Dottorato di Ricerca Internazionale
in
Ingegneria Agraria
XXVI CICLO

Andrea Giummarra

UWB-BASED REAL-TIME LOCATION SYSTEM PERFORMANCES FOR COW IDENTIFICATION AND LOCALISATION AND COW'S LOCATION DATA ANALYSIS AND MANAGEMENT IN FREE-STALL BARNS

Tesi per il conseguimento del titolo di
Dottore di Ricerca

Tutor:
Ch.mo Prof. Claudia Arcidiacono

Coordinatore:
Ch.mo Prof. Consoli Simona

Università degli Studi di Catania
Dipartimento di Gestione dei Sistemi Agroalimentari e Ambientali
Sezione Costruzioni e Territorio
Catania, dicembre 2013

UWB-BASED REAL-TIME LOCATION SYSTEM PERFORMANCES FOR COW IDENTIFICATION AND LOCALISATION AND COW'S LOCATION DATA ANALYSIS AND MANAGEMENT IN FREE-STALL BARNS

Andrea Giummarra

Dicembre 2013

To Cristina.
My energy, my smile, my life and soon...
...my wife.

Contents

List of Figures

List of Tables

Abstract

The main objective of this study was to evaluate the localisation and identification performances of a Real-Time Location System (RTLS) based on Ultra Wide Band (UWB) technology within a free-stall barn which represents a particularly harsh environment for the functioning of this kind of system. Each dairy cow was equipped with an active tag applied to one ear. A video-recording system was installed in the barn to perform the assessment of the RTLS. Top-view camera images of the area of the barn under study were rectified and synchronised with the RTLS. Each position of the cow computed by the RTLS was validated by performing cow visual recognition on the camera images.

To perform this validation a software specifically designed for the purpose was utilized. It is a quick, accurate, automatic and interactive tool which includes selection and control tabs for data management, visualization and labelling of the images with the aim of the computation of the tag true positions.

Localisation and identification performances of the RTLS were assessed by applying an outlier data cleaning technique to tag localisation errors and using precision and sensitivity indices. Trade-off between these performances was found through the computation of three performance metrics.

The results showed that, in the environmental conditions of the barn, the RTLS produced errors which were comparable to the errors declared by the RTLS producer for the fixed reference tag whereas localisation errors related to the tags in movement were higher and, in detail, a mean error of 0.56 m and an error at the 90th percentile of 1.03 m were obtained.

Outlier data cleaning produced significant improvements of the results by reducing the average localisation error of about 0.046 m for the cows' tags and made data distribution more homogeneous. Trade-off of RTLS performances yielded an average localisation error equal to about 0.52 m with a position accuracy of 98% for cows' tag and an error of about 0.11 m with a nearly 100% accuracy for the reference tag.

RTLS performances in the considered environment proved to be generally not dependent on cow behaviour, as it is observed for other systems, and that RTLS is suitable to determine the occupancy level of the different functional areas of the barn, compute cow behavioural indices, and track each animal of the herd.

An automatic and real-time software tool for the visual analysis of the cows' location data in free-stall barns acquired by the RTLS was designed and developed. A visual representation used for visualizing two dimensional data of each cow using colours was implemented as a functionality of this tool. The different colour and colour intensity denoted the difference in sample density at a location. Measurement of the instant speed of each cow over the time, represented through an interactive graph, was another functionality implemented in this software.

The results obtained by a use case of this software tool, which was carried out in order to acquire useful informations related to the occurrence of estrus, showed that a pattern, related to the behaviour of the cow analyzed, can be identified when the state of estrus occurs. Moreover, since it was designed to monitor cow behavior in real time, it offers the ability, by adding new control modules, to notify any inconvenience through alert messages as a result of changes in dairy cow behaviour and, therefore, it is possible to alert the farmer in real time.

Keywords: UWB tag; RTLS; Animal behaviour; Behavioural indices; Cow localisation; Cow speed analysis.

Part I

INTRODUCTION

Chapter 1

PREFACE

The increasing demand for livestock products such as meat, eggs, milk and derivatives, represents an important issue in today's society and has significant implications on agricultural production methods. Industrialisation of agricultural production, in fact, represents a direct consequence of this concept by assuming an ever-increasing scale directly linked to globalisation.

Intensive livestock farming involves thousands of animals, such as cows, pigs, laying hens and broilers, in highly controlled conditions. However, nutritional systems based on products of industrial nature are often adopted and housing systems are organised in functional areas which are not properly managed or provide insufficient space to the animals. These conditions are able to increase the risk of causing health problems. Therefore, the population growth of animals raised in such circumstances, leads to a higher probability of creating zoonotic pandemics, i.e. animal infectious diseases that can be transmitted to humans, such as avian flu or swine flu. The monitoring of animal behaviour should involve the control of growth and production of the herd but also include control of these forms of disease, transmitted through direct contact with the skin, hair, eggs, blood or secretions or, indirectly, through ingestion of contaminated food.

Another problem which is strictly correlated to the prevention of diseases and the treatment of infections is the use of antibiotics. Antibiotics, however, are used at the same time as a tool to enhance and accelerate growth of the animals. This management practice often causes excessive use of these substances. The direct consequence of this phenomenon is that certain bacteria become resistant to these antibiotics, surviving at the exposure to these substances. Therefore, the animals affected by certain diseases could not profit from the therapies adopted and tend to increase, in general, the rate of unproductive animals or even mortality rate of the herd.

New methods that aim at eliminating the use of such substances have been recently introduced in order to reduce costs related to the treatment of animal diseases and adoption of antibiotics to accelerate their growth. The development of new approaches based on the use of automated monitoring systems has featured a highly technological breakthrough capable of ensuring a growing condition of animal welfare and, therefore, food safety for consumers (Tullo et al., 2013).

Within these new approaches, Precision Livestock Farming (PLF) is a technology that includes and promotes techniques (based on sensors, contactless sensing with image analysis or sound analysis, wireless data transmission, traceability techniques, etc.) to make smart farming a reality and create added value for many stakeholders: animals, farmers, veterinarians, feed and product suppliers, health services, policy makers, the media and of course the consumer (Berckmans, 2013).

According to Wathes (2010), PLF relies on four essential elements:

1) the continuous sensing of the process responses at an appropriate frequency and scale with a continuous exchange of information with the process controller;

2) a compact mathematical model, which predicts the dynamic responses of each process output to variation of the inputs and can be and is best estimated online in real time;

3) a target value and/or trajectory for each process output, e.g. a behavioural pattern, pollutant emission or growth rate;

4) actuators and a model-based predictive controller for the process inputs.

In general, the reliability of the PLF is determined mainly by the animals and from all the relevant physiological variables that can be measured continuously, as the weight, the respiratory rate or heart rate, body temperature, motor activity, behavior, the produced noise, food intake, etc. Possible approaches to automatic monitoring systems can be based on sound, images, location and environmental data collection.

Information Technology (IT) is continuously doing significant progress in terms of technical efficiency. In particular, the new techniques of electronic design, reduction of size and power consumption of the devices, and the growing diffusion of open source software have made the technology affordable and accessible. When IT and the sensors are combined with the use of internet, it's possible to implement new technologies able to support the farmer by providing him with early warning systems for the automatic detection of potential production, health and welfare problems (Tullo et al., 2013).

Today, internet is used as a global platform for machines and human communication (web, chats, emails, etc.) and for the interconnection of smart objects. As a result, it is expected that in the near future, there will be a large class of objects natively interconnected which will allow for new uses, including new ways of working, new ways of entertainment and animation, new ways of living, and, last but not least important, new ways of farming. A recently emerging computing concept, namely Internet of Things (IoT), could be included as part of this scenario, bypassing the interoperability problems often typical of highly specialized proprietary devices, which are generally designed to be integrated into systems that are extremely 'closed' themselves. This new approach considers all sources of information (RFID tags, Real-Time Location Systems, sensors, actuators, mobile phones, etc.), such as assets, which are uniquely indexed by an address assigned by a centralized system that can interact and cooperate with each of them to achieve common goals (Scalera et al., 2013).

Chapter 2

OBJECTIVE OF THE STUDY

In the research described in this thesis work, a Real-Time Location System (RTLS) based on the UWB technology was utilised for the identification and localisation of a group of dairy cows housed in a free-stall barn. The RTLS used in this work (Ubisense, UK), is currently recognised to provide one of the highest accuracies of all the RTLSs (Linde, 2006; Weichert et al., 2010). The achievable accuracy, which Ubisense declares for the localisation of moving objects in real time, is 15 cm in the three dimensions (x, y, z), i.e., about 12.25 cm in two dimensions. However, in field tests accuracy of the Ubisense system was found to vary between 30 cm and 100 cm in the two dimensions x and y in dependence on the applications, e.g. agriculture, transit yard management, and personnel safety (Mok et al., 2010; Ward, 2010).

In this context, the main purpose of this thesis was to evaluate the RTLS performances in the identification and localisation of cows during feeding and lying behaviours in the breeding environment considered, which included the resting area and the feeding area of the barn. To this aim, RTLS performances need to be analysed for fixed tags applied to the building structure and for tags in movement applied to the cows.

A further aim of this research was the reduction of localisation errors through the application of data cleaning techniques and trade-off of RTLS identification and localisation performances.

Then, this research study considered the analysis of possible applications of the RTLS to the study of dairy cow behaviour.

On the basis of these considerations, another objective of this study was the development of an automatic and real-time software tool for the visual analysis of the cows' location data in free-stall barns acquired by the RTLS, designed in order to have the features needed to be integrated into the world of IoT.

The developed of a use case of this software tool, applied to data acquired by the RTLS in the same free-stall barn where RTLS performances were evaluated, was the ultimate goal of this thesis. The purpose of this test was to assess the performances of the implemented features and to verify the effectiveness of the results obtained from the use of the tool in order to find useful informations related to the occurrence of the physiological state of estrus.

Chapter 3

WORK ORGANIZATION

Part II of this thesis work shows a review of studies carried out in the field of precision livestock farming. In particular, ICT applications for indoor animal identification and localisation are listed in order to illustrate the state of the art in the introduction of new approaches based on the use of high-tech automated monitoring systems in the study of animal behaviour.

Part III contains the materials and methods of the research. In detail, chapter 5 describes the properties of an Ultra Wideband Real-Time Location System. In the last section (5.3) attention was focused, in the description of the system adopted in this research. Furthermore, this part gives a detailed description of the case study (chapter 6) carried out within of a dairy house located in the province of Ragusa (Sicily, Italy). In particular, after a description of the area of the barn under study (section 6.1), the analysis and installation of the Ubisense UWB RTLS (section 6.2) are outlined in detail. Subsequently, the multi-camera video-recording system (section 6.3), adopted to assess the accuracy of the localisation and identification achieved through the use of the RTLS, is described. Finally, section 6.4 shows the materials and methods relative to the computation of localisation and identification performances of Ubisense UWB RTLS, while, section 6.5 describes the automatic and real-time software tool developed for the visual analysis of the cows' location data in free-stall barns acquired by the RTLS.

Part IV describes the results achieved in this research study. In particular, chapter 7 shows results relative to the study of localisation and identification performances of the RTLS (section 7.1) and those relative to the software tool for the visual analysis (section 7.2). Finally, chapter 8 reports in detail the comments on obtained results by making comparisons with other similar works and the possible further improvements of the research in this field.

Finally, part V, is dedicated to the outcomes achieved in the research work described in this thesis.

Part II

STATE OF THE ART

Chapter 4

ICT APPLICATIONS FOR ANIMAL IDENTIFICATION AND LOCALISATION INDOOR

The automatic detection of changes in dairy cows behaviour, combined with the concept of IoT introduced in chapter 1, can be used as input to non-invasive warning systems which alert the farmer when some health problems (e.g., lameness) or a particular physiological status (e.g., estrus) occur. In this context, the real-time identification and localisation of each dairy cows of the herd play an important role and several studies in the field of precision livestock farming have been carried out.

4.1 Animal identi cation

With regard to cow identification, Rorie et al. (2002) studied the application of commercially available electronic estrus detection technologies to reproductive management of cattle and demonstrated that pedometers are the most applicable devices to lactating dairy cattle and have a higher accuracy and efficiency when combined with visual observation. Also Roelofs et al. (2005) adopted a pedometer for oestrous detection as predictor for time ovulation in dairy cattle and demonstrated that pedometers can accurately detect estrus, though not in real time.

The collection of estrus data in real time is of considerable importance to avoid delayed inseminations of cows which could reduce cow fertilization rate and, as a consequence, have negative economic impacts on farm budget and costs. To this aim, recent research (Brehme et al., 2008; Jónsson et al., 2011) achieved excellent results by adopting real-time pedometers for estrus detection in dairy cows. However, pedometers do not allow cow localisation and cannot be used to distinguish some behavioural patterns (e.g., standing vs. feeding/perching) that are crucial data as it has been proved that the daily incidence of standing behaviour is of considerable importance in estrus detection (Firk et al., 2002) and early diagnosis of lameness (Pastell et al., 2009).

Cow identification has also been frequently obtained by means of Radio Frequency Identification (RFID) technology which makes use of two main electronic components, tags and readers, that exchange information through radio waves. Schwartzkopf-Genswein et al. (1999) adopted a RFID system (GrowSafe System R) for monitoring the feeding patterns of feedlot cattle and demonstrated that RFID technology provides the ability to better understand feeding patterns, thereby aiding in the selection of those management practices that improve animal performance. The same system was adopted by Sowell et al. (1998) who demonstrated that RFID technology has the potential to identify sick animals and improve feedlot health management practices. In another study, Voulodimos et al. (2010) utilised RFID tags for the storage of information suitable to identify the animal (sheep, cattle, pigs, etc.) in case it gets lost, or even collect some basic data (e.g. behaviour against other animals). In the same year, Samad et al. (2010) demonstrated that the use of RFID-based identification and data retrieval offers the value-added benefit of data security and would prevent insurance-related frauds. Ilie-Zudor et al. (2011) carried out a survey of applications and requirements of identification systems and RFID techniques. They reported that a new cattle tracking system, which was introduced by the Spanish cattle farmers' association FEVEX, utilized RFID tags injected into the cows' hooves to allow a quick and accurate animal data retrieval. Furthermore, the Canadian Cattle Identification Agency replaced barcode tags with RFID tags to identify a bovine's herd of origin and this was used for the traceability of packing plant meat. Finally, RFID tags were used by Feng et al. (2013) to perform tracking and storing of dairy cattle breeding data in order to implement a real-time traceability management system along cattle production flow.

4.2 Animal localisation

With regard to cow localisation, some research studies investigated the possibility of using different indoor positioning systems for cow tracking. Huhtala et al. (2007) demonstrated that a system which uses Wireless Local Area Network technology and Time Difference of Arrival (TDoA) algorithms provided a localisation error of about 1 m (with a position accuracy of 70%)

in undisturbed conditions (no moving cows, clear sights between antennae and tags, etc.). Tøgersen et al. (2010) used the Bluetooth wireless technology to monitor dairy cows in a barn and demonstrated that is possible to locate a cow with a precision of approximately 0.6 m in stationary conditions. Ipema et al. (2013) obtained an average accuracy of 0.30 m with a standard deviation of 0.25 m, in a static accuracy test carried out in a cattle barn by applying an indoor localisation system based on Ultra-High Frequency (UHF) technology.

Another group of studies (Barbari et al., 2008; Porto et al., 2012; Reiners et al., 2009) has been carried out with the aim of demonstrating that RFID technologies based on High Frequency (HF) and UHF could locate individual animals inside specific functional areas of buildings for intensive animal breeding. The outcomes of these studies demonstrated that it possible to detect the animals in specific functional areas of the breeding environment (e.g., feeding area and resting area), though high localisation errors are found with these systems.

The localisation accuracy provided by RFID systems based on HF and UHF technologies could be improved by using Ultra Wide Band (UWB) technology (Álvarez & Cintas, 2010). The advantage of this technology is provided by the signal transmission mode which takes place by means of short duration pulses. This mode of transmission provides UWB-based systems with a low sensitivity to interference due to the absence of reflection of the wave itself. Furthermore, the low radiation used in UWB technology has allowed utilisation of a UWB system for remote monitoring and measuring the patients' motion in short distance within hospitals (Yong et al., 2007).

Part III

MATERIALS AND METHODS

Chapter 5

THE ULTRA WIDEBAND REAL-TIME LOCATION SYSTEM

As already seen in chapter 4, localisation accuracy provided by RFID systems based on HF and UHF technologies could be improved by using Ultra Wide Band (UWB) technology (Álvarez & Cintas, 2010). The advantage of this technology is provided by the signal transmission mode which takes place by means of short duration pulses. This mode of transmission provides UWB-based systems with a low sensitivity to interference due to the absence of reflection of the wave itself.

In this chapter the features of an ultra wide band real-time localisation system are described in detail.

5.1 ULTRA WIDEBAND SYSTEMS

A traditional UWB system is composed of two electronic components, tags and readers, which exchange information through radio waves and UWB pulses. A tag is a device capable of transmitting a unique identification number when requested by a particular signal; a reader is a device that, by sending radio waves, urges the transmission of tag and receives the information transmitted. The tags can be active, passive, semi-passive or semi-active.

A system using UWB technology for telecommunications and for localisation has the following features:

Simplicity of implementation: in a UWB system the modulation of the signal, i.e. the set of techniques with which it is possible to adapt, to the communication channel used, an electrical or electromagnetic signal to be transmitted without altering its information content, takes place directly in the antennae. Such feature fulfill, from the point of view of the manufacturers, the ability to realize inexpensive receivers (Di Noia, 2010).

High bit-rate: bandwidth occupied by UWB systems allows the transmission of large quantities of data in a given time interval (*bitrate*), reaching, in short distances, a speed greater than or equal to 0.5 Gbit/s, which is a decidedly high speed considering that it is a wireless telecommunications system (Di Noia, 2010).

Immunity to multipath signals: UWB pulses are pulses in broadband radio frequency of extremely limited duration (from a few tens of picoseconds to a few nanoseconds). This feature allows a better resolution of the multiple paths. In fact, due to the short pulse duration, it is unlikely that a reflected signal interferes with a direct one. Therefore it favours the identification and subsequent exclusion of the reflected signals. This property of UWB signals is the main reason for which this technology is used by the Real-Time Location Systems (RTLS) for locating the objects in closed environments (or with high density of obstacles) (Di Noia, 2010).

Ability to penetrate through obstacles: the absorption spectrum of various materials (metals in particular) is concentrated on narrow frequency bands. The UWB signals, as already seen, are broadband signals that are not strongly attenuated by the obstacles, unlike other narrow band signals (Di Noia, 2010).

Localisation and communication simultaneously: perform simultaneously locating operations and data transmission in communication systems usually involves the presence of numerous collisions between the two traffic flows, with the result to obtain a low bit-rate and a high imprecision in the estimation of distances. The use of an extremely wide bandwidth, as that of UWB, make it possible to divide the traffic flows properly and avoid collisions (Di Noia, 2010).

Security: the very low level of power emitted by the tags delivers high security within a system that uses UWB technology. A UWB tag transmits, in fact, at a power level of 10000 times lower than a typical mobile phone (Domain, 2013).

No interference with other devices: the low power and wide range of frequencies used for a UWB signal ensures the almost complete absence of interference with other wireless technologies such as Wi-Fi, Bluetooth, UHF or handheld passive readers (PDA passive readers) (Figure 5.1). Tests showed that the interference impact on UWB from a wireless technology like Wi-Fi is essentially negligible ($< 0.3\%$) (Domain, 2013).

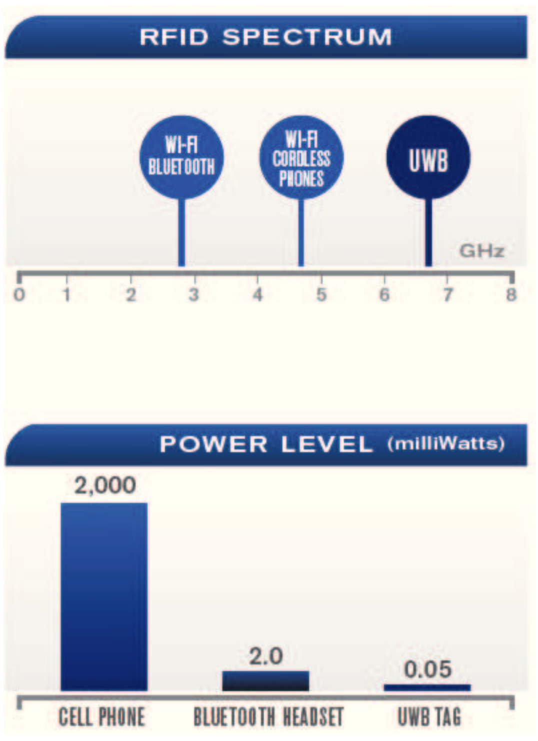


Figure 5.1: Frequency spectrum and power level of UWB signal in comparison to other wireless technologies (Domain, 2013).

A system based on UWB technology is, therefore, characterized by high bandwidth pulses (greater than 500 MHz) for communication and/or for the localisation, obtaining high transmission speed, a high resolution of the multiple paths, high robustness to interference with other wireless technologies and the presence of obstacles, high security and low cost of implementation.

5.1.1 Ultra Wideband signals

According to the Federal Communications Commission (FCC) (FCC, 2002), an UWB signal is defined as any signal that has a fractional bandwidth greater than 0.2 or occupies 500 MHz or more of spectrum. The main feature of UWB signals is that they occupy a much wider frequency band than conventional signals; hence, they need to share the existing spectrum with incumbent systems.

5.1.1.1 Signal bandwidth

The absolute bandwidth is calculated as the difference between the upper frequency f_H of the -10 dB emission point and the lower frequency f_L of the -10 dB emission point:

$$B = f_H - f_L, \tag{5.1}$$

which is also called -10 dB bandwidth (Figure 5.2). On the other hand, the fractional bandwidth is defined as

$$B_{frac} = \frac{B}{f_c}, \tag{5.2}$$

where f_c is the center frequency and is given by

$$f_c = \frac{f_H + f_L}{2}. \tag{5.3}$$

From 5.1 and 5.3, the fractional bandwidth B_{frac} in 5.2 can be expressed as

$$B_{frac} = \frac{2(f_H - f_L)}{(f_H + f_L)}. \tag{5.4}$$

In conclusion, as already mentioned before, according to the FCC, an UWB system with f_c larger than 2.5 GHz must have an absolute bandwidth larger than 500 MHz, and a UWB system with f_c smaller than 2.5 GHz must have a fractional bandwidth larger than 0.2 (Figure 5.2) (Sahinoglu et al., 2008).

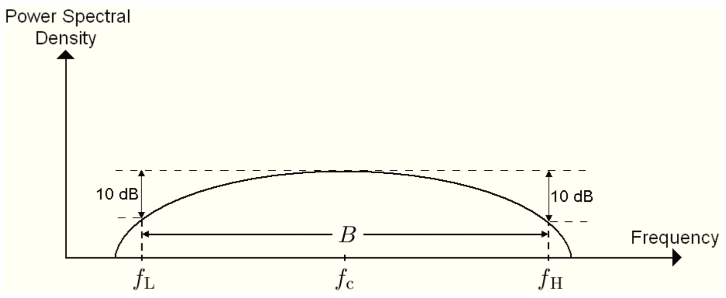


Figure 5.2: Characteristics of a UWB signal: absolute bandwidth B is at least 500 MHz, or fractional bandwidth $b_{frac} = B/f_c$ is larger than 0.2 (Gezici et al., 2005).

5.1.1.2 Shape of the impulse

The duration of the waveform in a UWB system is typically very short, usually on the order of a nanosecond (from 0.25 ns to 1.25 ns), due to its large bandwidth. These ultra-short pulses have a low duty cycle. In other words, the ratio between the pulse transmission instant and the average time between two consecutive transmissions is usually kept small. The typical shape of the pulse used in the UWB technology is that of the double Gaussian (gaussian doublet). Some common UWB pulse shapes include derivatives of the Gaussian pulse, wavelet pulses, Raised Cosine pulse and pulses based on modified Hermite polynomials. For example, the second derivative of the Gaussian pulse is expressed as

$$\omega(t) = A\left(1 - \frac{4\pi t^2}{\zeta^2}\right)e^{-2\pi t^2/\zeta^2}, \tag{5.5}$$

where $A > 0$ and ζ are parameters that determine the energy and the width of the pulse, respectively. In Figure 5.3, a unit energy pulse is plotted according to 5.5 ($\zeta = 0.4$ ns and width of around 1 ns) (Sahinoglu et al., 2008).

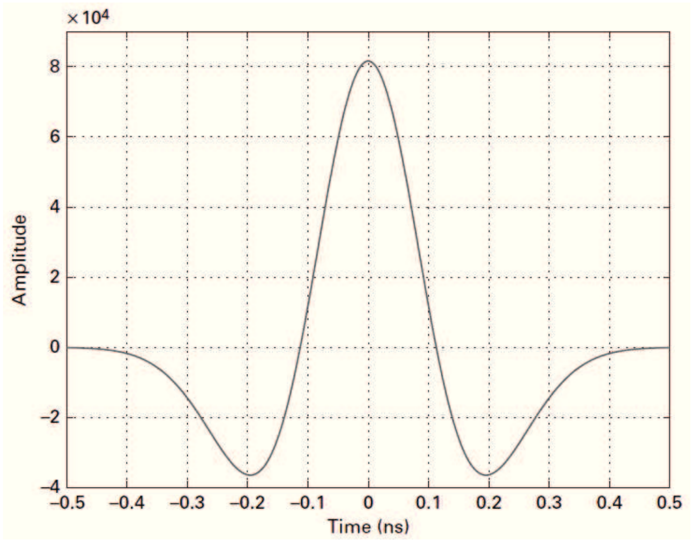


Figure 5.3: The UWB pulse (gaussian doublet) according to Equation 5.5 with $\zeta = 0.4$ ns and width of around 1 ns (Sahinoglu et al., 2008).

5.1.1.3 Spectral analysis

A communication can not be established by sending a single pulse since it alone is not capable of transmitting a quantity of information. Therefore, a train of pulses, that appear randomly in time to optimize the spectrum usage, represents a transmission that carries information (Figure 5.4) (Mei, 2003).

Due to their very large portion in the frequency spectrum, UWB signals are likely to cause interference with other systems. As seen in Figure 5.4, the pulse train frequency takes the form of a noise signal (noise like) which minimizes the effect of the interference of the UWB signal if there are other signals that have been transmitted within the bandwidth occupied by UWB system (Di Noia, 2010). For example, frequency allocation of some wireless systems is shown in Figure 5.5 (Sahinoglu et al., 2008).

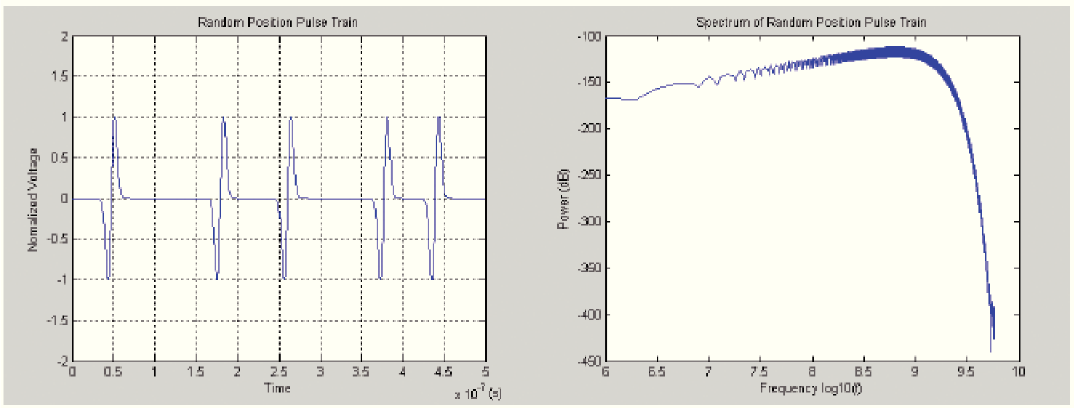


Figure 5.4: Random positioned pulse train and its frequency spectrum (noise like) (Mei, 2003).

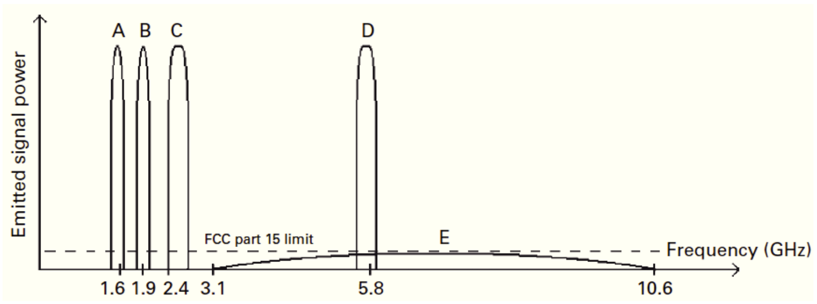


Figure 5.5: Frequency spectrum allocation of various wireless systems. A: Global Positioning System (GPS) (1.56 - 1.61 GHz), B: Personal Communication System (PCS) (1.85 - 1.99 GHz), C: Microwave ovens, cordless phones, bluetooth, IEEE 802.11b (2.4 - 2.48 GHz), D: IEEE 802.11a (5.725 - 5.825 GHz), E: UWB (3.1 - 10.6 GHz). The bandwidths and power levels of various systems are not drawn to scale (Sahinoglu et al., 2008).

If UWB signals were allowed to transmit over the range of frequencies of these systems without any restrictions, all these systems could be jammed by UWB emission. Therefore, the power spectral density must not overcome - 41.3 dBm/MHz for frequency ranges from 3.1 to 10.6 GHz, and it must be even lower outside this band, depending on the specific application. In other words, the FCC spectral mask specifies a useful spectrum of 7.5 GHz for most UWB systems (Sahinoglu et al., 2008).

5.2 LOCALISATION IN ULTRA WIDEBAND SYSTEMS

5.2.1 Properties of UWB indoor localisation systems

As regards the concept of positioning and functioning of the UWB system is necessary to carefully analyze the environment in which localisation is carried out, to evaluate some variables that may affect the system and fix the limits of the area to be analyzed in relation to the capacity of the instrument.

A unit of measure for distance must be established in order to determine the location of a target moving within such an environment. Furthermore, a two-dimensional or three-dimensional reference system must be defined in the environment in order to locate the target. For instance, a system of Cartesian or polar coordinates can be used in order to describe the x, y and z coordinates or latitude, longitude and altitude, respectively. In some cases it may be more appropriate to adopt a location logical model. For example, in a context in which we want to determine the presence or absence of a cow in a particular functional area within a livestock environment, we can use a labeling system (feeding alley, service alley, bunks, etc.) to refer to such areas. This logic model can be derived from a location geometric model.

The scale of the localisation system must not exceed the threshold of a few hundred meters. The localisation systems adopted in a restricted area are usually referred to as Local Positioning Systems (LPS), unlike to those used on a large scale such as 2G/3G cellular network positioning systems, or the Global Navigation Satellite Systems (GNSS). The indoor location systems fall into the category LPS.

In general, the positioning can be performed using relative or absolute localisation methods, or even by making use of both methods (Linde, 2006).

A further analysis of the environment concerns the study of the physical features, i.e. environmental conditions where we want to implement the localisation system. It is important to take into account the obstacles and the material of which they are made (e.g., if there are cement walls, metal doors, stairs, beams, etc.), in order to determine the possible degree of reflection, scattering and attenuation they produce, in the presence of electromagnetic waves. Furthermore, a study on the level of air temperature and relative humidity, which should be suitably measured, is recommended in order to determine the parameters necessary for the choice of the location technology to be adopted.

5.2.2 Performance measures in UWB indoor localisation

A localisation system should fulfill certain features relating to its performance in order to be considered suitable for their intended use.

The following parameters are the main performance measures:

Accuracy. Measurement accuracy reflects the closeness between the measurement result and the true value of the measurand. Accuracy is a positive characteristic of the measurement, but in reality it is expressed through a dual negative characteristic - inaccuracy - of the measurement. The inaccuracy reflects the unavoidable imperfection of a measurement. The inaccuracy of a measurement is expressed as the deviation of the measurement result from the true value of the measurand (this value is called measurement error) or as an interval that covers the true value of the measurand (Rabinovich, 2010).

Precision. This parameter indicates the degree of reproducibility of a measurement. Typically, the accuracy is considered as an additional measure of accuracy (for example, 15 cm on the accuracy of 90% of cases). The precision of a measurement system is the degree to which repeated measurements under unchanged conditions show the same results. Two components of the precision are the reproducibility and the repeatability, described below.

Reproducibility: The quality of measurement that reflects the closeness of the results of measurements of the same quantity performed under different conditions, i.e., in different laboratories (at different locations) and using different equipment, is called the reproducibility of measurement. Good reproducibility indicates that both the random and systematic errors are small (Rabinovich, 2010).

Repeatability. The quality of measurement that reflects the closeness of the results of measurements of the same quantity performed under the same conditions is called the repeatability of measurements. Good repeatability indicates that the random errors are small (Rabinovich, 2010).

Recall. The recall, also called sensitivity or true positive rate, measures how often we find what we are looking for. In other words is the proportion of actual correct measures which are correctly identified as such.

Cost. In cases where a substantial number of elements must be equipped with devices for the localisation, the cost is an important aspect. A good remedy is the use of devices equipped with temporary sleep in situations in which the measurement is not needed (for example, when the object to be localised is in a stationary position, the device may set itself automatically in a state of stand-by, until it receives the next movement through appropriate motion sensors) in order to optimize the costs due to the consumption of the batteries.

Power consumption vs update rate. An important detail also relates to the maintenance of an optimum trade-off between the energy consumption of the batteries and the update frequency of the measurement. A solution to this problem requires the ability to set an update rate, regarding the acquisition of the measurement, which ensures the battery life within the desired time period but which respects, at the same time, the desired update frequency of the measurement.

The estimation cycle length. This feature in particular cases is not to be considered of minor importance. Depending on your goals, in fact, can be a very important factor. Just think in certain situations in which you must automate the occurrence of an event as a result of another directly connected. For instance, when activating an actuator, such as the closure of a fence following the movements of the animals located within a livestock environment whose movements are tracked by a localisation system, an update cycle rather long may cause delay in the closure of the fence.

Scalability. A good localisation system must ensure a trade-off between the extensibility of coverage, in cases in which you want to add additional space to perform the localisation in neighboring areas, and the capacity to be able to locate a growing number of the mobile target simultaneously.

Robustness. This parameter for evaluating the performance of a localisation system expresses the ability of a system to be immune to interference with other systems, but also to the variations of the environmental features, such as temperature and humidity.

Device size and weight. Especially in cases where we want to locate animals or humans, it is important to take into consideration the size and weight of the transmitter devices (active tags) that need to be transported. If these are bulky and heavy, the natural behavior of users could be modified (Linde, 2006).

5.2.3 Positioning

UWB systems, thanks to the brevity of the pulse that characterizes them, are ideal candidates for combining comunication and positioning at the same time. The duration of a pulse is inversely proportional to the bandwidth of the transmitted signal. If the arrival time of a pulse is known with little uncertainty, then it is possible to accurately estimate the distance covered by the impulse from the source. Combining the distances estimated by multiple receivers, we can use simple triangulation techniques to estimate the position of the source.

For UWB systems with a bandwidth of 7.5 GHz, the maximum resolution time of the pulse is of the order of 133 picoseconds. Therefore, when a pulse is detected, it is possible to know within 133 picoseconds the 'time-of-flight' impulse with an uncertainty of 4 cm. For systems with a bandwidth of 500 MHz, however, the corresponding time resolution is 2 nanoseconds, which corresponds to an uncertainty space of about 60 cm. With any of the UWB signal, therefore, it is potentially possible to obtain an accuracy less than one meter in the localisation (Oppermann et al., 2005).

There are several techniques to determine the position starting from estimates of the time-of-arrival, time-of-flight or angle-of-arrival. In the next section some of them will be analysed.

5.2.3.1 Positioning techniques for UWB systems

Most of the positioning schemes estimate the position of objects to locate establishing geometric relationships between the transmitters and the receivers. This is achieved through special techniques that are based on timing information (Time-of-Arrival (ToA) and Time Difference of Arrival (TDOA)), Received Signal Strength (RSS), and Angle of Arrival (AoA). There also exist approaches that utilize location-dependent characteristics, instead of geometric relations, of these measurements to find the position. Such techniques are usually called location fingerprinting (Yan, 2010).

Some techniques for UWB positioning systems are analysed in the following of the thesis. The object to be localised (transmitter) is defined as the target node and the receivers as reference nodes.

Angle of Arrival (AoA)

An AoA measurement provides information about the direction of a signal sent from a target node and received by a reference node, particularly about the angle between the two nodes, as shown in Figure 5.6.

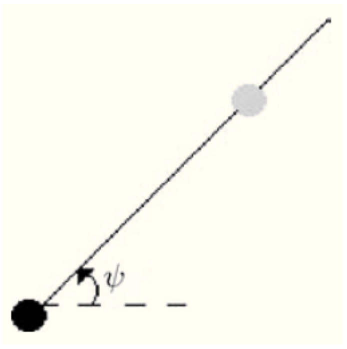


Figure 5.6: Definition of AoA between two nodes: the reference node (black node) measures the AoA by determining the angle ψ between itself and the target node (gray node).

Commonly, antenna arrays are employed in order to measure the AoA of a signal. The information about the angle is obtained at an antenna array by measuring the differences in arrival times of an incoming signal at different antenna elements. An example is illustrated in Figure 5.7 for AoA estimation at a uniform linear array (ULA).

When the distance between the target and the reference nodes is sufficiently large, the incoming signal can be modeled as a planar wave-front. This results in $l\ sin\ \psi/c$ seconds difference between the arrival times at consecutive array elements, where l is the inter-element spacing, ψ is the AoA and c represents the speed of light. Therefore, estimation of the time differences of arrivals provides angle information. More advanced array structures, such as uniform circular arrays (UCAs) and rectangular lattices, operate on the same basic principle as the ULA.

For a narrowband signal, time difference can be represented as a phase shift. Therefore, the combinations of the phase-shifted versions of received signals at array elements can be tested for various angles in order to estimate the direction of signal arrival.

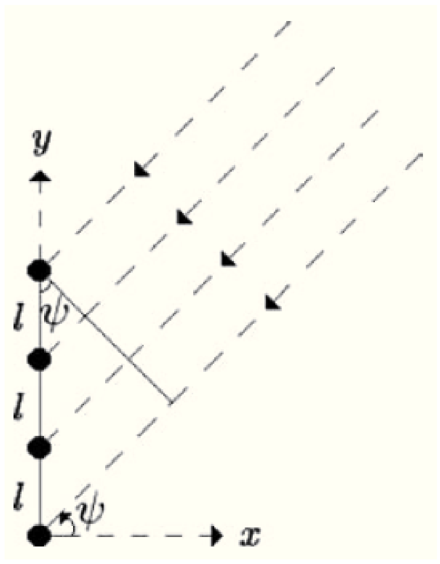


Figure 5.7: Signal arrival at a ULA having a spacing l between the array elements, and relation between arrival time differences and AoA.

In order to obtain theoretical lower bounds on the achievable accuracy of AoA measurements, consider a ULA, as shown in Figure 5.7, with N_a antenna elements. Let $r_i(t)$ denote the received signal at the ith element, which is expressed as

$$r_i(t) = \alpha s(t - \tau_i) + n_i(t), \tag{5.6}$$

for $i = 1, ..., N_a$, where $s(t)$ is the transmitted signal, α is the channel coefficient, τ_i is the delay for the signal arriving at the ith antenna element, and $n_i(t)$ is white Gaussian noise with zero mean. The delay τ_i can be expressed as

$$\tau_i \approx \frac{d}{c} + \frac{l_i sin\psi}{c}, \tag{5.7}$$

with

$$l_i = l\left(\frac{N_a + 1}{2} - i\right), \tag{5.8}$$

for $i = 1, ..., N_a$, where d is the distance between the transmitter and the center of the antenna array at the receiver, and l is the inter-element spacing (Sahinoglu et al., 2008).

Time difference of arrival (TDoA)

TDoA measurements can be obtained even in the absence of synchronization between the target node and the reference nodes, if there is synchronization among the reference nodes. In this case, the difference between the arrival times of the signals traveling between the target node and the reference nodes is estimated. The setup of the resulting localisation principle which is commonly referred to as 'hyperbolic localisation' is shown in Figure 5.8.

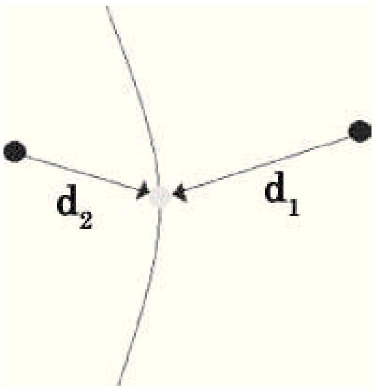


Figure 5.8: Hyperbolic localisation scheme in TDoA measurements.

One way to obtain a TDoA measurement is to estimate the ToA[1] at each reference node and then to obtain the difference between the two estimates. Specifically, if the received signals are given by $r_1(t)$ and $r_2(t)$ as in (5.6), τ_1 is estimated from $r_1(t)$ and τ_2 is estimated from $r_2(t)$. Since the target node and the reference nodes are not synchronized, the ToA estimates at the reference nodes include a timing offset in addition to the time of flight. As the reference nodes are synchronized, the timing offset is the same for each ToA estimation. Therefore, the TDoA measurement can be obtained as

$$\tau_{TDoA} = \tau_1 - \tau_2, \tag{5.9}$$

where τ_1 and τ_2 denote the ToA estimates at the first and second nodes, respectively.

Another way to obtain a TDoA measurement is to perform cross-correlations of the received signals $r_1(t)$ and $r_2(t)$, and to calculate the delay correspond-

[1]A ToA measurements provide information about the distance between two nodes by estimating the time of flight of a signal that travels from one node to the other. To prevent ambiguity in ToA estimates, the two nodes must have a common clock, or they must exchange timing information via certain protocols, such as a two-way ranging protocol.

ing to the largest cross-correlation value. The cross-correlation function can be expressed as (Sahinoglu et al., 2008):

$$\pi_{1\,2}(\ \ ) = \frac{1}{T}\ {}_{0}^{T}\ r_1(t)r_2(t+\ \ )dt \tag{5.10}$$

where T is the observation interval, and the TDoA estimate is given by

$$\hat{}_{TDoA} = \arg\max\ \pi_{1\,2}(\ \ ) \tag{5.11}$$

5.3 UBISENSE ULTRA WIDEBAND REAL-TIME LOCATION SYSTEM

As already seen in chapter 2, the RTLS used in this thesis (Ubisense, UK), is currently recognised to provide one of the highest accuracies of all the RTLSs (Linde, 2006; Weichert et al., 2010). The achievable accuracy, which Ubisense declares for the localisation of moving objects in real time, is 15 cm in the three dimensions (x, y, z) (Steggles & Gschwind, 2005), i.e., about 12.25 cm in two dimensions. However, in field tests accuracy of the Ubisense system was found to vary between 30 cm and 100 cm in the two dimensions x and y in dependence on the applications, e.g. agriculture, transit yard management, and personnel safety (Mok et al., 2010; Ward, 2010).

In the next sections the Ubisense RTLS will be analyzed in detail.

5.3.1 Platform architecture

Ubisense has developed a platform for building *Smart Space applications*. This platform meets the basic requirements for building *Smart Spaces*: accurate 3D positioning, scalable realtime performance, development and deployment tools.

Accurate 3D localisation. In a typical open environment, a location accuracy of about 15cm can be achieved across 95% of readings.

Scalable real-time performance. The Ubisense RTLS is designed to support multiple, integrated, real-time location applications that work over large areas for large numbers of users.

Development and deployment tools. The Ubisense platform provides a number of visual configuration and development tools, which simplify the generation of end-user applications (Steggles & Gschwind, 2005).

Furthermore, Ubisense platform allows any location or sensor system to be plugged into its platform, protecting users being locked to a single vendor. This reduces the cost of ownership.

Ubisense platform architecture consists of three interconnected main 'layers': i) hardware components, ii) software middleware and iii) software applications, as shown in Figure 5.9 (Heathcote, 2011).

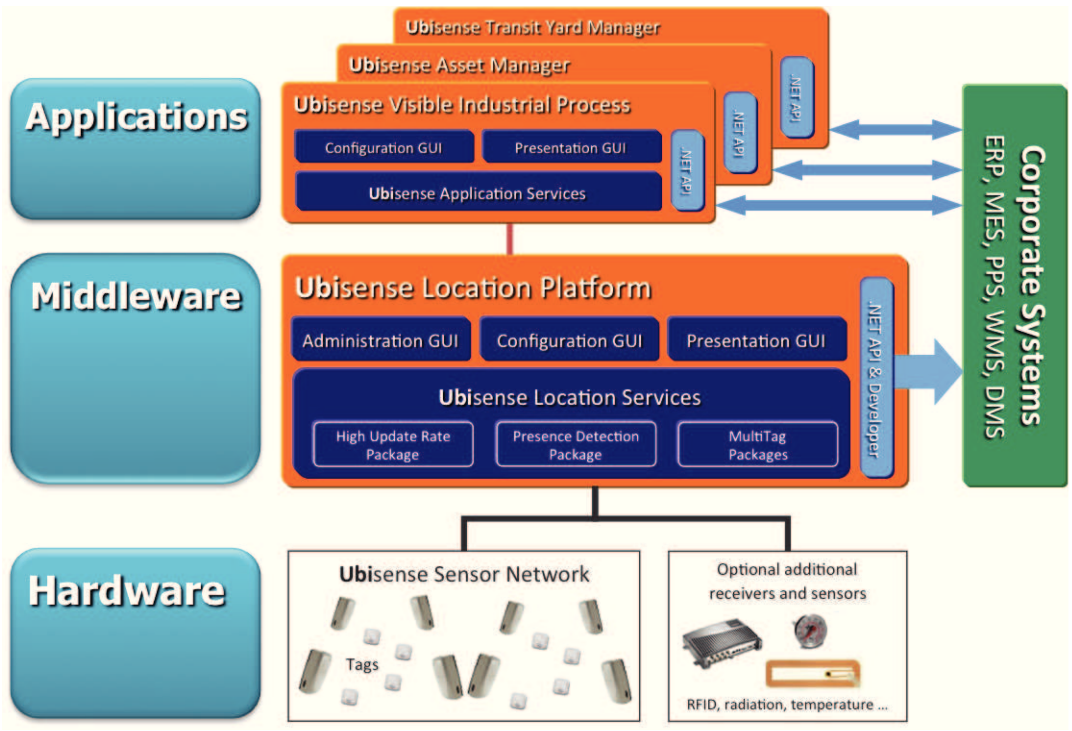


Figure 5.9: Ubisense platform architecture. Hardware level contains the Ubisense Sensor Network and optional receivers and sensors. Middleware level contains the Ubisense Location Platform which allows the interfacing of RTLS with Corporate Systems and the Applications level that contains the software applications customized to the use case in question (Heathcote, 2011).

