Universidade de Aveiro
2009
Departamento de Electrónica e Telecomunicações e Informática

Luís Filipe Nunes Quaresma de Oliveira

NAVEGAÇÃO EM ROBÔS MÓVEIS BASEADA EM COMUNICAÇÃO RF AD-HOC

MOBILE ROBOT NAVIGATION BASED ON AD-HOC RF COMMUNICATION

Universidade de Aveiro
2009
Departamento de Electrónica e Telecomunicações e Informática

Luís Filipe Nunes Quaresma de Oliveira

NAVEGAÇÃO EM ROBÔS MÓVEIS BASEADA EM COMUNICAÇÃO RF AD-HOC

MOBILE ROBOT NAVIGATION BASED ON AD-HOC RF COMMUNICATION

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Prof. Doutor Luís Almeida, Professor Associado da Faculdade de Engenharia da Universidade do Porto e co-orientação científica do Prof. Doutor Paulo Pedreiras, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho a todos os meus amigos e família por todo o apoio e amizade dados ao longo destes anos.

O júri

Presidente

Professor Doutor Nuno Miguel Gonçalves Borges de Carvalho
Professor Associado da Universidade de Aveiro

Vogais

Professor Doutor Paulo José Lopes Machado Portugal
Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Luís Miguel Pinho de Almeida (Orientador)
Professor Associado da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Paulo Bacelar Reis Pedreiras (Co-Orientador)
Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Agradecimentos

Aqui deixo os agradecimentos a todos os que contribuíram para o desenvolvimento desta dissertação:

Em primeiro lugar, gostaria de agradecer ao meu orientador, Prof. Doutor Luís Almeida. Apesar de se encontrar a longa distância sempre se disponibilizou a prestar o apoio necessário.

Em segundo lugar, ao meu Co-orientador, Prof. Doutor Paulo Pedreiras. Pelo constante apoio e pelas “dicas” que sempre ajudaram a resolver problemas que foram surgindo.

Gostaria de agradecer também à “malta” da 234, em especial Nuno Marujo, Luís Farinha, Rui Sancho e José Santos. Todos eles sempre disponíveis para ajudar e grandes companheiros da “hora da bucha”.

Agradeço também ao meu colega de casa Pedro Brochado por me aturar até nos dias de pior humor.

Não me posso esquecer dos companheiros de laboratório Ricardo, Alexandre, Rui e Milton pelos bons momentos passados no decorrer deste ano.

Ao meu amigo Pedro Silva pelo papel fundamental no final deste trabalho.

Aos amigos e colegas Jorge Rodrigues e Margarida Fernandes pelos fins-de-semana de descontracção proporcionados no decorrer deste ultimo ano.

A toda a minha