5.3.1.1 Hardware architecture

The hardware architecture of the Ubisense system consists mainly of: sensors, tags and optional additional receivers and sensors.

Sensors

Ubisense sensors series 7000 (Figure 5.10, Table ??) are tools for precise measurements, based on an array of 5 UWB antennas to receive radio pulses.

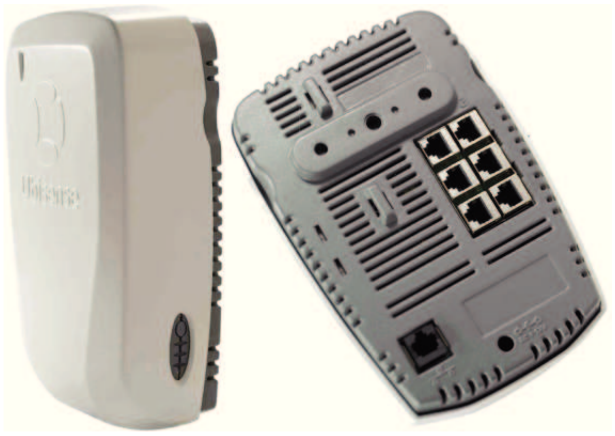


Figure 5.10: Ubisense Series 7000 IP30 Sensors.

The sensors calculate the position of tags on the bases of the reception of Ubisense UWB signals (pulses) transmitted by the tag .

Each sensor independently determines, via the antenna array, both the azimuth and elevation (AoA) of the UWB signal, obtaining a vector for each tag. The TDOA is determined between pairs (or more) of sensors connected to each other via a synchronization cable. This combination of measurement techniques (AoA and TDOA) provides a flexible tool for a powerful and robust localisation system, which allows both an accurate 2D position, determined by a single sensor, and a precise 3D position, determined by two or more sensors receiving the UWB signal pulse. This reduces infrastructure requirements, reducing costs, while achieving high reliability and robustness of the system.

Belowe there are some typical features of Ubisense sensors:

Reactivity in Real Time: Each sensor is able to sustain a continuous update frequency of 160 Hz, which means that the tag can be seen every 6.25 milliseconds from each sensor or cell.

Flexible and scalable installations: The Ubisense system uses cellular architecture for scalability, from small to large installations. Thousands of sensors can be integrated into a single enterprise wide system to monitor an unlimited area and manage thousands of tags.

Wired or wireless: The sensors can be connected with standard Ethernet cable or via Wi-Fi wireless adapters, by using the existing infras-

Sensors technical specification	
Dimensions:	20 cm x 13 cm x 6 cm
Weight:	650 g
Temperature range:	from 20 C to 60 C
Humidity range:	from 0 to 95%, non condensing
Standard operative range:	$\geqslant$ 160 m
Precision:	$\geqslant$ 15 cm in 3D
UWB channel:	6 GHz - 8 GHz
Telemetry channel:	Narrow-band 2.45 GHz
Certifications:	US: FCC part 15; EU: CE
Power supply:	Power-over-Ethernet IEEE 802.3af
	Low voltage 12V DC @ 10W (optional)
Mounting options:	Adjustable mounting bracket

Table 5.1: Ubisense sensors technical specification.

tructure such as access points, Ethernet switches and CAT5 cabling, for the communication between sensors and servers.

Two-way communication: Ubisense sensors support two-way communication at 2.45 GHz with Ubisense tags. This allows the system to dynamically manage optimally tags, for example, to dynamically change the update rate in real time depending on the position, send feedback to users via tag's LEDs, query tag battery status information and button press action. Two-way communication is also used for presence detection, control and telemetry.

Ease of maintenance: The sensors are managed remotely via TCP/IP and Ethernet standards for communication and configuration. The sensor firmware is easily upgradeable via the network to allow the installation of software release updated.

Tags

Ubisense tags (Figure 5.11, Table ??) are small and robust devices that, when applied to assets or people, enable their real-time location in 3D with an accuracy of 15 centimeters. In addition to the localisation features, tags include additional features such as a LED for easy identification, a buzzer to provide messaging features, a motion detector to instantly activate the

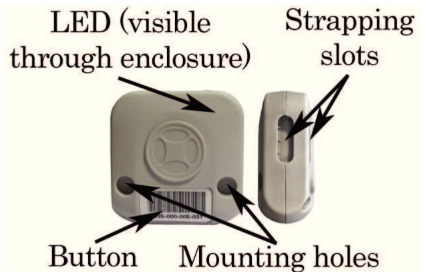


Figure 5.11: Ubisense Series 7000 Compact tag.

tags and buttons that can be associated with specific events defined by the software.

Tags technical specification	
Dimensions:	38 cm x 39 cm x 16 cm
Weight:	25 g
Temperature range:	from $-20°$C to $60°$C
Humidity range:	from 0 to 95%, non condensing
Update rate:	from 0.00225 Hz to 33.75 Hz (can be varied dynamically under software control)
Peripherals:	LEDs (application controllable) Push button (application controllable) Motion detector
Certifications:	US: FCC part 15 FCC IDs SEATAG22, SEATAG22HH EU: CE Canada: RSS-Gen, RSS-102, RSS-210, RSS-220 Industry Canada ID 8673A-TAG22HH Singapore: IDA TS-SRD, IDA TS-UWB
Power supply:	3V coin cell (CR2477)
Mounting options:	Industrial adhesive pad (supplied), industrial Velcro and screw mountings.

Table 5.2: Ubisense tags technical specification.

Below there are some typical features of Ubisense tags:

Precise localisation: The tag transmits radio pulses, in UWB modality, which are used by the Ubisense location system to determine the 3D position of the tag within 15 cm. By Using UWB technology, the precision of the system is maintained even in complex indoor environments.

Two-way communication: Ubisense tags uses a dual-radio system, in addition to the one-way UWB radio communication used for the spatial detection. The tags use a conventional two-way communication at 2.45 GHz for the control and telemetry.

Flexible Update Rate: The software platform allows to dynamically change the update rate of the tag depending on the activity. If a tag is moving fast, we can have a high update for a more accurate localisation, but if it moves slowly, the update rate can be reduced to conserve battery life. When are at rest, the tags are put in power save mode, via an embedded motion sensor, which ensures the restart in the movement events.

Interactive Buttons: The Compact tag provides a button to allow context-sensitive input in systems that require interactivity. Applications can use the location of the tag when the button is pressed and perform functions based on an event - for example, the activation of a production process of a machine when we press the button, but only if the user is located in a safe place. The application can also send feedback to the user via the LED.

Resistant and adaptable: The tags are designed to withstand to harsh industrial environments. They resist to dust, water, moisture and impacts and can be safely installed on mechanical or electronic instrumentation and animals as well as humans.

Battery life: The techniques of low energy or power management affect the battery life. In a typical application where a tag is used to identify a user every three seconds, the battery has an average life of four years. Charge level of the battery is shown by the system, to allow a planned replacement (the actual duration will depend upon the type of battery, tag and blink rate) (Ubisense, 2013b).

Optional additional receivers and sensors

The Ubisense platform allows the integration of additional devices, such as measuring systems, transducers, sensors, detectors (e.g., GPS, 802.11, RFID, temperature sensors, radiometers). This allows a usage of a variety of equipment to meet the needs of users while avoiding blocking the user to a single vendor.

Integration of devices and sensors can be achieved by two methods: for more straightforward tasks, many parts of the platform are accessible via Application Programming Interface (API) presented in C++ or COM; for more complex tasks where tighter integration is required, it is possible directly to use the Ubisense data modelling language used to build the rest of the Ubisense system (Steggles & Gschwind, 2005).

5.3.1.2 Middleware software architecture

Middleware software architecture is constituted by the Ubisense Location Platform, a suite of software components that enable location sensors and tags to be set-up, calibrated and configured into cells and objects using a simple Graphical User Interface (GUI).

Ubisense tag locations are sent via standard Ethernet cable or wireless LAN to the Location Engine which processes the data and passes the information via an industry-standard API to applications.

The Location Platform scales seamlessly from a laptop to a multi-CPU cluster. Exactly the same software can be run on a single machine, for software development or demos, or deployed across a cluster to support huge installations generating tens of thousands of location events per second.

The Location Platform has a service oriented architecture. All services are crash-tolerant, restart-tolerant and cluster friendly and have been proven through years of testing in large scale real-time control applications in assembly plants, transit depots and military sites where high reliability and low latency are essential.

The Location Platform includes a suite of software tools for easy system configuration and its open API covers every service feature so that anything that can be done using a tool can also be done via the API.

Location Engine Configuration

The Location Engine Configuration tool is used to control and monitor operation of location sensors and tags as well as to tune the location system behaviour. The tool enables:

Monitoring and control of the sensors that are running across the platform;

Monitoring and control of the tags that have been registered to the platform;

Defining the cell functionality e.g. conventional radio settings and cell boundaries;

Defining the sensor functionality e.g. master/slave operation and thresholds;

Sensor calibration;

Configuration of the filter mechanisms for the location calculation in the master sensors;

Processing presence information from a combination of complementary indoor/outdoor positioning technologies.

Map

The map tool allows visualization of objects tracked using the Ubisense system. It includes two main elements:

2D Viewing to view objects in a map-like plan view;

3D Viewing to view objects in a perspective or 'virtual reality' view.

Platform Control

Platform Control is used to control the core services that make up the Ubisense Location Platform. The tool interacts with the core platform component of the Ubisense Domain Object Model. This core component consists of:

The platform itself, which coordinates between controller nodes on the sensor network;

Controller nodes, which provide the operating environment for the various services running on each machine.

Security Manager

Ubisense provides security and access control through the Security Manager. The Security Manager allows the definition of a security policy for the Ubisense Location Platform, to protect platform data from unauthorized access. It is designed to ensure that only authorized users are able to modify the Ubisense Location Platform state.

Service Installer

The Service Installer is a tool that allows extra services or service upgrades to be installed into the Ubisense Location Platform. An intuitive Service Installation Wizard guides the user through the installation process. Platform services are installed from a user-defined distribution directory.

Service Manager

The Service Manager is used to administer the distributed services that make up a Ubisense platform. The tool supports:

Monitoring the services that are running across the platform;

Starting and stopping services;

Deploying services on service controllers;

Requesting that services backup their data to the core server.

Site Manager

The Ubisense Site Manager simplifies the configuration of the properties of types, objects, and building data. It gives the administrator control over objects, building models, visualization and geometry (Ubisense, 2013a).

La Table ?? mostra alcune specifications della Ubisense location platform.

Specifications	
Supported platform:	Windows: XP, Vista Business Edition, 2003, Server, 2008 Server and Windows 7. (.NET API requires .NET 2.0 or higher. Visualization tools require DirectX 9.0 or higher) Linux: 32-bit or 64-bit, 2.6 kernels
Security:	Supports AES/256 encryption for all data
Open APIs:	.NET 2.0 API to all features as standard C++ API (Windows or Linux) also available.

Table 5.3: Ubisense location platform specifications.

5.3.1.3 Software applications

As mentioned in the previous section, Ubisense software platform allows you to create applications interfacing directly to the Ubisense Location Platform and to make calls to all the services, that were already described in the previous sections, through the use of Ubisense .NET APIs.

In particular, it is possible to create console applications, Windows Forms applications, web-service applications, etc, in different programming languages (C#, C++, Visual Basic, etc) by using Microsoft R Visual Studio Express, a programming environment free distributed by Microsoft R . To include reference to Ubisense, to be assembled within the project that you want to develop, just select the file with *.dll* extension in the folder where the Ubisense software was installed. To perform this operation, follow the standard procedure of including a library, as shown in Figure 5.12 and in Figure 5.13.

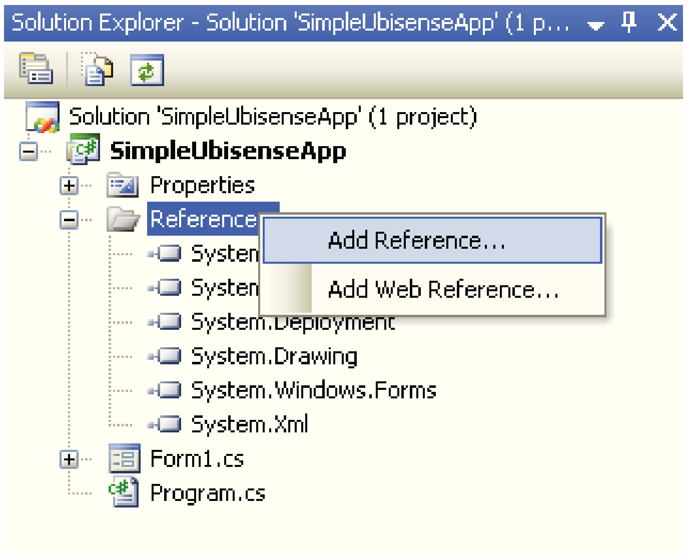
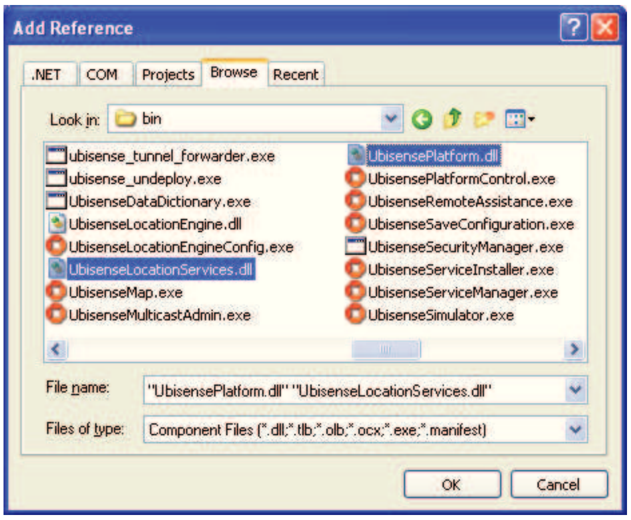


Figure 5.12: Add references in a Microsoft R Visual Studio Express project. Step 1: first click with the right mouse button on the 'References' of a project and then the left mouse button on 'Add Reference...'.



ubisense_tunnel_forwarder.exe
ubisense_undeploy.exe
UbisenseDataDictionary.exe
UbisenseLocationEngine.dll
UbisenseLocationEngineConfig.exe
UbisenseLocationServices.dll
UbisenseMap.exe
UbisenseMulticastAdmin.exe
UbisensePlatform.dll
UbisensePlatformControl.exe
UbisenseRemoteAssistance.exe
UbisenseSaveConfiguration.exe
UbisenseSecurityManager.exe
UbisenseServiceInstaller.exe
UbisenseServiceManager.exe
UbisenseSimulator.exe

File name: "UbisensePlatform.dll" "UbisenseLocationServices.dll"

Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe;*.manifest)

Figure 5.13: Add references in a Microsoft R Visual Studio Express project. Step 2: select the desired file with .dll extension, present in one of the folder, to install the software Ubisense, and click 'OK'. Links to the references in the project in order to be able to use the classes implemented in these libraries will be automatically created.

Once you have set the project, you can implement the desired functions. The Ubisense .NET API Reference Manual, available in Visual Studio help if you have installed the Ubisense software, is the golden reference for any detailed information about the API.

Ubisense company developed a number of software applications, such as Visible Industrial Process, Asset Manager, Transit Yard Manager, etc, by offering innovative solutions to prestigious customers in different sectors (industry, manufacturing and automotive, logistics, military, etc) (Heathcote, 2011).

Finally, as already seen in Figure 5.9, it is possible to interface various types of Corporate Systems (e.g., ERP, MES, PPS, WMS and DMS) with the Ubisense Location Platform and/or with specifically created software applications, in order to integrate the system Ubisense in an information technology management system (called 'information system' in information technology) and place it in relation to other relevant processes which are already available in it.

5.3.2 Operating speci cations

In the previous sections, the features of the main components (hardware and software) of the Ubisense system were described.

In this section, the operating specifications, i.e. the principle of functioning of the Ubisense system will be explained. In other words, you will see how Ubisense components interact during the localisation process, according to the theoretical principles detailed above.

Figure 5.14 shows the way in which communication occurs between sensors and tags, and an idea of how sensors use combined positiong technologies to calculate the location of the tag in a precise manner.

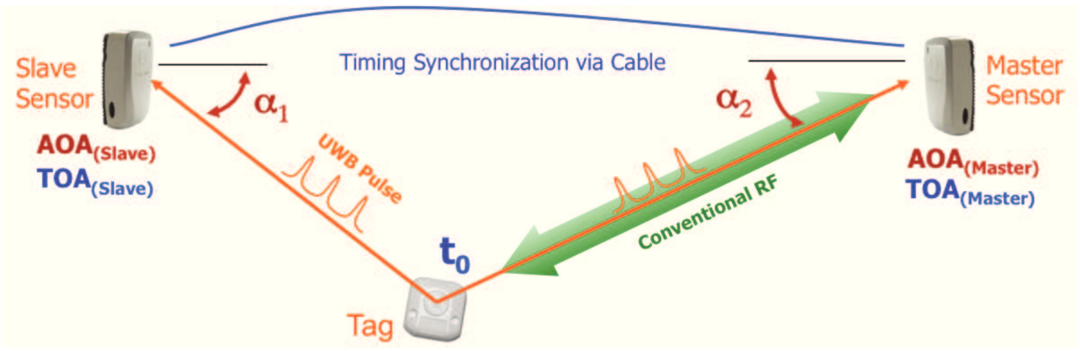


Figure 5.14: Principle of functioning of the Ubisense system. The sensors are asking the tags to generate a UWB pulse through conventional radio signal (RF). At the time t_0 the tag transmits the pulse and, once received by the sensors, calculates the Angles-Of-Arrival (AOAs) α_1 and α_2. Sensors are synchronized via cable, then Time-Difference-Of-Arrival (TDOA) between each pair of sensors is calculated. From the combination of TDOA and AOA, position of the tag is calculated (Heathcote, 2011).

We can summarize the process of estimating the location of a tag through the following steps:

1) Sensors, within the cellular Ubisense network, communicate with the tags on conventional Radio Frequency (RF) signals (green arrow in Figure 5.14) and ask to generate a UWB pulse depending on their profile up to 20 times a second.

2) At the time t_0, tags answer by sending UWB signals that are received from the sensors and are used to calculate the Angles-Of-Arrival (AOAs) α_1 and α_2. Therefore only two sensors receiving a tag signal are already able to deliver a 3D location.

3) Sensors are synchronized via cable (Timing Sinchronization via Cable in Figure 5.14), then Time-Difference-Of-Arrival (TDOA) between each pair of Sensors is calculated. This makes the location robust.

4) The Master Sensor calculates a coordinates based on all raw information delivered, taking into account tuning parameters such as motion model filtering, and sends it to the Location Platform.

Both TDOA and AOA are used to increase robustness of 3D-Position computations.

Once the location of one or more tags is estimated, map window of Location Engine Configuration tool, already described in section 5.3.1.2, allows their visualization. Among other functionalities, the tool provides the ability to view a map of the sensors network in which the AOA vectors are shown in real time through green lines coming from each sensor, the hyperbolic localisation (TDOA) with blue hyperbolic curves and the position of the tag with a red dot.

The Figure 5.15 shows two screenshots of the Location Engine Configuration tool.

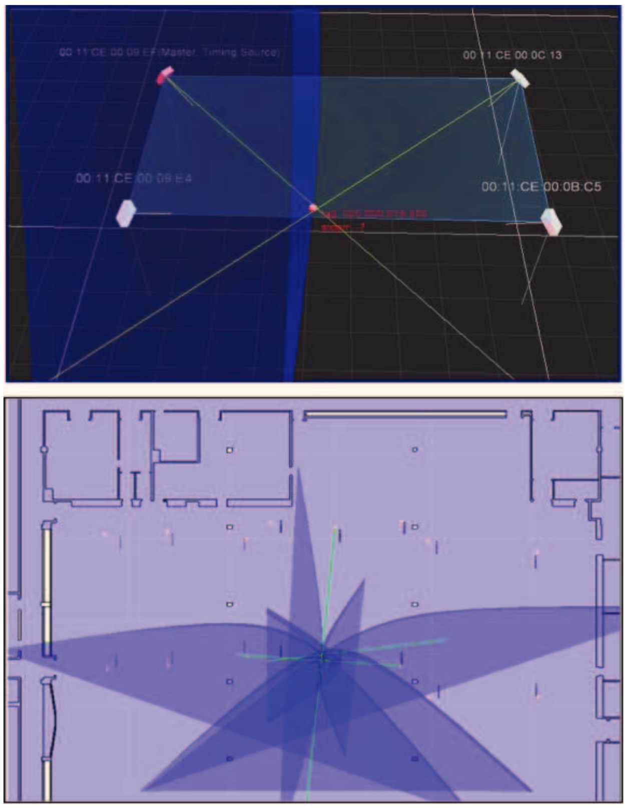


Figure 5.15: Screenshots del Location Engine Configurator tool for the visualization of 3D e 2D maps, respectively. Green lines represent the AOA vectors from each sensor. Blue hyperbolic curves represent the hyperbolic localisation (TDOA). Red dot is the tag position (Heathcote, 2011).

5.3.3 Installation procedure

The installation phase is an activity which requires a lot of care and attention. An installation carried out inaccurately, without following the recommendations provided by Ubisense, could lead to malfunctioning or alter the performance of the system.

This section describes the main steps of the installation process generalized to all case studies. More detailed information could be found in the online Ubisense How-To articles (Ubisense, 2008-2010).

To develop the system, a path divided into five steps should be followed, as shown in Figure 5.16.

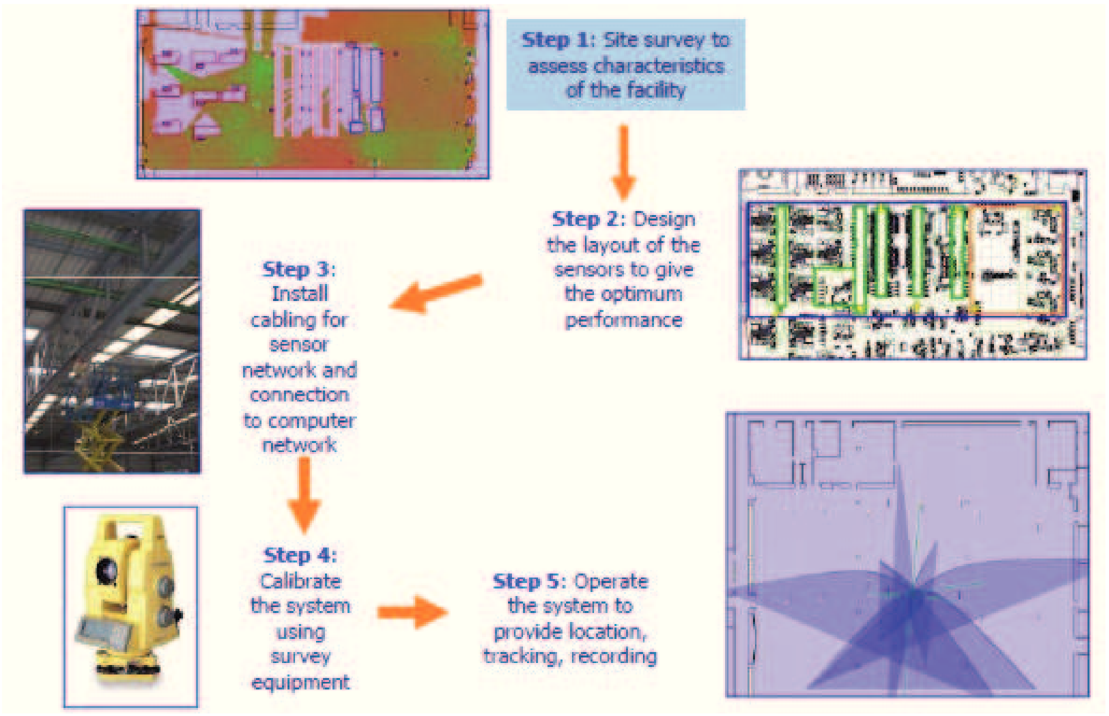

Step 1: Site survey to assess characteristics of the facility
Step 2: Design the layout of the sensors to give the optimum performance
Step 3: Install cabling for sensor network and connection to computer network
Step 4: Calibrate the system using survey equipment
Step 5: Operate the system to provide location, tracking, recording

Figure 5.16: Installation Procedure of Ubisense system structured in five steps (Ward, 2010).

Referring to Figure 5.9, the installation procedure described below will regard the hardware and middleware of the Ubisense platform since they constitute the basic elements for the development of the system. The software interfacing with other systems or applications, which is specifically developed for specific objectives, will not be discussed in this section because it depend on the case study which the system is applied to.

In the initial step of the installation process a survey should be performed to verify the environmental features of the space where you want to implement the localisation system. As described in section 5.2.1, a careful analysis is recommended with regard to the physical features of the location, and the indeed use of the system, in order to determine the possible constraints to be taken into account in the next phases.

Subsequently, the design of the sensors layout through the preliminary definition of their geometrical arrangement is recommended. This step is crucial to get the optimal performance of the system because, if done inaccurately, it could compromise the proper functioning of the RTLS. For instance, if you have four sensors, a simple configuration is a square with sides in the range of 10-30 m. The sensors should be mounted at the corners of the square, near the ceiling for good line of sight across the tracked space.

After defining the layout of the sensors, a PC that performs the functions of the server for the RTLS must be configured. To do this, you need to install the software provided by Ubisense and a DHCP server, as indicated in the reference guide (Ubisense, 2008-2010). After that, it's possible to perform the physical installation of the sensors as determined in the previous step. The sensors should point towards the floor in the middle of the space, and should have no roll. In addition, it is necessary to connect the sensors to the server PC through a PoE-enabled network switch. 'Master' and 'Slave' sensors are physically the same. The minimum requirement for the network cables is CAT5e (Power-over-Ethernet (PoE)). Through the DHCP server it will be, then, possible to assign the desired IP address to the sensors in order to monitor and send input commands to each individual sensor. The sensors must be also connected to each other through the 'timing' Cables (CAT5e cables) to allow time synchronization among them by calculating the TDOA between each pair of sensors.

The next step involves the calibration process of the RTLS. However, you must first perform a virtual reconstruction of the sensors map through the Location Engine Configuration software tool, as described in section 5.3.1.2. A reference system must be determined within the graphical section of this software tools in order to precisely mapping the sensors. The position of each sensor must be manually inserted by bringing the values corresponding to those of the real reference system where the sensors were physically installed. Therefore, it will be possible to carry out the true calibration process which provide the system with the missing parameters in order to perform the localisation. The steps to be followed for the calibration of the system are the following: sensor orientations and cable offsets. In the first phase of calibration, also called 'Orientation calibration', once a Ubisense tag is put in a known position within the coverage area of the sensors and the coordinates of the position in the considered reference system are written in the Location Engine Configuration tool, then the system takes readings of the AoA from the tag and yield the orientation of the sensor from the average of the measurements set. This calibration step must be performed on each sensor in order to set their parameters of orientation (pitch, yaw and roll) within the network, in each of them. The second phase of calibration, also called 'Cable calibration', is similar to orientation calibration, except that a cable offset is between two sensors. Therefore, the calibration uses two sensors: the sensor getting its cable offset and the reference sensor, which is ideally the master. Cable offsets are just some arbitrary numbers indicating the offset from the master, so that the master achieve a null value. For this reason this latter calibration phase, unlike the previous one, must be carried out only on each of the slave sensors. At the end of the operations just described, assuming that the sensors are disposed in such a way as to form a square and the tag used for the calibration has been placed at the point where they intersect its diagonals, in the map of the Location Engine tool it will be noted that all sensors are oriented toward the reference point, as shown in Figure 5.17.

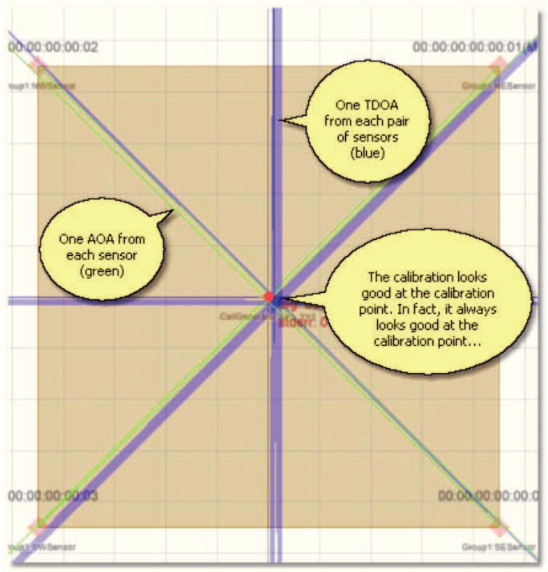

00:00:00:00:02
00:00:00:00:00:01
One TDOA from each pair of sensors (blue)
One AOA from each sensor (green)
The calibration looks good at the calibration point. In fact, it always looks good at the calibration point...
00:00:00:00:03
00:00:00:00:00

Figure 5.17: Map of sensor network after calibration operations. Each sensor is oriented toward the calibration point (the center) where the tag used for calibration (red dot) is placed. The green lines, outgoing from each sensor, focus the tag calibration in order to calculate the AOA between the tag and the sensor. The blue curves indicate the TDOA between each pair of sensors (Ubisense, 2008-2010).

Once the installation procedure is completed, you can use the RTLS for the localisation and tracking of tags within the area covered by the sensors.

Chapter 6

THE CASE STUDY

The trial was carried out from 1st August to 10th September 2011 and from 1st June to 30th September 2013 within a dairy house located in the province of Ragusa (Sicily, Italy), which is one of the most important livestock breeding areas in the country (Figure 6.1).

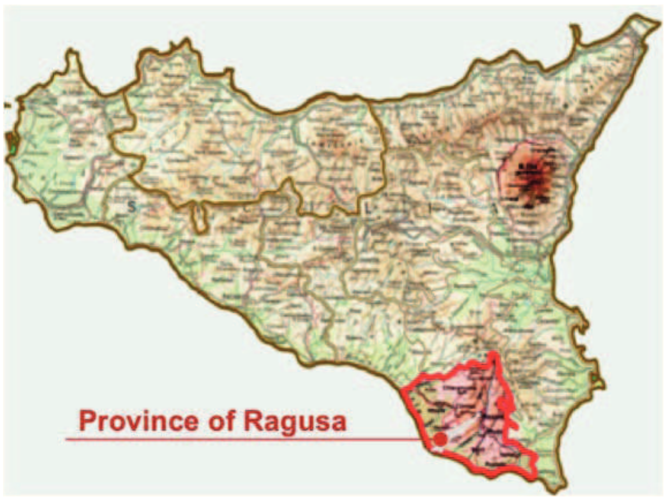


Figure 6.1: Province of Ragusa (Sicily, Italy), one of the most important livestock breeding areas in the country.

6.1 THE AREA OF THE BARN UNDER STUDY

The barn was characterized by a rectangular plan of about 55.6 m x 20.7 m with three sides completely open, i.e. without outside walls (Figure 6.2). The roof was symmetric and covered by fibro-cement sheets supported by a bearing structure made of steel trusses and purlins. The feeding alley of about 55.75 m x 3.50 m was adjacent to the resting area that was arranged with two rows of 64 stalls faced head-to-head equipped with sand beds. Service alleys allowed the easy access of the cows from the feeding alley to the service alley for the second row of stalls. The side of the barn at the back of the second row of stalls was completely open. From an interview with the breeder and the direct observation of the breeding environment, knowledge of the management activities carried out in the barn was obtained. Feed was delivered to the cows once a day at approximately 06:30 a.m. and was moved closer to the feed barrier later in the day at 4:30 p.m.. Milking occurred twice a day at 06:00 a.m. and 05:00 p.m. Furthermore, in the alleys there was presence of slurry accumulation as the cleaning is not automated but it is carried out 1-2 times a day by a scraper. Within the barn a reduced but complete breeding area of about 15.40 m x 11.50 m was considered for the experimental trial (Figure 6.3). In this area a group of 15 Holstein dairy cows was housed. The breeding environment was composed of a resting area of about 10.40 m x 4.30 m with two rows of stalls arranged head-to-head and sand beds, a feeding alley of about 15.40 m x 3.50 m adjacent to the resting area, a service passage and two service alleys.

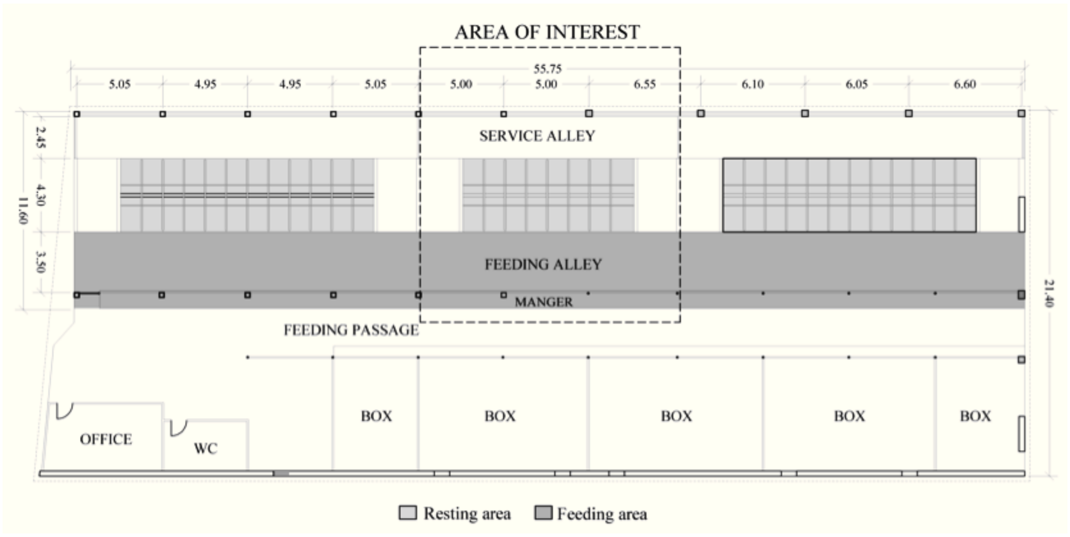


Figure 6.2: Plan of the free-stall barn showing the area of interest for the experimental trial.

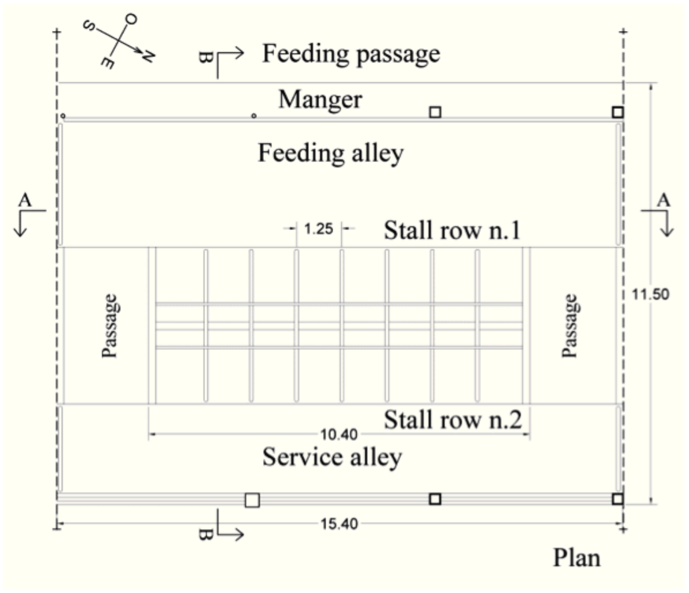


Figure 6.3: Plan of the study area.

6.2 ANALYSIS AND INSTALLATION OF THE UBISENSE ULTRA WIDEBAND REAL-TIME LOCATION SYSTEM

In this section a detailed requirements analysis (section 6.2.1) and the individual steps of the RTLS installation (section 6.2.3) will be described.

6.2.1 Requirements analysis

The requirements for a system are the descriptions of what the system should do, the services that it provides and the constraints on its operation. These requirements reflect the needs of users for a system that serves a certain purpose such as controlling a device or a peripherical, or finding information (Sommerville, 2010).

In this study, two kinds of requirements were analysed: functional and non-functional requirements.

6.2.1.1 Functional requirements

Functional requirements are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations (Sommerville, 2010).

In this case study the RTLS would have to meet the following requirements:

Client side

View location of sensors, on a grid matrix where the map of the barn is reproduced.

Identify the sensors.

Activate the UWB tags.

Calibrate the system.

View graphics and statistics of location information.

Server side

A computer with optimum operating system and hardware.

A permanent Internet connection to communicate with the client.

Maintain a database of all known APIs.

Maintain location informations of all clients and update this whenever possible.

Display this information graphically.

Send this informations to the clients.

A fully functional Graphical User Interface (GUI).

6.2.1.2 Non-functional requirements

Non-functional requirements, as the name suggests, are requirements that are not directly concerned with the specific services delivered by the system to its users. They may relate to system properties such as reliability, response time, store occupancy and the constraints on the system implementation such as the capabilities of I/O devices or the data representations used in interfaces with other systems (Sommerville, 2010).

In this case study the non-functional requirements are:

Provide a pleasing interface and offer a robust and reliable service.

Be flexible because the users requests change over time.

A permanently available connection between the client and server.

A response time as low as possible.

Be portable for the user in order to allow the use of the same application on different devices without requiring performing an installation each time.

6.2.1.3 Hardware and software requirements

The system would have to require the following hardware:

Ubisense hardware components as seen in 5.3.1.1 section.

A server to send and receive location information.

A PoE-enabled network switch and CAT5e cables to build the sensors network.

A DHCP server to assign the desired IP addresses to the sensors.

An internet connection.

The following software requirements should be available to run in a stable Microsoft R operative system:

Ubisense software components (5.3.1.2).

Software applications (5.3.1.3).

DHCP server.

6.2.2 Analysis of the RTLS functionality

A detailed study on the RTLS functionality, implemented in the case study in question, was carried out before the beginning of the system installation step.

In particular, an analysis of *Visual Modeling* was carried out by adopting the Unified Modeling Language (UML). The Visual Modeling is a process that provides a graphic representation of a model, using a well-defined set of graphical elements that, depending on the used modeling language, constitute the main components for the representation of particular diagrams.

In this study, as already mentioned, the UML was used, i.e., a graphical language used in object-oriented development that includes several types of system models able to provide different views of a system. The UML is the standard for object-oriented modeling (Sommerville, 2010).

In the following sections some UML diagrams produced for the RTLS functionality analysis will be analyzed individually: use case diagram (section 6.2.2.1), class diagram (section 6.2.2.2), state diagram (section 6.2.2.3) and sequence diagram (section 6.2.2.4).

Since it comes to high-level diagrams, therefore, all the features of the properties and operations related to the system will not be described but the most important steps that constitute the process of localisation of the RTLS will analysed without going into details.

6.2.2.1 Use case diagram

A use case diagram shows the interactions between a system and its environment. Use cases are collections of scenarios, involving the use of the system, where each scenario describes a sequence of events. The main element of a use case diagram is the actor, i.e. the entity that interacts with a use case by starting the sequence of actions described by the use case itself, and possibly getting precise answers from the system. It can be a person or another system. Graphically, as shown in Figure 6.4, an ellipse represents a use case and a little man is an actor. The name of the actor appears just below the representation of the same actor as the name of the use case appears inside the ellipse. An association line finally connects an actor to the use case and represents the communication between the actor and the use case.

In this case study, the actor is represented from the User, who has the task of assigning tags to the cows to locate. It also has the task to activate the tags, i.e. to perform the operations necessary to ensure that tags are working properly. Once the 'Activate tags' use case is activated, a sequence of use cases is triggered. In particular, this use case is an extension[1] of the 'Locate tags' use case, while the tag battery is discharged, which represents the main use case diagram. In fact, it includes[2] 'Compute TDOA' and 'Compute AOA' use cases in order to indicate the two positioning techniques used by the system, and extends in 'Activate sensors' use case while the cables are connected and

[1]The extend relation allows you to create a new use case by adding more steps to an existing use case.

[2]The include relation allows you to define one or more use cases within another use case.

the DHCP server runs. Furthermore, it is an extension of 'Update and store data', in order to indicate that, once location is carried out, data are updated and saved in memory by the system. This last use case is an extension of 'Load software' use case.

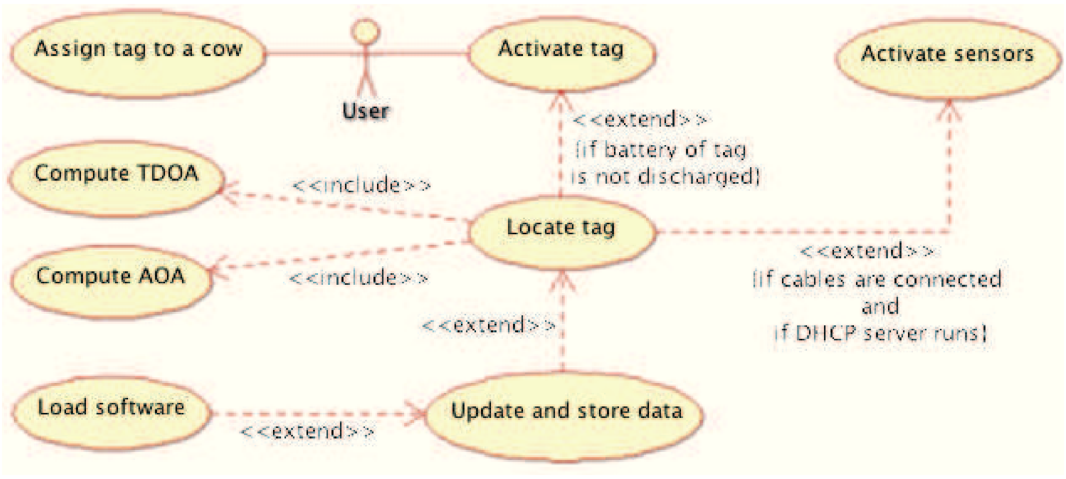


Figure 6.4: Use case diagram.

6.2.2.2 Class diagram

Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes.

A class is a category or group of objects that have similar attributes and similar behaviors. Graphically, as shown in Figure 6.5, a class is represented by a rectangle. The class name, by convention, is a word with an initial capital letter and appears near the top of the rectangle.

An attribute is a property of a class. It describes a set of values that the property can have when the objects of that class are instantiated. A class can have zero or more attributes. The list of attributes of a class is graphically separated from the name of the class to which it belongs by a horizontal line. The attribute name can be followed by two points and the data type with which you want to represent (e.g., string, float, int, bool, etc). In Figure 6.5 the Sensor class, for example, has the attribute 'MacAddress: String'. Each sensor, in fact, is characterized by a unique string that represents its MAC address.

A method is an action that objects of a certain class can accomplish. Like the name of the attributes, the name of a method is written with lowercase characters. If the method name consists of several words, then those words

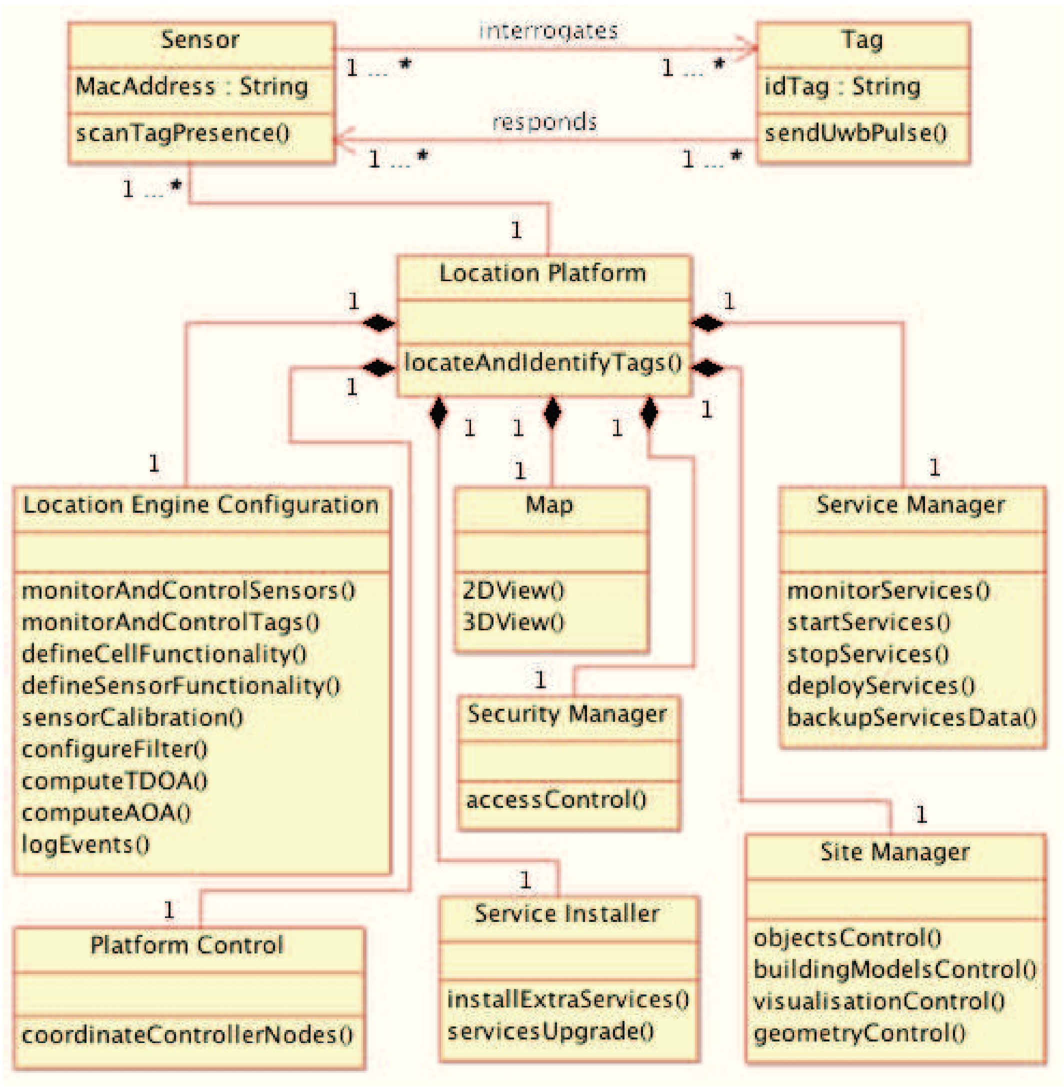


Figure 6.5: Class diagram.

are joined together and each of them, except the first, is written with the first letter capitalized. The list of methods is graphically represented below the list of attributes and separated from this by a horizontal line. The methods may have additional informations. In the parentheses that follow the name of a method, in fact, you can show any required parameters to the method together with their type. Finally, if the method is a function, the return type must also be specified. In Figure 6.5 Location Engine Configuration class, for example, is constituted by nine methods including *sensorCalibration()*, which represents the calibration function of the system.

When several classes are conceptually connected with one another, such connection is called association. Graphically, an association is represented with a line connecting the two classes, with the name of the association just above it. The multiplicity is a special type of association in which you show the number of objects belonging to a class that interacts with the number of objects of the associated class. There are different types of association, and each of them is represented by a default symbol.

In Figure 6.5 the class diagram of the case study in question is shown. The Sensor class is connected to the Tags class in two ways: the first indicates the association that is established when a sensor interrogates a tag and the second indicates the response of the tag. The multiplicity of these associations is 1 ... * since each sensor interrogates one or more tags, and each tag, in turn, responds to the sensors that sent the request. The Sensor class is also connected to the Location Platform class with multiplicity 1 ... * because a system can have one or more sensors, while the Location Platform class has a relationship of multiplicity 1 against the Sensor class since a RTLS can be constituted by a single Location Platform. This, finally, results to be a class consisting of more than one class with multiplicity 1 to 1 since there is only one Location Platform and every class, which composes it, has no other instances. In particular, the classes associated with the composition relationship with Location Platform class are: Location Engine Configuration, Map, Service Manager, Security Manager, Platform Control, Service Installer and Site Manager.

6.2.2.3 State diagram

In UML language a state diagram is used to indicate the model type that shows the states of a system and the events that trigger a transition from one state to another. This diagram shows how a system responds to external and internal events. It is based on the assumptions that a system has a finite number of states and that events may cause a transition from one state to another (Sommerville, 2010).

Graphically, as shown in Figure 6.6, the states of a state diagram are represented by a rectangle with rounded corners. The symbol that shows the transition from one to another state is drawn by using a continuous line that ends with an arrow.

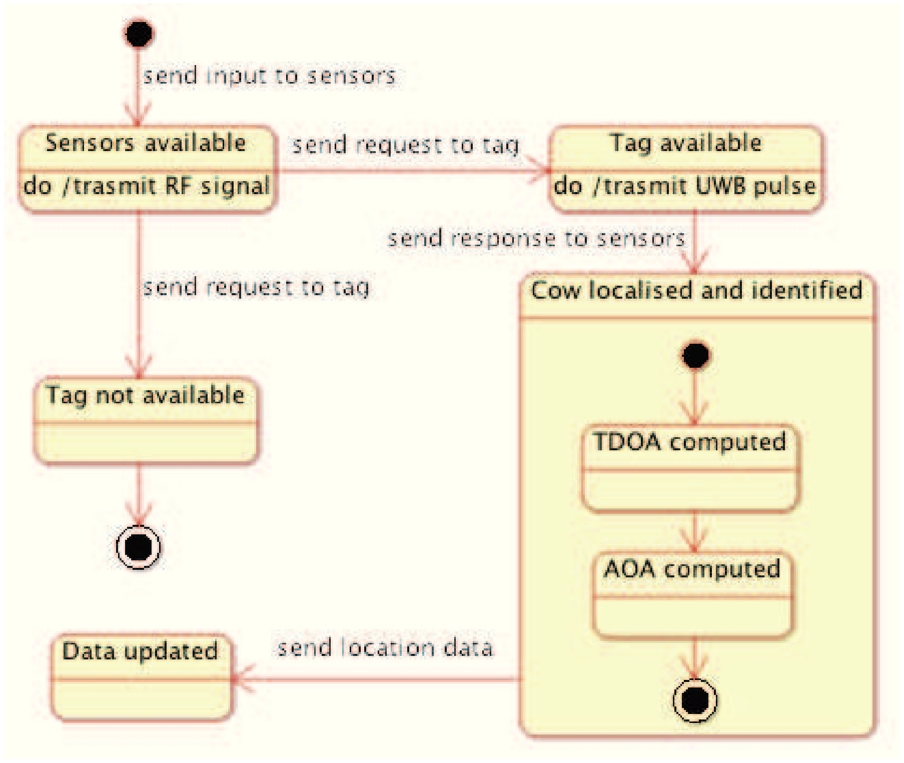


Figure 6.6: State diagram.

A solid black circle represents the starting point of a sequence of states while the point of arrival is represented by two concentric circles in which the innermost is filled with the black color.

In each state its name is shown, in the upper part of the rectangle, and activities (i.e., what happens when the system enters a given state) in the lower.

In Figure 6.6 the state diagram of the case study in question is shown. By proceeding from the starting point, once that the input is sent to the sensor, you switch to the 'Sensors available' state in which the activity 'transmit RF signal' occurs. From this state you can switch in two other states through the 'send request to tags' transition which leads: *i)* to the 'Tag not available' state if the tag does not respond and then, consequently, at the point of arrival or *ii)* to the 'Tag available' state if the tag answers. Subsequently, you switch to the 'Cow Localised and Identified' state, through the 'send response to sensors' transition, which contains a sub-state diagram which represents the implementation of the techniques of positioning adopted by the RTLS. We move finally to the 'Data updated' state through the 'send location data' transition.

6.2.2.4 Sequence diagram

A sequence diagram is a UML model that shows the sequence of interactions required to compose some operations. In particular, the interactions between actors and the system and between system components are represented (Sommerville, 2010). The key idea in this type of diagram is that the interactions between the objects are carried out by following a precise order and that this sequence occurs, in time, from the start to the end. The Sequence diagram consists of objects represented by rectangles with a name, messages represented by continuous lines with an arrow at the end and the time represented as a vertical progression.

Objects are arranged in sequence from left to right. From each rectangle a dotted line, called 'lifeline', is drawn downwards. Along the lifeline there is a small rectangle called "activation". The activation is the execution of an operation to which the object is responsible. The length of the rectangle, however, represents the duration of the activation.

A message that travels from one object to another, is drawn from the lifeline of the object from which the message leaves to the lifeline of the object to which the message is directed. It can also occur that an object sends a message to itself, that is, a message that starts from its lifeline and arrives to the same lifeline.

When viewed in the vertical direction, the sequence diagram represents the flow of time. Graphically, the time starts at the base of each object and continue to the bottom.

In Figure 6.7 the sequence diagram of the case study in question is shown. The main objects are three: Location Platform, Sensor and Tags. The Location Platform is always active during the entire process of localisation, in fact, the activation of the Location Platform object has a height equal to that of the lifeline.

The first operation is carried out, in chronological order, from the Location Platform object, which sends the 'startSensors()' message to the object Sensor in order to activate it. Once activated, the object Sensor sends the 'interrogateTag(RFSignal)' message to the Tag object, to denote the operation of the activation request made by the sensor through an RF signal to the tag. Once processed the received message, the object Tag transmits

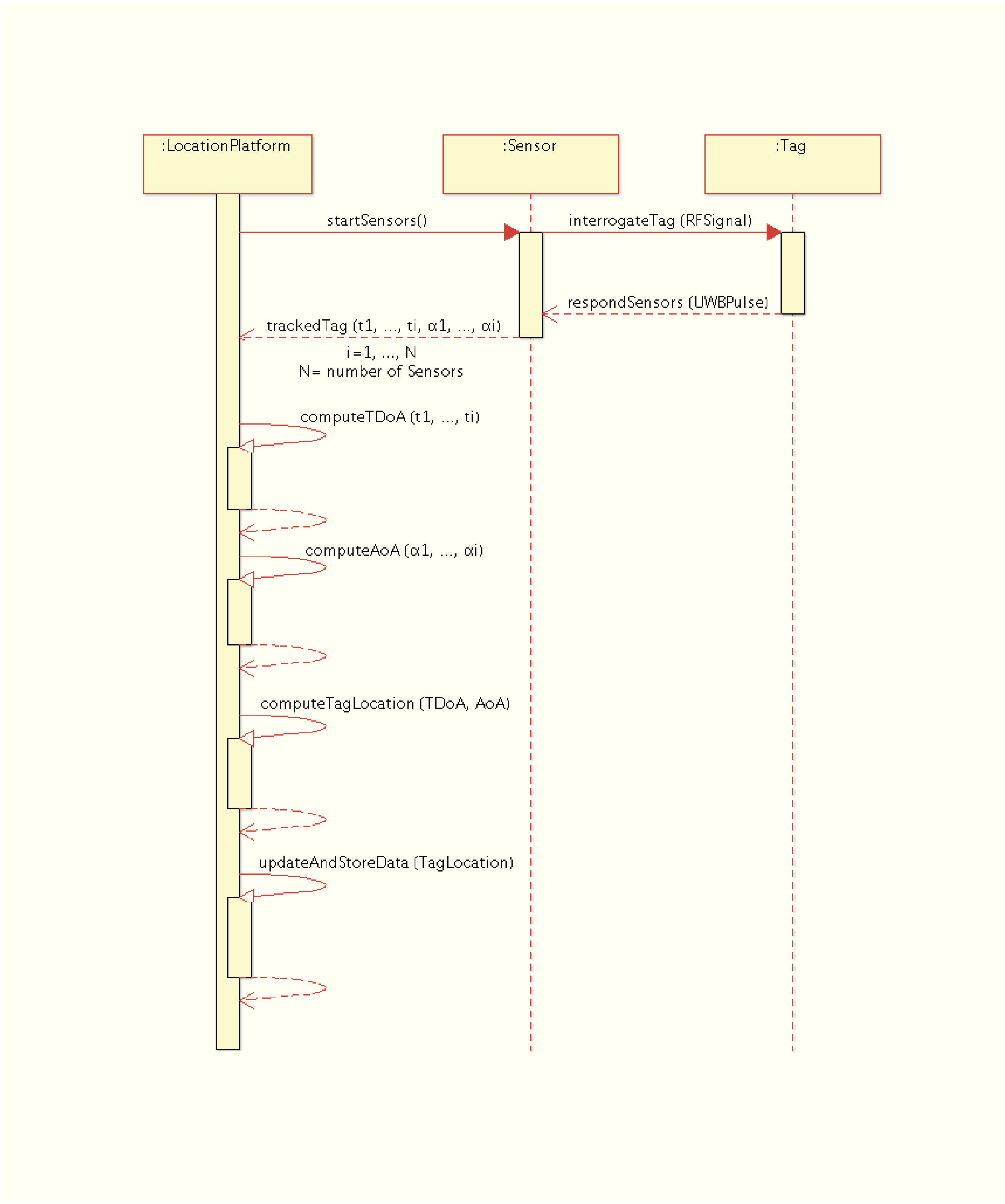

:LocationPlatform
:Sensor
:Tag
startSensors()
interrogateTag (RFSignal)
respondSensors (UWBPulse)
trackedTag (t1, ..., ti, α1, ..., αi)
i=1, ..., N
N= number of Sensors
computeTDoA (t1, ..., ti)
computeAoA (α1, ..., αi)
computeTagLocation (TDoA, AoA)
updateAndStoreData (TagLocation)

Figure 6.7: Sequence diagram.

the ’respondSensors(UWBPulse)’ response message to the object Sensor, to indicate the operation of UWB impulse transmission from the tags to the sensors. At this point, the activation relative to the object Tag is stopped since the tag task in the localisation process ends. Therefore, the next operation is carried out by the object Sensor, which, once processed the received message by the Tag object, sends the ’trackedTag($t1$ ti 1 i)’ (for $i = 1$ N and $N =$ number of sensors) message to the Location Platform object. This message indicates the transmission of the angles of arrival (i) and time of arrival (ti) received by the tags. Also the activation relative to the object Sensor, at this point, is interrupted since the task of the sensors in the localisation process ends.

All the remaining operations are performed by the Location Platform object, without interacting with other objects; in fact, the next sent messages are all recursive. These transactions consist in *i)* the computation of the two positioning techniques used by the RTLS: (’computeTDOA($t1$ ti)’ and ’computeAOA(1 i)’), *ii)* computation of the tag position in the three coordinates x, y and z (’computeTagLocation(TDOA, AOA)’) and *iii)* updating and storing of the new data acquired (’UpdateAndStoreData(TagLocation)’).

As soon as, the last step of the localisation process is completed both the activation of the Location Platform object and the lifeline of the entire process will be interrupted and the diagram is complete.

6.2.3 RTLS installation

After the study of the system requirements and functionality it was possible to proceed with the installation phase of the RTLS.

As described in section 5.3.3, the installation process is a very important step in which a special care should be provided in order to avoid to compromise the system performances.

In the following sections the description of each step of this process is reported.

6.2.3.1 Site survey

The initial step of the installation process consisted in carrying out an inspection at the candidate location to be the object of the study, in order to assess the environmental and ambient features where it was decided to implement the localisation system.

Figure 6.8 shows a picture taken during this phase. The presence of metallic structures, walls and slurry accumulation involved the preparation of an appropriate protective equipment from any accidents during the assembly phase but however it did not result in threats against the performances of the system due to the eventual reflection, scattering and attenuation degree in presence of electromagnetic waves.

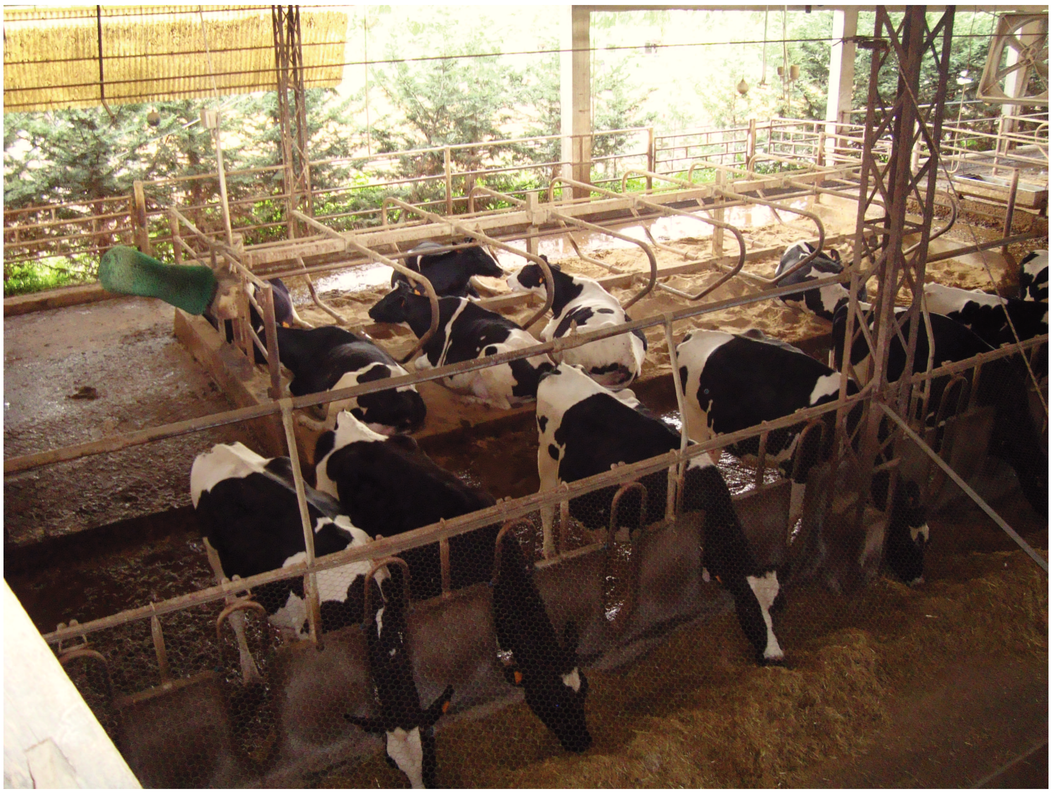


Figure 6.8: Photo of the study area.

In this initial phase, a further study on the environmental features was carried out within the area in question. A datalogger connected to temperature and humidity sensors, anemometers and globe thermometers (shown in Figure 6.9) was installed in order to determine the suitability of the location to be the object of the experiment. The comparison of the measured values over a period of about a month with those declared in the hardware features tables of the system (section 5.3.1.1), confirmed the feasibility of the test in this environment.

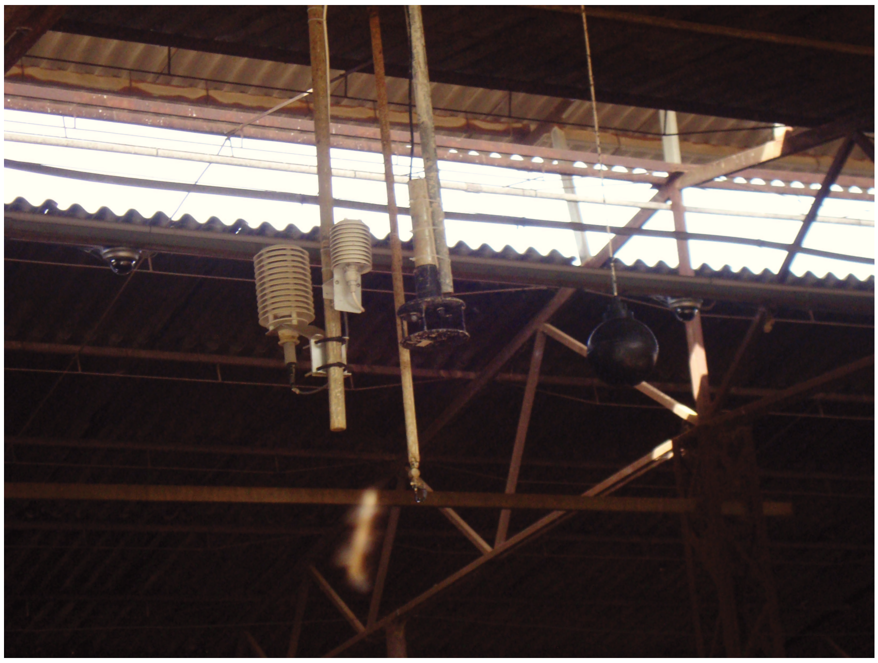


Figure 6.9: Temperature and humidity sensors, anemometers and globe thermometers within the study area.

A final analysis on the scale of the system was conducted during this preliminary phase. Given the size of the study area, it was possible to determine the number of sensors (4) to purchase, which were sufficient to cover the entire space involved in the trial and the length of the connection cables to be installed. Finally, adjacent to the study area and sufficiently isolated from the livestock environment, locals were identified where to install the rest of the hardware (e.g., PCs, switches and modems).

6.2.3.2 Sensors layout

During this phase the layout of the sensors through the preliminary definition of their geometric arrangement was designed.

Sensors $Sens_{SO}$, $Sens_{NO}$, $Sens_{NE}$ and $Sens_{SE}$ were placed at the four corners of a rectangle outside the area studied, which had a size of 15.10 x 13.95 meters, at a height of 3.85 meters above the ground, as shown in Figure 6.10.

The layout of the 4 sensors was not casual since the error committed by the system can also depend on the layout of the sensors, as described by Shahi et al. (2012). In this work it is recommended that the optimal receiver layout should incorporate spatial variability of the UWB receivers in the three principal directions. Thus, rectangular or polygonal receiver arrangements are preferable to linear or otherwise constricted layouts. Where possible, it is desirable to incorporate spatial variability in the z-direction (height) of the receiver locations as well.

The position of sensor $Sens_{SO}$ was chosen as origin of the Cartesian coordinate system used in the real environment.

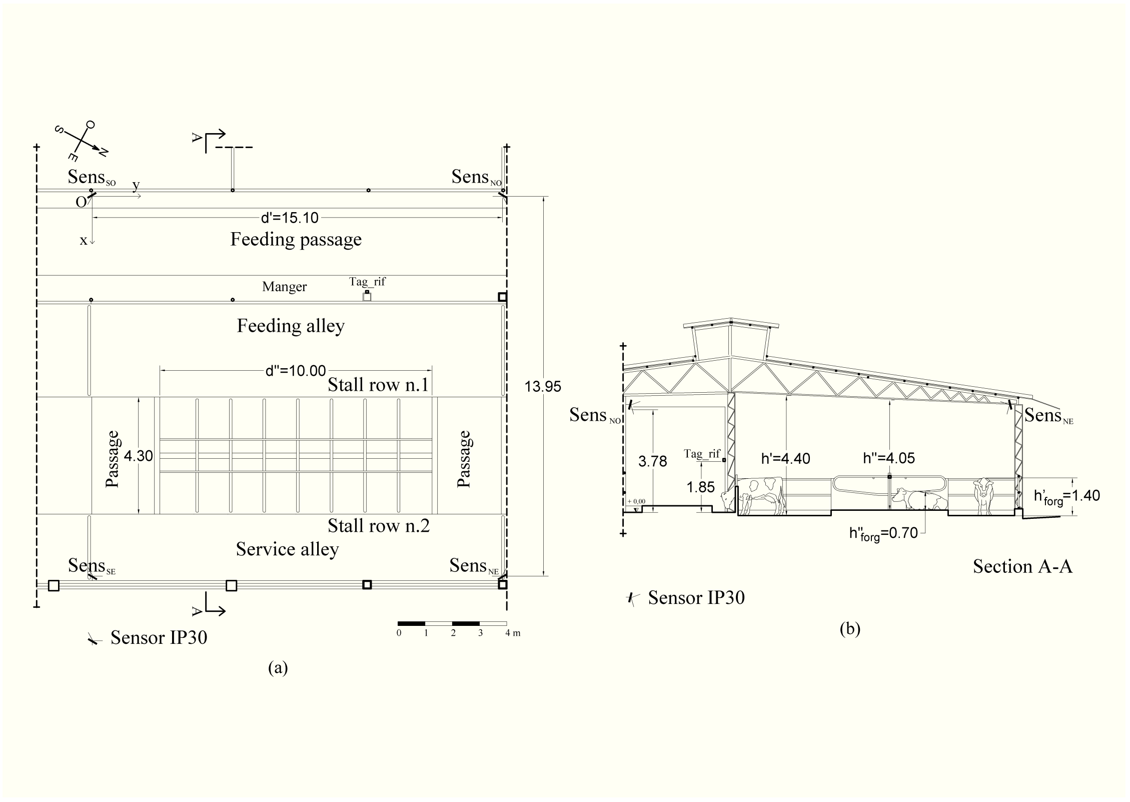


Figure 6.10: Plan (a) and section (b) of the area of the barn under study with sensors layout.

6.2.3.3 Hardware installation

At this point of the installation process, it was possible to perform the physical installation of the hardware components of the RTLS. The following devices were used in this case study:

4 Sensors IP30 Serie 7000.

10 Compact Tag IP65.

1 PoE-enabled network switch.

1 PC with an Intel R Core$^{\text{TM}}$2 Quad CPU Q6700 @ 2.66 Ghz processor and operating system Windows Vista R Business.

The sensors were fixed with mounting brackets, supplied by Ubisense, on the metal beams that support the roof of the barn, through plastic cable ties. A spirit level was used across the back of the sensors case to ensure there was no roll and the pitch and yaw angles were calculated in order to direct the sensors towards the floor in the middle of the area under study. Once fixed their position, thin films and flexible plastic material were used to entirely cover the sensors in order to protect them from the surrounding environment (Figure 6.11 c).

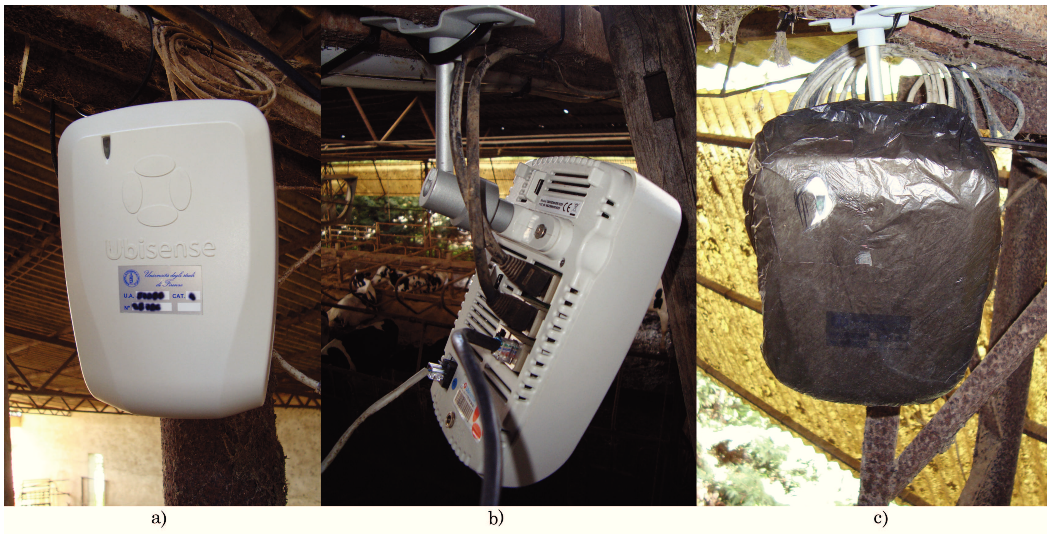


Figure 6.11: Sensors IP30 Serie 7000 used in the trial. a) front side of a sensor; b) back side of Master sensor with 'Timing Cables' (black) and power cable (white); c) sensor covered with protective plastic material.

A wired network, consisting of CAT5 cables and RJ45 modular connector was designed and installed to the support structure of the barn roof in order to connect the sensors to the PC via the PoE switch. This network allowed both the power of the sensors and the data transmission between sensors and PC.

Another network was installed in order to connect the sensors to each other through the 'Timing Cables' (CAT5e cables) to allow their temporal synchronization, by calculating the TDOA between each pair of sensors. As already mentioned in section 5.3.2, the Ubisense system is composed of two types of sensors: 'Master' and 'Slave'. Although Master and Slave sensors are physically the same, the function of calculating the TDOA between each pair of sensors is performed considering the master as reference sensor for time synchronization. For this reason, the connection of the 'Timing Cables' must follow a definite scheme.

The physical scheme of the networks described above is shown in Figure 6.12, while a picture of the connection of the Timing Cables in the master sensor is shown in Figure 6.11 b.

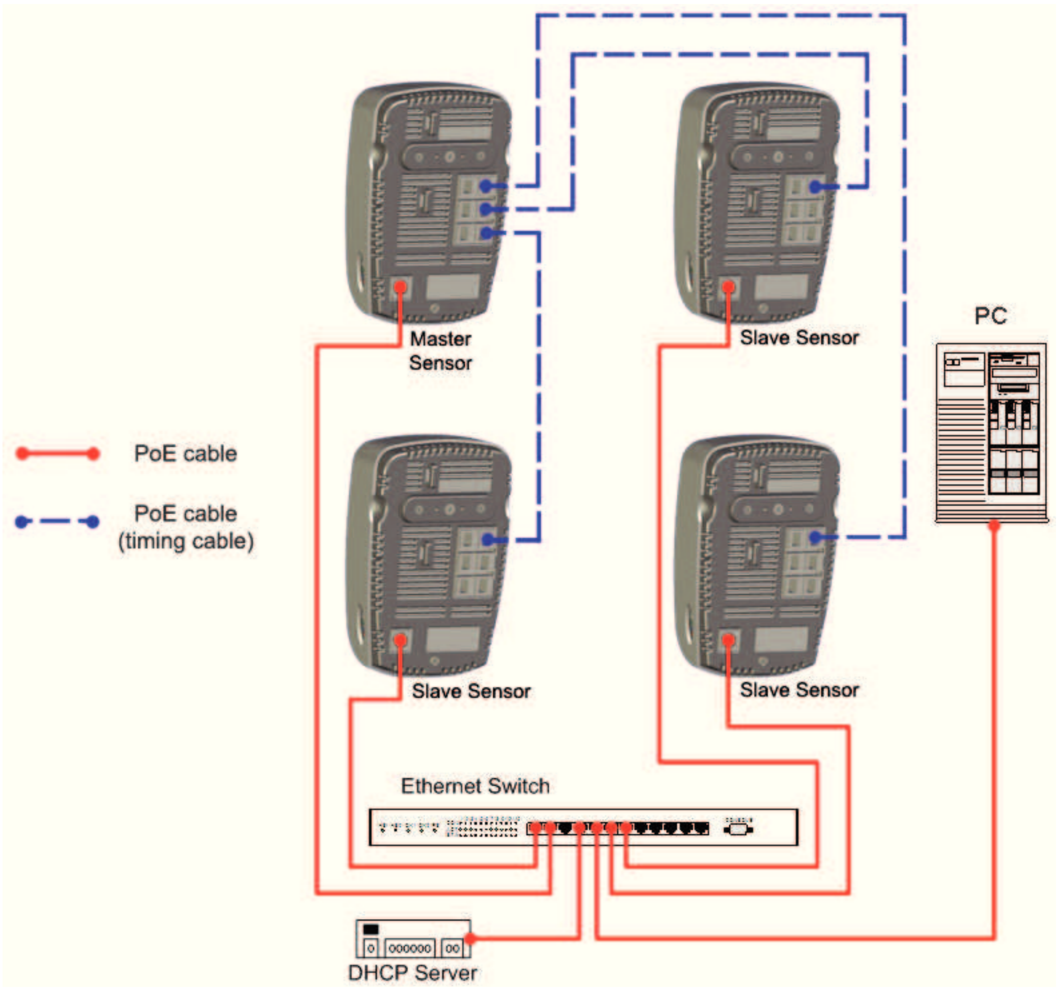


Figure 6.12: Scheme of the RTLS hardware used in the trial.

Both the PC and the switch were placed in a room adjacent to the barn, where the two wired networks were converged, in order to avoid direct exposure to the livestock environment. Although not necessary for the functioning of the RTLS, also a modem for ADSL connection was connected to the network in order to allow remote control of the entire system.

6.2.3.4 Middleware software installation

Completed hardware system operations the software configuration was carried out.

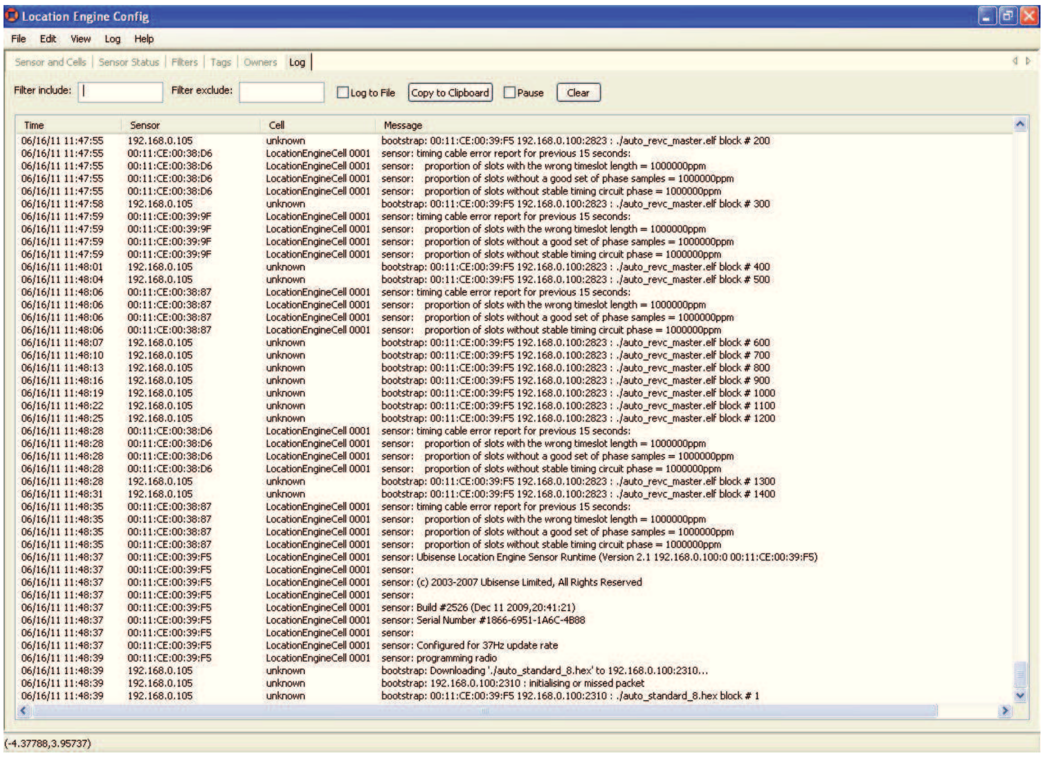
Although in Figure 6.12 DHCP Server and PC were represented by two separate physical devices, both of them were managed by the same machine. To do that, a virtual machine, where the software provided by Ubisense was installed, was configured on the PC, while the DHCP server was run on the physical machine. The features of the virtual machine was the following: Intel R Core^{TM}2 Quad CPU Q6700 @ 2.66 Ghz processor and operating system Windows R XP Professional 2002 SP3. Once installed, the DHCP Server assigned to the sensors a pool of static IP addresses manually set; each of them was uniquely associated to the MAC address of a sensor, in order to separately send input commands to each sensor and recognize the sender of the messages received.

At this point, referring to the user manual provided by Ubisense (Ubisense, 2008-2010), the suite of software components, Ubisense Location Platform, which is described in section 5.3.1.2, was installed. Once completed the installation wizard, the sensors, which were initialized by the software, using the Configuration Engine Location (LEC) application, were booted. During this operation, a log window showed the messages issued by the system, as a result of any transaction carried out in order to communicate the details of the communication with the sensors to the user (Figure 6.13).

However, in Figure 6.14 is shown the status of the sensors after the boot operation. The table in the figure shows the MAC address of the sensors, the cell to which they belong, the current status, date and time in which they were active before the last update, the IP address and any flags, respectively.

Subsequently, a power threshold that is related to the signal received by the sensors was set in the LEC. If the received signal strength fell below the threshold, it was rejected as random noise. In Figure 6.15 power thresholds (red horizontal lines) are shown in relation to each sensor and noise signals are displayed (green waves). This operation made it possible to distinguish, in the presence of a tag, the signal noise from a UWB signal associated with a tag.

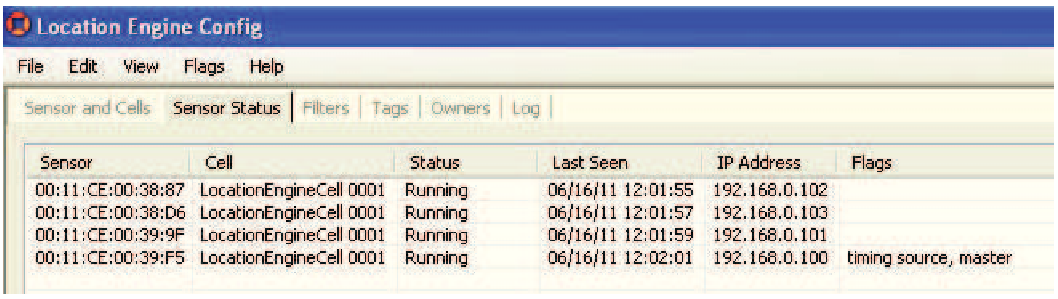
The last step before the calibration of the system consisted of the virtual reconstruction of the map of the sensors inside the LEC (Figure 6.16). A reference system was determined within the graphical section of this software



```
Time               Sensor              Cell                     Message
06/16/11 11:47:55  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2823 : ./auto_revc_master.elf block # 200
06/16/11 11:47:55  00:11:CE:00:38:D6   LocationEngineCell 0001  sensor: timing cable error report for previous 15 seconds:
06/16/11 11:47:55  00:11:CE:00:38:D6   LocationEngineCell 0001  sensor:    proportion of slots with the wrong timeslot length = 1000000ppm
06/16/11 11:47:55  00:11:CE:00:38:D6   LocationEngineCell 0001  sensor:    proportion of slots without a good set of phase samples = 1000000ppm
06/16/11 11:47:55  00:11:CE:00:38:D6   LocationEngineCell 0001  sensor:    proportion of slots without stable timing circuit phase = 1000000ppm
06/16/11 11:47:58  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2823 : ./auto_revc_master.elf block # 300
06/16/11 11:47:59  00:11:CE:00:39:9F   LocationEngineCell 0001  sensor: timing cable error report for previous 15 seconds:
06/16/11 11:47:59  00:11:CE:00:39:9F   LocationEngineCell 0001  sensor:    proportion of slots with the wrong timeslot length = 1000000ppm
06/16/11 11:47:59  00:11:CE:00:39:9F   LocationEngineCell 0001  sensor:    proportion of slots without a good set of phase samples = 1000000ppm
06/16/11 11:47:59  00:11:CE:00:39:9F   LocationEngineCell 0001  sensor:    proportion of slots without stable timing circuit phase = 1000000ppm
06/16/11 11:48:01  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2823 : ./auto_revc_master.elf block # 400
06/16/11 11:48:04  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2823 : ./auto_revc_master.elf block # 500
06/16/11 11:48:06  00:11:CE:00:38:87   LocationEngineCell 0001  sensor: timing cable error report for previous 15 seconds:
06/16/11 11:48:06  00:11:CE:00:38:87   LocationEngineCell 0001  sensor:    proportion of slots with the wrong timeslot length = 1000000ppm
06/16/11 11:48:06  00:11:CE:00:38:87   LocationEngineCell 0001  sensor:    proportion of slots without a good set of phase samples = 1000000ppm
06/16/11 11:48:06  00:11:CE:00:38:87   LocationEngineCell 0001  sensor:    proportion of slots without stable timing circuit phase = 1000000ppm
06/16/11 11:48:07  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2823 : ./auto_revc_master.elf block # 600
06/16/11 11:48:10  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2823 : ./auto_revc_master.elf block # 700
06/16/11 11:48:13  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2823 : ./auto_revc_master.elf block # 800
06/16/11 11:48:16  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2823 : ./auto_revc_master.elf block # 900
06/16/11 11:48:19  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2823 : ./auto_revc_master.elf block # 1000
06/16/11 11:48:22  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2823 : ./auto_revc_master.elf block # 1100
06/16/11 11:48:25  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2823 : ./auto_revc_master.elf block # 1200
06/16/11 11:48:28  00:11:CE:00:38:D6   LocationEngineCell 0001  sensor: timing cable error report for previous 15 seconds:
06/16/11 11:48:28  00:11:CE:00:38:D6   LocationEngineCell 0001  sensor:    proportion of slots with the wrong timeslot length = 1000000ppm
06/16/11 11:48:28  00:11:CE:00:38:D6   LocationEngineCell 0001  sensor:    proportion of slots without a good set of phase samples = 1000000ppm
06/16/11 11:48:28  00:11:CE:00:38:D6   LocationEngineCell 0001  sensor:    proportion of slots without stable timing circuit phase = 1000000ppm
06/16/11 11:48:28  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2823 : ./auto_revc_master.elf block # 1300
06/16/11 11:48:31  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2823 : ./auto_revc_master.elf block # 1400
06/16/11 11:48:35  00:11:CE:00:38:87   LocationEngineCell 0001  sensor: timing cable error report for previous 15 seconds:
06/16/11 11:48:35  00:11:CE:00:38:87   LocationEngineCell 0001  sensor:    proportion of slots with the wrong timeslot length = 1000000ppm
06/16/11 11:48:35  00:11:CE:00:38:87   LocationEngineCell 0001  sensor:    proportion of slots without a good set of phase samples = 1000000ppm
06/16/11 11:48:35  00:11:CE:00:38:87   LocationEngineCell 0001  sensor:    proportion of slots without stable timing circuit phase = 1000000ppm
06/16/11 11:48:37  00:11:CE:00:39:F5   LocationEngineCell 0001  sensor: Ubisense Location Engine Sensor Runtime (Version 2.1 192.168.0.100:0 00:11:CE:00:39:F5)
06/16/11 11:48:37  00:11:CE:00:39:F5   LocationEngineCell 0001  sensor:
06/16/11 11:48:37  00:11:CE:00:39:F5   LocationEngineCell 0001  sensor: (c) 2003-2007 Ubisense Limited, All Rights Reserved
06/16/11 11:48:37  00:11:CE:00:39:F5   LocationEngineCell 0001  sensor:
06/16/11 11:48:37  00:11:CE:00:39:F5   LocationEngineCell 0001  sensor: Build #2526 (Dec 11 2009,20:41:21)
06/16/11 11:48:37  00:11:CE:00:39:F5   LocationEngineCell 0001  sensor: Serial Number #1866-6951-1A6C-4B88
06/16/11 11:48:37  00:11:CE:00:39:F5   LocationEngineCell 0001  sensor:
06/16/11 11:48:37  00:11:CE:00:39:F5   LocationEngineCell 0001  sensor: Configured for 37Hz update rate
06/16/11 11:48:39  00:11:CE:00:39:F5   LocationEngineCell 0001  sensor: programming radio
06/16/11 11:48:39  192.168.0.105       unknown                  bootstrap: Downloading './auto_standard_8.hex' to 192.168.0.100:2310...
06/16/11 11:48:39  192.168.0.105       unknown                  bootstrap: 192.168.0.100:2310 : initialising or missed packet
06/16/11 11:48:39  192.168.0.105       unknown                  bootstrap: 00:11:CE:00:39:F5 192.168.0.100:2310 : ./auto_standard_8.hex block # 1
```

(-4.37788,3.95737)

Figure 6.13: Log window of the LEC.


Location Engine Config
File Edit View Flags Help
Sensor and Cells | Sensor Status | Filters | Tags | Owners | Log

Sensor	Cell	Status	Last Seen	IP Address	Flags
00:11:CE:00:38:87	LocationEngineCell 0001	Running	06/16/11 12:01:55	192.168.0.102	
00:11:CE:00:38:D6	LocationEngineCell 0001	Running	06/16/11 12:01:57	192.168.0.103	
00:11:CE:00:39:9F	LocationEngineCell 0001	Running	06/16/11 12:01:59	192.168.0.101	
00:11:CE:00:39:F5	LocationEngineCell 0001	Running	06/16/11 12:02:01	192.168.0.100	timing source, master

Figure 6.14: The Sensor Status shown in the LEC after sensors boot operation. The table in the figure shows the MAC address of the sensors, the cell to which they belong, the current status, date and time in which they were active before the last update, the IP address and any flags, respectively.

tool in order to precisely mapping the sensors. The position of each sensor was inserted manually by bringing the values corresponding to those of the real reference system where the sensors were physically installed.

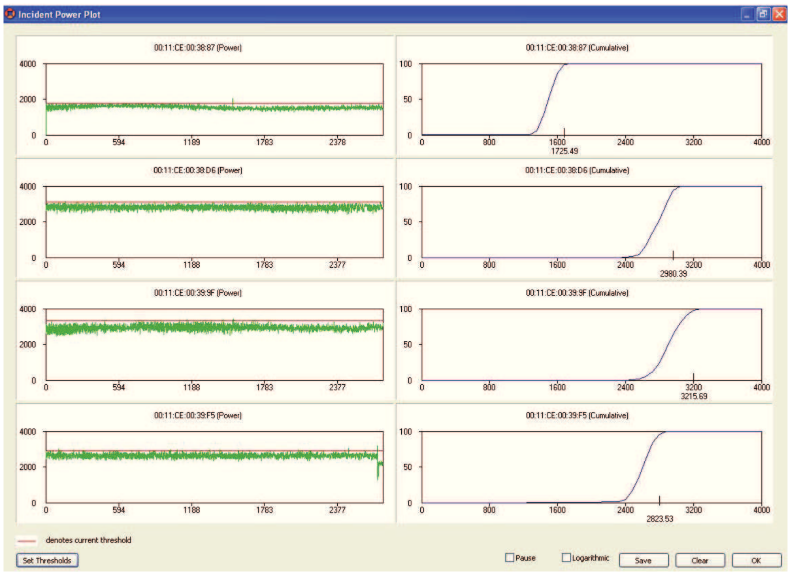

Incident Power Plot
00:11:CE:00:38:87 (Power)
00:11:CE:00:38:87 (Cumulative)
1725.49
00:11:CE:00:38:D6 (Power)
00:11:CE:00:38:D6 (Cumulative)
2980.39
00:11:CE:00:39:9F (Power)
00:11:CE:00:39:9F (Cumulative)
3215.69
00:11:CE:00:39:F5 (Power)
00:11:CE:00:39:F5 (Cumulative)
2823.53
denotes current threshold
Set Thresholds
Pause
Logarithmic
Save
Clear
OK

Figure 6.15: The power thresholds of each sensor, indicated with a red horizontal line, allow to distinguish the noise signal (green wave) from the UWB pulses transmitted by the tags.

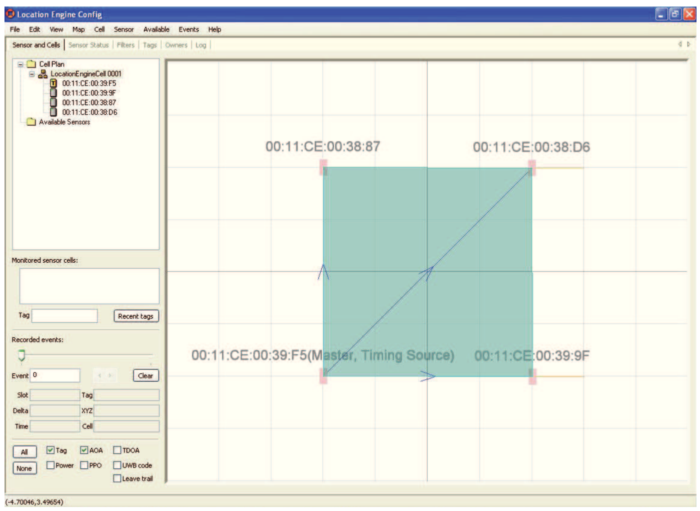


Figure 6.16: 2D sensors map in the graphical section of the LEC. Each sensor is identified by its MAC address represented by a string of 12 hexadecimal digits.

At the sensor having MAC Address corresponding to that of the Master sensor, chosen during hardware installation, the 'Master' and the 'Timing Source' flags were attributed.

6.2.3.5 System calibration

At this point, the system was able to identify and detect the presence of a tag placed in proximity to at least one of the sensors but, however, was not able to accurately determine the relative position in space. Therefore, we proceeded with the calibration of the system within the study area by using the calibration procedure in order to attribute the missing parameters of localisation to the system. The steps that were followed for the calibration of the system were two: sensor orientations and cable offsets. As already introduced in section 5.3.3, the basic principle of the first phase of calibration, also called 'Orientation calibration', once a Ubisense tag was put in a known position within the coverage area of the sensors and the coordinates of the position in the considered reference system were written in the Location Engine Configuration tool, then the system took readings of the AoA from the tag and yielded the orientation of the sensor from the average of the measurements set. This calibration step was performed on each sensor in order to set their parameters of orientation (pitch, yaw and roll) within the network. The position of the tags used for this operation is shown in Figures 6.22. A central point within the study area (close to the barycentre of the rectangle having as vertices the points where the sensors were installed) was chosen, in order to orient the sensors as symmetrically as possible between them. As the cows mainly held the most central areas of the test area, perform a calibration using as reference point a point placed on the central area would have ensured the best performance of the system for the case in question. The second phase of calibration, also called 'Cable calibration', was similar to orientation calibration, except that a cable offset was between two sensors. Therefore, the calibration used two sensors: the sensor getting its cable offset and the reference sensor, which was ideally the master. Cable offsets were just some arbitrary number indicating the offset from the master, so the master had a value of zero. For this reason this latter calibration phase, unlike the previous one, was only carried out on each of the slave sensors.

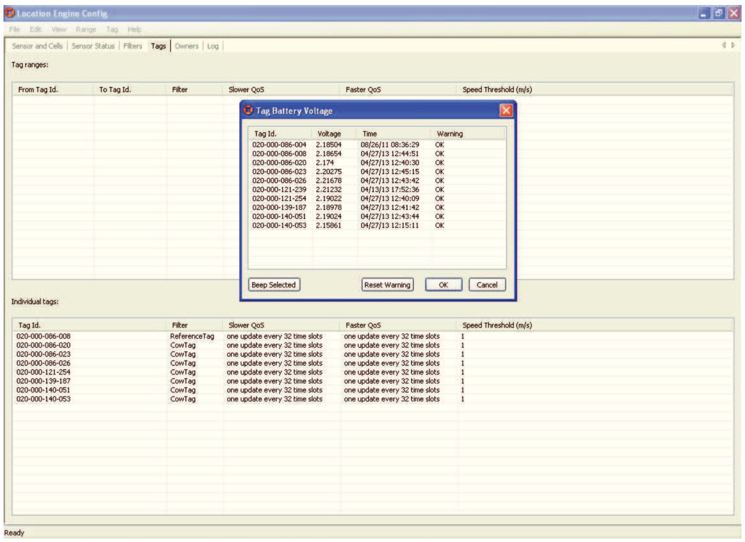
The calibration process described above was carried out only once since the components layout of the structure used for the test remained unaltered during the whole period of observations. Otherwise it would have been necessary to perform further calibrations whenever the layout of the elements present in the area under study had been altered. This observation belong to in the recommendations made in the work carried out by Shahi et al. (2012).

6.2.4 Data acquisition

After having carried out the operations above described, the RTLS was able to identify and determine the location of a tag placed at any point belonging to the coverage area of the sensors.

In this phase the choice of the acquisition time interval of the tags position, during the test was crucial. In fact, as mentioned in section 5.2.2, it's important to maintain an optimal trade-off between the batteries energy consumption and the update frequency of the measurement. A solution to this problem consists in being able to set an update rate, related to the acquisition of the measure, which ensures a battery life within the desired time period but respects, at the same time, the desired update frequency of the measurement.

On the basis of this concept, an acquisition time of 0.865 seconds was chosen in order to ensure a battery life for a period of at least eight weeks, for the tags applied to the cows. This choice was based on the idea that the average speed of a cow in the environment in question is less than 0.5 m/s, and, therefore, it would have been possible to perform the tracking of the animal without losing important information in any context related to the behavior (e.g., lying, standing and feeding). The chosen time rate was manually set by the software thanks to the specific functionality offered by the LEC software tool. In Figure 6.17 the section of the LEC dedicated to the management of the acquisition time and the real-time monitoring of the battery status associated with each tag is shown.


Location Engine Config
File Edit View Range Tag Help
Sensor and Cells | Sensor Status | Filters | Tags | Owners | Log
Beep Selected
Reset Warning
OK
Cancel
Ready

Tag ranges:

From Tag Id.	To Tag Id.	Filter	Slower QoS	Faster QoS	Speed Threshold (m/s)

Tag Battery Voltage

Tag Id.	Voltage	Time	Warning
020-000-086-004	2.18504	08/26/11 08:36:29	OK
020-000-086-008	2.18654	04/27/13 12:44:51	OK
020-000-086-020	2.174	04/27/13 12:40:30	OK
020-000-086-023	2.20275	04/27/13 12:45:15	OK
020-000-086-026	2.21678	04/27/13 12:43:42	OK
020-000-121-239	2.21232	04/13/13 17:52:36	OK
020-000-121-254	2.19022	04/27/13 12:40:09	OK
020-000-139-187	2.18978	04/27/13 12:41:42	OK
020-000-140-051	2.19024	04/27/13 12:43:44	OK
020-000-140-053	2.15861	04/27/13 12:15:11	OK

Individual tags:

Tag Id.	Filter	Slower QoS	Faster QoS	Speed Threshold (m/s)
020-000-086-008	ReferenceTag	one update every 32 time slots	one update every 32 time slots	1
020-000-086-020	CowTag	one update every 32 time slots	one update every 32 time slots	1
020-000-086-023	CowTag	one update every 32 time slots	one update every 32 time slots	1
020-000-086-026	CowTag	one update every 32 time slots	one update every 32 time slots	1
020-000-121-254	CowTag	one update every 32 time slots	one update every 32 time slots	1
020-000-139-187	CowTag	one update every 32 time slots	one update every 32 time slots	1
020-000-140-051	CowTag	one update every 32 time slots	one update every 32 time slots	1
020-000-140-053	CowTag	one update every 32 time slots	one update every 32 time slots	1

Figure 6.17: Battery status and update rate of each tag.

Since there is only a single UWB channel, only one tag can be located at a time. The Location Platform divides time up into a number of 'time slots', and allocates appropriate time slots to tags, so that their update rate is as close as possible to the rate requested for the tag. The conventional radio frequency is used to manage this location schedule, to receive beep and flash requests, and to send button presses and battery power status information. The UWB channel is used only when generating a location, at the scheduled time for each tag.

The tags supports an automatic sleep feature to save battery. When a tag is still for a few moment it will automatically enter 'sleep' mode if this feature is activated. To wake up the tag from this state, it is necessary to simply move it or press the button. This feature, however, was not used in our test because small movements or vibrations of the body of the cow, even when it had not moved, would activate the accelerometer present in the tag, and then 'woke up' the tag.

The LEC software tool provides the user with the management of the events recorded by the RTLS, i.e. informations related to a particular tag being localized in a specific section of its User Interface (UI), as shown in Figure 6.18.

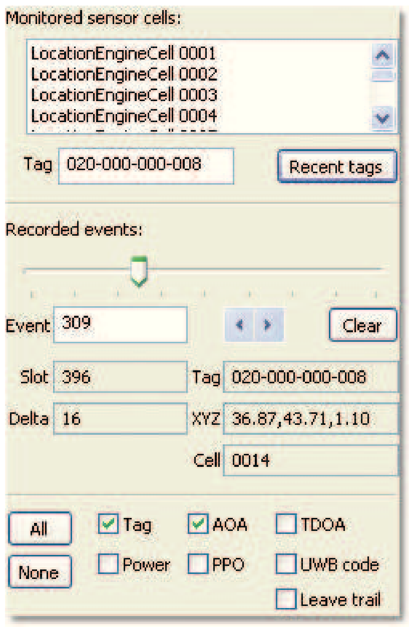

Monitored sensor cells:
LocationEngineCell 0001
LocationEngineCell 0002
LocationEngineCell 0003
LocationEngineCell 0004
Tag 020-000-000-008
Recorded events:
Event 309
Slot 396
Tag 020-000-000-008
Delta 16
XYZ 36.87,43.71,1.10
Cell 0014
[✓ Tag] [✓ AOA] TDOA
Power PPO UWB code
Leave trail

Figure 6.18: LEC monitoring Controls.

The Monitored sensor cells list shows which cells are being monitored.

The Tag text box allows you to filter the tags shown on the Cell Map. The Recent tags button, when clicked, shows a history list of the most recent tag IDs you have entered. The slider and associated controls allow you to rewind recorded events. Every location acquisition by the tag generates an event in this list. The Clear button clears all recorded events. The text boxes show information about the selected event. The slot number refers to the timeslot. Delta means the difference from the previous timeslot for an event for that tag. The tag, XYZ and cell boxes show the tag id, its co-ordinates and the cell id. The check-boxes at the bottom determine what information about events is shown in the Cell Map. The All and None buttons select all check-boxes and no check-boxes, respectively.

6.2.5 Data storage

The software provided by Ubisense, however, does not allow the disk storage of the acquired data from the RTLS.

For this reason, a special software was developed. As already introduced in section 5.3.1.3 you can create software applications interfacing directly with the Ubisense Location Platform and make calls to all services through the use of Ubisense .NET APIs.

According to this consideration, using Microsoft R Visual Studio, a free programming environment distributed by Microsoft R (.Net framework), it was possible to automate the necessary operations for data storage on disk so that to they could, subsequently, be processed.

This tool, which was running on the virtual machine where it was installed the software provided by Ubisense, by executing a connection to the Ubisense Location Platform at 1-s interval, requested the latest data acquired by the sensors and wrote the following information to a file with .csv extension:

acquisition date;

acquisition time;

tag identification number (ID);

location in space, expressed by the coordinates x, y, and z;

standard error.

The main instructions used for the implementation of the software tool are listed below:

Importing Ubisense libraries:

```
using Ubisense.UBase;
using Naming = Ubisense.UName.Naming;
using Ubisense.ULocation;
using Ubisense.ULocation.CellData;
using System.Media;
```

Creation of naming schema object and filewriter output:

```
        // create naming schema object
        private static Ubisense.UName.Naming.Schema
           naming_schema = null;

        // filewriter output
        TextWriter log = new StreamWriter("date.csv",true);
```

Initialization of naming schema object and connection as a client:

```
naming_schema = new Ubisense.UName.Naming.Schema(false);
naming_schema.ConnectAsClient();
```

Creation of multicell object and loading of all Ubisense cells:

```
      MultiCell multicell = new MultiCell();
      SortedDictionary<string, Cell> cells =
         multicell.GetAvailableCells();
      foreach (Cell _cell in cells.Values)
        {
            multicell.LoadCell(_cell, true);
        }
```

Getting all objects out of Ubisense DB:

```
      using (ReadTransaction xact =
         multicell.Schema.ReadTransaction())
       foreach (Location.RowType r in
          Ubisense.ULocation.CellData.Location.object_(xact))
        LocationOutput(r);
```

Definition of update event handler used if an Ubisense object moves:

```
   Ubisense.ULocation.CellData.Location.AddUpdateHandler(
      multicell.Schema, CellData_Update);
```

Event-Handler of Ubisense database location update:

```
      private void CellData_Update(Location.RowType old_row,
         Location.RowType new_row)
```

```
            {
            if (this.InvokeRequired)
                this.BeginInvoke(new MethodInvoker(delegate()
                    { CellData_Update(old_row, new_row); }));
            else
                LocationOutput(new_row);
        }
```

Output of the location of the object within Location.RowType r:

```
        private void LocationOutput(Location.RowType r)
        {
            string id = r.object_.Id.ToString().Substring(15);
            string type = r.object_.DynamicType.Name;
            string stdErr = r.accuracy_.stderr_.ToString();
            double x = Math.Round(r.position_.P.X, 3);
            double y = Math.Round(r.position_.P.Y, 3);
            double z = Math.Round(r.position_.P.Z, 3);
            string datetime = r.time_.ToLocalTime().ToString() +
                "," + r.time_.Millisecond;
            object[] values = { datetime, id, x, y, z, type };
            StringBuilder sb = new StringBuilder(datetime);

            sb.Append(";").Append(id).Append(";").Append(x).
                Append(";").Append(y).Append(";").Append(z).
                Append(";").Append(stdErr).Append(";");

            log.WriteLine(sb.ToString());
            log.Flush();

            insertInList(id);
        }
```

Acquisition of the name of the given object:

```
        static protected string object_name(UObject obj)
        {
            if (naming_schema != null)
                using (Ubisense.UName.Naming.ReadTransaction
                    xact = naming_schema.ReadTransaction())
                     foreach
                        (Ubisense.UName.Naming.ObjectName.RowType
                        row in Ubisense.UName.Naming.ObjectName.
                        object_name_(xact, obj))
                         return row.name_;

            return "(unnamed " + obj.DynamicType + ")";
        }
```

Furthermore, the software allows the visualisation of a 2D-reproduction of the plan view of the study area, which was implemented by using the functions provided by the following libraries of 'Emgu Computer Vision' (EmguCV, 2013):

Importing Emgu libraries:

```
        using Emgu.CV;
        using Emgu.CV.UI;
        using Emgu.CV.CvEnum;
        using Emgu.CV.Structure;
```

In Figure 6.19 the Emgu CV Architecture Overview is shown.

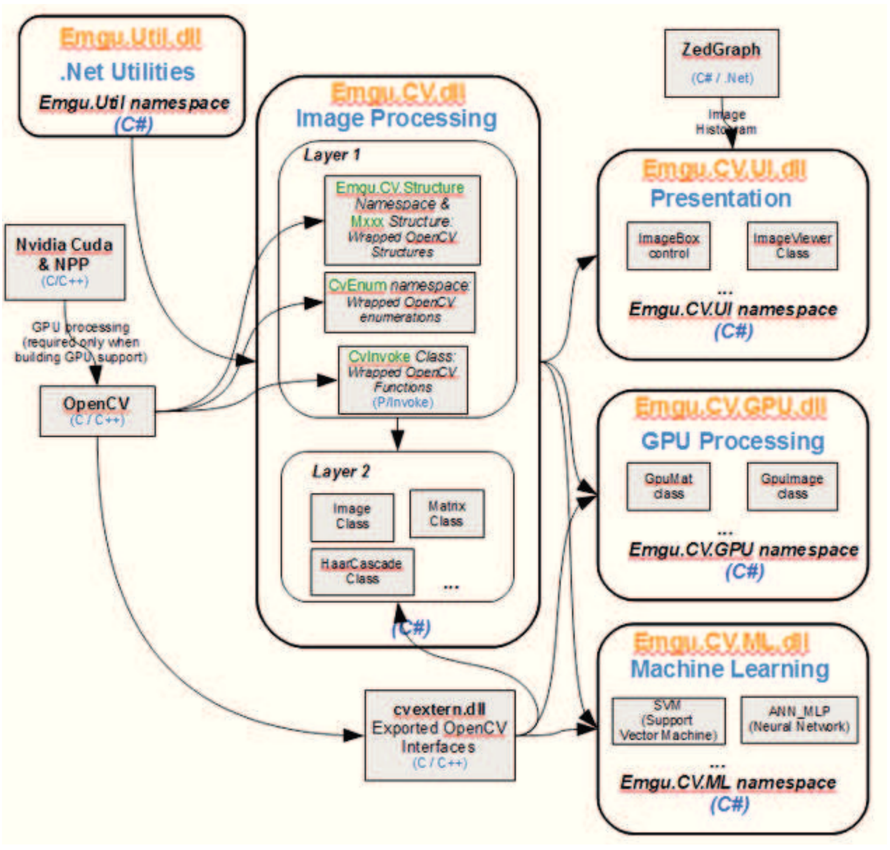
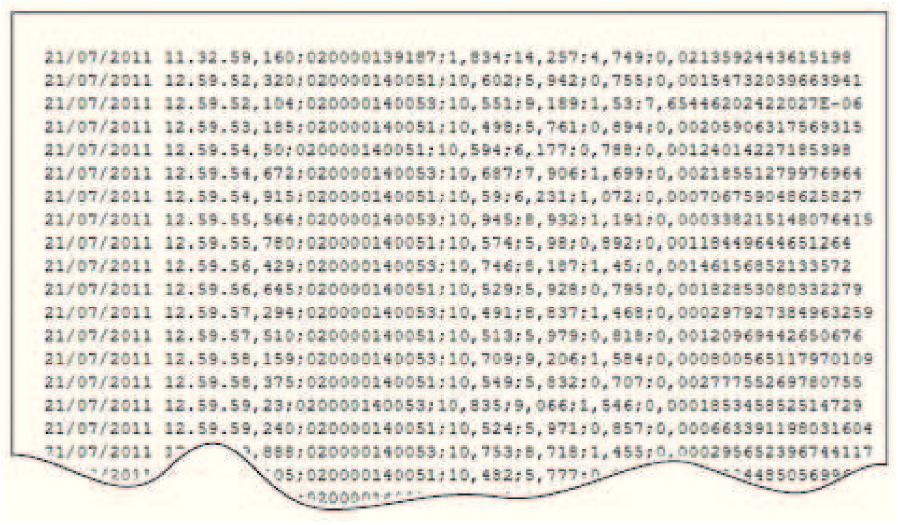


Figure 6.19: Emgu CV Architecture Overview (EmguCV, 2013).

The output produced by the tool is presented in the format shown in Figure 6.20.



```
21/07/2011 11.32.59,160;020000139187;1,834;14,257;4,749;0,0213592443615198
21/07/2011 12.59.52,320;020000140051;10,602;5,942;0,755;0,0015473203966394l
21/07/2011 12.59.52,104;020000140053;10,551;9,189;1,53;7,65446202422027E-06
21/07/2011 12.59.53,185;020000140051;10,498;5,761;0,894;0,0020590631756931S
21/07/2011 12.59.54,50;020000140051;10,594;6,177;0,788;0,0012401422718539B
21/07/2011 12.59.54,672;020000140053;10,687;7,906;1,699;0,00218551279976964
21/07/2011 12.59.54,915;020000140051;10,59;6,231;1,072;0,000706759048625827
21/07/2011 12.59.55,564;020000140053;10,945;8,932;1,191;0,000338215148076415
21/07/2011 12.59.55,780;020000140051;10,574;5,98;0,892;0,00118449644651264
21/07/2011 12.59.56,429;020000140053;10,746;8,187;1,45;0,00146156852133572
21/07/2011 12.59.56,645;020000140051;10,529;5,928;0,795;0,00182853080332279
21/07/2011 12.59.57,294;020000140053;10,491;8,837;1,468;0,000297927384963259
21/07/2011 12.59.57,510;020000140051;10,513;5,979;0,818;0,00120969442650676
21/07/2011 12.59.58,159;020000140053;10,709;9,206;1,584;0,000800565117970109
21/07/2011 12.59.58,375;020000140051;10,549;5,832;0,707;0,00277755269780755
21/07/2011 12.59.59,23;020000140053;10,835;9,066;1,546;0,000185345852514729
21/07/2011 12.59.59,240;020000140051;10,524;5,971;0,857;0,000663391198031604
21/07/2011 1    ,888;020000140053;10,753;8,718;1,455;0,000295652396744117
  /2011         05;020000140051;10,482;5,777;0                448505699
                 020000
```

Figure 6.20: Format of the output produced by the tool: 'acquisition date acquisition time; tag ID; x; y; z; standard error'.

6.2.5.1 Populating Database

Simultaneously with 'write' operations on the file aimed to disk storage of data acquired by the RTLS, an extension of the software described above was specifically developed in this thesis work in order to automatically populate a database with these data and allow its subsequent processing.

Figure 6.21 shows the flow chart related to the operations implemented by the software.

Initially, database updating was automated in order to allow the backup of data in such data structure. By using a timer, a time is set which the follows operations are cyclically iterated:

- Getting all object out of Ubisense database.
- Event-handler of Ubisense Location Update.
- Storing cow location data in a temporary data structure.

The first two operations, as described in the previous section, implement the functions of reading the records in the Ubisense database and updating the variables where the extracted data are stored, respectively. Subsequently, this information was stored in a temporary data structure.

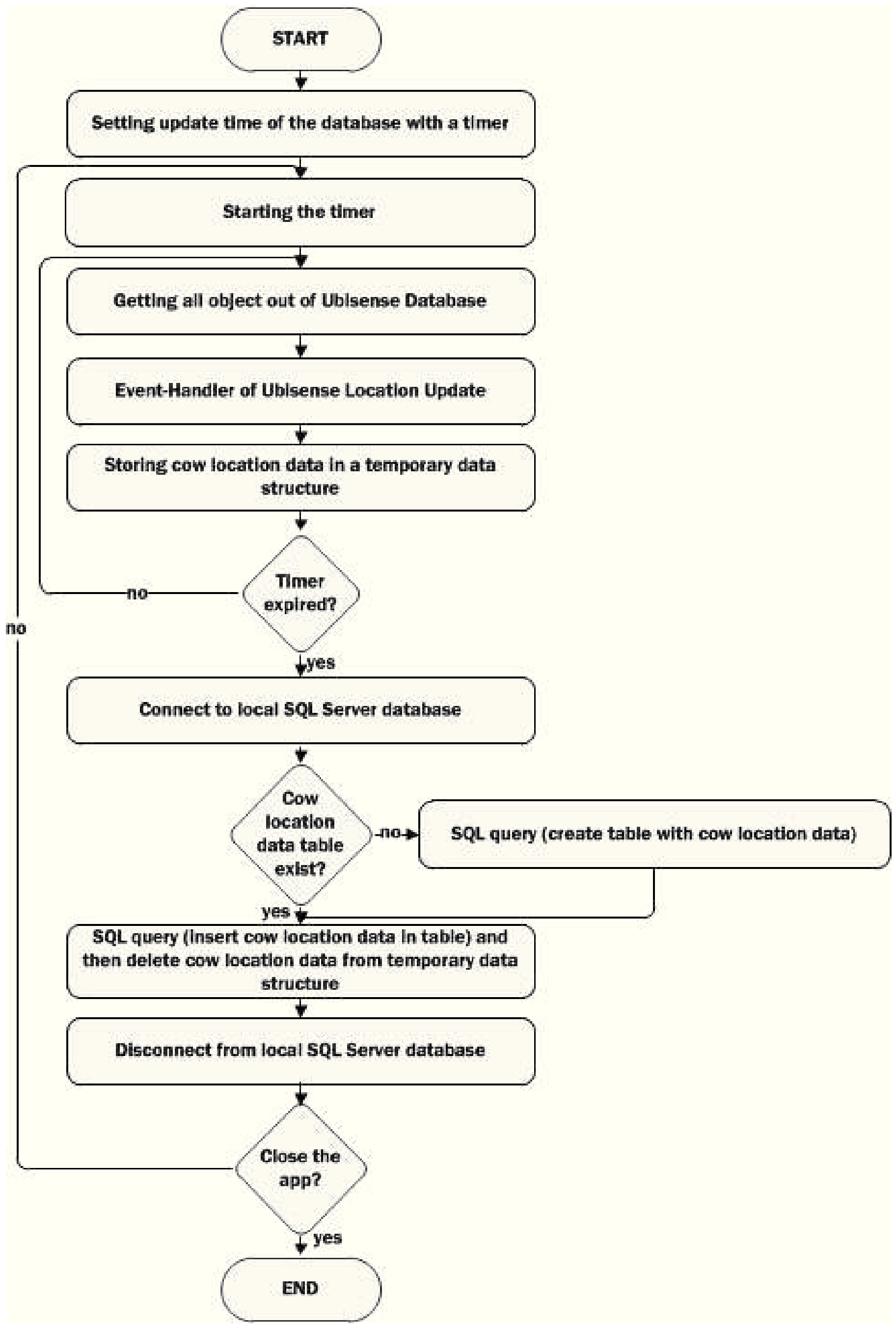

START
Setting update time of the database with a timer
Starting the timer
Getting all object out of Ubisense Database
Event-Handler of Ubisense Location Update
Storing cow location data in a temporary data structure
Timer expired?
no
yes
Connect to local SQL Server database
Cow location data table exist?
no
SQL query (create table with cow location data)
yes
SQL query (insert cow location data in table) and then delete cow location data from temporary data structure
Disconnect from local SQL Server database
Close the app?
no
yes
END

Figure 6.21: Flow chart of the software for the automatic storage of data acquired by the RTLS on a database.

When the time set in the timer expires, the software starts a connection to a SQL Server database, previously installed locally, i.e. in the same virtual machine where the software and the Ubisense Location Platform were launched.

Therefore, through a SQL query a table is created (if it not existing) where the cow location data through an insert query can be stored. After this operation the information is automatically removed from the data temporary structure.

Finally, after disconnecting from the SQL Server database, all the steps described above are repeated until the end of the application.

6.2.6 Static test

An accurate operation of testing, after a careful calibration phase and after the development and installation of the software for the storage of data acquired by the RTLS, was performed inside the study area. This test consisted in placing a tag in a number of 25 known points and then record the location data acquired by the RTLS for a few minutes. This test allowed to evaluate the localisation performance of the system under static conditions along the entire area and determine any malfunctions related to the geometric conditions. Another objective of this test was to store the results obtained in order to subsequently compare them with those obtained in a different scenario, where, on the contrary, the system would have to simultaneously locate multiple tags in unchanged geometrical conditions.

As already mentioned, a series of well determined points within the area in question was defined. By using an accurate laser rangefinder and referring to the cartesian reference system determined in the design phase of the sensors layout (section 6.2.3.2), the coordinates in three-dimensional Euclidean space of these points were measured. Next, in the considered area a tag was introduced and, at intervals of two minutes, it was placed in each of the selected points allowing the RTLS to capture and record its position at 2-seconds intervals. Obviously, the measurements acquired by the RTLS in the instants where the tag was moved from one point to another were discarded.

Figure 6.22 shows the layout of the points chosen for the realization of this test.

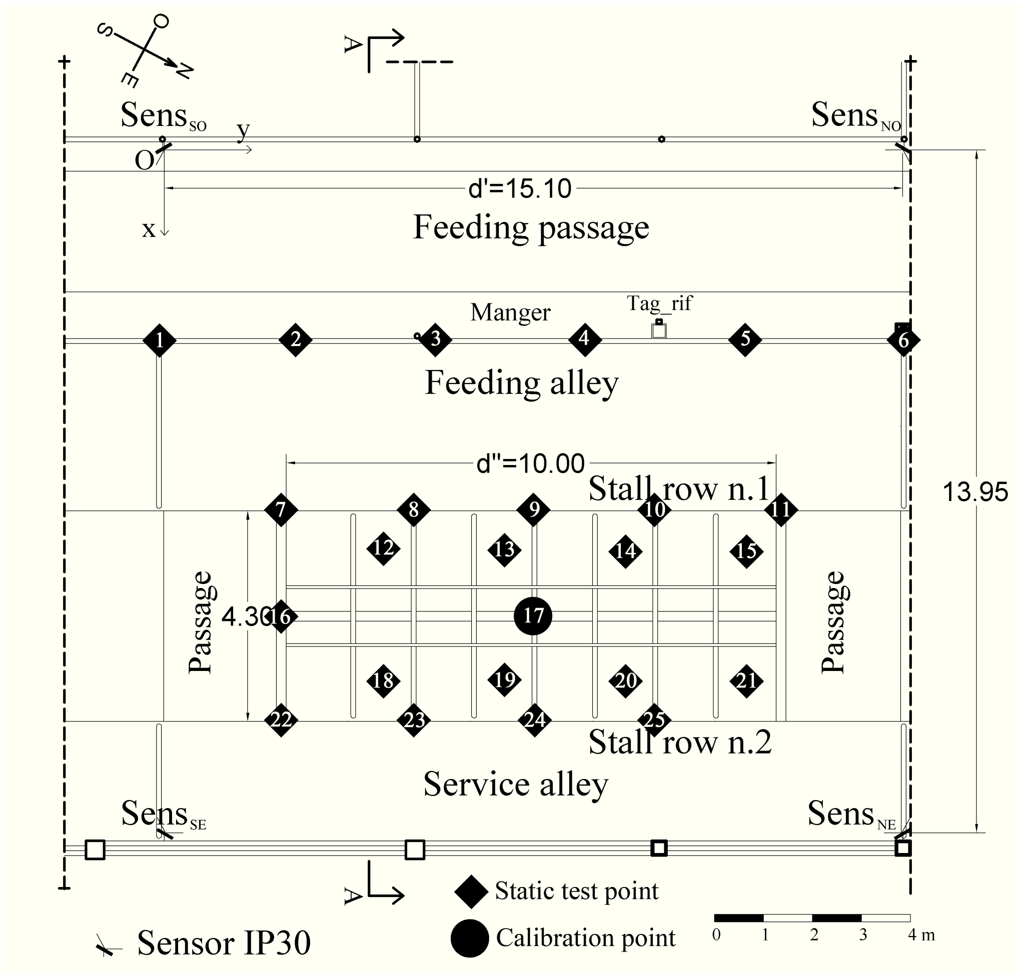


Figure 6.22: Static test points and calibration point in the area of the barn under study.

6.2.7 Cow tags application

Once the static test was carried out, eight active tags were installed on the eight cows. In literature different techniques was used to apply RFID tags to the body of the animals (Finkenzeller, 2003; Ruiz-Garcia & Lunadei, 2011). In order to avoid introducing foreign objects into the animal body, the small size and weight of the tags made it possible to fix them in the animals' ears, by means of a suitable plier, as shown in Figure 6.23(a). To ensure resistance to water, mud, humidity and impact tags, tags were covered with adhesive insulating material and placed in a waterproof bag constituted by synthetic leather and plastic material, as shown in Figure 6.23(b).

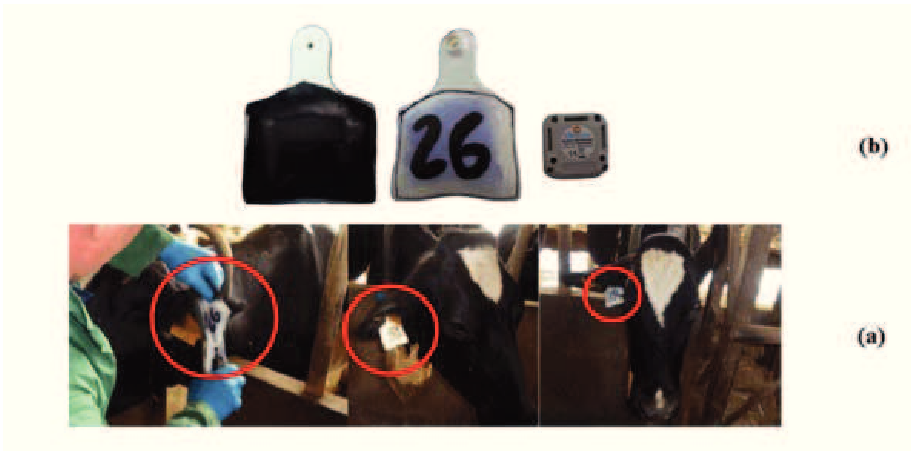


Figure 6.23: (b) Protecting wrapping of tags (front and back) and a Compact Ubisense Tag. (a) Application of the tags to three cows.

However, referring to another study where the tag application method regarded tag positioning on cow's collar and on special bands (Huhtala et al., 2007), the use of an additional technique of tag application was assessed in order to avoid infections or modifications in cow's behavior due to the weight on the ear.

The application of the tags on cow's collar is part of an other trial of this research which will be described in section 6.5.4.

By using plastic cable ties, eight Compact Ubisense tags were applied to eight collars, and subsequently covered by adhesive tape to isolate them from water, mud and moisture. In Figure 6.24 the result is shown: (a) a tag installed in a collar, (b) the outer side of the collar, (c) the inner side of the collar, (d) the eight collars after having isolated tags with adhesive tape.

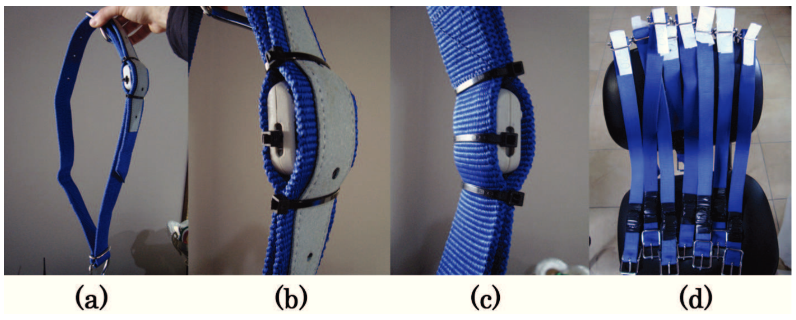


Figure 6.24: Preparation of the collars: (a) a tag installed in a collar using plastic cable ties, (b) the outer side of the collar, (c) the inner side of the collar, (d) the eight collars after having isolated the tags with an adhesive tape.

In Figure 6.25, instead, a photo of the installation process of a collar, equipped with a Ubisense tag, to a cow is shown.

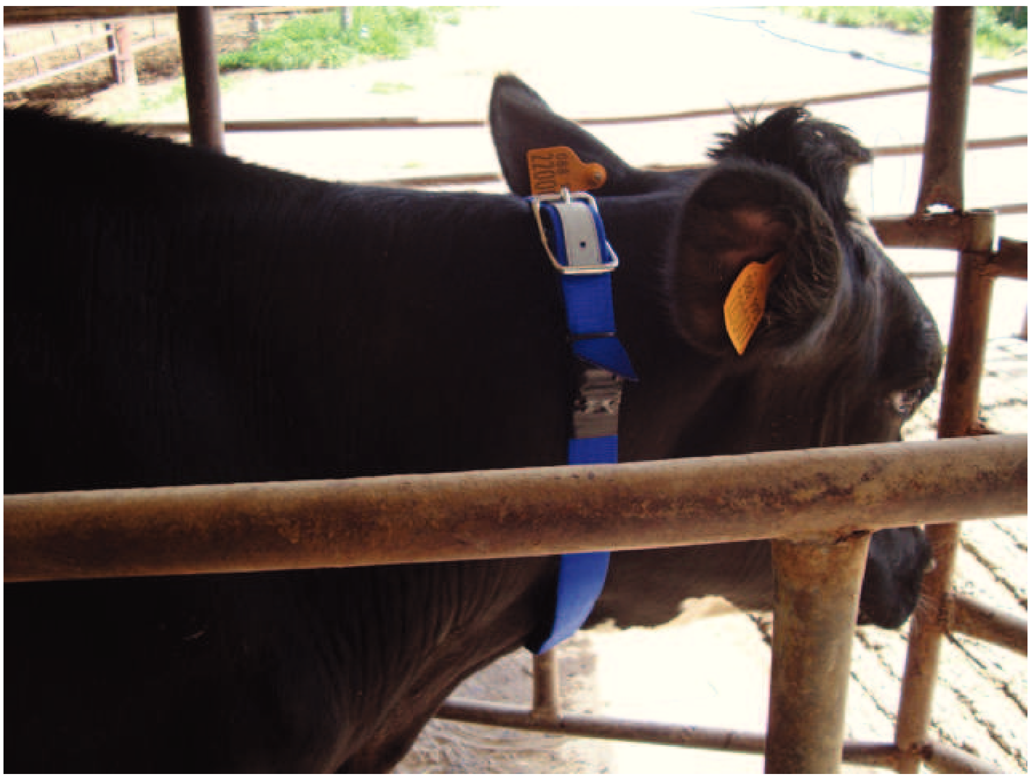


Figure 6.25: Application of a collar equipped with tag to a cow.

6.2.8 Filtering algorithms

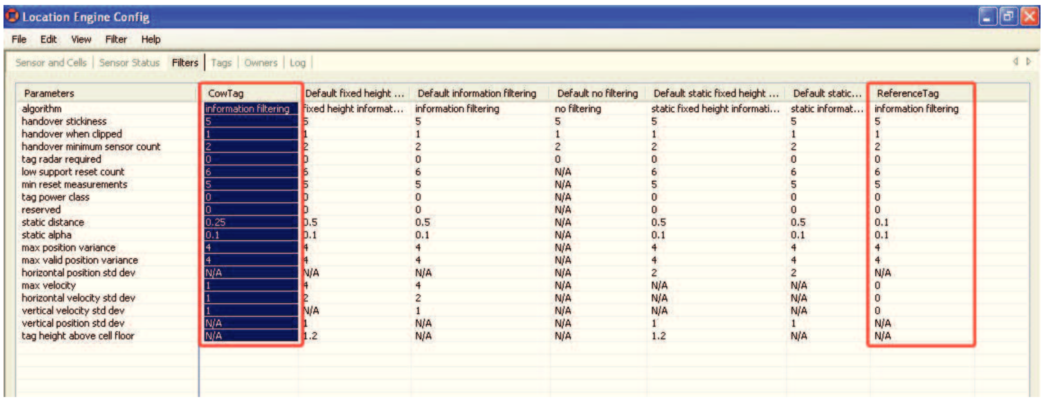
Before proceeding with the storage of data acquired by the RTLS related to the localisation of the cows' tags, a final setting operation was performed by the software in the LEC tool.

The Location Engine supports different algorithms for estimating tag positions from sensor measurements. Each algorithm can have a number of parameters that control the behaviour of the algorithm, and can include known constraints on the motion of tracked tags. The Location Engine allows a set of parameters to be stored as a 'filter', that can be applied to individual tags or a range of tags. Filters are defined in the Filters tab of Location Engine Config (Ubisense, 2008-2010).

In other words, you can customize the system according to the context in which it's installed, in order to optimize resources and to improve performance. It allows you to set a number of constraints of speed, fixed or variable height, max position variance, position standard deviation, etc., related to the tags, in order to filter measures on the basis of the elements that characterize the usage scenario.

For our test, two filter algorithms ('CowTag' Filter and 'ReferenceTag' Filter) were customized in order to associate them to the tags applied to cows and to a tag selected to be installed in a fixed point inside the study area, respectively.

In Figure 6.26 it is possible to observe the set parameters for the two filters in question, highlighted by a red box.


Location Engine Config
File Edit View Filter Help
Sensor and Cells | Sensor Status | Filters | Tags | Owners | Log

Parameters	CowTag	Default fixed height ...	Default information filtering	Default no filtering	Default static fixed height ...	Default static...	ReferenceTag
algorithm	information filtering	fixed height informat...	information filtering	no filtering	static fixed height informati...	static informat...	information filtering
handover stickiness	5	5	5	5	5	5	5
handover when clipped	1	1	1	1	1	1	1
handover minimum sensor count	2	2	2	2	2	2	2
tag radar required	0	0	0	0	0	0	0
low support reset count	6	6	6	N/A	6	6	6
min reset measurements	5	5	5	N/A	5	5	5
tag power class	0	0	0	N/A	0	0	0
reserved	0	0	0	N/A	0	0	0
static distance	0.25	0.5	0.5	N/A	0.5	0.5	0.1
static alpha	0.1	0.1	0.1	N/A	0.1	0.1	0.1
max position variance	4	4	4	N/A	4	4	4
max valid position variance	4	4	4	N/A	4	4	4
horizontal position std dev	N/A	N/A	N/A	N/A	2	2	N/A
max velocity	1	4	4	N/A	N/A	N/A	0
horizontal velocity std dev	1	2	2	N/A	N/A	N/A	0
vertical velocity std dev	1	N/A	1	N/A	N/A	N/A	0
vertical position std dev	N/A	1	N/A	N/A	1	1	N/A
tag height above cell floor	N/A	1.2	N/A	N/A	1.2	N/A	N/A

Figure 6.26: Filters tab of Location Engine Config and customized filters in the red rectangles.

One of the differences between two filters, for example, was the maximum speed of the tags to locate, set to 1 m/s relative to the 'CowTag' Filter, since a cow can hardly exceed this speed in these environmental conditions, and to 0 m/s for the 'ReferenceTag' Filter, because the tag was physically installed in a fixed point.

Further details about all the other parameters, could be found in the Ubisense guide (Ubisense, 2008-2010).

6.2.9 Cow localisation

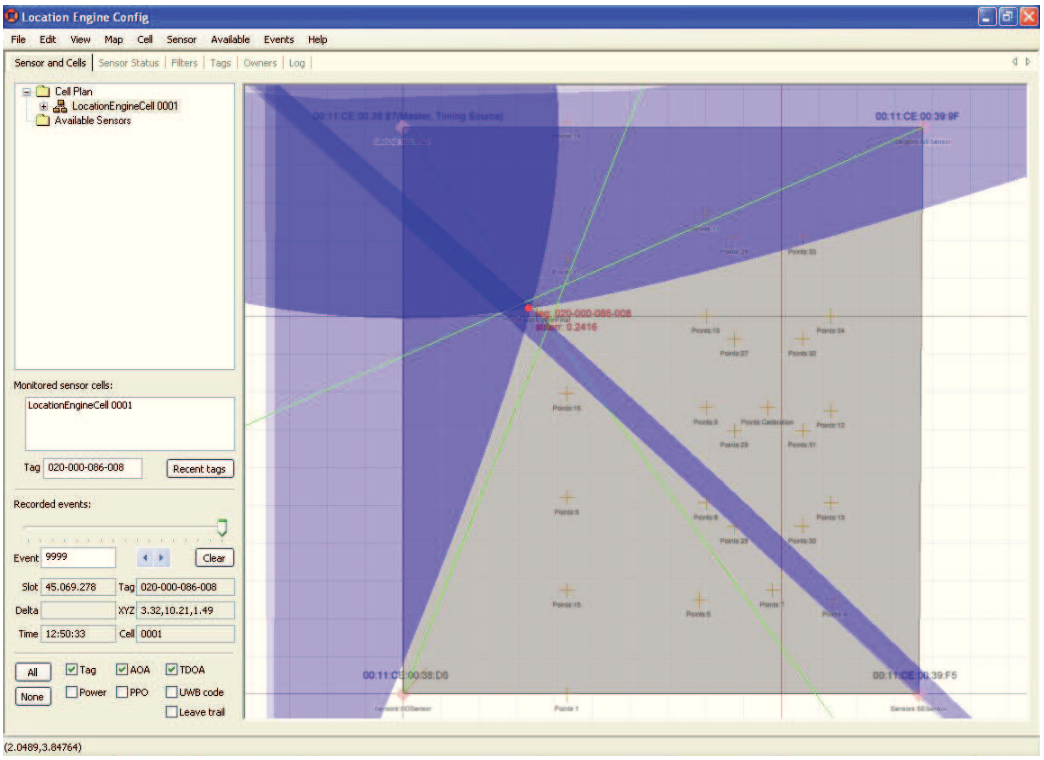
After the filter setting it was possible to carry out the operation of cow localisation.

During this phase, by starting the LEC tool, it was possible to monitor in real time all the data, captured by the RTLS, related to the cows' tags and the reference tag.

Figures 6.27, 6.28, 6.29 and 6.30 show screenshots of the LEC during this phase. In Particular, the Figure 6.27 shows the 2D view of the area of the test where the reference tag is represented by a graphic element with a shape of a red circle. Green lines represent the angle of arrival AOA and blue curves represent the time difference of arrival TDOA. The elements represented by the symbol ’+’ are the points used for static test (section 6.2.6) and for calibration, which were manually entered into the software.

In Figure 6.28, instead, in the 3D reproduction of the study area is shown and the displayed tag is one of the tags applied to the cows.

Figures 6.29 and 6.30 show, instead, the position of the reference tag estimated by the RTLS relative to reference tag, which corresponds exactly to the point indicated by the symbol ’+’. This point, in turn, corresponds to the point where was physically installed that tag, measured in the real environment.



Monitored sensor cells: LocationEngineCell 0001

Tag 020-000-086-008

Recorded events:

Event 9999

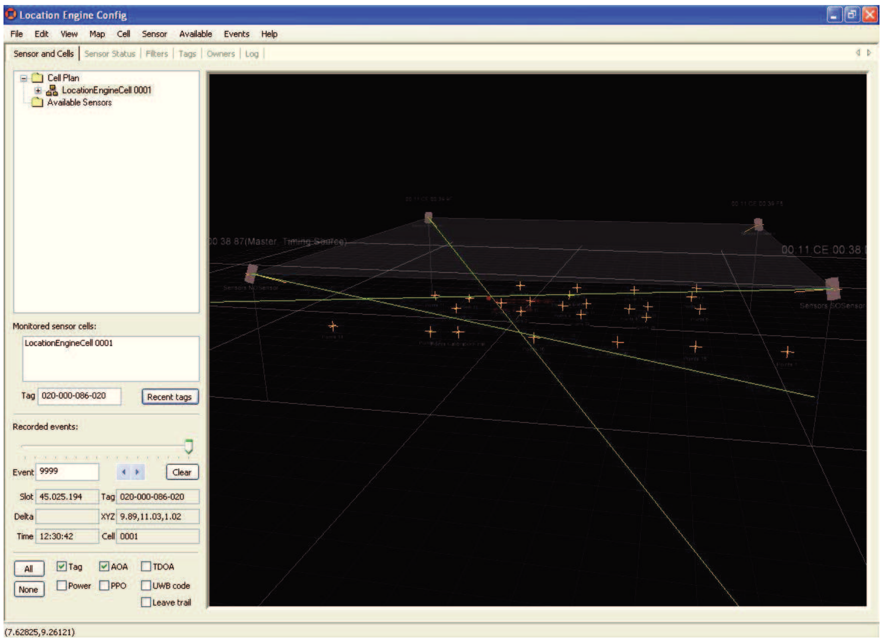
Slot 45.069.278 Tag 020-000-086-008

Delta XYZ 3.32,10.21,1.49

Time 12:50:33 Cell 0001

(2.0489,3.84764)

Figure 6.27: 2D view of the object of the test in the LEC tool.


Location Engine Config
File Edit View Map Cell Sensor Available Events Help
Sensor and Cells | Sensor Status | Filters | Tags | Owners | Log
Cell Plan
LocationEngineCell 0001
Available Sensors
Recent tags
Clear
All
None
0 38 87(Master, Timing Source)
00 11 CE 00 38

Monitored sensor cells:
LocationEngineCell 0001

Tag 020-000-086-020

Recorded events:

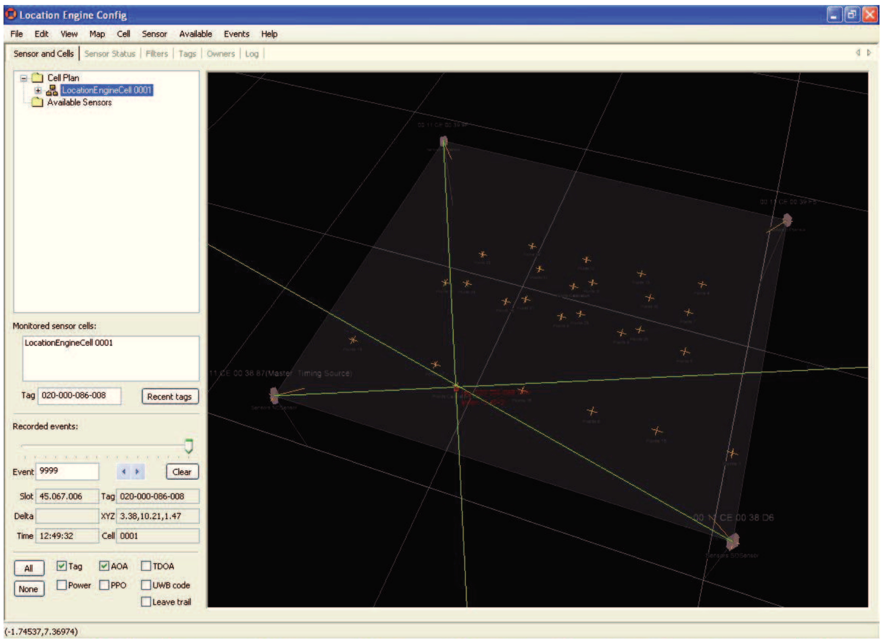
Event 9999

Slot 45.025.194 Tag 020-000-086-020
Delta XYZ 9.89,11.03,1.02
Time 12:30:42 Cell 0001

✓ Tag ✓ AOA ☐ TDOA
☐ Power ☐ PPO ☐ UWB code
☐ Leave trail

(7.62825,9.26121)

Figure 6.28: 3D view of the study area. The displayed tag is one of the tags applied to the cows.


Location Engine Config
File Edit View Map Cell Sensor Available Events Help
Sensor and Cells | Sensor Status | Filters | Tags | Owners | Log
Cell Plan
LocationEngineCell 0001
Available Sensors
Recent tags
Clear
All
None
11 CE 00 38 87(Master, Timing Source)
00 11 CE 00 38 D6

Monitored sensor cells:
LocationEngineCell 0001

Tag 020-000-086-008

Recorded events:

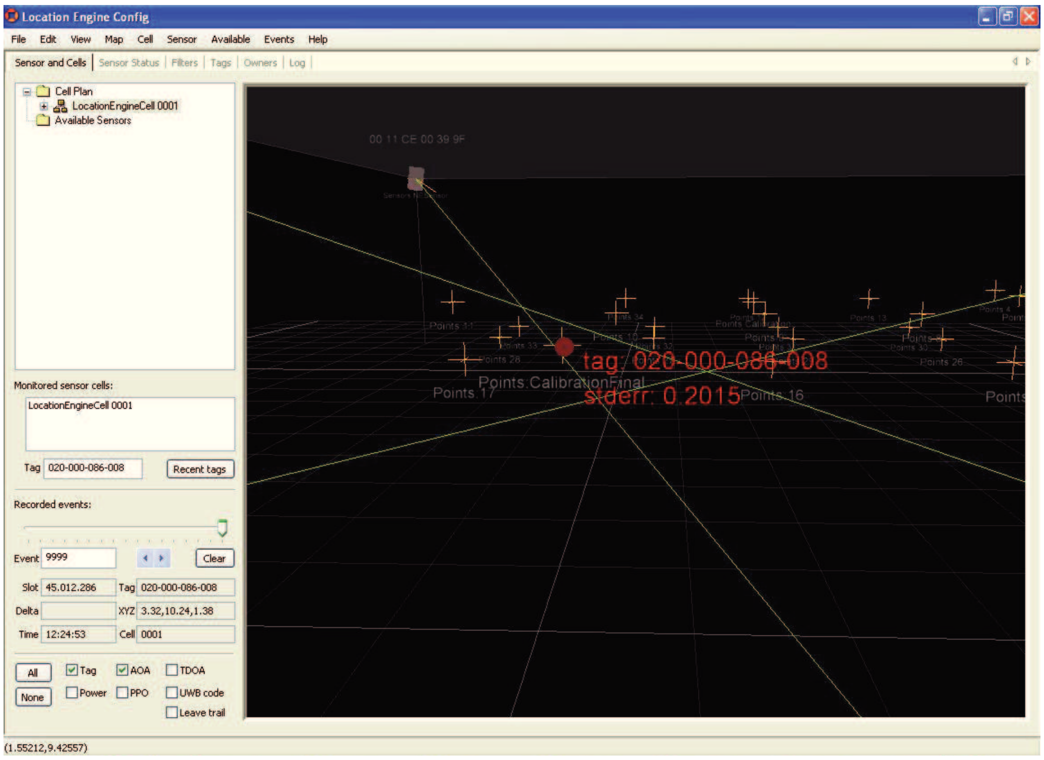
Event 9999

Slot 45.067.006 Tag 020-000-086-008
Delta XYZ 3.38,10.21,1.47
Time 12:49:32 Cell 0001

✓ Tag ✓ AOA ☐ TDOA
☐ Power ☐ PPO ☐ UWB code
☐ Leave trail

(-1.74537,7.36974)

Figure 6.29: 3D view of the study area. The displayed tag is the reference tag.



Monitored sensor cells:
LocationEngineCell 0001

Tag 020-000-086-008

Recorded events:

Event 9999

Slot 45.012.286 Tag 020-000-086-008

Delta XYZ 3.32,10.24,1.38

Time 12:24:53 Cell 0001

[✓ Tag] [✓ AOA] ☐ TDOA

☐ Power ☐ PPO ☐ UWB code

☐ Leave trail

(1.55212,9.42557)

Figure 6.30: Another point of view of the reference tag in LEC.

6.3 THE MULTI-CAMERA VIDEO-RECORDING SYSTEM

The accuracy of the localisation and identification achieved through the use of the RTLS was assessed by using information from a multi-camera video-recording system that was installed in the area of the barn under study (Figure 6.31). Among possible views obtainable from a video-recording system, those providing plan views of the barn are the most suitable for the recognition of cow behavioural activities. Plan views of the barn make it possible to distinguish each cow from the others and determine the real position of each cow within the barn. To achieve these objectives, the video-recording system, which was designed according to a previously defined methodology (Porto et al., 2013), provided panoramic rectified top-view images of the barn to obtain real dimensions of cows, physical spaces, and equipment. The application of this system (Porto et al., 2013) proved that it is suitable to detect cow lying and feeding behaviours in free-stall barns.

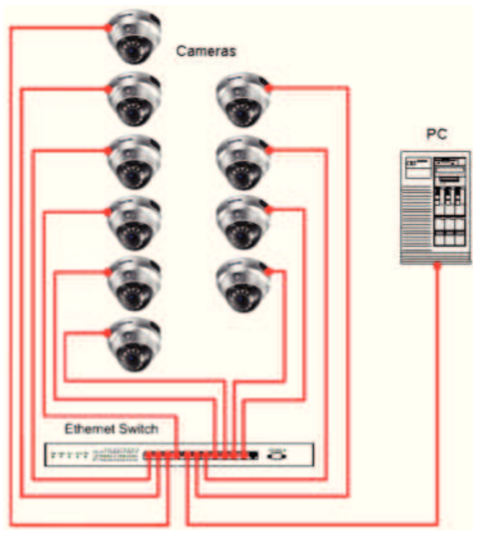


Figure 6.31: Hardware scheme of the video-recording system.

6.4 LOCALISATION AND IDENTIFICATION PERFORMANCES OF UBISENSE UWB RTLS

6.4.1 Automatic data preprocessing

The construction of panoramic plan-view images of the area under study made it possible to verify the planimetric position of each tag, provided by the RTLS.

The data analysed in the following section were obtained from the video-recordings carried out on 2nd August, 2011 during a time interval of about 54 minutes. Within this time interval two different cow behaviours were observed. The first time interval between 06h:26m:49s and 06h:53m:39s (about 27 minutes) included cow localisation during the feeding activity at the manger, whereas the second time interval between 11h:35m:37s and 12h:02m:27s (about 27 minutes) regarded cow lying behaviour in the stalls.

According to one of the objectives of this thesis work, which consisted in the evaluation of RTLS accuracy to monitor the feeding and lying activities of the cows, the service alley adjacent to the second row of stalls was not framed by the multi-camera system. Therefore, if the cow was in the service alley adjacent to the second row of bunks and therefore was not caught on the camera, the operator had the ability to report such an event and the corresponding measurements were excluded from subsequent analyses.

Firstly, two different datasets were defined: one contained the tag positions measured by the RTLS (section 6.4.1.1), whereas the other contained the panoramic plan-view images recorded by the multi-camera system (section 6.4.1.2). The information of the two datasets was linked together (section 6.4.1.3).

6.4.1.1 The dataset of the tags

As already described in section 6.2.5, the structure of the records relating to the data acquired by the RTLS, stored on disk by the tool specially developed, is the following:

```
(DD/MM/YYYY HH.MM.SS; ID_TAG; X; Y; Z; STD_ERROR)
```

Importing data on database

By using Microsoft® Office Access software a database was created and it was then populated by entering all the data acquired by the RTLS in a single table. This operation has facilitated the manipulation of data.

In fact, formatting operations (i.e., change the time format), elimination of duplicate queries and sorting filters by date, were carried out.

Exporting data from database

The database was exported to a text file through a tool specially developed in C#. This application initially performs a database connection and then performs a query to the records, which are gradually written to a text file. The code for database connection and query of select data execution is reported below.

```
// Select query
string strAccessSelect = "SELECT * FROM TagTable";

// Dataset creation:
DataSet myDataSet = new DataSet();

OleDbConnection myAccessConn = null;
try
{
    // Database connection
    myAccessConn = new
        OleDbConnection(strAccessConn);
}
catch(Exception ex)
{
```

```
                Console.WriteLine("Error: Failed to create a
                    database connection. \n{0}", ex.Message);
                return;
            }

            try
            {
                OleDbCommand myAccessCommand = new
                    OleDbCommand(strAccessSelect, myAccessConn);
                OleDbDataAdapter myDataAdapter = new
                    OleDbDataAdapter(myAccessCommand);

            // Opening database connection
            myAccessConn.Open();
                    myDataAdapter.Fill(myDataSet, "TagTable");
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error: Failed to retrieve the
                    required data from the DataBase.\n{0}",
                    ex.Message);
                return;
            }
            finally
            {
                // closing database connection
                myAccessConn.Close();
            }
```

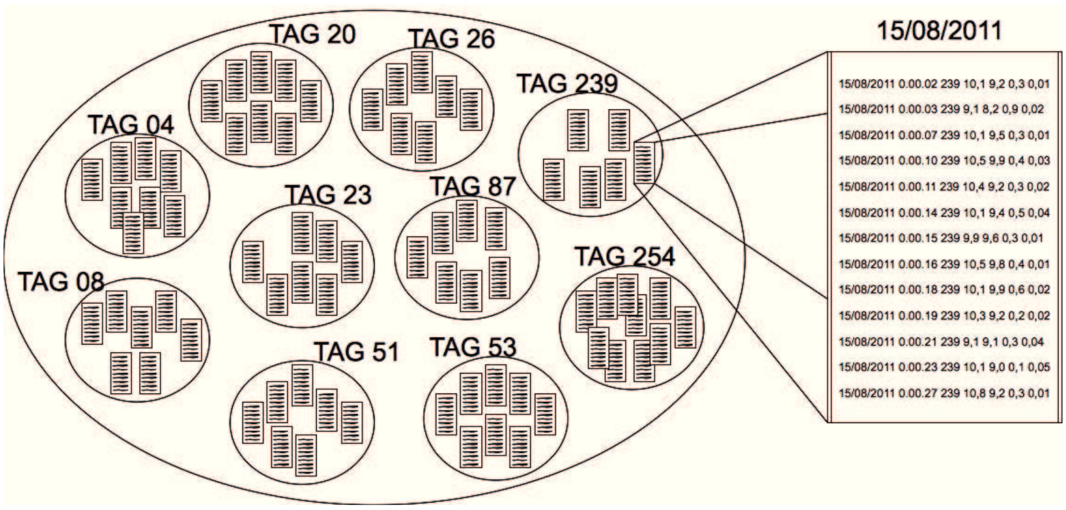
Therefore, the selected records were written to a text file by using a StreamWriter.

The structure of the records within these files is the following:

```
DD/MM/YYYY HH.MM.SS ID_TAG X Y Z STD_ERROR
```

Reorganisation of the dataset

In addition, through the software tool it was possible to automate the partitioning of the data into groups, each representing a particular tag. Finally, additional operation of data organization were automated, creating subgroups, each of them related to a day of recording. In other words, the dataset of the tag was constituted by a number of folders equal to the number of tags, each of them containing a number of text files equal to the number of days of registration of the tag to which they were attached. The record structure remained unchanged. In Figure 6.32, the structure of the tags dataset, obtained after the execution of such software, is shown.



```
15/08/2011 0.00.02 239 10,1 9,2 0,3 0,01
15/08/2011 0.00.03 239 9,1 8,2 0,9 0,02
15/08/2011 0.00.07 239 10,1 9,5 0,3 0,01
15/08/2011 0.00.10 239 10,5 9,9 0,4 0,03
15/08/2011 0.00.11 239 10,4 9,2 0,3 0,02
15/08/2011 0.00.14 239 10,1 9,4 0,5 0,04
15/08/2011 0.00.15 239 9,9 9,6 0,3 0,01
15/08/2011 0.00.16 239 10,5 9,8 0,4 0,01
15/08/2011 0.00.18 239 10,1 9,9 0,6 0,02
15/08/2011 0.00.19 239 10,3 9,2 0,2 0,02
15/08/2011 0.00.21 239 9,1 9,1 0,3 0,04
15/08/2011 0.00.23 239 10,1 9,0 0,1 0,05
15/08/2011 0.00.27 239 10,8 9,2 0,3 0,01
```

Figure 6.32: Tag dataset structure. Each subset constitutes a folder representing a tag which contains a number of text files equal to the number of days of registration of the tag in question. In the figure is also reported an example of the contents of one of these text files corresponding to the day August 15, 2011.

6.4.1.2 The dataset of the images

The tool suitable for recording the images, which are captured by the cameras, saved the frames in "jpg" format in separate folders (one for each day) by assigning them a unique name corresponding to a number that will be the identifier.

Therefore, to organize these folders a software tool, written in C++ language, was developed which, using the recursive algorithm "quicksort" (Demetrescu et al., 2004), performed first a sorting (by date of acquisition) of the images names, and then returned in output a text file[3] for each day of acquisition where the identifier in each row of the current image is followed by the acquisition date in the following format:

```
ID IMAGE_YYYY_MM_DD_H_M_SS
```

Below is the code used in the implementation of the algorithm quicksort contained in the specifically developed tool:

```
void quicksort(array< fileStructType >^ a, int
    left , int right)
{
    if (left < right)
    {   N++;
```

[3] called 'FileIndex_[yyyy]_[mm]_[dd].txt'

```
                int pivot = partition(a, left , right);

                quicksort(a, left , pivot  1);
                quicksort(a, pivot+1, right);
            }
        }
        int partition(array< fileStructType >^ a, int left ,
           int right)
        {
            __int64 pivot = a[left].date;
            while (true)
            {
                while (a[left].date < pivot) left++;
                while (a[right].date > pivot) right   ;
                if (left < right)
                {
                    String ^tempName;
                    __int64 tempDate;
                    tempName = a[left].FileName;
                    tempDate = a[left].date;
                    a[left].FileName = a[right].FileName;
                    a[left].date = a[right].date;
                    a[right].FileName = tempName;
                    a[right].date = tempDate;
                }
                else
                {
                    return right;
                }
            }
        }
```

Importing data on database

Once the folder containing all the index files generated in the previous step was obtained, a database was created by using the software Microsoft R Office Access, which was populated by inserting all the records belonging to those files.

Therefore, delete queries for duplicate data and sorting filters by date were carried out.

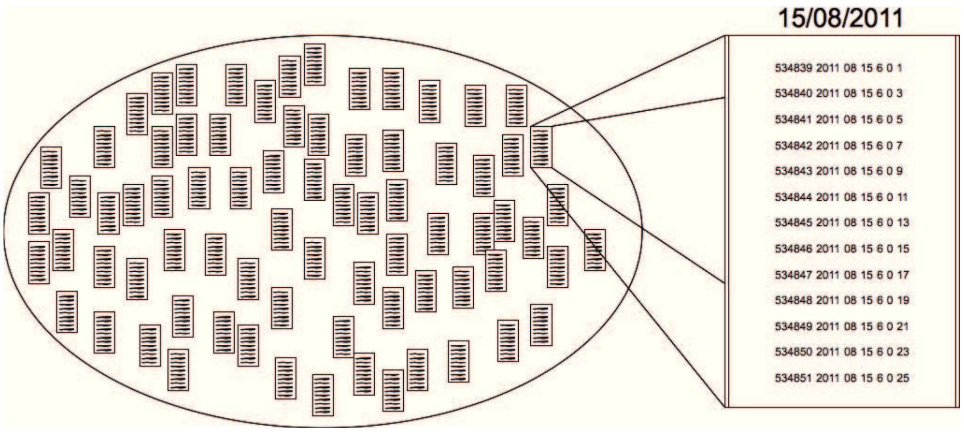
Exporting data from database

Database was exported to a number of text files equal to the number of days of image recordings by using the tool already described in section 6.4.1.1, suitably adapted to the dataset of images. This application performed an initial connection to the database and then a query for the selection of records, which were gradually written to a text file.

The structure of the records stored in these files is the following:

```
ID_IMAGE YYYY MM DD H M SS
```

In Figure 6.33, is shown the structure of the dataset of images obtained after the execution of this software.



```
534839 2011 08 15 6 0 1
534840 2011 08 15 6 0 3
534841 2011 08 15 6 0 5
534842 2011 08 15 6 0 7
534843 2011 08 15 6 0 9
534844 2011 08 15 6 0 11
534845 2011 08 15 6 0 13
534846 2011 08 15 6 0 15
534847 2011 08 15 6 0 17
534848 2011 08 15 6 0 19
534849 2011 08 15 6 0 21
534850 2011 08 15 6 0 23
534851 2011 08 15 6 0 25
```

Figure 6.33: Images dataset structure. The number of text files is equal to the number of the days when image recordings were carried out. In the figure is also reported an example of the contents of one of these text files corresponding to the day August 15, 2011.

6.4.1.3 Matching between tag dataset and image dataset

In this phase the matching between the two datasets built in the previous sections was carried out, and in particular, between the dates of acquisition of the images and those of the tags in order to intersect the data acquired by both systems and verify the planimetric position of each tag, provided by the RTLS. Therefore, to each record belonging to the dataset of images a record belonging to the dataset of tags was coupled based on the following criteria:

> Each panoramic plan-view image ($panoramic_j$) characterised by an acquisition time t_j was associated to tag positions which were recorded at t_j.
>
> When it was not possible to associate to a $panoramic_j$ the position of all the tags present in the scene because of differences in acquisition time, the excluded tags were even then associated with $panoramic_j$ by using for them the positions acquired at the time t_j closest to t_j within the interval $[t_j \quad 3s \ t_j + 3s]$. This time interval was selected according to the acquisition time rate of the RTLS.

The tag positions acquired at the time t_j not included in the interval $[t_j \quad 3s\ t_j + 3s]$ were discarded.

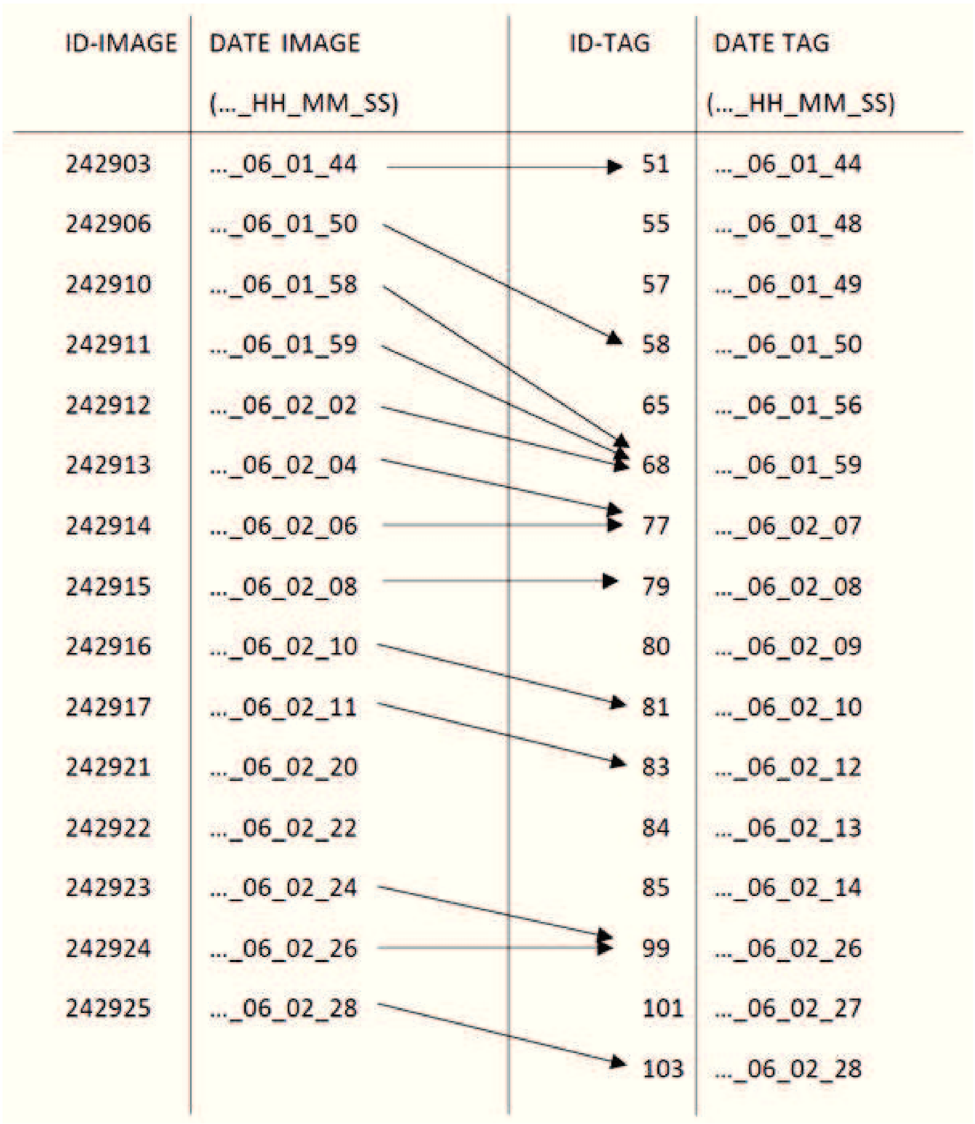
The problem of matching these datasets was addressed by developing a special software in the C++ programming language. This application, by using the iterative binary search algorithm (Demetrescu et al., 2004), computed the matching between the records of the two datasets to intersect, distinguishing the three cases described above. The input of this application were two folders: one containing the text file generated during the 'Reorganisation of the dataset' phase, described in 6.4.1.1 section, and one that contains the text file generated in 'Exporting data from database' phase, described in 6.4.1.2 section. This software tool returned in output a folder for each tag containing as many text file[4] as were the days in which the matching of records belonging to the dataset of tags and records belonging to the dataset of images occurred.

Below is the code of the implementation of the iterative binary search algorithm used in the specifically developed tool:

```
public: bool binarySearch (array< FileStructType >^
    v, int dimvect, __int64 sought){
    int i, m;
    found = false;
    i = 0;
    j = dimvect  1;
    do {
        m = (i+j)/2;
        if(sought == v[m].date){
            found = true;
            index = m;
        }
        else if(sought == v[i].date){
            found = true;
            index = i;
        }
        else if(sought == v[j].date){
            found = true;
            index = j;
        }
        else if (sought < v[m].date) j = m  1;
        else i = m+1;
    }
    while((i <= j) && (found == false));
    return found;
}
```

[4] called 'Matching_[tagName]_[yyyy]_[mm]_[dd].txt'

In Figure 6.34 is shown an example of computation of the matching between the date of the records related to the dataset of tags and the date of the records related to the dataset of images. The arrow shows the matching occurred between records for which one of the first two criteria described above was true.



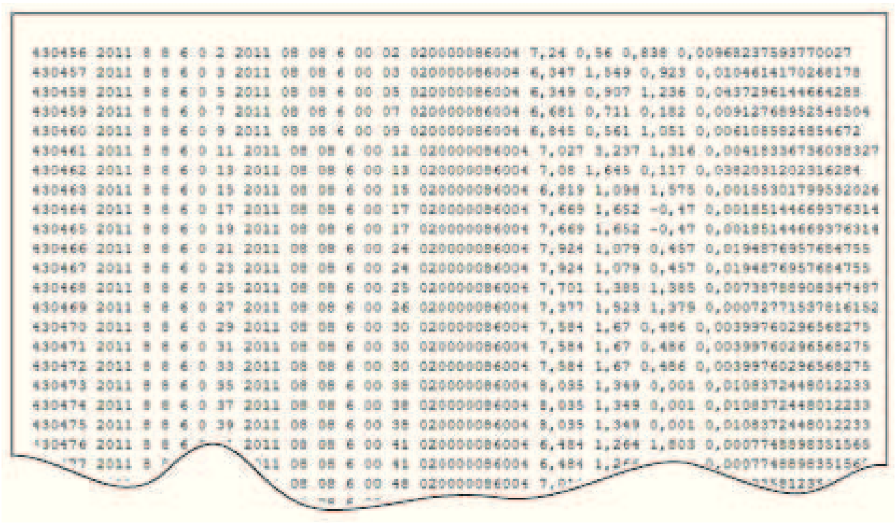
ID-IMAGE	DATE IMAGE (..._HH_MM_SS)	ID-TAG	DATE TAG (..._HH_MM_SS)
242903	..._06_01_44	51	..._06_01_44
242906	..._06_01_50	55	..._06_01_48
242910	..._06_01_58	57	..._06_01_49
242911	..._06_01_59	58	..._06_01_50
242912	..._06_02_02	65	..._06_01_56
242913	..._06_02_04	68	..._06_01_59
242914	..._06_02_06	77	..._06_02_07
242915	..._06_02_08	79	..._06_02_08
242916	..._06_02_10	80	..._06_02_09
242917	..._06_02_11	81	..._06_02_10
242921	..._06_02_20	83	..._06_02_12
242922	..._06_02_22	84	..._06_02_13
242923	..._06_02_24	85	..._06_02_14
242924	..._06_02_26	99	..._06_02_26
242925	..._06_02_28	101	..._06_02_27
		103	..._06_02_28

Figure 6.34: Example of computation of the matching between the date of the records related to the dataset of tags and the date of the records related to the dataset of images. The arrow shows the matching occurred between records for which one of the first two criteria described above was true.

The structure of the records obtained after the matching operation is the following:

```
ID_IMAGE YYYY_IMAGE M_IMAGE D_IMAGE H_IMAGE M_IMAGE
    S_IMAGE YYYY_TAG MM_TAG DD_TAG H_TAG MM_TAG
    SS_TAG ID_TAG X Y Z STD_ERROR
```

The text files returned in output to the end of this phase presented the format shown in Figure 6.35.



```
430456 2011 8 8 6 0 2 2011 08 08 6 00 02 020000086004 7,24 0,56 0,838 0,00968237593770027
430457 2011 8 8 6 0 3 2011 08 08 6 00 03 020000086004 6,347 1,549 0,923 0,0104614170246178
430458 2011 8 8 6 0 5 2011 08 08 6 00 05 020000086004 6,349 0,907 1,236 0,0437296144664288
430459 2011 8 8 6 0 7 2011 08 08 6 00 07 020000086004 6,681 0,711 0,182 0,00912768952548504
430460 2011 8 8 6 0 9 2011 08 08 6 00 09 020000086004 6,845 0,561 1,051 0,0061085924854672
430461 2011 8 8 6 0 11 2011 08 08 6 00 12 020000086004 7,027 3,237 1,316 0,00418336736038327
430462 2011 8 8 6 0 13 2011 08 08 6 00 13 020000086004 7,08 1,645 0,117 0,0382031202316284
430463 2011 8 8 6 0 15 2011 08 08 6 00 15 020000086004 6,819 1,098 1,575 0,00155301799532026
430464 2011 8 8 6 0 17 2011 08 08 6 00 17 020000086004 7,669 1,652 -0,47 0,00185144669376314
430465 2011 8 8 6 0 19 2011 08 08 6 00 17 020000086004 7,669 1,652 -0,47 0,00185144669376314
430466 2011 8 8 6 0 21 2011 08 08 6 00 24 020000086004 7,924 1,079 0,457 0,0194876957684755
430467 2011 8 8 6 0 23 2011 08 08 6 00 24 020000086004 7,924 1,079 0,457 0,0194876957684755
430468 2011 8 8 6 0 25 2011 08 08 6 00 25 020000086004 7,701 1,385 1,385 0,00738788908347497
430469 2011 8 8 6 0 27 2011 08 08 6 00 26 020000086004 7,377 1,523 1,379 0,000727715378616152
430470 2011 8 8 6 0 29 2011 08 08 6 00 30 020000086004 7,584 1,67 0,486 0,00399760296568275
430471 2011 8 8 6 0 31 2011 08 08 6 00 30 020000086004 7,584 1,67 0,486 0,00399760296568275
430472 2011 8 8 6 0 33 2011 08 08 6 00 30 020000086004 7,584 1,67 0,486 0,00399760296568275
430473 2011 8 8 6 0 35 2011 08 08 6 00 38 020000086004 8,035 1,349 0,001 0,0108372448012233
430474 2011 8 8 6 0 37 2011 08 08 6 00 38 020000086004 8,035 1,349 0,001 0,0108372448012233
430475 2011 8 8 6 0 39 2011 08 08 6 00 38 020000086004 8,035 1,349 0,001 0,0108372448012233
430476 2011 8 8 6      2011 08 08 6 00 41 020000086004 6,484 1,264 1,803 0,0007748898351565
   77 2011 8         11 08 08 6 00 41 020000086004 6,484 1,26        0,000774889835156
                        08 08 6 00 48 020000086004 7,0                 3581235
```

Figure 6.35: Output format after the matching operation.

6.4.2 The dataset of the true positions of UWB tags

A specific software, which was developed by using Microsoft® Visual C# Express (framework .NET), allowed visualisation of tags within each panoramic image by using graphic elements (points). For each tag, the coordinates of the related graphic element in a $panoramic_j$ were expressed in number of pixels and obtained through a linear conversion of the real tag coordinates $P_i(x, y)$ (for $i = 1, \ldots, n$ where n is the number of measurements obtained by the RTLS), expressed in meters. The visual recognition of the position of each tag was carried out by an operator via the software. An identikit of each dairy cow of the herd was made available to the operator in a section of the software interface in order to facilitate this activity. Once a cow had been identified, the operator checked that the graphic element associated to the tag was displayed in the right position. When the operator did not find this condition, he moved the graphic element in the correct position and the software recorded a new position of the tag $P_i^T(x \ y)$. After a linear conversion from pixels to meters, these new positions of each tag for each measurement constituted the dataset of the true positions of the tags during the monitoring period.

6.4.3 Computation of the planimetric localisation errors of the tags

For each tag, a planimetric localisation error $_i$ was obtained by computing the Euclidean distance between the position provided by the RTLS and that verified by the operator:

$$_i = \ P_i \quad P_i^T \ = \ \overline{(x_i \quad x_i^T)^2 + (y_i \quad y_i^T)^2} \quad (i = 1 \qquad n) \tag{6.1}$$

These errors were utilised to compute localisation error at 90th percentile, mean localisation error, and related standard deviation.

Identification and filtering of anomalous measurements, which were highly different from the central data distribution values, were carried out by adopting an outlier data cleaning technique (Peck et al., 2011). The measurements higher than $q_3 + w \quad (q_3 \quad q_1)$ and lower than $q_1 \quad w \quad (q_3 \quad q_1)$, where q_1 and q_3 are the 25th and the 75th percentiles, respectively, and $w = 1\ 5$, were discarded.

6.4.4 Assessment of the localisation and identi cation performances of the RTLS

Localisation performance of the RTLS was evaluated by using mean localisation error and standard deviation, whereas RTLS identification performance was assessed by computing precision and sensitivity. The precision of a measurement system, also called reproducibility or repeatability, is the degree to which repeated measurements under unchanged conditions show the same results. It was calculated through the relation:

$$precision = \frac{number \quad of \quad TP}{number \quad of \quad TP \quad + \quad number \quad of \quad FP} \tag{6.2}$$

where, $TP = True \quad Positive$ and $FP = False \quad Positive.$

The sensitivity, instead, also called recall or true positive rate, measures the proportion of actual positives which are correctly identified as such. It was calculated through the following relation:

$$sensitivity = \frac{number \quad of \quad TP}{number \quad of \quad TP \quad + \quad number \quad of \quad FN} \tag{6.3}$$

where, $FN = False \quad Negative.$

For each tag, the number of true positives was obtained by counting the number of times that the cow was successfully detected in its position by the RTLS, the number of false positives was obtained by counting the number of times that the cow was wrongly detected in its position by the RTLS, and the number of false negatives was obtained by counting the number of times that the cow was not detected by the RTLS though present in the framed scene.

Finally, three performance metrics (Metric A, Metric B, and Metric C) were defined to establish the trade-off between the localisation and identification performances of the RTLS.

Metric A: for each tag, all the were used to compute the mean localisation error and the related standard deviation. In this metric, a true positive was assigned when the RTLS detected the tag in the panoramic top-view image. Therefore, the number of true positives was obtained by counting the number of times that the tag was present in the framed scene. The number of false positives and the number of false negatives were assumed to be equal to 0.

Metric B: for each tag, all the obtained after the outlier data cleaning process were used to calculate the mean localisation error and the related standard deviation. In this metric, the number of true positives was obtained by counting the number of times that the tag was present in the framed scene and the related were not filtered out by the data cleaning process. The number of false positives and the number of false negatives were assumed to be equal to the number of measurements considered as outliers.

Metric C: a threshold value of 0.50 m was established for localisation error. All $_i$ less than or equal to the threshold were used to calculate the mean localisation error and the related standard deviation. In this metric, the number of true positives was obtained by counting the number of times that the RTLS detected the tag with a localisation error lower than the threshold. The number of false positives and the number of false negatives were obtained by counting the number of times that the RTLS detected the tag with a localisation error greater than the threshold. The choice of the value of 0.50 m for the threshold is founded on the following consideration. Since the distance between the extremities of cow's ears is equal to about 0.60 m on average and since the tags were applied some to the left ear and others to the right ear, the center point of the cow head was chosen as the origin and 0.30 m from the origin as threshold in each direction (x and y), which corresponds to about 0.42 m in two directions. Finally, it was decided to further increase the margin of error by choosing a threshold value equal to 0.50 m. In Figure 6.36 the area of the green square with sides of 0.60 m, represents the considered surface for the computation of the threshold adopted in the Metric C.

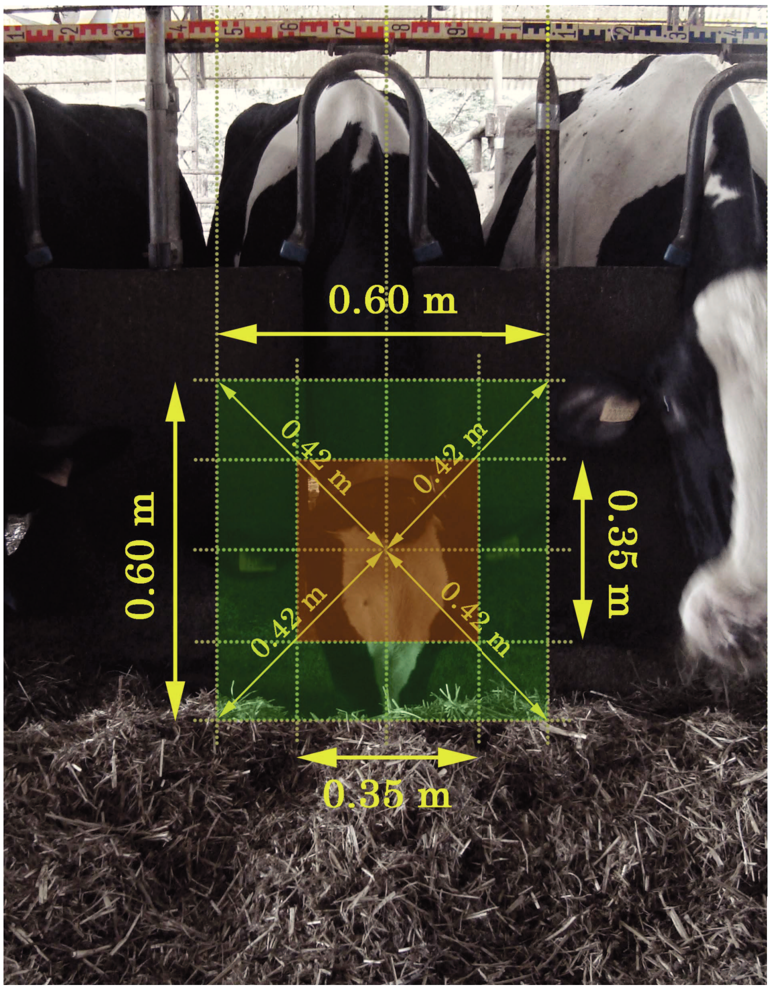


Figure 6.36: Computation of the threshold established for localisation error in Metric C.

Furthermore, another area was determined (red square in Figure 6.36) in order to perform the same analysis of the performances in the case when the technique of application of the tag using the collars (subject of a study in progress) was adopted. In this case the average width of the head (about 0.35 m) was considered as the starting point for the computation of the relative threshold.

6.5 AN AUTOMATIC AND REAL-TIME SOFTWARE TOOL FOR THE VISUAL ANALYSIS OF THE COWS LOCATION DATA IN FREE-STALL BARNS

A specific software, which was developed by using Microsoft R Visual C# Express (framework .NET), allowed visualisation of cow's location and speed data.

In particular, with regard to the cow's location data, the creation of a '*CowHeatMap*' was implemented to allow the visualization of the spatial distribution in two dimensions of the data acquired from the RTLS within the study area. Two dimensions represent cartesian coordinates (x and y values) and the third dimension is used for showing the intensity of a datapoint in relative comparison to the absolute maximum of the dataset. The different color and color intensity denote the difference in sample density at a location. Usually red (hot) is used for the maxima and blue (cold) for minima (Wied, 2011-2013).

As regards, instead, the cow's speed data, the building of a '*CowSpeedGraph*' was implemented to show trend, over time, of the cow's instantaneous speeds, sampled at regular time intervals, relatively to data acquired by the RTLS. In fact, Bell et al. (2013) established that deterioration of walking speed is one of the characteristic symptoms of lameness. Furthermore, measurement of the speed of each cow showed to be well-correlated with the cow's mobility score (Bell et al., 2013). In addition, Martinez-Ortiz et al. (2013) calculated the speeds of approximately 190 dairy cows inspected by a video tracking system over a number of weeks. The relative speed of a cow with respect to the group on its own is not sufficient to detect lameness; a cow may be consistently slower than the group due to old age or simply due to its own preferred pace of walking. Therefore, they proposed to detect lameness by monitoring each individual cow's speed over a time interval to look for consistent changes in mean speed. On the basis of this considerations, the

CowSpeedGraph function makes it possible to monitor each individual cow's speed over a time interval selected by the user.

This software, described in detail in the next sections (sections 6.5.1, sections 6.5.2 and sections 6.5.3), was designed to have the necessary features to be integrated into the world of IoT, giving to the user the ability to work in real time by monitoring the data, acquired by the RTLS, updated at short intervals of time. In fact, the software can be launched during the execution of the location platform and the tool for recording data in the database and automatically takes records of the constantly updated database as input. By adding new control modules, this feature has the ability to raise alert messages from changes in dairy cow behaviour and then to alert immediately the farmer when some health problems (e.g., lameness) or a particular physiological status (e.g., estrus) occur.

In Figure 6.37 the flow chart of the algorithms implemented in this software is shown.

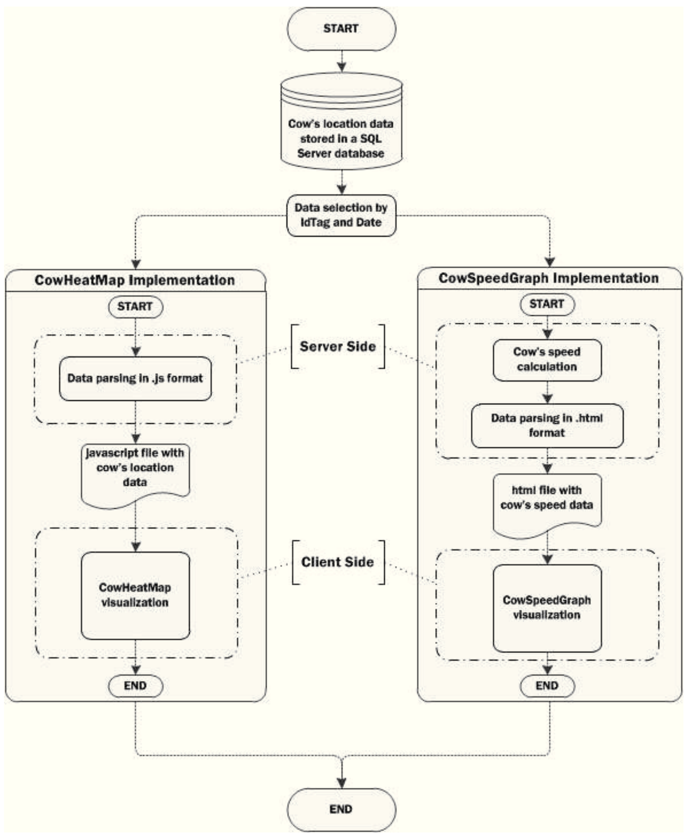

START
Cow's location data stored in a SQL Server database
Data selection by IdTag and Date
CowHeatMap Implementation
START
Data parsing in .js format
javascript file with cow's location data
CowHeatMap visualization
END
Server Side
Client Side
CowSpeedGraph Implementation
START
Cow's speed calculation
Data parsing in .html format
html file with cow's speed data
CowSpeedGraph visualization
END
END

Figure 6.37: Flow chart of the algorithms implemented in the software tool for the visual analysis of the cows' location data.

6.5.1 Data selection

As described in section 6.2.5.1, during the acquisition phase of the data provided by the RTLS, a database was populated. The software described in this section was designed in order to perform access to that database and extract the data required for the analysis.

The user has the possibility to filter data, by setting certain parameters, in order to perform elaborations on them. These elaborations will be provided by the subsequent operations automated by the software. In other words, it is possible to choose on which cows you want to perform the visual analysis, indicating the ID of the tag associated to the cow, and which period of acquisition. In particular, let 'h' be the current time, the software requires a number of 'x' hours in order to select the data belonging to the interval [h - x, h].

Therefore, the informations acquired by the RTLS software in the last 'x' hours related to a particular cow will be provided.

Below is the implementation of the operations described above.

```
        private void CreateTableInDBFromAllTagsByIdTag(string idTag, string connectionString)
                {

                    string ctStr = "SELECT * INTO [dbo].[" + idTag + "] FROM [UwbTagDB].[dbo].[data] WHERE idTag='" + idTag + "'";

                    SqlConnection conn = new SqlConnection(connectionString);
                    SqlCommand command = conn.CreateCommand();
                    command.CommandText = ctStr;
                    conn.Open();
                    command.ExecuteNonQuery();
                conn.Close();

            }
```

The 'CreateTableInDBFromAllTagsByIdTag' method takes as input a string containing the ID of the tag, from which you want to select the information and performs a SQL SELECT query. This query selects the rows in the 'date' table containing all the data acquired by the RTLS and inserts them into a table called 'idTag' (if doesn't exist is it created) filtering for the ID value of the tag passed as input.

```
                private void CreateTableFromSelectedTagByTimePeriod(string idTag, double hours, string
                    connectionString)
                {
                    string startTime =
                        DateTime.Now.AddHours( hours).ToString();
                    string ctStr = "SELECT * INTO [dbo].[Last" +
                        hours + "hours] FROM [UwbTagDB].[dbo].["
                        + idTag + "] WHERE Date BETWEEN '" +
                        startTime + "' AND '" + DateTime.Now +
                        "'";

                    SqlConnection conn = new
                        SqlConnection(connectionString);
                    SqlCommand command = conn.CreateCommand();
                    command.CommandText = ctStr;
                    conn.Open();
                    command.ExecuteNonQuery();
                    conn.Close();

                }
```

The 'CreateTableFromSelectedTagByTimePeriod' method takes as input a string containing the ID of the tag and a numerical value corresponding to the number of hours to be subtracted from the current time and performs a SQL SELECT query. Through this query, the rows in the 'idTag' table which have, in the 'Dates' field, a date that belongs to the interval $[h - x, h]$, where x is the parameter corresponding to the hours passed in the input and h is the current time, are selected.

6.5.2 CowHeatMap implementation

As it is shown in the flow chart of Figure 6.37, the implementation of the CowHeatMap was managed through two separate modules: the server side and the client side.

Sections 6.5.2.1 and 6.5.2.2 describe the operations performed by the individual modules.

6.5.2.1 Server side

The server side software module was designed with the purpose of providing services to the client side module, implemented in order to satisfy the requests received. In particular, after receiving a request from the client side module, which ask to view the HeatMap relative to the data selected in the previous section (6.5.1), a conversion from meters to pixels of the coordinates 'x' and 'y' is performed on the data extracted from the database with the following format: 'acquisition date acquisition time; tag ID; x; y; z; standard error'. This operation aims at transforming the data in the correct format in order to perform a mapping of these on a panoramic image of the barn. According to the pixel dimensions of the image, the relations 6.8 and 6.7 are defined to get $xtagPx$ and $ytagPx$, i.e., the 'x' coordinate value and the 'y' coordinated value in pixels within the image, respectively.

$$xtagPx = 47 + xtag \quad k \tag{6.4}$$

$$ytagPx = 613 \quad ytag \quad k \tag{6.5}$$

where

$xtag$ and $ytag$ are 'x' and 'y' coordinates in meters, respectively.

$k = 35\ 67$

This operation is iterated for each record automatically extracted.

Once the sequence of the couple ($xtagPx$, $ytagPy$) is obtained, the software executes a specially created script to get an output file, with extension js, containing the structure of the dataset to be provided as input to the client in order to create the HeatMap. Below is the source code of the method that implements these steps:

```
private void WriteJsWithLocationData(string idCow,
    LocationEvent[] locEvent)
{

            string fileName = "C:\\datasetHeatMap_" + idCow +
                ".js";
            StringBuilder content = new StringBuilder();
            content.AppendLine("(function(global){");
            content.AppendLine("    var events = " +
                locEvent.Length + ";");
            content.AppendLine("    var locations = new
                Array(events);");
            content.AppendLine("    for (i=0; i<events; i++){");
            content.AppendLine("        locations[i]=new
                Array(2);");
            content.AppendLine("    }");
            content.AppendLine("    locations = [");
            double x;
            double y;
            for (int i = 0; i < locEvent.Length   1; i++)
            {
                x = 47 + (locEvent[i].X   35.67);
```

```
                    y = 613   (locEvent[i].Y   35.67);
                    content.AppendLine("            [\"" +
                        x.ToString().Replace(",", ".") + "\",\"" +
                        y.ToString().Replace(",", ".") + "\"], //" +
                        locEvent[i].Date.ToString());
                }
                x = 47 + (locEvent[locEvent.Length   1].X   35.67);
                y = 613   (locEvent[locEvent.Length   1].Y   35.67);
                content.AppendLine("            [\"" +
                    x.ToString().Replace(",", ".") + "\",\"" +
                    y.ToString().Replace(",", ".") + "\"] //" +
                    locEvent[locEvent.Length   1].Date.ToString());
                content.AppendLine("    ];");
                content.AppendLine("    global.locations =
                    locations;");
                content.AppendLine("    global.events = events;");
                content.AppendLine("}(window));");

                File.WriteAllText(fileName, content.ToString());
}
```

Therefore, the server-side module after processeing the request, sends the javascript file to the client side module.

6.5.2.2 Client side

The client side software module was designed with the purpose of sending requests to the server side module and to interpret the answers received.

As already introduced in section 6.5.2.1, client receives from the server a file with extension *.js* containing the dataset to be processed for the construction of HeatMap.

This file is in the following format:

```
(function(global){

// cardinality of the dataset (in this example is 735)
var events = 735;
var locations = new Array(events);
for (i=0; i<events; i++){
    locations[i]=new Array(2);
}

locations = [
            // Dataset ["x", "y"]
            ["382.47635","468.78619"],
            ["383.54645","470.03464"],
            ["383.86748","464.68414"],

                        .
                        .
                        .

              ["384.15284","474.45772"],
              ["366.88856","458.22787"],
              ["377.05451","481.62739"],
              ["379.58708","475.70617"],
          ];

global.locations = locations;
global.events = events;

}(window));
```

Then, the software performs the construction of the HeatMap by reading the contents of the file containing the dataset ('dataset.js') and by using an open source JavaScript library called 'heatmap.js' (Wied, 2011-2013). It uses the power and flexibility of the HTML5 canvas element to draw heatmaps based on the dataset.

This operation was implemented through the following steps:

```
<script type="text/javascript"
    src="../../../src/heatmap.js"></script>
<script type="text/javascript" src="dataset/dataset.js"></script>
<script type="text/javascript">


window.onload = function(){
    var xx = h337.create(
                {
                "element":document.getElementById("heatmapArea"),
                "radius":6, "visible":true,
                "opacity":50,
                "legend":
                    {
                    "position":'br',
                    "title":'Number of Events Distribution'
                    }
                });

    var locations = window.locations;
    var pos;
    for (i=0; i<window.events; i++) {
            pos = locations[i];
            xx.store.addDataPoint(pos[0],pos[1]);
    }

};
```

The 'heatmap.js' library creates a global object called 'h337'. This global object has a function 'create' that takes one argument 'config' and returns a heatmap instance. In the Javascript programming language, each primitive is represented by an object. An object is just a special kind of data, with properties and methods. Properties are the values associated with an object whereas methods are the actions that can be performed on objects (Crockford, 2008). In this case, the object 'h337' executes the action to create a heatmap instance invoking the method 'create' and stores the return value in the variable 'xx'.

6.5.3 CowSpeedGraph implementation

As it is shown in the flow chart of Figure 6.37, the implementation of CowSpeedGraph, such as the CowHeatMap, was managed through two separate modules: the server side and client side.

Sections 6.5.3.1 and 6.5.3.2 describe the operations performed by the individual modules.

6.5.3.1 Server side

The server-side module is designed to provide services to the client side module, which forwards to it the requests dictated by the end user.

Such requests, in particular, concern the viewing of the cowSpeedGraph, in relation to the parameters set by the user.

In order to calculate the instantaneous speed relative to the data selected in section 6.5.1, the following data structures are implemented through the classes 'LocationEvent', 'ScalarVelocityEvent' and 'ScalarVelocityNDEvent'.

In particular, the 'LocationEvent' class was implemented to contain the date, tag ID and the values for the three position coordinates x, y and z for each record extracted from the database. The following source code is related to the class just described:

```
public class LocationEvent
{
        #region properties
        public DateTime Date;
        public string IdTag;
        public double X;
        public double Y;
        public double Z;
        #endregion

        #region constructors
        internal LocationEvent(
            DateTime date,
            string idTag,
            double x,
            double y,
            double z)
        {
            this.Date = date;
            this.IdTag = idTag;
            this.X = x;
            this.Y = y;
            this.Z = z;
        }
         #endregion
}
```

The 'ScalarVelocityEvent' class, however, was implemented to contain the date, tag ID and distinct instantaneous velocities for each coordinate (x, y and z). The following source code is related to the class just described:

```
public class ScalarVelocityEvent
{
        #region properties
        public DateTime Date;
        public string IdTag;
        public double? X;
        public double? Y;
        public double? Z;
        #endregion

        #region constructors
        internal ScalarVelocityEvent(
            DateTime date,
            string idTag,
            double? x,
            double? y,
            double? z)
        {
            this.Date = date;
            this.IdTag = idTag;
            this.X = x;
            this.Y = y;
            this.Z = z;
        }
         #endregion
}
```

The 'ScalarVelocityNDEvent' class, finally, was implemented to contain the date, the ID of the tag and the instantaneous velocity with respect to that instant of sampling. The following source code refers to the class just described:

```
public class ScalarVelocityNDEvent
{
        #region properties
        public DateTime Date;
        public string IdTag;
        public double InstantSpeed;
        #endregion

        #region constructors
        internal ScalarVelocityNDEvent(
            DateTime date,
            string idTag,
            double instantSpeed)
        {
            this.Date = date;
            this.IdTag = idTag;
            this.InstantSpeed = instantSpeed;
        }
         #endregion
}
```

At this point of the process, a sampling of the selected data was performed. In particular, a sampling time of 5 seconds was chosen, in order to calculate the average positions for each interval in the three coordinates x, y and z.

Once the sampling was performed, instantaneous speeds in two dimensions (x, y) for each sampling instant i were calculated according to the following relation:

$$v_i = \frac{ds}{dt} = \frac{d}{dt}\left(\sqrt{dx^2 + dy^2}\right) = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} \tag{6.6}$$

The 'instantSpeed2D' object was allocated in order to contain the instantaneous speeds:

```
List<ScalarVelocityNDEvent> instantSpeed2D = new
    List<ScalarVelocityNDEvent>();
```

Equation 6.6 was implemented in the following code:

```
            ScalarVelocityNDEvent instantSpeedND2 = new
                ScalarVelocityNDEvent(meanData[i].MiddleDate,
                meanData[i].IdTag,
                Math.Sqrt(Math.Pow(((meanData[i].X   meanData[i
                1].X) / window), 2) + Math.Pow(((meanData[i].Y
                meanData[i   1].Y) / window), 2)));
```

where *window* is a variable containing the number of seconds of the sampling instant set. In our case, *window* = 5.

After the instantaneous speeds are calculated, an *XMLParser* was implemented in order to get a file with extension '*Xml*' containing the data to be plotted which are related to the computed velocities.

Here is the code of the method that implements this function:

```

private void WriteXmlWithVelocityData()
{
        XmlTextWriter writer = new
            XmlTextWriter("C:\\data.xml", null);
        writer.Formatting = Formatting.Indented;
        writer.WriteStartDocument();

        // open plot tag with id=2
        writer.WriteStartElement("plot");
        writer.WriteAttributeString("id", "2");

        // open curves tag
        writer.WriteStartElement("curves");

        // open curve tag with id=1
        writer.WriteStartElement("curve");
        writer.WriteAttributeString("id", "1");

        // open points tag
        writer.WriteStartElement("points");

        string date;
        for (int i = 0; i < instantSpeed2D.Count; i++)
        {
            // open point tag
            writer.WriteStartElement("point");
            date =
                instantSpeed2D[i].Date.ToString().Replace(".",":");
            writer.WriteAttributeString("x", date);
            writer.WriteAttributeString("y",
                instantSpeed2D[i].InstantSpeed.ToString());
            writer.WriteAttributeString("ymin", "0");
            writer.WriteAttributeString("ymax",
                instantSpeed2D[i].InstantSpeed.ToString());
            // close point tag
            writer.WriteEndElement();
        }

        // close points tag
        writer.WriteEndElement();

        // open and close label tag
        writer.WriteElementString("label", "SPEED");

        // close curve tag
        writer.WriteEndElement();

        // close curves tag
        writer.WriteEndElement();

        // open and close description tag
        writer.WriteElementString("description",
           "&lt;![CDATA[Cow Speed Graph.]]&lt;");

        // close plot tag
        writer.WriteEndElement();

        writer.WriteEndDocument();
        writer.Close();

}
```

The file obtained with the instructions just shown is in following format:

```
<?xml version="1.0"?>
<plot id="2">
  <curves>
    <curve id="1">
      <points>
        <point x="17/04/2013 6:16:03" y="0" ymin="0" ymax="0" />
        <point x="17/04/2013 6:16:08" y="0,0123" ymin="0"
           ymax="0,0123" />
        <point x="17/04/2013 6:16:13" y="0,0083" ymin="0"
           ymax="0,0083" />
        <point x="17/04/2013 6:16:18" y="0,0297" ymin="0"
           ymax="0,0297" />

                                .
                                .
                                .

        <point x="17/04/2013 9:15:43" y="0,0119" ymin="0"
           ymax="0,0119" />
        <point x="17/04/2013 9:15:48" y="0,043" ymin="0"
           ymax="0,043" />
        <point x="17/04/2013 9:15:53" y="0,0441" ymin="0"
           ymax="0,0441" />
      </points>
```

```
        <label>SPEED</label>
      </curve>
    </curves>
    <description><![CDATA[Cow Speed Graph.]]></description>
</plot>
```

A plot is defined by an identifier ('*id* = 2' in this example), zero or multiple curves ('*id* = 1' in this example) and a '*points*' list of points made up of objects of '*point*' type. Each of these objects is composed by the couple (x, y) to be plotted, where x corresponds to the sampling instant and y corresponding to the value of the instantaneous velocity. There are also the *ymin* and *ymax* values used to indicate the minimum point and the maximum point, respectively, of y axis in order to highlight a specific part of the graph. For instance, by always setting $ymin = 0$ and $ymax = y$ the area under the curve plotted will be highlighted in the graph. In addition, there is a '*label*' and a '*description*'.

At this point of the process, data were ready to be plotted, and then were displayed for the client in the form of speed graph. To this end, two data structures are implemented through the 'Point' and 'Curves' classes. In particular, the 'Point' class, which has the following source code, was implemented to contain the coordinates x and y to be plotted, where x is the value of the instantaneous velocity and y is relative to the corresponding sampling instant.

```
public class Point
{
        #region Constructors and Destructors

        public Point(DateTime x, float y)
        //public Point(string x, float y)
        {
            X = x;
            Y = y;
        }

        public Point(DateTime x, float y, float ymin, float
            ymax)
        //public Point(string x, float y, float ymin, float
            ymax)
            : this(x, y)
        {
            YMin = ymin;
            YMax = ymax;
        }

        #endregion

        #region Public Properties

        public DateTime X { get; private set; }
        //public string X { get; private set; }
        public float Y { get; private set; }
        public float YMin { get; private set; }
        public float YMax { get; private set; }

        #endregion
}
```

The class 'Curve', which has the source code shown below, was implemented in order to contain the labels of the axes in the graph and return the list of points to be plotted.

```
public class Curve
{
        #region Constructors and Destructors

        public Curve(string label, IList<Point> points)
        {
            Label = label;
            Points = points;
        }

        #endregion

        #region Public Properties

        public string Label { get; private set; }
        public IList<Point> Points { get; private set; }

        #endregion
}
```

Finally, the 'Plot' class was implemented in order to contain all the data required to build the speed graph, such as, the 'Title', the 'ylabel', the 'Description', the 'Curves', etc.

In addition, the 'Load ()' public method was created to implement the required operations for the preparation of the data to be plotted by using the data structures described above.

The source code for the class Plot and the Load() method is shown below.

```
public class Plot
{
  #region Constructors and Destructors

```

```
public Plot(int id, string title, string ylabel, string
    dataPath, string thumbnailPath, bool isRelativePath)
{
      Id = id;
      Title = title;
      YLabel = ylabel;
      DataPath = dataPath;
      ThumbnailPath = thumbnailPath;
      IsRelativePath = isRelativePath;
      Description = string.Empty;
      Curves = new List<Curve>();
      IsLoaded = false;
}

#endregion

#region Public Properties

      public IList<Curve> Curves { get; private set; }
      public string DataPath { get; private set; }
      public string Description { get; private set; }
      public int Id { get; private set; }
      public bool IsLoaded { get; private set; }
      public bool IsRelativePath { get; private set; }
      public string ThumbnailPath { get; private set; }
      public string Title { get; private set; }
      public string YLabel { get; private set; }

#endregion

#region Public Methods

public void Load()
{
      // load the curves from the XML file
      XDocument data = XDocument.Load(IsRelativePath ?
          HttpContext.Current.Server.MapPath(DataPath) :
          DataPath);

       // Initialize the Curves collection
       Curves = new List<Curve>();

       // Retrieve the plot current plot element from the XML
           file
       XElement xplot = data.Element("plot");

       // Retrieve the current Description
       Description = xplot.Element("description").Value;

```

```
        // Retrieve the current curves
        IEnumerable<XElement> xcurves =
            xplot.Descendants("curve");

        // Enumerate curves
        // Build a the Curve object
        // and Add it to the current Curves collection
        foreach (XElement xcurve in xcurves)
        {
          // Retrive the label
          string label = xcurve.Element("label").Value;

          // Initialize a new Set
          // Retrieve the points from the XML file
          IEnumerable<XElement> xpoints =
              xcurve.Descendants("point");

          // Enumerate the points
          // Parse and add their values to a new Point object
          // Build the Set

          IFormatProvider culture = new CultureInfo("fr-FR",
              true);
          List<Point> set = (from xpoint in xpoints
                             let x = DateTime.
                                 Parse(xpoint.Attribute("x").Value,
                                 culture)
                             let ymin = (float)Math.Round(float.
                                 Parse(xpoint.Attribute("ymin").
                                 Value.Replace(',', '.'),
                                 CultureInfo.CurrentCulture),4)
                             let y = (float)Math.Round(float.
                                 Parse(xpoint.Attribute("y").
                                 Value.Replace(',', '.'),
                                 CultureInfo.CurrentCulture),4)
                             let ymax = (float)Math.Round(float.
                                 Parse(xpoint.Attribute("ymax").
                                 Value.Replace(',', '.'),
                                 CultureInfo.CurrentCulture),4)
                             select new Point(x, y, ymin,
                                 ymax)).ToList();

            // Add the curve to the Curves collection
            Curves.Add(new Curve(label, set));
        }

             // Set data as loaded
             IsLoaded = true;
         }
```

```

#endregion
}
```

The just reported instructions implement data extraction to be plotted from the ' *Xml*' file in order to create a file '*.Html*' to be sent to the client.

Being a client-server system, the choice to use a html file for the transmission of information processed by the server to the client, was based on the fact that the typical features of such a data structure guarantee high performance in terms of transmission speed, as it comes to small files, and in terms of compatibility with the majority of client systems.

The following source code shows the implementation of the operations performed by the software to create the html file:

```
private void WriteHtmlWithVelocityData(string idCow)
{

            string fileName = "C:\\cowSpeedGraph_" + idCow +
               ".html";
            StringBuilder content = new StringBuilder();
            content.AppendLine("<html>");
            content.AppendLine("<head>");
            content.AppendLine("<script type=\"text/javascript\"
               src=\"dygraph-combined.js\"></script>");
            content.AppendLine("</head>");
            content.AppendLine("<body>");
            content.AppendLine("<div id=\"graphdiv\"
               style=\"width:800px; height:500px;\"></div>");
            content.AppendLine("<script
               type=\"text/javascript\">");
            content.AppendLine("    g = new Dygraph(");
            content.AppendLine("
               document.getElementById(\"graphdiv\"),");
            content.AppendLine("    [");
            string date;
            string[] splitDate;
            string[] splitDate1;
            string time;
            string day;
            string month;
            string year;
            for (int i = 0; i < instantSpeed2D.Count  1; i++)
            {
                date = instantSpeed2D[i].Date.ToString().
                   Replace(".",":");
                splitDate = date.Split(' ');
                time = splitDate[1];
                splitDate1 = splitDate[0].Split('/');
```

```
                    day = splitDate1[0];
                    month = splitDate1[1];
                    year = splitDate1[2];
                    date = year + "/" + month + "/" + day + " " +
                        time;
                    content.AppendLine("          [new Date(\"" + date
                        + "\"), " +
                        instantSpeed2D[i].InstantSpeed.ToString().
                        Replace(",",".") + " ],");
                }
                date = instantSpeed2D[instantSpeed2D.Count
                    1].Date.ToString().Replace(".", ":");
                splitDate = date.Split(' ');
                time = splitDate[1];
                splitDate1 = splitDate[0].Split('/');
                day = splitDate1[0];
                month = splitDate1[1];
                year = splitDate1[2];
                date = year + "/" + month + "/" + day + " " + time;
                content.AppendLine("          [new Date(\"" + date +
                    "\"), " + instantSpeed2D[instantSpeed2D.Count
                    1].InstantSpeed.ToString().Replace(",", ".") + "
                    ]");
                content.AppendLine("     ],");
                content.AppendLine("     {");
                content.AppendLine("          labels: [ \"Date\",
                    \"COW SPEED\" ],");
                content.AppendLine("          fillGraph: true");
                content.AppendLine("     });");
                content.AppendLine("</script>");
                content.AppendLine("</body>");
                content.AppendLine("</html>");

                File.WriteAllText(fileName, content.ToString());

}
```

Therefore, the server-side module processed the request from the client side module, which asked to view the speedGraph relative to the data selected in the previous section (6.5.1), and transmitted the file *Html* to the client side module.

6.5.3.2 Client side

The client-side module of the software was designed with the aim to send to the server-side module requests made by the user in the interface section dedicated to cowSpeedGraph and to interpret the answers received.

As already introduced in section 6.5.3.1, the client receives from the server a file with extension *Html* containing the dataset to be processed for the construction of cowSpeedGraph.

This file is in the following format:

```
<html>
<head>
<script type="text/javascript"
  src="dygraph-combined.js"></script>
</head>
<body>
<div id="graphdiv"></div>
<script type="text/javascript">
  g = new Dygraph(

      // containing div
      document.getElementById("graphdiv"),

      [
          [ new Date("2013/04/17 6:16:03"), 0.0123 ],
          [ new Date("2013/04/17 6:16:08"), 0.0083 ],

                              .
                              .
                              .

          [ new Date("2013/04/17 6:16:13"), 0.0297 ],
          [ new Date("2013/04/17 6:16:18"), 0.0521 ]
      ],
      {
           labels: [ "Date", "COW SPEED" ],
           fillGraph: true
      });
</script>
</body>
</html>
```

The instructions just shown implement the functions that can be interpreted by a web browser for the construction of cowSpeedGraph.

In particular the fast, flexible, and open source JavaScript charting library called *dygraph combined.js* (Dygraph, 2013) it is used.

6.5.4 A use case of the software tool

A use case of the software tools described in this section was carried out in order to assess the performances of the implemented features and to verify the effectiveness of the results obtained from the use of the tool in order to find useful informations related to the occurrence of the physiological state of estrus.

The date analysed in this test were acquired by the RTLS from 10th August 2013 to 16th August 2013. In particular, only the location data related to the tag with ID 020 were used as the cow, which that tag was associated, manifested the state of estrus, visually observed by the farmer at around 9:30 am on 12th August 2013.

For each of the selected days, two time intervals were identified:

1) **CFTime (Cow Feeding Time)**, where the CFI (Cow Feeding Index), calculated as the ratio between the number of cows that are to the manger and the total number of cows present in the barn (Equation 6.7), took on average the maximum values, as shown in Figure 6.38 (green area), in the same period considered in this test and in the same livestock environment in a study carried out by Agosta (2012).

$$CFI = \frac{cows \quad in \quad feeding}{tot \quad cows} \tag{6.7}$$

In particular, CFTime is composed of two time intervals: the first between 06:00 and 09:30 and the second between 18:00 and 19:30.

2) **CLTime (Cow Lying Time)**, where the CLI (Cow Lying Index), defined as the ratio between the number of cows in decubitus position inside of the bunks and the total number of cows present in the barn (Equation 6.8), assumed on average maximum values (Figure 6.38, yellow area), also in this case by reference to the work conducted by Agosta (2012) in the same period considered in this test and in the same livestock environment.

$$CLI = \frac{cows \quad in \quad lying}{tot \quad cows} \tag{6.8}$$

In particular, CLTime is composed of two time intervals: the first included between 00:00 and 05:00 (not visible in Figure 6.38) and the second between 11:00 and 14:30.

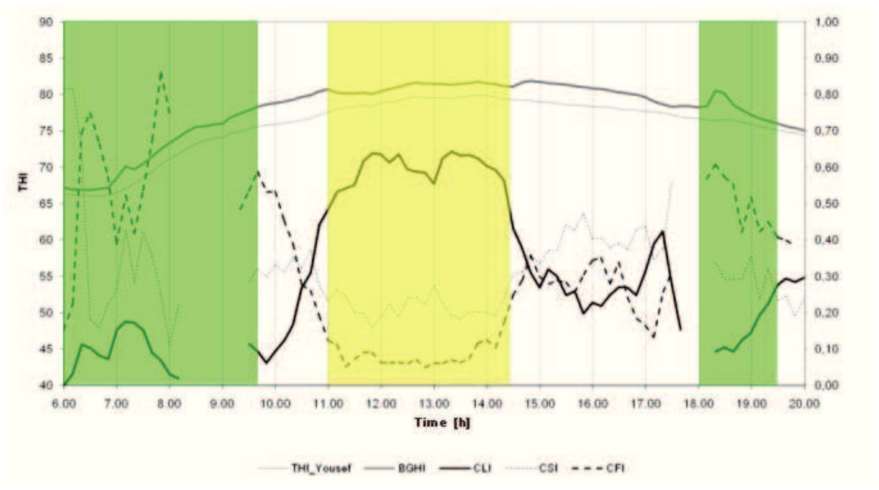


Figure 6.38: CFTime (green area) and CLTime (yellow area) defined according to the CFI and the CLI obtained from Agosta (2012).

Through the use of the software tools described in this section, the data for the selected time period, and in the intervals CFTime CLTime were analyzed.

In particular, for each of the seven days in question and for each time interval established (CFTime and CLTime), a CowHeatMap and a CowSpeedGraph were obtained.

In addition, the CowSpeedGraphs related to the tag with ID 020, distinct by day, where the instantaneous velocities calculated by the software in relation to all the 24 hours are represented, were analyzed.

Part IV

RESULTS AND DISCUSSION

Chapter 7

RESULTS

7.1 LOCALISATION AND IDENTIFICATION PERFORMANCES OF THE RTLS

7.1.1 The RTLS operativity in the barn environment

The RTLS showed a good performance in relation to the harsh environment analysed. In fact, no connection or communication failure occurred and few damages due to oxidation of PoE cables plugs were observed.

The protecting wrapping of the tags resulted suitable to the operative conditions of the trial since it avoided localisation data losses due to device failure. Furthermore, no infections occurred to cows' ears due to tag application. However, a further improvement was obtained by using the methodology of application with collars, because it's a totally non-invasive method, offers the possibility to integrate additional devices for other studies, guarantees a longer duration in time, allow for easier interventions in case of maintenance of tags due to the simplicity of insertion/removal of the collars. The collars are, also, accessories easily commercially available and do not require high costs.

Since tag battery consumption is directly related to system configuration, the RTLS acquisition time rate chosen in this study determined a tag battery duration of about two months.

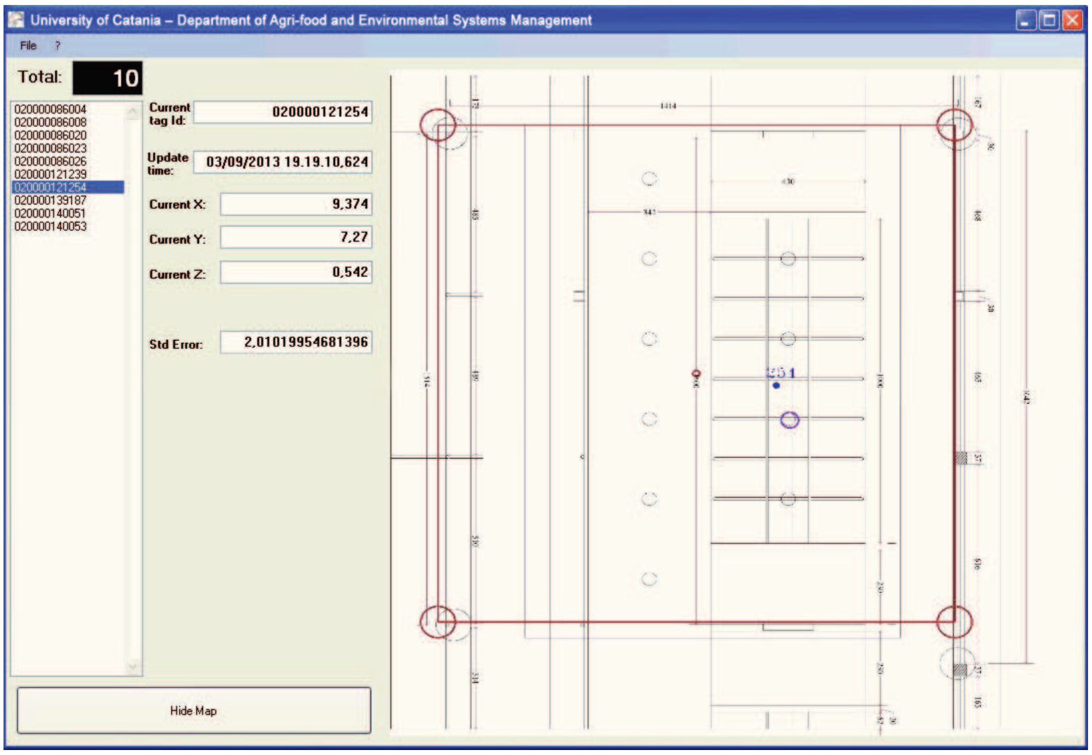
7.1.2 The software for storage of the data acquired by the RTLS

During cow localisation phase, in order to store the data acquired by the RTLS, the specifically developed software, described in section 6.2.5, was launched.

This software, at the same time, allowed the user to view real-time informations acquired from the RTLS through an interface. Figure 7.1 shows a screenshot of the UI of the tool in the running state.

On the left, the tag IDs, which are currently present within the monitored area, are listed and the total number of them is shown immediately above. Selecting one of the IDs in the list the corresponding values of: i) 'Current Tag ID', ii) 'Update Time', iii) 'Current X', iv) 'Current Y', v) 'Current Z', vi) 'Standard Error' are shown in the near textbox.

On the right the section 'Map', which is a reproduction of the plan view of the study area, is shown. The coverage area of the sensors (the four red circles) is bounded by thick red lines. The position of the selected tag is represented by a graphic element (blue filled circle) above which the three last digits of the ID are shown.


University of Catania – Department of Agri-food and Environmental Systems Management
File
?
Hide Map
251

Total: 10

020000086004
020000086008
020000086020
020000086023
020000086026
020000121239
020000121254
020000139187
020000140051
020000140053

Current tag Id: 020000121254

Update time: 03/09/2013 19.19.10,624

Current X: 9,374

Current Y: 7,27

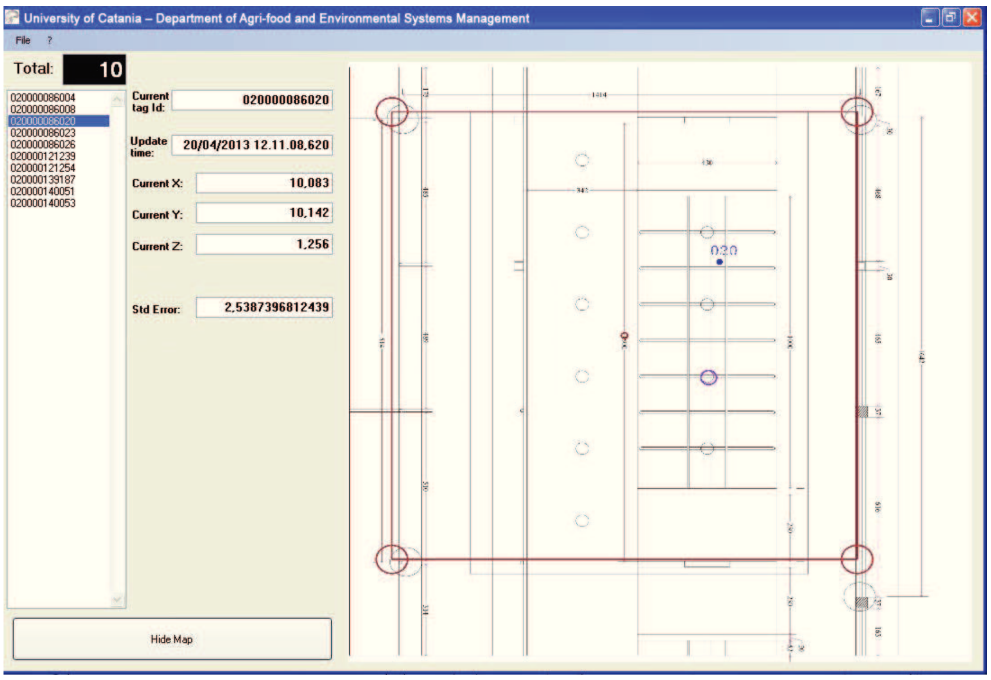
Current Z: 0,542

Std Error: 2,01019954681396

Figure 7.1: UI of the tool for RTLS data storage.

In Figures 7.2 and 7.3, instead, two screenshots of the UI of this tool in two particular cases are shown.

Figure 7.2, in particular, shows the data acquired from the RTLS for one of the tags applied to the cows in the case where the cow in question was in perching. The software is able to clarify the cow behaviour observed because it shows the graphic element corresponding to the tag located in a cubicle and, at the same time, the value of the z-coordinate equal to 1.256 m, which are typical features of the behaviour associated with perching.


University of Catania – Department of Agri-food and Environmental Systems Management

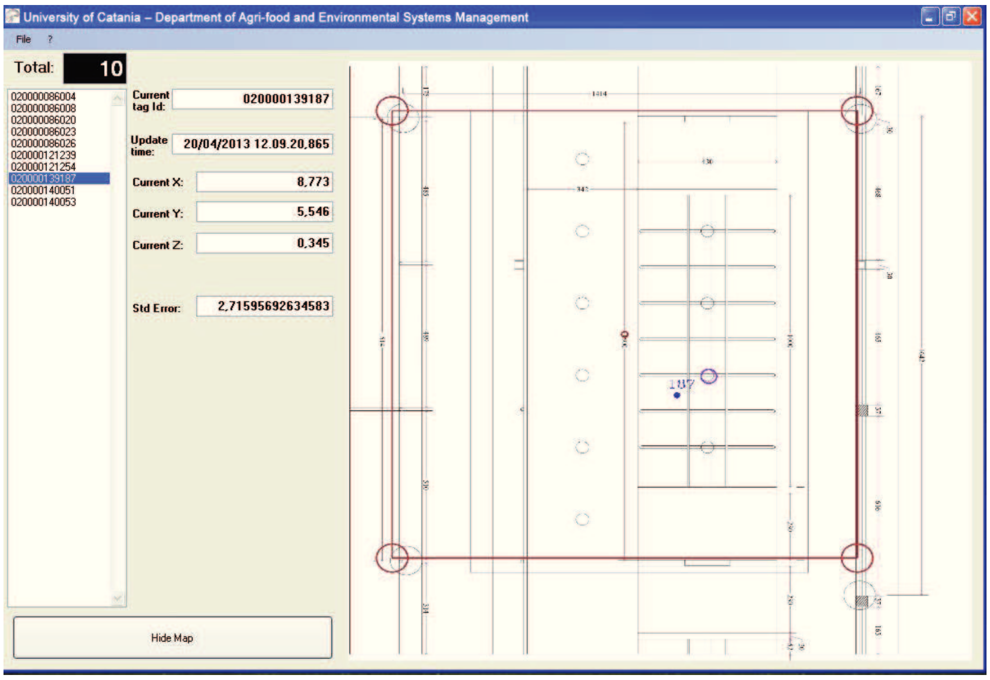
Total: 10

020000086004
020000086008
020000086020
020000086023
020000086026
020000121239
020000121254
020000139187
020000140051
020000140053

Current tag Id: 020000086020
Update time: 20/04/2013 12.11.08,620
Current X: 10,083
Current Y: 10,142
Current Z: 1,256
Std Error: 2,5387396812439

Figure 7.2: UI of the software for storing data acquired by the RTLS. The displayed tag is applied to a cow in perching.

Figure 7.3, instead, shows the data relating to one of the cows tags in the case where the cow in question was in Lying. The software, in fact, shows the graphic element, corresponding to the monitored tag, within a cubicle and, simultaneously, the value of the z coordinate equal to 0.345 m, which are typical features of the behavior associated with the Lying.


University of Catania – Department of Agri-food and Environmental Systems Management
File
?
Hide Map

Total: 10

020000086004
020000086008
020000086020
020000086023
020000086026
020000121239
020000121254
020000139187
020000140051
020000140053

Current tag Id:	020000139187
Update time:	20/04/2013 12.09.20,865
Current X:	8,773
Current Y:	5,546
Current Z:	0,345
Std Error:	2,71595692634583

Figure 7.3: UI of the software for storing data acquired by the RTLS. The displayed tag is applied to a cow in Lying.

7.1.3 The sequence of panoramic top-view images and the building of the dataset of tags true positions

To obtain the fully rectified panoramic top-view images of the two selected functional units. i.e, feeding area and resting area, the installation of 10 cameras was required, according to a previously defined methodology (Porto et al., 2013).

In the chosen time intervals, the verification of 10742 tag positions was carried out through the use of 1600 panoramic images acquired by the video-recording system. The dataset of tags true positions was obtained by using the specifically developed software.

By using this software, the accuracy assessment took about 40 hours of operator's work.

In the next section the user interface of this software is described.

7.1.3.1 User Interface of the software for the computation of tags true positions dataset

The software specifically developed for the computation of the dataset of the true positions of UWB tags has a user-friendly User Interface. This software provides a quick, accurate, automatic and interactive tool designed to take advantage of the technologies used in this study for the localisation and the video recording.

At program startup it is necessary to enter the input data, which consists of the folder that contains the text files, identified by the day of recording, related to the location systems and the video-recording system when the matching occurred and also grouped by tags. This input is, in fact, the output of the software which carried out the matching between the tags dataset and the images dataset, described in section 6.4.1.3.

The input loading operation is described in Figure 7.4.

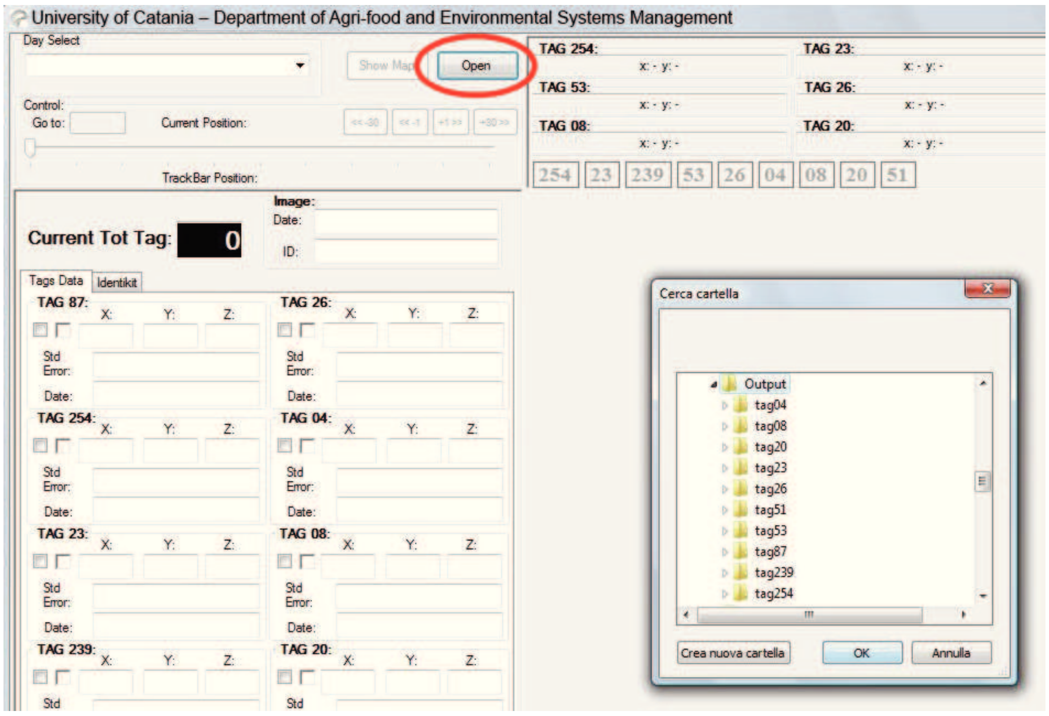


Figure 7.4: User Interface at program startup. By clicking on the 'Open' button a window, that allows you to search for the folder containing the input of the program, is opened.

After the input data are loaded, the software make it possible to choose the day of recording of the two systems for which you want to perform the visual recognition of the position of each tag.

In Figure 7.5 a screenshot of the phase described above is shown.

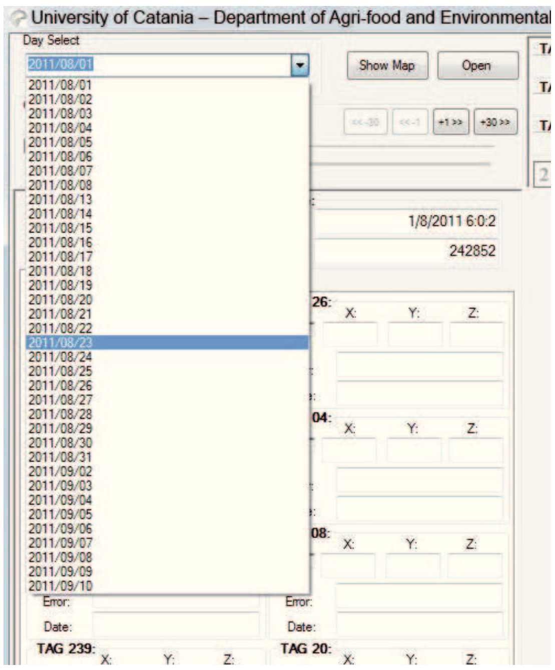
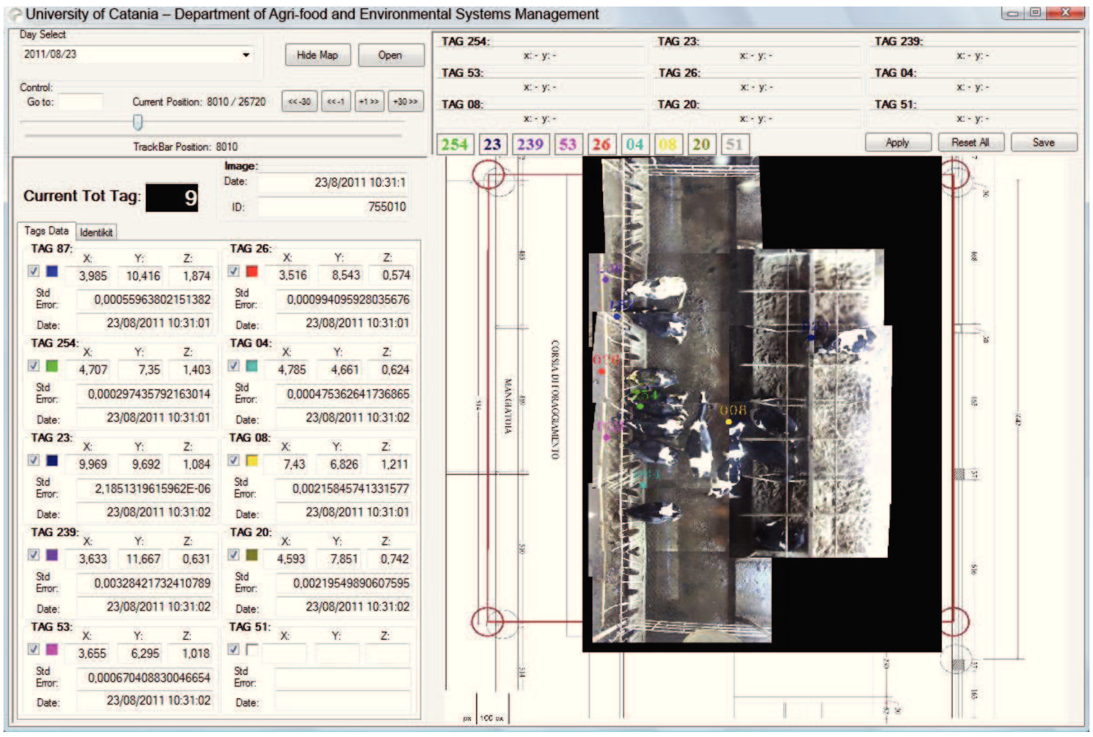


Figure 7.5: Day Select section of the tool. A 'select box' allows you to select the day of stored recording of the two systems for which you want to perform the visual recognition of the position of each tag.

Once that the date is selected, the software loads and displays the information captured by the two systems in relation to the first record in the input file corresponding to the selected day.

The software UI is composed of different sections, which are utilised to display some information or interact with it by sending some commands (Figure 7.6).



University of Catania – Department of Agri-food and Environmental Systems Management

Day Select: 2011/08/23

Control:
Go to:
Current Position: 8010 / 26720
TrackBar Position: 8010

Current Tot Tag: 9

Image:
Date: 23/8/2011 10:31:1
ID: 755010

		X:	Y:	Z:	Std Error:	Date:
TAG 87:	✓ 🟦	3,985	10,416	1,874	0,00055963802151382	23/08/2011 10:31:01
TAG 26:	✓ 🟥	3,516	8,543	0,574	0,000994095928035676	23/08/2011 10:31:01
TAG 254:	✓ 🟩	4,707	7,35	1,403	0,000297435792163014	23/08/2011 10:31:01
TAG 04:	✓ 🟦	4,785	4,661	0,624	0,000475362641736865	23/08/2011 10:31:02
TAG 23:	✓ 🟦	9,969	9,692	1,084	2,1851319615962E-06	23/08/2011 10:31:02
TAG 08:	✓ 🟨	7,43	6,826	1,211	0,00215845741331577	23/08/2011 10:31:01
TAG 239:	✓ 🟪	3,633	11,667	0,631	0,00328421732410789	23/08/2011 10:31:02
TAG 20:	✓ 🟫	4,593	7,851	0,742	0,00219549890607595	23/08/2011 10:31:02
TAG 53:	✓ 🟪	3,655	6,295	1,018	0,000670408830046654	23/08/2011 10:31:02
TAG 51:	✓ ☐					

TAG 254: x: - y: -	TAG 23: x: - y: -	TAG 239: x: - y: -
TAG 53: x: - y: -	TAG 26: x: - y: -	TAG 04: x: - y: -
TAG 08: x: - y: -	TAG 20: x: - y: -	TAG 51: x: - y: -

Figure 7.6: Software UI for the computation of the true positions of UWB tags.

At the top left corner of the UI, under the 'Day Select' tab, there is the 'Control' tab, which consists of:

'Go to:' textbox suitable to enter, from the keyboard, the number of records relative to the loaded input file. In this textbox it is possible to write the position required for the visual recognition of the position of each tag. The 'Current Position' is displayed below.

'Trackbar', to move to the desired record to be loaded, by moving the cursor along the horizontal line, directly with the mouse. The current 'Trackbar Position' is displayed below.

Four buttons to move 30 record backwards, 1 record backwards, 1 records forward or 30 records forward from the current record, respectively. The software automatically checks, with respect to the position of the current record, when enabling or disabling these buttons appropriately. If, for example, the current record is the last record of the selected day, the last two buttons are automatically disabled because, since no other records to follow, any pressing of one of the two buttons in question would cause an error.

Under the 'Control tab' there is a section dedicated to the display of informations of the current record. In particular, it is constituted by:

'Image' tab, that shows 'Date' and 'ID' of the current image used for the visual recognition of the position of each tag.

'Tags Data' tab, which shows, for each tag, 'X', 'Y' and 'Z' coordinates, 'Standard error' and 'Date' related to the current record.

'Identikit' tab, which allows the visualization of two images of each cow associated to the identification number of the tag (see Figure 7.7).

'Current Tot Tag' textbox, which shows the total number of tags associated to the current image in the operation of matching between the two datasets.

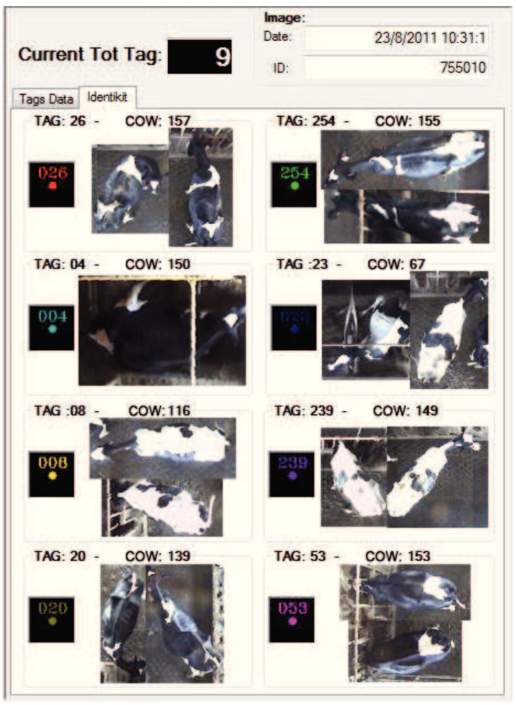


Figure 7.7: 'Identikit' tab of the Software tool. It allows the visualization of two images of each cow associated to the identification number of the tag.

The section located on the right side of the UI is dedicated to the management of the commands suitable for the visual recognition of the position of each tag and any manual correction of that position.

In particular, a graphical representation of the study area (called 'Map'), which consists in the overlap of the top-view panoramic image corresponding to the current record and the reproduction of the plan view of the study area, is available for the user. The software allows the visualization of current tags positions, within the 'Map', by using graphic elements (points) above which the last three digits of the respective tag are shown.

The software also provides the user with nine labels, each corresponding to one of the tags applied to cows. These labels are located just above the Map, as it is shown in Figure 7.8.

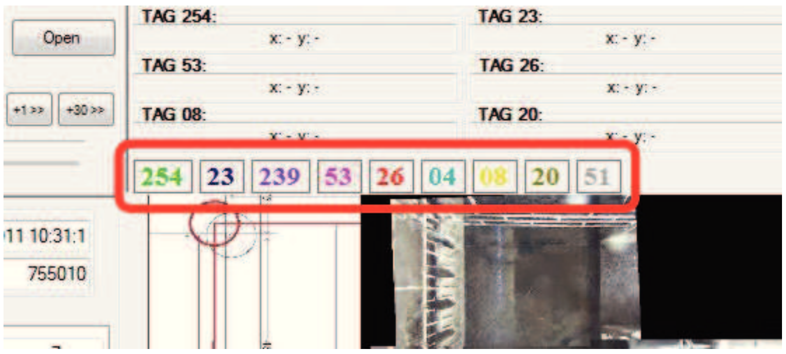


Figure 7.8: The nine labels, each corresponding to one of the tags applied to cows. By using these labels the user has the possibility to accurately determine the true position of each tag by placing them appropriately to the desired position by a drag and drop operation.

Through a visual recognition of the Map and the use of these labels, the user has the possibility to accurately determine the true position of each tag, by appropriately placing these labels to the desired position through a drag and drop operation. During this operation, on the top-right corner of the UI, the coordinates (x, y) of the true positions of the tags are shown.

During this phase, the identikit tab is an important support tool for the user, because it makes it possible to check whether the tag was successfully associated to the cow in question by the RTLS.

In this section there are three buttons:

'Apply': it makes it possible to apply the changes without saving on disk. The software saves in a temporary data structure the new positions determined by the user to keep any changes in memory.

'Reset All': it makes it possible to undo the changes in the current record. The software, after pressing this button, replaces all the labels in the initial position by resetting the changes.

'Save': it makes it possible to save all the changes. The software saves on the disk the information related to the true locations of tags which was placed in the temporary data structure. This information constitutes the output of the software and, therefore, the dataset of the true positions of UWB tags.

In the design phase of this software, a specific colour was established and associated with some elements of the UI to allow for an easy visual recognition of the position of each tag carried out by the user. To this purpose, ten distinct colours were associated to the ten tags. Subsequently, these colours were used to highlight the elements of the UI related to the tags. In particular:

In Tags Data tab, near to the coordinates of each tag, a graphic element which has the shape of a square and is filled with the colour associated to the related tag, was inserted.

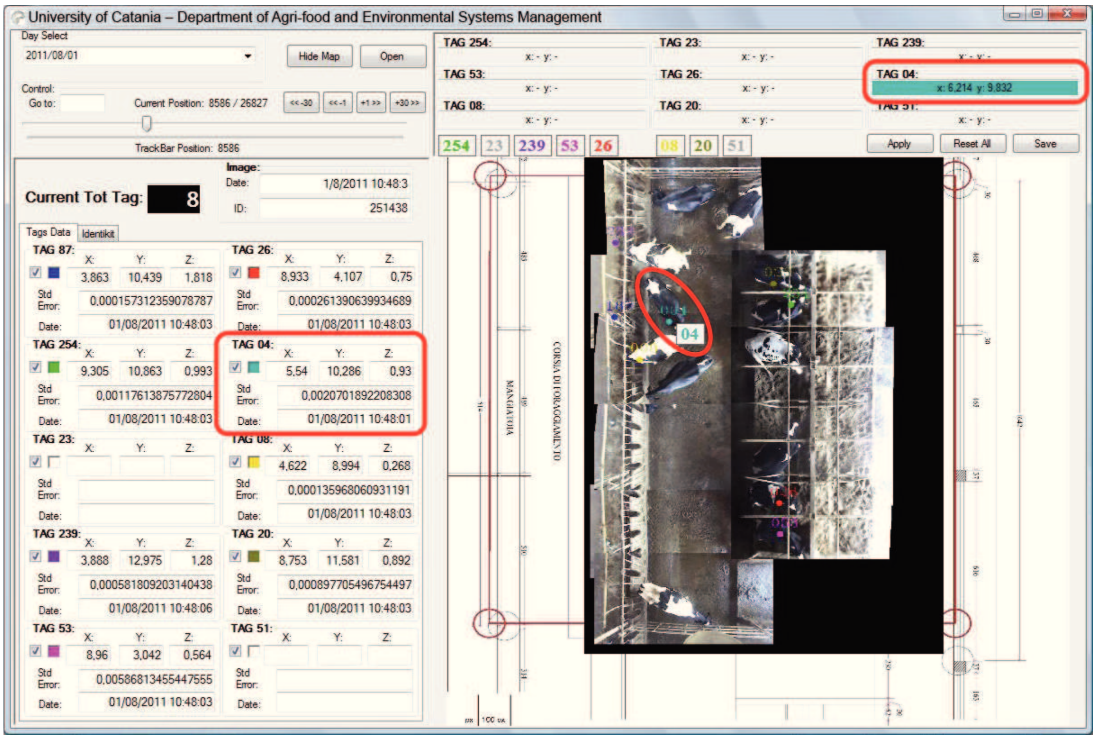
In Identikit tab, two images of each cow are associated to the identification number of the tag and a graphic element with the shape of a circle coloured according to the specific tag colour.

In the Map, the graphic elements (points), which have above them the last three digits of the respective tags, are coloured according to the specific tag colour.

Each label, which is used to determine the true position of a tag (Figure 7.8), is characterized by the ID of the tag to which it is associated and from the colour chosen for the text according to the tag colour.

The coordinates of the cow true position displayed on the top-right corner of the UI are highlighted according to the tag colour.

In Figure 7.9, the computation of the true position of tag with ID 004 was exemplified. The association of a label to each cow make it possible for the operator to place the label 04 on the cow's head by a drag and drop operation. The cow true positions, shown in the top-right section of the UI, were automatically stored after clicking on the Save button.


University of Catania – Department of Agri-food and Environmental Systems Management

Day Select: 2011/08/01

Control:
Go to:
Current Position: 8586 / 26827
TrackBar Position: 8586

Current Tot Tag: 8

Image:
Date: 1/8/2011 10:48:3
ID: 251438

	X:	Y:	Z:	Std Error:	Date:
TAG 87:	3,863	10,439	1,818	0,000157312359078787	01/08/2011 10:48:03
TAG 26:	8,933	4,107	0,75	0,000261390639934689	01/08/2011 10:48:03
TAG 254:	9,305	10,863	0,993	0,00117613875772804	01/08/2011 10:48:03
TAG 04:	5,54	10,286	0,93	0,0020701892208308	01/08/2011 10:48:01
TAG 23:					
TAG 08:	4,622	8,994	0,268	0,000135968060931191	01/08/2011 10:48:03
TAG 239:	3,888	12,975	1,28	0,000581809203140438	01/08/2011 10:48:06
TAG 20:	8,753	11,581	0,892	0,000897705496754497	01/08/2011 10:48:03
TAG 53:	8,96	3,042	0,564	0,00586813455447555	01/08/2011 10:48:03
TAG 51:					

TAG 254: x: - y: -
TAG 23: x: - y: -
TAG 239: x: - y: -
TAG 53: x: - y: -
TAG 26: x: - y: -
TAG 04: x: 6,214 y: 9,832
TAG 08: x: - y: -
TAG 20: x: - y: -
TAG 51: x: - y: -

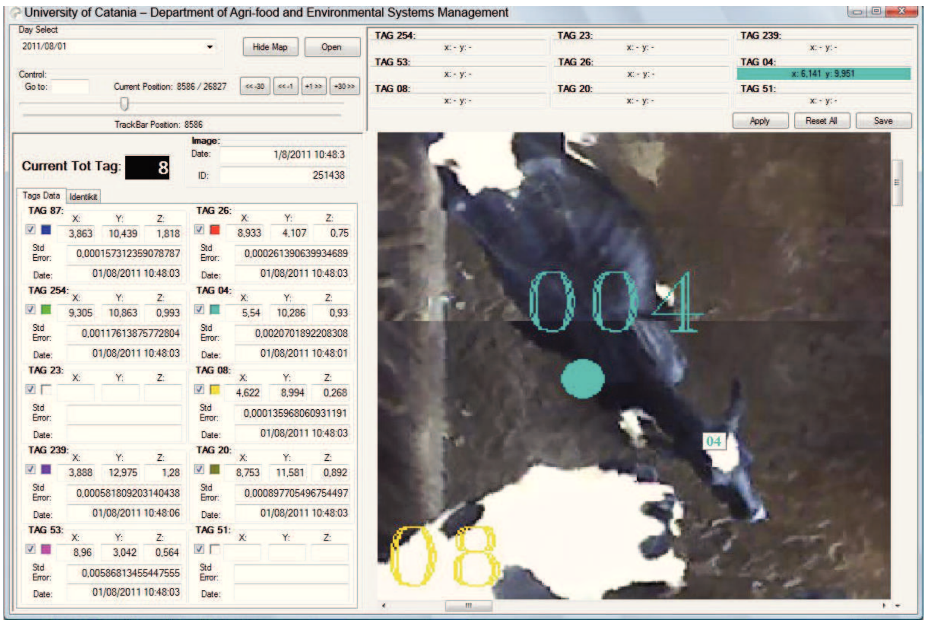
Figure 7.9: The computation of the true position of tag with ID 004. The association of a label to each cow made it possible for the operator to place the label 04 on the cow's head by a drag and drop operation. The cow true positions, shown in the top-right section of the UI, were automatically stored after clicking on the Save button.

Another important feature was implemented in the software in order to provide greater precision to the operations of computation of the true position of tags.

This feature allows you to perform pan and zoom operations by using the mouse scroll wheel in the Map section as shown in Figure 7.10.

Another example of the computation of the true position of tags is exemplified in Figure 7.11.

The format and structure of the output of the software described above are the same as its input, as described in section 6.4.1.3. The only differences between the output text files and the input ones are the values of the two coordinates (x and y), in the case where the operator makes and saves changes.



University of Catania – Department of Agri-food and Environmental Systems Management

Day Select: 2011/08/01

Control:
Go to:
Current Position: 8586 / 26827
TrackBar Position: 8586

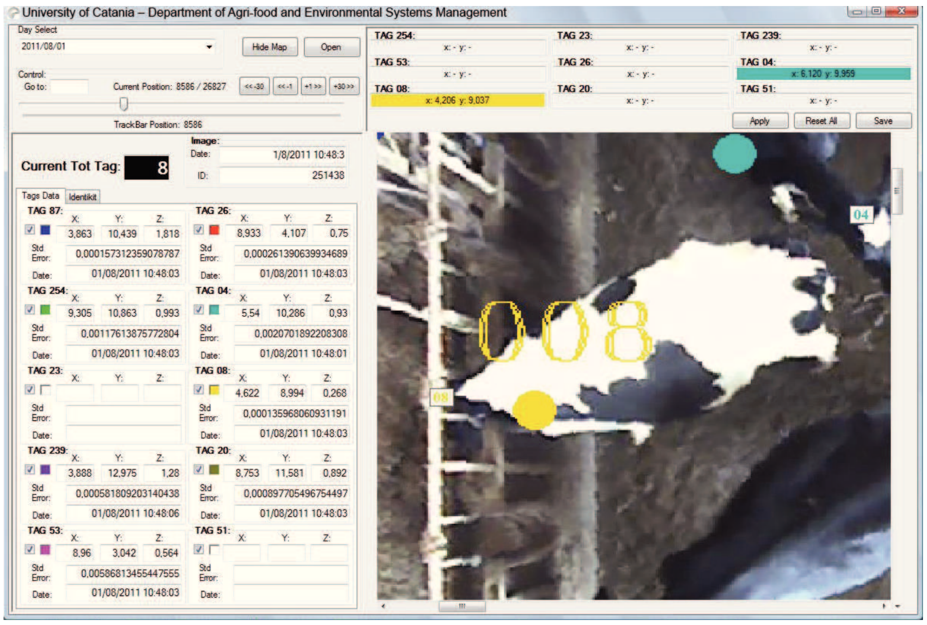
Current Tot Tag: 8

Image:
Date: 1/8/2011 10:48:3
ID: 251438

		X:	Y:	Z:	Std Error:	Date:
TAG 87:	✓ 🟦	3,863	10,439	1,818	0,000157312359078787	01/08/2011 10:48:03
TAG 26:	✓ 🟥	8,933	4,107	0,75	0,000261390639934689	01/08/2011 10:48:03
TAG 254:	✓ 🟩	9,305	10,863	0,993	0,00117613875772804	01/08/2011 10:48:03
TAG 04:	✓ 🟦	5,54	10,286	0,93	0,0020701892208308	01/08/2011 10:48:01
TAG 23:	✓ ☐					
TAG 08:	✓ 🟨	4,622	8,994	0,268	0,000135968060931191	01/08/2011 10:48:03
TAG 239:	✓ 🟪	3,888	12,975	1,28	0,000581809203140438	01/08/2011 10:48:06
TAG 20:	✓ 🟩	8,753	11,581	0,892	0,000897705496754497	01/08/2011 10:48:03
TAG 53:	✓ 🟪	8,96	3,042	0,564	0,00586813455447555	01/08/2011 10:48:03
TAG 51:	✓ ☐					

TAG 254: x: - y: -	TAG 23: x: - y: -	TAG 239: x: - y: -
TAG 53: x: - y: -	TAG 26: x: - y: -	TAG 04: x: 6,141 y: 9,951
TAG 08: x: - y: -	TAG 20: x: - y: -	TAG 51: x: - y: -

Figure 7.10: In Map section, an example of pan and zoom operation.



University of Catania – Department of Agri-food and Environmental Systems Management

Day Select: 2011/08/01

Control:
Go to:
Current Position: 8586 / 26827
TrackBar Position: 8586

Current Tot Tag: 8

Image:
Date: 1/8/2011 10:48:3
ID: 251438

		X:	Y:	Z:	Std Error:	Date:
TAG 87:	✓ 🟦	3,863	10,439	1,818	0,000157312359078787	01/08/2011 10:48:03
TAG 26:	✓ 🟥	8,933	4,107	0,75	0,000261390639934689	01/08/2011 10:48:03
TAG 254:	✓ 🟩	9,305	10,863	0,993	0,00117613875772804	01/08/2011 10:48:03
TAG 04:	✓ 🟦	5,54	10,286	0,93	0,0020701892208308	01/08/2011 10:48:01
TAG 23:	✓ ☐					
TAG 08:	✓ 🟨	4,622	8,994	0,268	0,000135968060931191	01/08/2011 10:48:03
TAG 239:	✓ 🟪	3,888	12,975	1,28	0,000581809203140438	01/08/2011 10:48:06
TAG 20:	✓ 🟩	8,753	11,581	0,892	0,000897705496754497	01/08/2011 10:48:03
TAG 53:	✓ 🟪	8,96	3,042	0,564	0,00586813455447555	01/08/2011 10:48:03
TAG 51:	✓ ☐					

TAG 254: x: - y: -	TAG 23: x: - y: -	TAG 239: x: - y: -
TAG 53: x: - y: -	TAG 26: x: - y: -	TAG 04: x: 6,120 y: 9,959
TAG 08: x: 4,206 y: 9,037	TAG 20: x: - y: -	TAG 51: x: - y: -

Figure 7.11: An example of the computation of the true position of the tags with ID 004 and 008.

7.1.4 Planimetric position errors obtained by the RTLS

In this section, the planimetric position errors obtained by the RTLS in the static test (Table 7.1) and these obtained for each one of the eight tags analysed and for the reference tag (ID 187), before outlier data cleaning (Table 7.2) and after outlier data cleaning (Table 7.3) were shown.

The performance of the RTLS resulted higher in static conditions than in the case of moving objects. Tables 7.2 and 7.3 shows that the localisation error made by the RTLS for the reference tag is definitively lower than the errors computed for the moving tags. An analogous observation was gathered in a test carried out in a railway tunnel (Mok et al., 2010). The static test, carried out in this study (section 6.2.6), confirmed this concept. In fact, the average value   standard deviation of the means relative to x, y and z coordinates resulted about 0.13 m   0.14 m, 0.11 m   0.11 m and 0.16 m   0.14 m, respectively. Table 7.1 shows mean x, mean y and mean z obtained in this test for each point analysed.

Tables 7.2 and 7.3 show the minimum value, the mean value, the maximum value, and the value at the 90th percentile of tag planimetric position errors obtained before and after outlier data cleaning, respectively.

Before applying outlier data cleaning technique to the dataset, the average values of the mean error and the error at the 90th percentile of the eight tags resulted about 0.56 m and 1.03 m, respectively.

Outlier data cleaning allowed reduction of the localisation errors made by the RTLS and made data distribution more homogeneous without losing any relevant information for the analysis performed in this study. The data discarded by applying this technique (about 1.9% of the error dataset) corresponded to measures clearly unreliable. Figure 7.12 shows the box plot representation of the error computed on the data provided by the RTLS, for each one of the eight tags and the reference tag analysed. The measures discarded by the outlier data cleaning were indicated with the symbol '+'. In this figure, the localisation error at the 25th percentile for the cows' tags ranges between 0 m and 0.65 m with an average value of 0.21 m whereas the localisation error at the 75th percentile varies between 0.62 m and 0.97 m with an average value of 0.77 m.

Static Test Point	**Error (m)**			**Points (n.)**
	Mean x	**Mean y**	**Mean z**	
1	0.32	0.26	0.53	277
2	0.25	0.29	0.08	80
3	0.06	0.07	0.05	79
4	0.07	0.0009	0.02	78
5	0.01	0.13	0.21	80
6	0.48	0.36	0.25	21
7	0.04	0.01	0.24	122
8	0.21	0.02	0.10	41
9	0.13	0.23	0.09	51
10	0.06	0.04	0.07	53
11	0.01	0.16	0.19	41
12	0.09	0.17	0.004	77
13	0.07	0.11	0.29	76
14	0.07	0.06	0.20	77
15	0.11	0.05	0.09	79
16	0.04	0.04	0.13	54
17	0.02	0.02	0.001	621
18	0.43	0.04	0.06	79
19	0.15	0.04	0.12	81
20	0.41	0.16	0.42	77
21	0.23	0.34	0.34	41
22	0.03	0.10	0.27	38
23	0.01	0.006	0.04	39
24	0.04	0.04	0.11	55
25	0.001	0.01	0.01	52

Table 7.1: Planimetric position errors computed on the data provided by the RTLS in the static test.

Tag	Error (m) before outlier data cleaning				Points
ID	Min	Mean	Max	90^{th} perc.	(n.)
004	0.12	0.59	4.41	1.12	1029
008	0.15	0.42	2.96	0.98	1582
020	0.17	0.42	4.16	0.87	1377
023	0.18	0.83	2.62	1.20	1453
026	0.16	0.73	6.84	1.25	1358
053	0.09	0.49	3.97	0.90	1362
239	0.14	0.45	3.97	0.96	1209
254	0.17	0.56	5.31	0.97	1372
187*	0.01	0.11	0.56	0.17	3672

Table 7.2: Planimetric position errors computed on the data provided by the RTLS, for each one of the eight tags analysed and for the reference tag (ID 187), before outlier data cleaning.

Tag	Error (m) after outlier data cleaning				Points
ID	Min	Mean	Max	90^{th} perc.	(n.)
004	0.12	0.53	1.52	1.04	1029
008	0.15	0.39	1.66	0.93	1582
020	0.17	0.38	1.56	0.82	1377
023	0.18	0.79	1.44	1.12	1453
026	0.16	0.66	1.71	1.17	1358
053	0.09	0.45	1.68	0.87	1362
239	0.14	0.41	1.87	0.91	1209
254	0.17	0.51	1.33	0.88	1372
187*	0.01	0.11	0.24	0.17	3672

Table 7.3: Planimetric position errors computed on the data provided by the RTLS, for each one of the eight tags analysed and for the reference tag (ID 187), after outlier data cleaning

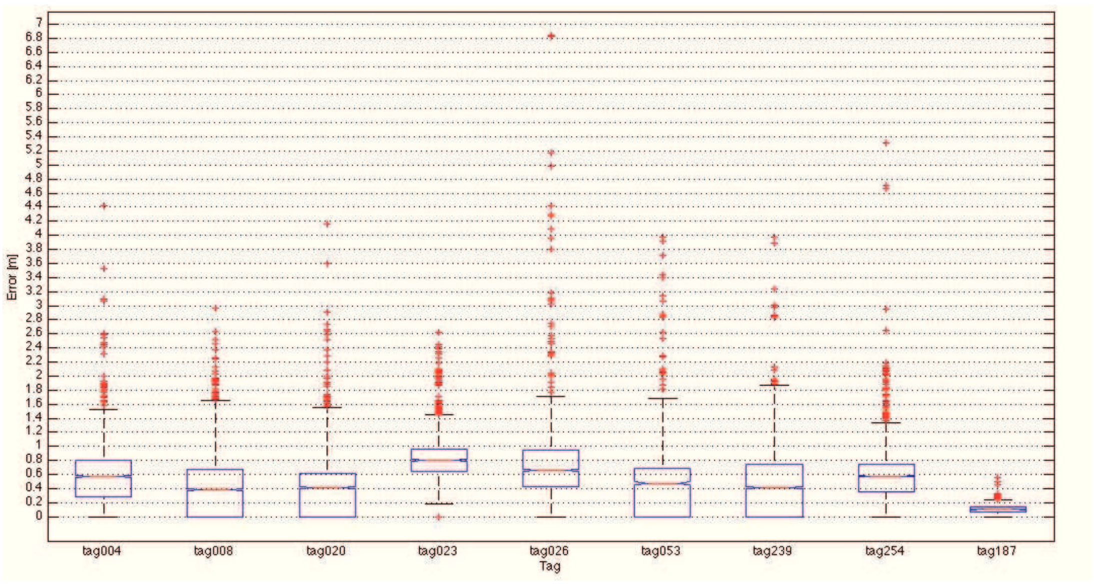


Figure 7.12: Box plot of the errors computed on the data provided by the RTLS, for each one of the nine tags analysed.

After the outlier data cleaning technique was applied to the dataset, the average values of the mean error and the error at the 90^{th} percentile of the eight tags resulted equal to about 0.51 m and about 0.97 m, respectively.

The number of points verified for each of the eight tags applied to the cows were lower than those verified for the reference tag (Tables 7.2 and 7.3). This was due to the fact that all the localisation data of the reference tag provided by the RTLS were considered, since the position of the tag in the barn was known and fixed during the trial. Therefore, the errors were obtained by computing the Euclidean distance between the point deriving from the localisation of the tag and its known position. On the other hand, the number of points verified for each one of the eight tags inserted in the cows' ears resulted lower because it depended on the assessment of tag position in the panoramic image and the rule of automatic association between the image dataset and tag position dataset. In this rule it was established that if the cow was outside the areas monitored by the video-cameras (i.e., the feeding alley or the stalls), or else the acquisition time of the tag was not within the time interval defined to determine the matching with an image, the measure was discarded.

7.1.5 Trade-o between localisation and identi cation performances of the RTLS

Tables 7.4, 7.5 and 7.6 shows the identification and localisation performances of the RTLS relative to Metric A, Metric B and Metric C, respectively. In particular, with reference to each metric, precision and sensitivity are shown for identification performance and mean error and standard deviation are shown for localisation performance.

Tag ID	Metric A			
	Identification		Localisation	
	Precision	Sensitivity	Mean Error	Std (m)
004	1	1	0.59	0.49
008	1	1	0.42	0.47
020	1	1	0.42	0.44
023	1	1	0.83	0.34
026	1	1	0.73	0.59
053	1	1	0.49	0.46
239	1	1	0.45	0.49
254	1	1	0.56	0.44
187*	1	1	0.11	0.05

Table 7.4: RTLS identification and localisation performances computed for each one of the eight tags analysed and for the reference tag (ID 187), for the Metric A.

The average values of precision, sensitivity, and mean error standard deviation of the eight tags resulted about equal to the following values, respectively:

Metric A: 1, 1, 0.56 0.46,

Metric B: 0.98, 0.98, 0.52 0.36,

Metric C: 0.47, 0.47, 0.18 0.20.

From the analysis of these values a high improvement of the localisation performance was observed for Metric C when compared with Metric A and Metric B. On the contrary, identification performance worsened from Metric A to Metric B and particularly in Metric C. The comparison between the

Tag ID	Metric B			
	Identification		Localisation	
	Precision	Sensitivity	Mean Error	Std (m)
004	0.97	0.97	0.53	0.38
008	0.98	0.98	0.39	0.41
020	0.98	0.98	0.38	0.34
023	0.96	0.96	0.79	0.27
026	0.97	0.97	0.66	0.39
053	0.98	0.98	0.45	0.35
239	0.98	0.98	0.41	0.40
254	0.96	0.96	0.51	0.32
187*	0.99	0.99	0.11	0.05

Table 7.5: RTLS identification and localisation performances computed for each one of the eight tags analysed and for the reference tag (ID 187), for the Metric B.

system performances obtained by adopting the Metric A and those obtained by adopting the Metric B showed that the best trade-off between localisation and identification performances was achieved when Metric B was adopted.

Precision and sensitivity resulted almost equal in Metric A and Metric B. In fact, the number of false positives and the number of false negatives were set equal to 0 in Metric A, and these numbers resulted in an irrelevant quantity equal to about 1.9% of all the detections performed by the RTLS in the case where Metric B was adopted. Furthermore, localisation performance was higher in Metric B than in Metric A since the value of mean error standard deviation obtained with the Metric B was lower than that of Metric A.

Similarly, the comparison between the performances obtained by adopting the Metric B and those obtained by adopting the Metric C showed that the best trade-off between localisation and identification performances was obtained when Metric B was adopted. Although localisation performance obtained with Metric C was better than that obtained with Metric B, the number of false positives and the number of false negatives obtained with the Metric C were much higher than those of Metric B since they were equal to about the 39.7% of the RTLS detections.

On the basis of these considerations, the best trade-off between localisation and identification performances of the RTLS was obtained when adopting Metric B.

Regarding the reference tag, precision and sensitivity resulted 1 for Metric A and 0.99 for Metric B and Metric C, and the values of mean error standard deviation resulted 0.11 0.05 for all the metrics (Tables 7.4, 7.5 and 7.6).

Therefore, excellent results were obtained for both localisation and identification performances, regardless of the metric adopted. This mean error is in good agreement with the value of 0.1225 m declared by Ubisense.

Tag ID	Metric C			
	Identification		Localisation	
	Precision	Sensitivity	Mean Error Std (m)	
004	0.42	0.42	0.18	0.20
008	0.61	0.61	0.12	0.19
020	0.61	0.61	0.17	0.20
023	0.11	0.11	0.31	0.20
026	0.32	0.32	0.24	0.20
053	0.53	0.53	0.20	0.21
239	0.61	0.61	0.15	0.20
254	0.54	0.54	0.20	0.20
187*	0.99	0.99	0.11	0.05

Table 7.6: RTLS identification and localisation performances computed for each one of the eight tags analysed and for the reference tag (ID 187), for the Metric C.

7.2 THE AUTOMATIC AND REAL-TIME SOFTWARE TOOL FOR THE VISUAL ANALYSIS OF THE COWS LOCATION DATA

7.2.1 Features of the software

As described in section 6.5, the software specially developed for the real-time visual analysis of the cows' location data in free-stall barns was principally based on the implementation of two instruments: the CowHeatMap and the CowSpeedGraph. The sections 7.2.1.1 and 7.2.1.2 describe the main features offered by these tools, respectively.

7.2.1.1 CowHeatMap

CowHeatMap contains an archive suitable to colorize the heatmap based on relative data. It means that, if you are adding only a single datapoint to the dataset it will be displayed as the hottest (red) spot, then adding another point with a higher count, CowHeatMap will dynamically recalculate the color intensity.

At the configuration phase it is possible to specify the following properties in order to customize the heatmap instance:

radius (optional), Type: Number. It is the radius of a single datapoint in the CowHeatMap (measured in pixels).

element (required), Type: String HTML Element. It either provide an element's id or the element itself which should contain the heatmap.

visible (optional), Type: Boolean. It specifies whether the CowHeatMap is visible or not.

gradient (optional), Type: Object. It is an object which contains colours ranging from 0 to 1.

opacity (optional), Type: Number [0-100]. It provide the opacity of the CowHeatMap measured in percent.

7.2.1.2 CowSpeedGraph

In the CowSpeedGraph the chart is interactive. In fact, it is possible to execute the following actions on the chart:

Mouse-over to highlight individual values.

Click and drag to zoom.

Double-clicking will execute a zoom (back) out.

Shift-drag will pan.

The main features of CowSpeedGraph are:

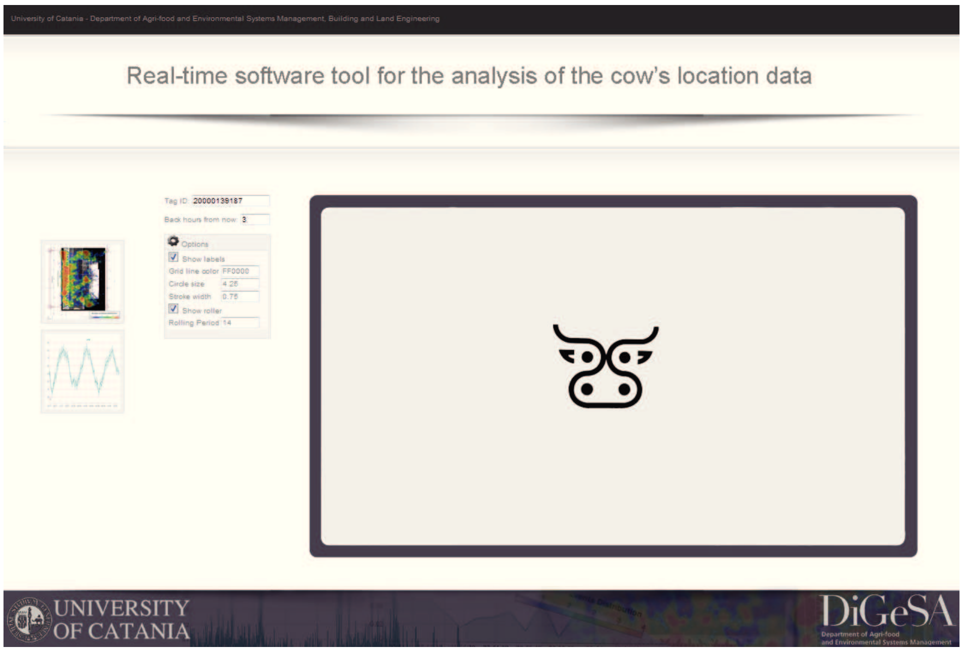
Handles huge data sets: dygraphs plots millions of points without getting bogged down.

Interactive out of the box: zoom, pan and mouse-over are on by default.

Dygraphs is highly compatible: it works in all major browsers (including IE8). It is possible even to pinch in order to zoom on mobile/tablet devices.

7.2.2 Software User Interface

The UI of the software, at boot time, is shown in Figure 7.13.


University of Catania - Department of Agri-food and Environmental Systems Management, Building and Land Engineering

Real-time software tool for the analysis of the cow's location data

Tag ID: 20000139187

Back hours from now: 3

Options

☑ Show labels

Grid line color FF0000

Circle size 4.25

Stroke width 0.75

☑ Show roller

Rolling Period 14

Figure 7.13: A screenshot of the software UI at boot time.

The main content of the UI is divided into two sections: the '*controls*' section and the '*visualisation*' section.

The *controls* section contains two clickable elements that allow the selection of the type of function being performed (CowHeatMap or CowSpeedGraph), two *text- eld* for entering tag *ID* and the number of *back hours from now* which can be chosen for carrying out the analysis and a *setting panel* through which you can set a variety of parameters related to the CowSpeedGraph display, such as:

Show labels: it lets you choose whether or not display the value, on the top right corner, of the point on the graph where the pointer is located.

Grid line color: it allows you to set the color of the grid lines.

Circle size: it allows you to set the size of the circle displayed at the pointer location when positioned within the area of the chart.

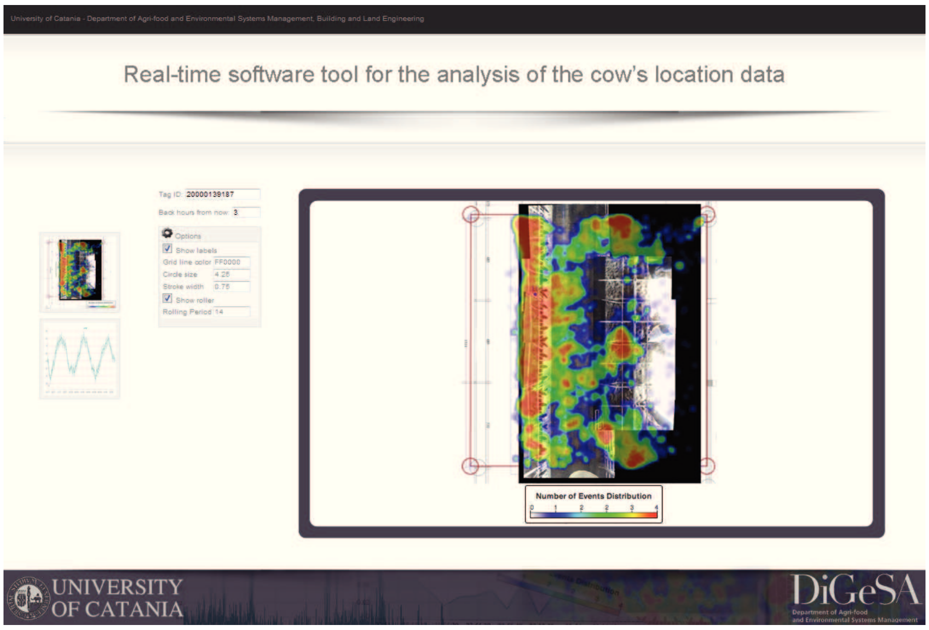
Stroke width: it allows you to set the thickness of the line representing the plotted curve.

Show roller: it lets you choose whether or not display the *roller*, i.e. the value of the display scale.

Rolling Period: it allows you to set the value of the roller.

To view the CowHeatMap, relatively to the parameters selected, you must click on the element, which is associated to a HeatMap icon, situated in the controls section. Figure 7.14 shows an example of CowHeatMap. The CowHeatMap will appear in the visualisation section. The number of events distribution varies depending on the dataset of the CowHeatMap and the color varies from red (maximum concentration of the location events in the point) to transparent (null concentration of the location events). The minimum concentration of events location is indicated by the blue color.

To view the CowSpeedGraph, related to the selected parameters, you must click on the item which is associated to a SpeedGraph icon (in the controls section). Figure 7.15 shows an example of CowSpeedGraph. As described in section 7.2.1.2, the CowSpeedGraph is interactive. In Figure 7.15, in fact, you can see the zoom operation performed on a region of the graph through the combined click and drag actions. Figure 7.16 shows the view obtained after the zoom operation carried out in the previous figures.


University of Catania - Department of Agri-food and Environmental Systems Management, Building and Land Engineering

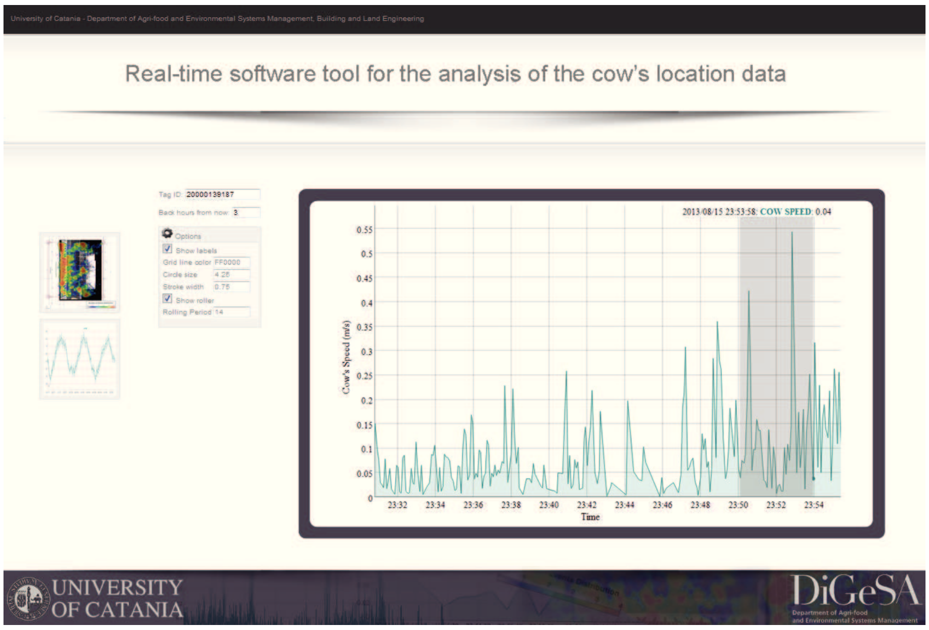
Real-time software tool for the analysis of the cow's location data

Tag ID: 20000139187
Back hours from now: 3

Options
☑ Show labels
Grid line color FF0000
Circle size 4.25
Stroke width 0.75
☑ Show roller
Rolling Period 14

Number of Events Distribution
0
1
2
3
4
UNIVERSITY OF CATANIA
DiGeSA
Department of Agri-food and Environmental Systems Management

Figure 7.14: An example of CowHeatMap displayed in the visualisation section of the UI.


University of Catania - Department of Agri-food and Environmental Systems Management, Building and Land Engineering

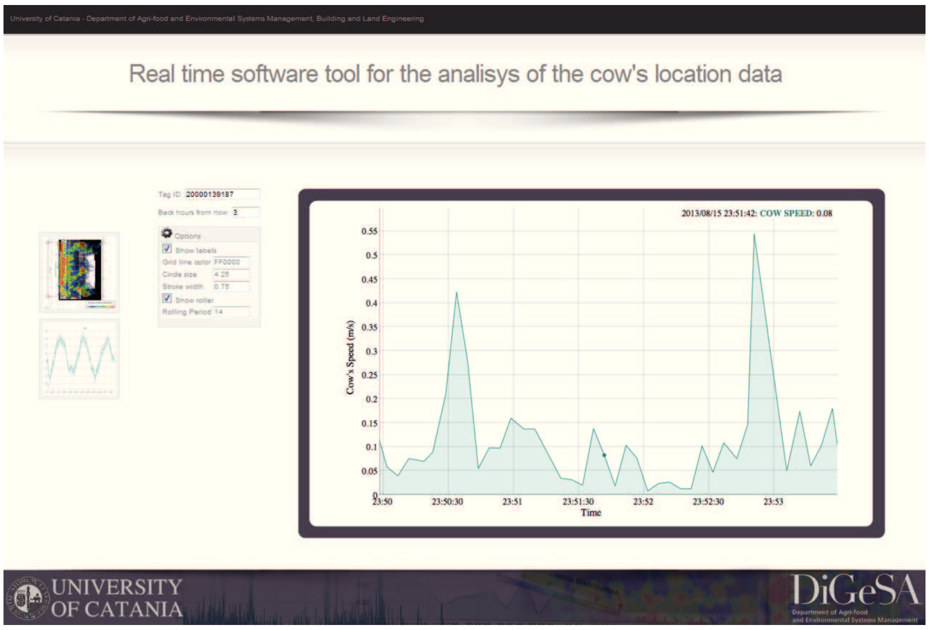
Real-time software tool for the analysis of the cow's location data

Tag ID: 20000139187
Back hours from now: 3

Options
☑ Show labels
Grid line color FF0000
Circle size 4.25
Stroke width 0.75
☑ Show roller
Rolling Period 14

2013/08/15 23:53:58: COW SPEED: 0.04

Figure 7.15: An example of CowSpeedGraph displayed in the visualisation section of the UI.



University of Catania - Department of Agri-food and Environmental Systems Management, Building and Land Engineering

Real time software tool for the analisys of the cow's location data

Tag ID: 20000139187

Back hours from now: 3

Options

✓ Show labels

Grid line color FF0000

Circle size 4.25

Stroke width 0.75

✓ Show roller

Rolling Period 14

2013/08/15 23:51:42: COW SPEED: 0.08

Figure 7.16: CowSpeedGraph obtained after the zoom operation carried out in Figure 7.15.

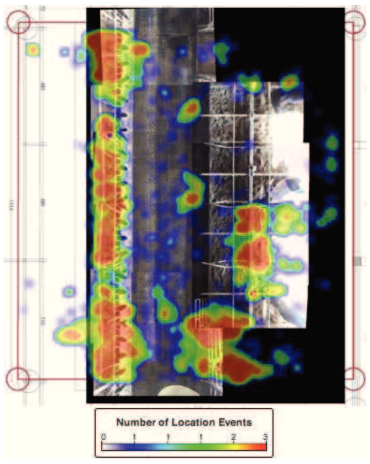
7.2.3 Use case of the software tool

The results obtained from the use case of the the software tool, described in section **??**, are shown in Figures 7.17, 7.18 and 7.19.

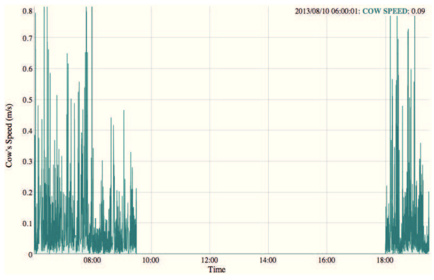
Figure 7.17, in particular, shows the CowHeatMaps and the CowSpeedGraphs relative to the tag with ID 020 in the CFTime interval distinct by day. As already introduced in section 6.5.4, seven days ranging from 10th August 2013 to 16th August 2013 were chosen since in one of these (12th August 2013), the cow having the tag in question showed the status of estrus, which was visually observed by the farmer at around 9:30 am.

By observing the CowHeatMaps, the position of the location points in the space maintained on average a pattern that was repeated nearly in a systematic manner in each of the days selected for the analysis with the exception of 12th August 2013, where a distribution of points in a much wide area was observed.

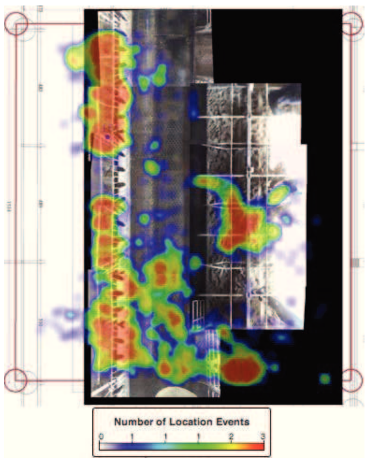
In particular, since the data were related to the CFTime interval, the most frequented zones, in the study area, by cow in question were mainly located at the manger, as expected and, in particular, at the left passage lane (probably due to the presence of the watering hole) and the cubicles (in the middle or the left, on the resting area in particular), by referring



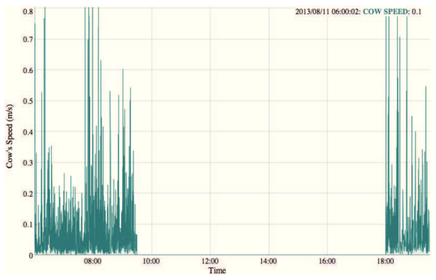
(a) Tag Id: 020; Date: 10/08/2013; Interval: CFTime.



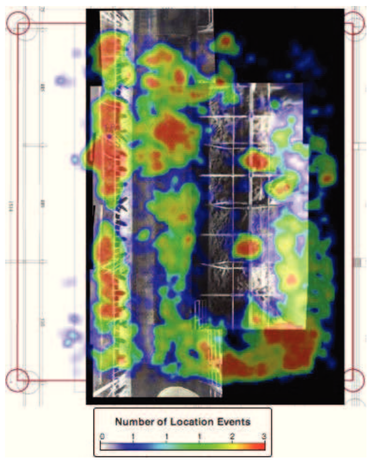
(b) Tag Id: 020; Date: 10/08/2013; Interval: CFTime.



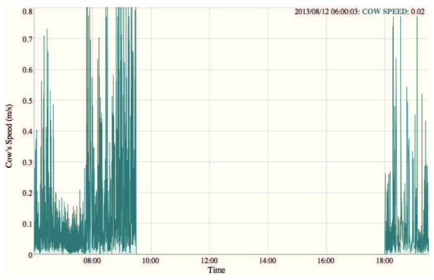
(c) Tag Id: 020; Date: 11/08/2013; Interval: CFTime.



(d) Tag Id: 020; Date: 11/08/2013; Interval: CFTime.

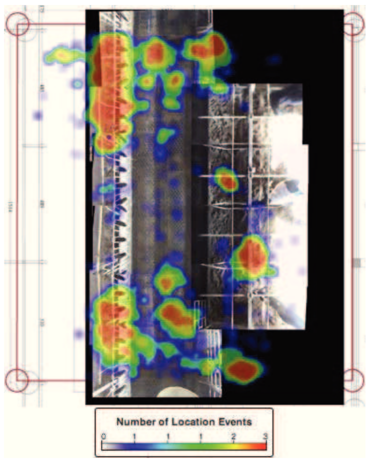


(e) Tag Id: 020; Date: 12/08/2013; Interval: CFTime.

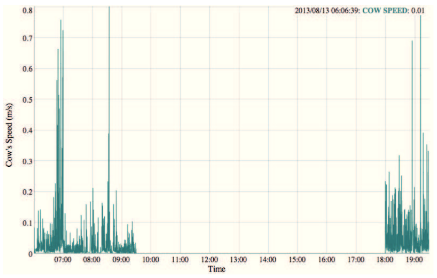


(f) Tag Id: 020; Date: 12/08/2013; Interval: CFTime.

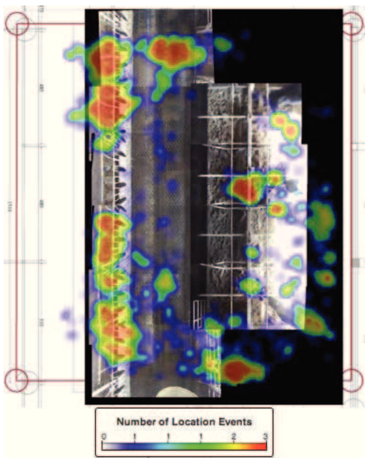
Figure 7.17: CowHeatMaps and CowSpeedGraphs relative to the tag with ID 020 in CFTime interval distinct by day (1/3).



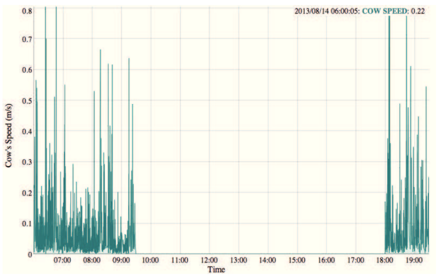
(g) Tag Id: 020; Date: 13/08/2013; Interval: CFTime.



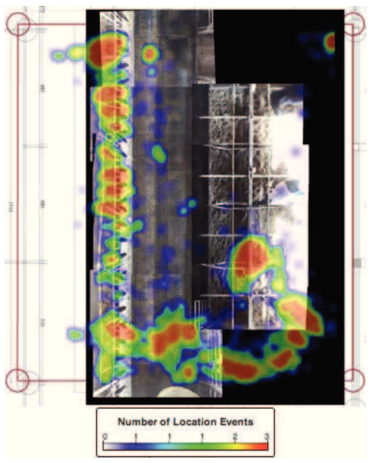
(h) Tag Id: 020; Date: 13/08/2013; Interval: CFTime.



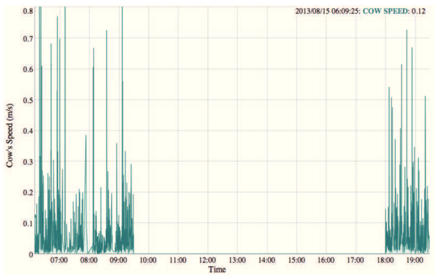
(i) Tag Id: 020; Date: 14/08/2013; Interval: CF-Time.



(j) Tag Id: 020; Date: 14/08/2013; Interval: CFTime.

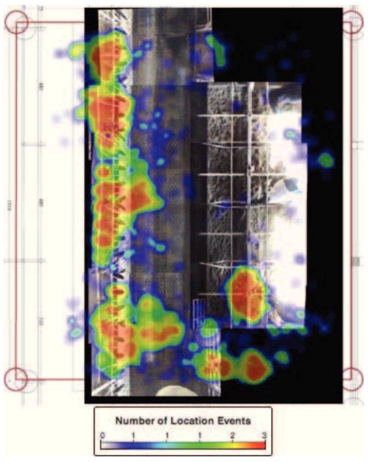


(k) Tag Id: 020; Date: 15/08/2013; Interval: CFTime.

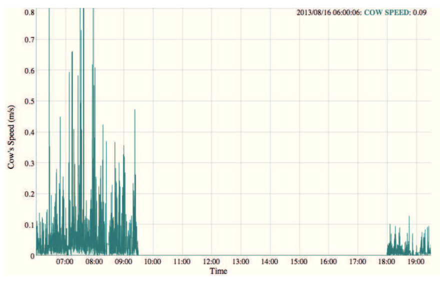


(l) Tag Id: 020; Date: 15/08/2013; Interval: CFTime.

Figure 7.17: CowHeatMaps and CowSpeedGraphs relative to the tag with ID 020 in CFTime interval distinct by day (2/3).



(m) Tag Id: 020; Date: 16/08/2013; Interval: CFTime.



(n) Tag Id: 020; Date: 16/08/2013; Interval: CFTime.

Figure 7.17: CowHeatMaps and CowSpeedGraphs relative to the tag with ID 020 in CFTime interval distinct by day (3/3).

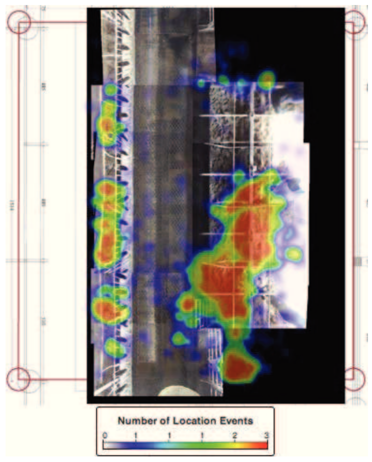
the Figure 6.10. On 12th August 2013, instead the distribution of the points in space mainly involved the manger, the service alley, the left and right passage lanes and the feeding alley. In addition, by observing the coloration of these points, it was clear that, in the CowHeatMap of 12th August 2013, the percentage of red points cover the total number of points represented in the CowHeatMap of that day, was significantly lower than the same quantity measured in other days.

Furthermore, in Figure 7.17, on the right of each CowHeatMap, the relative CowSpeedGraphs are shown, where the instantaneous speeds calculated by the software were plotted. By observing these graphs, also in this case, on 12th August 2013 higher values of instantaneous speed over time were recorded, especially in the time interval ranging from 8:00 am to 10:00 am, compared to those recorded in other days. This proves to be a confirmation that the motor activity manifested by the cow having tag ID 020 in 12th August 2013 was notable compared to that recorded in the other days observed.

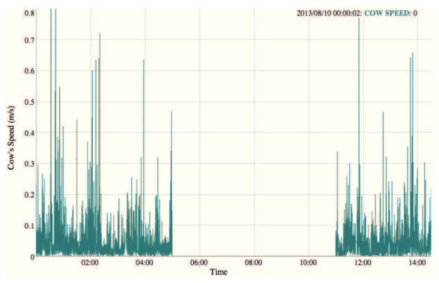
Similarly, Figure 7.18 shows CowHeatMaps and CowSpeedGraphs relative to the tag with ID 020 in CLTime interval distinct by day.

Also in this case, the CowHeatMaps show that the distribution of the location point in the space followed a pattern that on average was repeated in all the days chosen for the test with the exception of 12th August 2013, when, once again a distribution of points in a much wider area was observed.

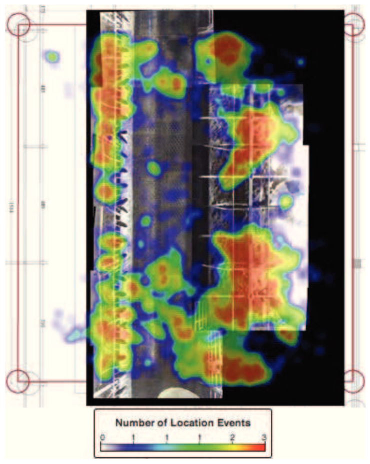
In particular, as expected, in this case the most frequented zones in the



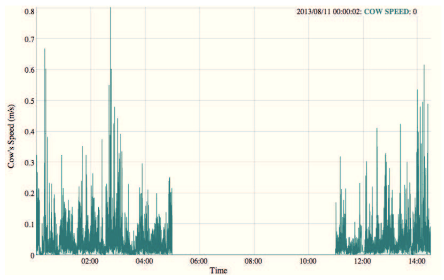
(a) Tag Id: 020; Date: 10/08/2013; Interval: CLTime.



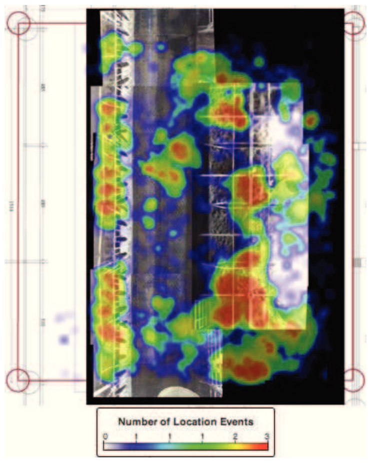
(b) Tag Id: 020; Date: 10/08/2013; Interval: CLTime.



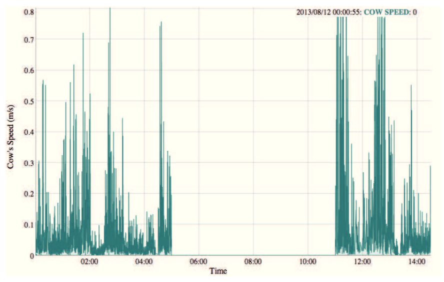
(c) Tag Id: 020; Date: 11/08/2013; Interval: CLTime.



(d) Tag Id: 020; Date: 11/08/2013; Interval: CLTime.

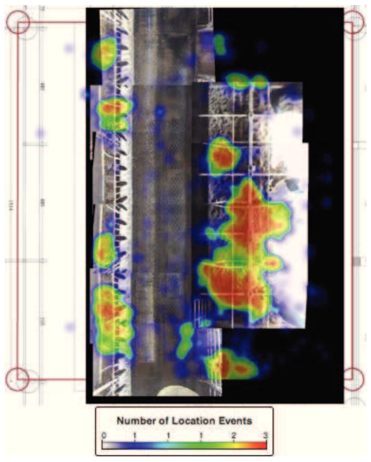


(e) Tag Id: 020; Date: 12/08/2013; Interval: CLTime.

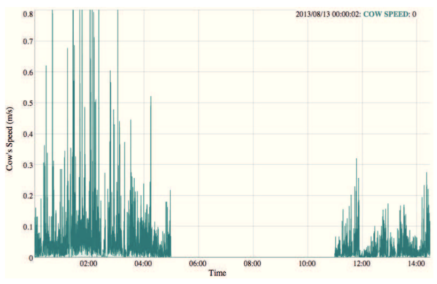


(f) Tag Id: 020; Date: 12/08/2013; Interval: CLTime.

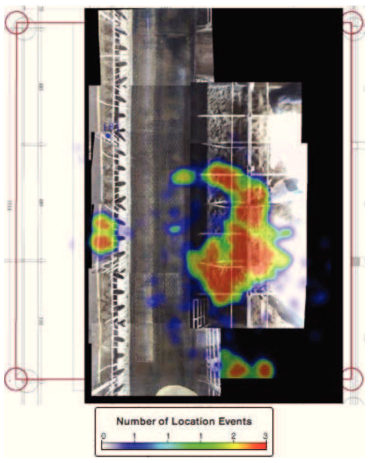
Figure 7.18: CowHeatMaps and CowSpeedGraphs relative to the tag with ID 020 in CLTime interval distinct by day (1/3).



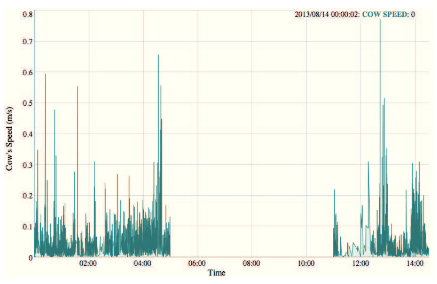
(g) Tag Id: 020; Date: 13/08/2013; Interval: CLTime.



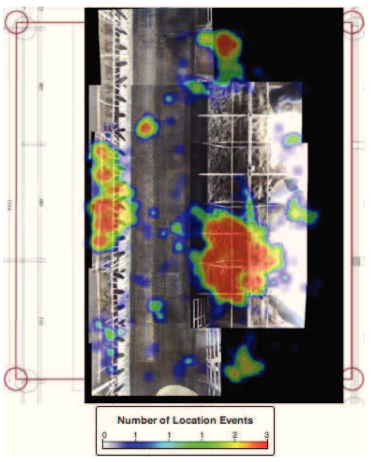
(h) Tag Id: 020, Date: 13/08/2013; Interval: CLTime.



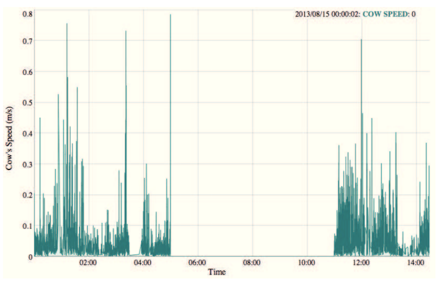
(i) Tag Id: 020; Date: 14/08/2013; Interval: CLTime.



(j) Tag Id: 020; Date: 14/08/2013; Interval: CLTime.

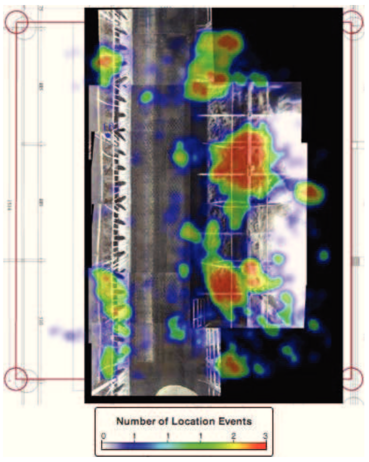


(k) Tag Id: 020; Date: 15/08/2013; Interval: CLTime.

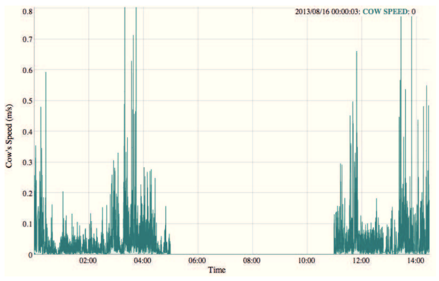


(l) Tag Id: 020; Date: 15/08/2013; Interval: CLTime.

Figure 7.18: CowHeatMaps and CowSpeedGraphs relative to the tag with ID 020 in CLTime interval distinct by day (2/3).



(m) Tag Id: 020; Date: 16/08/2013; Interval: CLTime.



(n) Tag Id: 020; Date: 16/08/2013; Interval: CLTime.

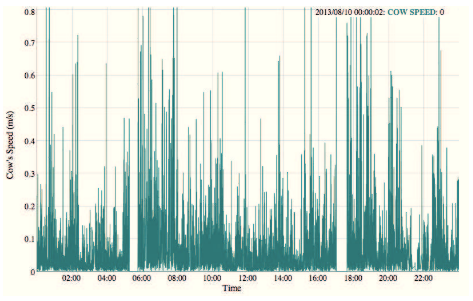
Figure 7.18: CowHeatMaps and CowSpeedGraphs relative to the tag with ID 020 in CLTime interval distinct by day (3/3).

study area by the cow in question were mainly those corresponding to the cubicles (in the middle or in the left, in particular) and in smaller quantities at the manger referring, always, to the Figure 6.10. On the day of 12th August 2013, however, the distribution of points in the space mainly involved, as well as at the manger, the service alley, the left and right passage lane and the feeding alley, as already observed in the CFTime interval. Furthermore, also in this case, by observing the coloration of these points, it was evident that the animal to which the tag with ID 020 was associated on 12th August 2013 showed a motor activity more sustained than that recorded in the other days.

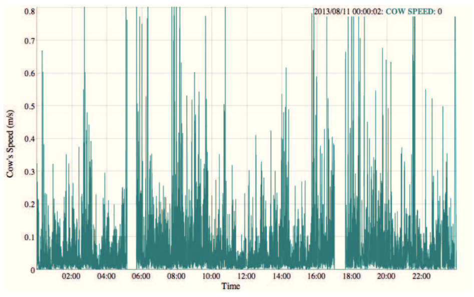
Furthermore, in Figure 7.18 the CowSpeedGraphs associated to the CowHeatMap are shown. By observing these graphs, also in this case, especially in the time interval ranging from 11:00 am and 11:30 am, when the CLI usually shows maximum values, on 12th August 2013 higher values of instantaneous speed were recorded, compared to those obtained in the other days.

Finally, figure 7.19 shows the CowSpeedGraphs relative to the tag with ID 020, distinct by day, where the instantaneous velocities calculated by the software in relation to all the 24 hours are represented.

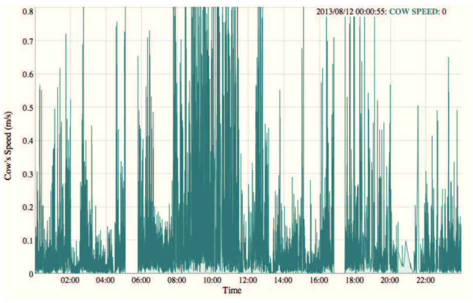
By comparing these charts is even more evident that the curve of the instantaneous velocity on 12th August 2013 shows higher values compared to those recorded in other days, especially in the time interval ranging from the 08:00 am to the 11:30 am.



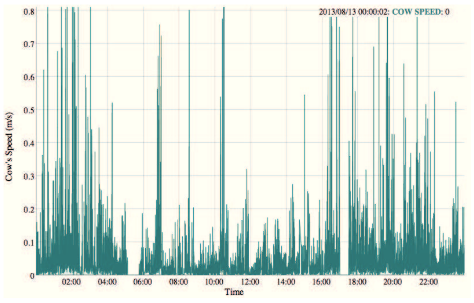
(a) CowSpeedGraph. Tag Id: 020; Date: 10/08/2013; Interval: 24h.



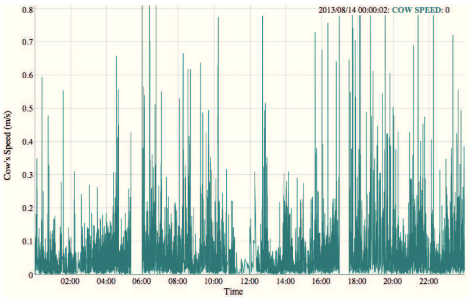
(b) CowSpeedGraph. Tag Id: 020; Date: 11/08/2013; Interval: 24h.



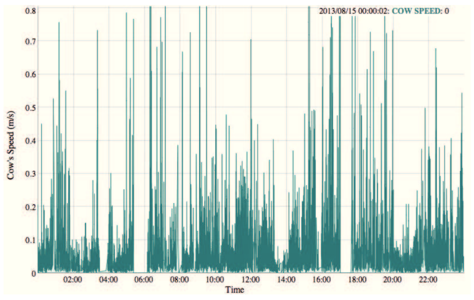
(c) CowSpeedGraph. Tag Id: 020; Date: 12/08/2013; Interval: 24h.



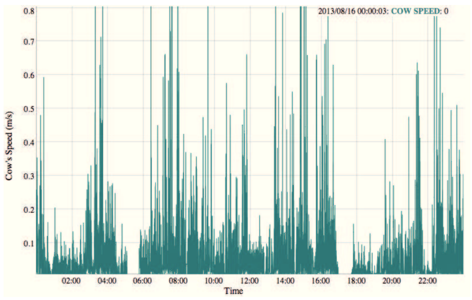
(d) CowSpeedGraph. Tag Id: 020; Date: 13/08/2013; Interval: 24h.



(e) CowSpeedGraph. Tag Id: 020; Date: 14/08/2013; Interval: 24h.



(f) CowSpeedGraph. Tag Id: 020; Date: 15/08/2013; Interval: 24h.



(g) CowSpeedGraph. Tag Id: 020; Date: 16/08/2013; Interval: 24h.

Figure 7.19: CowSpeedGraphs of to the tag with ID 020, distinct by day, relative to all the 24 hours.

Chapter 8

DISCUSSION

The environment of the barn was characterised by the presence of slurry accumulation in the alleys as the cleaning was not automated but it was carried out once a day. This condition facilitates production of gases that may enhance corrosion of materials. On the other hand, this effect is reduced by ventilation which is higher in barns with open sides. In relation to the analysed environment, the physical features of the Ubisense tags and the technique adopted to protect them in this trial allowed a good usability of the system. This demonstrates that some problems mainly due to the dimensions and the weight of the RFID UHF tags, which were highlighted in a previous study (Barbari et al., 2008), could be solved by using the type of tag utilised in this trial. Since the application of tags in cow's ears could cause infections or modifications in cow's behaviour due to the weight on the ear, further improvement of the tag application methods regarded tag positioning on cow's collar. In fact, the application of the tags on cow's collar is a totally non-invasive method, offers the possibility to integrate additional devices for other studies, guarantees a longer duration in time, allow for an easier intervention in case of maintenance of tags due to the ease of insertion/removal of the collars. Furthermore, the collars are accessories easily commercially available and do not require high costs.

Although the outlier data cleaning was carried out for the eight tags and the reference tag, the results related to the reference tag were not included in the computation of the because this tag has lower errors due to its fixed position and, therefore, this would lower the mean error of the tags as a whole.

In the operative conditions considered in this research, the results proved that the RTLS produced an error in computing reference tag positions, which is comparable with that declared by Ubisense and smaller than that obtained by Stephan et al. (2009) under optimal operating conditions and under realistic shop-floor conditions. The results related to the tags applied to the cows showed that a higher error is achieved than that of the reference tag. However, this error keeps under 1 m which is the threshold that Mok et al. (2010) considered achievable by the system under adverse geometrical site conditions.

As regards the localisation error of fixed tags in dairy houses, Tøgersen et al. (2010) obtained a precision of approximately 0.6 m by using the Bluetooth wireless technology, Huhtala et al. (2007) obtained a precision of 1 m (for the 70% of the measures) and 2 m (for the 90% of the measures) by using a WLAN (Wireless Local Area Network). Very poor results were obtained (Huhtala et al., 2007) for tags in motion since errors were higher than 3 m. Another feasibility study (Zhou & Shi, 2009) of localisation with passive RFID tags using multilateration and Bayesian inference algorithms, which was carried out in an open field, obtained mean localisation errors of 0.19 m (with a standard deviation of 0.24 m) and 0.37 m (with a standard deviation of 0.11 m) for the two algorithms, respectively.

The comparison between these research outcomes and those obtained by the RTLS utilised in this study shows that the localisation performance of the RTLS based on UWB technology for tags in movement is comparable to that of other localisation systems for fixed tags.

Since the worst mean localisation error (tag ID 023), which was about 0.80 m, is small if compared with the average dimensions of a cow, the achieved results led to affirm that the RTLS could be used for studying some specific aspects of cow behaviour. For instance, this error would not affect the computation of some behavioural indices that do not require a high level of precision on the position of the cow, such as cow standing index and cow feeding index (Bava et al., 2012; Mattachini et al., 2011; Overton et al., 2002; Provolo & Riva, 2009). Moreover, a relatively small localisation error shows that the RTLS has the capability to give good description of the occupancy level of the different functional areas of the barn as well as that it could be utilised to track each animal of the herd with a good approximation.

Further improvements may contribute to the reduction of the mean localisation error through the application of predictive algorithms such as the 'Kalman filter', which deal with the problem to reduce the amount of noise present in a signal by comparison with an estimation of the desired noiseless signal (Kalman, 1960).

From visual analysis of the video recordings it was observed that different cow behaviours corresponded to similar values of mean error and error distribution. For instance, the tag having ID 008, which corresponded to a cow standing still in the area during the time interval considered, showed a mean error equal to that of the tag having ID 020, which was associated to a cow in movement, and a similar error distribution (Figure 5). Therefore, this would lead to affirm that RTLS performance is generally not dependent on cow behaviour as it was observed for other systems (Porto et al., 2012; Ipema et al., 2013).

The analysis of the behaviour of the two cows that were associated with the highest localisation errors (tag ID 023 and tag ID 026) showed that, especially in the second time interval (i. e., between 11h: 35m: 37s and 12h: 02m: 27s), the areas occupied by these two cows were almost equal in size and location and close to the boundaries. Therefore, a possible explanation of the higher errors obtained may be related to the low AoAs in that area that could cause lower performances of the RTLS.

However, as regards the software tool specifically developed for the visual analysis of the cows' location data in free-stall barns promising results were obtained with regard to both the performance of the implemented features and its use for the purpose of estrus' automatic identification.

The test, in fact, showed excellent performance in terms of:

Portability: it was able to work in different environments (it worked in all major browsers, including IE8). This fundamental aspect allows you to have economic advantages, as it can amortize the costs by transporting the application in different environments. Being a web application, this software provides a guarantee regarding this feature.

Real-time correctness: it ensured, at the same time, a satisfactory logical consistency of the operations carried out and an acceptable timeliness of results production. The system, considering its scope, can be called a 'soft real-time system' as it can tolerate an occasional violation of a deadline[1] causing, in the worst case, a irrelevant degradation of performance, but not the failure (Oshana, 2006). In fact, regardless of the chosen deadline value, the goal for which the system was designed does not require response times in the order of milliseconds.

Reliability: when using the system, no serious malfunctions were observed.

Robustness: the system behaved in a reasonable manner in unexpected situations which are not covered by the specifications. An example of an unexpected situation was the presence of certain records in the dataset containing the value of in the instantaneous speed field

[1]The results of the carried out processing in response to a certain event must be produced within a given time interval, said deadline.

due to a control which was not originally planned during the programming phase. The software computed the output correctly, discarding, however, this exception only in the case of CowHeatMap, while the CowSpeedGraph reported errors. The bug, however, was subsequently solved.

E ciency: resources (memory, CPU, etc.) were used in a proportionate way for the services that the software performed, i.e. no waste of resources was encoured.

Usability: The UI is very simple and user-friendly and does not require the support of a user manual.

An important feature of this software is that it allows you to perform the analysis for each animal. This property is particularly important for the cases where the behavior of a cow must be analysed separately, i.e., not in relation to the group it belongs. Martinez-Ortiz et al. (2013) calculated the speeds of approximately 190 dairy cows inspected by a video tracking system over a number of weeks. Moreover, the relative speed of a cow with respect to the group on its own is not sufficient to detect lameness; a cow may be consistently slower than the group due to old age or simply due to its own preferred pace of walking. Therefore, they proposed to detect lameness by monitoring each individual cow's speed over a number of days to look for consistent changes in mean speed.

Another important feature of this software is that, since data for all 24 hours for each day are available, you do not have the problem of lack of the data, which may contain useful information to analyse. This type of problem, for example, is often typical in cases where video tracking systems are used, since they can record only during daytime hours.

The use case carried out in this work showed, also, that a pattern related to the cow behavior analysed, i.e., when the state of estrus occurred, could be identified in both CowHeatMap and CowSpeedGraph.

In particular, if the CowHeatMaps are considered, the test showed that in general the cow occupied systematically nearly the same zones of the study area in the same time intervals analysed in each of the days chosen for the study, with the exception of the day where estrus occurred. In fact, during this day, a distribution of points in a much wider area was observed. In addition, by observing the color of these points, it was evident that the animal showed, during the day of estrus, a more sustained motor activity than that recorded in the other days. The use of CowSpeedGraphs confirmed this observation as, in the range where the cow was in estrus, instantaneous velocities above the average of those obtained on other days were recorded.

From these considerations, it could be deduced that the behavior of the cow during the week analysed took on average was ripetitive except for the day when the state of estrus manifested.

With regard to cow estrus detection, Rorie et al. (2002) studied the application of commercially available electronic estrus detection technologies to reproductive management of cattle and demonstrated that pedometers are the most applicable devices to lactating dairy cattle and have a higher accuracy and efficiency when combined with visual observation. Also Roelofs et al. (2005) adopted a pedometer for estrous detection as predictor for time ovulation in dairy cattle and demonstrated that pedometers can accurately detect estrus, though not in real time.

The collection of estrus data in real time is of considerable importance to avoid delayed inseminations of cows which could reduce cow fertilization rate and, as a consequence, have negative economic impacts on farm budget and costs. The estrus period, in fact, lasts for 11-16h on average (Firk et al., 2002). To this aim, recent research (Brehme et al., 2008; Jónsson et al., 2011) achieved excellent results by adopting real-time pedometers for estrus detection in dairy cows. However, pedometers do not allow cow localisation and cannot be used to distinguish some behavioural patterns (e.g., standing vs. feeding/perching) that are crucial data as it has been proved that the daily incidence of standing behaviour is of considerable importance in estrus detection (Firk et al., 2002) and early diagnosis of lameness (Pastell et al., 2009). The software developed in this study offers the ability to work in real time to the user by monitoring the data acquired by the RTLS updated at short time intervals. In fact, it can be launched during the execution of both the location platform and the tools for recording data in the database, and automatically takes constantly updated records from the database as input. By adding new control modules, this feature would have the advantage of being suitable to notify any inconvenience through alert messages as a result of changes in dairy cow behavior and then instantly alert the farmer. Furthermore, compared to pedometers that do not allow cow localisation, it can be used to distinguish behavioural patterns.

A further improvement about the features offered by the software could regard to the implementation of functions for the computation of some quantities useful to perform statistical operations, such as the *Maximum Speed*, *Minimum Speed*, *Average Speed*, *Maximum Acceleration*, *Minimum Acceleration*, *Average Acceleration*, *Moving Duration*, *Moving Distance*, *Moving Average Speed*, etc.

Part V

CONCLUSIONS

Chapter 9

Conclusions

The main objective of this thesis work which consisted in the evaluation of localisation and identification performances of a real-time location system based on ultra wide band technology was achieved through a number of activities. The system was analysed by performing a requirements analysis which includes functional and non-functional requirements analysis and hardware and software requirements analysis. The performances of the RTLS based on UWB technology considered in this research were assessed in operative conditions typical of the harsh environment of a free-stall barn, which was located in South-Eastern Sicily, Italy. A site survey was carried out to study the environmental features of the barn under study and the sensor layout was designed on the basis of the recommendations provided by the producer of the system. The system was then calibrated and subject to a static test to evaluate the performances in the specific breeding environment.

To perform the assessment of the RTLS, the installation of a multi-camera video recording system was carried out to acquire the panoramic top-view images of the barn area under study. Furthermore, specific software were implemented to allow building and linking of two datasets composed of the panoramic images obtained from video recordings and the tag positions measured by the RTLS, as well as allow visual recognition of tag position within the panoramic images. This software allows an interactive, accurate and automatic selection and control of the datasets as well as image visualization and labelling with the aim of the computation of the true tag positions.

Localisation and identification performances of RTLS were then computed by using suitable indices and metrics also based on an outlier data cleaning technique. On this basis, tag positions and identification errors as well as the suitable trade-off between performances were provided. Significant improvements of the results were achieved as the average localisation error reduced of about 0.046 m in dynamic conditions. Trade-off of RTLS performances yielded an average localisation error equal to about 0.52 m with a position accuracy of 98% for cows' tag and an error of about 0.11 m with a nearly 100% accuracy for the reference tag.

Further improvement of this research could regard the evaluation of the localisation error along the z direction in order to distinguish different behaviours, for instance, lying from perching with regard to the computation of cow lying index.

In addition, an automatic and real-time software tool for the visual analysis of the cows' location data acquired by the RTLS principally based on the implementation of two tools, the CowHeatMap and the CowSpeedGraph, was developed. This software uses the RTLS potentiality to provide the user with a useful tool in order to perform specific analyses, such as the estrus detection. Promising results with regard to both the performance of the implemented features and their use were obtained in a use case, when useful information related to the occurrence of the physiological state of estrus were automatically achieved. This test showed that a pattern related to the behavior of the analysed cow, when the state of estrus occurred, can be identified both in CowHeatMap and in CowSpeedGraph.

The use cases carried out in this study, however, was only one way to test the functionality of the software applied to real data to obtain significant achievements about the use of the tool in order to find useful information related to the occurrence of the estrus physiological state. It is clear that a more detailed study, carried out on a dataset composed of a greater number of data concerning a larger period and a greater number of cows, is recommended in order to ensure the scalability and the effectiveness of the software.

The results obtained in this test represent a starting point from which take inspiration for a study aiming at defining rules capable of generalize the potentiality offered by the software in multiple contexts such as, in addition to estrus detection, lameness detection, effects of cows mobility on milk production, etc.

References

Agosta, G. (2012). *De nizione di una metodologia di analisi del comportamento di bovine da latte allevate a stabulazione libera in relazione al microclima.* Unpublished doctoral dissertation, DiGeSA, University of Catania, Catania, p.141.

Álvarez, C., & Cintas, C. (2010). *Accuracy evaluation of probabilistic location methods in uwb-r d.* Unpublished master's thesis, Department of Electronic Systems. Aalborg University, Aalborg, Álvarez, C.N., Cintas, C.C., 2010. Accuracy evaluation of probabilistic location methods in UWB-RFID, Department of Electronic Systems. Aalborg University, Aalborg, p. 111.

Barbari, M., Conti, L., & Simonini, S. (2008). Spatial identification of animals in different breeding systems to monitor behavior. In *International livestock environment symposium.* (Vol. VIII, p. 1-6). Iguassu Falls, Brazil, 31 August - 4 September 2008.

Bava, L., Tamburini, A., Penati, C., Riva, E., Mattachini, G., Provolo, G., et al. (2012). Effects of feeding frequency and environmental conditions on dry matter intake, milk yield and behaviour of dairy cows milked in conventional or automatic milking systems. *Italian Journal of Animal Science*, *e42*(11).

Bell, N., Miedema, H., Blajan, I., & Mottram, T. (2013). Automatic measurement of mobility score in dairy cows using walking speed. *Computers and Electronics in Agriculture, in press.*

Berckmans, D. (2013). Editorial. In *Precision livestock farming 2013.* Leuven, Belgium.

Brehme, U., Stollberg, U., Holz, R., & Schleusener, T. (2008). Alt pedometer new sensor-aided measurement system for improvement in oestrus detection. *Computers and Electronics in Agriculture*, *62*, 73-80.

Crockford, D. (2008). *Javascript: the good parts* (First ed.; Yahoo!, Ed.). O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Demetrescu, C., Finocchi, I., & Italiano, G. F. (2004). *Algoritmi e strutture dati.* The McGraw-Hill Companies.

Di Noia, M. (2010). *Algoritmi di localizzazione in sistemi ultra wideband.*

Domain, T. (2013). *Ultra wideband r d.* Available from http://www.timedomain.com/datasheets/UWBInterference.pdf

Dygraph. (2013). *Opensource javascript dygraph library.* Retrieved 25/10/2013, from http://dygraphs.com/

EmguCV. (2013). *Emgu cv: Opencv in .net (c#, vb, c++ and more).* Retrieved September 2013, from http://www.emgu.com/wiki/index.php/Main_Page

FCC. (2002). *Before the federal communications commission washington, d.c. 20554.* Available from http://transition.fcc.gov/Bureaus/Engineering_Technology/Orders/2002/fcc02048.pdf

Feng, F. Z., J., Wang, Z., Xu, M., & Zhang, X. (2013). Development and evaluation on a rfid-based traceability system for cattle/beef quality safety in china. *Food Control*, *31*, 314-325.

Finkenzeller, K. (2003). *R d handbook: Fundamentals and applications in contactless smart cards and identi cation.* John Wiley & Sons, Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England.

Firk, R., Stamer, E., Junge, W., & Krieter, J. (2002). Automation of oestrus detection in dairy cows: A review. *Livestock Production Science*, *75*, 219-232.

Gezici, S., Tian, Z., Giannakis, G. B., Kobayashi, H., Molisch, A. F., Poor, H. V., et al. (2005). Localization via ultra-wideband radios. *IEEE Signal Processing Magazine*.

Heathcote, J. (2011). *Real-time location systems to track people and assets.* Retrieved June 2013, from https://connect.innovateuk.org/c/document_library/get_file?p_l_id=3171260&folderId=4121020&name=DLFE-34306.ppt

Huhtala, A., Suhonen, K., Mäkelä, P., Hakojärvi, M., & Ahokas, J. (2007). Evaluation of instrumentation for cow positioning and tracking indoors. *Biosystems Engineering*, *96*(3), 399 - 405. Available from http://www.sciencedirect.com/science/article/pii/S1537511006004053

Ilie-Zudor, E., Kemény, Z., Blommestein, F. van, Monostori, L., & Meulen, A. van der. (2011). A survey of applications and requirements of unique identification systems and rfid techniques. *Computers in Industry*, *62*, 227-252.

Ipema, A., Ven, T. van de, & Hogewerf, P. (2013). Validation and application of an indoor localization system for animals. In *Precision livestock farming 2013.* (p. 135-144). Leuven, Belgium.

Jónsson, R., Blanke, M., Poulsen, N., Caponetti, F., & Højsgaard, S. (2011). Oestrus detection in dairy cows from activity and lying data using online individual models. *Computers and Electronics in Agriculture*, *76*, 6-15.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME Journal of Basic Engineering*, *82*, 35-45.

Linde, H. (2006). *On aspects of indoor localization.* Unpublished master's thesis, Fakultät fur Elektro und Informationstechnik - Universität Dortmund.

Martinez-Ortiz, C., Everson, R., & Mottram, T. (2013). Video tracking of dairy cows for assessing mobility scores. In D. Berckmans & J. Vandermeulen (Eds.), *Precision livestock farming 13.* Leuven, Belgium 10-12 september 13.

Mattachini, G., Riva, E., & Provolo, G. (2011). The lying and standing activity indices of dairy cows in free-stall housing. *Applied Animal Behaviour Science*, *129*, 18-27.

Mei, H. (2003). *Modeling and performance evaluation of a bppm uwb system.* Unpublished master's thesis, Delft University of Technology.

Mok, E., Xia, L., Retscher, G., & Tian, H. (2010). A case study on the feasibility and performance of an uwb-aoa real time location system for resources management of civil construction projects. *Journal of Applied Geodesy*, *4*, 23.

Oppermann, I., Hämäläinen, M., & Iinatti, J. (2005). *Uwb: theory and applications.* John Wiley & Sons, Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO 19 8SQ, England.

Oshana, R. (2006). *Dsp software development techniques for embedded and real-time systems* (Elsevier, Ed.). 30 Corporate Drive, Suite 400, Burlington, MA 01803, USA. Linacre House, Jordan Hill, Oxford OX2 8DP, UK: Newnes.

Overton, M., Sischo, W., Temple, G., & Moore, D. (2002). Using time-lapse video photography to assess dairy cattle lying behavior in a free-stall barn. *Journal of Dairy Science*, *85*, 2407-2413.

Pastell, M., Tiusanen, J., Hakojärvi, M., & Hänninen, L. (2009). A wireless accelerometer system with wavelet analysis for assessing lameness in cattle. *Biosystems Engineering*, *104*, 545-551.

Peck, R., Olsen, C., & Devore, J. (2011). *Introduction to statistics and data analysis.* CengageBrain.com, Boston, MA 02210 USA.

Porto, S., Arcidiacono, C., Anguzza, U., & Cascone, G. (2013). A computer vision-based system for the automatic detection of lying behaviour of dairy cows in free-stall barns. *Biosystems Engineering*, *115*, 184-194.

Porto, S., Arcidiacono, C., Cascone, G., Anguzza, U., Barbari, M., & Simonini, S. (2012). Validation of an active rfid-based system to detect pigs housed in pens. *Journal of Food Agriculture & Environment*, *10*, 468-472.

Provolo, G., & Riva, E. (2009). One year study of lying and standing behaviour of dairy cows in a frestall barn in italy. *Journal of agricultural engineering*, *40*, 27-34.

Rabinovich, S. G. (2010). *Evaluating measurement accuracy.* Springerverlag New York.

Reiners, K., Hegger, A., Hessel, E., Böck, S., Wendl, G., & Weghe, H. Van den. (2009). Application of rfid technology using passive hf transponders for the individual identification of weaned piglets at the feed trough. *Computers and Electronics in Agriculture*, *68*, 178-184.

Roelofs, J., Eerdenburg, F. van, Soede, N., & Kemp, B. (2005). Pedometer readings for estrous detection and as predictor for time of ovulation in dairy cattle. *Theriogenology*, *64*(1690-1703).

Rorie, R., Bilby, T., & Lester, T. (2002). Application of electronic estrus detection technologies to reproductive management of cattle. *Theriogenology*, *57*, 137-148.

Ruiz-Garcia, L., & Lunadei, L. (2011). The role of rfid in agriculture: Applications, limitations and challenges. *Comput. Electron. Agric.*, *79*, 42-50.

Sahinoglu, Z., Gezici, S., & Guvenc, I. (2008). *Ultra-wideband positioning systems* (Vol. 2). Cambridge university press Cambridge, UK.

Samad, A., Murdeshwar, P., & Hameed, Z. (2010). High-credibility rfid-based animal data recording system suitable for small-holding rural dairy farmers. *Computers and Electronics in Agriculture*, *73*, 213-218.

Scalera, A., Conzon, D., Brizzi, P., Tomasi, R., Spirito, A., & Mertens, K. (2013). From animal monitoring to early warning systems through the internet of things. In *Precision livestock farming 2013, leuven, belgium.*

Schwartzkopf-Genswein, K., Huisma, C., & McAllister, T. (1999). Validation of a radio frequency identification system for monitoring the feeding patterns of feedlot cattle. *Livestock Production Science*, *60*, 27-31.

Shahi, A., Aryan, A., West, J. S., Haas, C. T., & Haas, R. C. (2012). Deterioration of uwb positioning during construction. *Automation in Construction*, *24*(0), 72 - 80. Available from http://www.sciencedirect.com/science/article/pii/S0926580512000246

Sommerville, I. (2010). *Software engineering* (9th ed.). Harlow, England: Addison-Wesley.

Sowell, B., Bowman, J., Branine, M., & Hubbert, M. (1998). Radio frequency technology to measure feeding behavior and health of feedlot steers. *Applied Animal Behaviour Science*, *59*, 277-284.

Steggles, P., & Gschwind, S. (2005). The ubisense smart space platform. *A Ubisense White Paper*.

Stephan, P., Heck, I., Kraus, P., & Frey, G. (2009). Evaluation of indoor positioning technologies under industrial application conditions in the smartfactorykl based on en iso 9283. In *Proceedings of the 13th ifac symposium on information control problems in manufacturing* (p. 874-879).

Tøgersen, F., Skjøth, F., Munksgaard, L., & Højsgaard, S. (2010). Wireless indoor tracking network based on kalman filters with an application to monitoring dairy cows. *Computers and Electronics in Agriculture*, *72*, 119-126.

Tullo, E., Fontana, I., & Guarino, M. (2013). Precision livestock farming: an overview of image and sound labelling. In *Precision livestock farming 2013* (p. 30-38). Leuven, Belgium.

Ubisense. (2008-2010). *Ubisense hot-to articles.* Retrieved June 2013, from http://eval.ubisense.net/howto/Contents/Contents.html

Ubisense. (2013a). *Ubisense location platform: Real-time location platform middleware and tools.* Retrieved June 2013, from http://www.ubisense.net/en/media/pdfs/factsheets_pdf/81357_ubisense_location_platform.pdf

Ubisense. (2013b). *Ubisense series 7000 compact tag fact sheet, rev 5 en120812.* Retrieved november 2013, from www.ubisense.net/en/resources

Voulodimos, A., Patrikakis, C., Sideridis, A., Ntafis, V., & Xylouri, E. (2010). A complete farm management system based on animal identification using rfid technology. *Computers and Electronics in Agriculture*, *70*, 380-388.

Ward, A. (2010). *Ultrawideband in-building location systems.* Retrieved June 2013, from http://nrc.simtech.a-star.edu.sg/rfid/slot/downloads/d401/deaf9251c_u1762.pdf

Wathes, C. M. (2010). The prospects for precision livestock farming. *Journal of the Royal Agricultural Society of England*, *171*, 26-32.

Weichert, F., Fiedler, D., Hegenberg, J., Müller, H., Prasse, C., Roidl, M., et al. (2010). Marker-based tracking in support of rfid controlled material flow systems. *Logistics Research*, *2*, 13-21.

Wied, P. (2011-2013). *Js heatmaps.* Retrieved 11/10/13 18.13, from http://www.patrick-wied.at/static/heatmapjs/index.html

Yan, J. (2010). *Algorithms for indoor positioning systems using ultra-wideband signals.* Unpublished doctoral dissertation, Delft University of Technology, Delft, The Netherlands.

Yong, X., Yinghua, L., Hongxin, Z., & Yeqiu, W. (2007). An overview of ultra-wideband technique application for medial engineering. In *Complex medical engineering, 2007* (p. 408-411). IEEE/ICME International Conference on. IEEE.

Zhou, J., & Shi, J. (2009). Rfid localization algorithms and applications - a review. *J Intell Manuf*, *20*, 695-707.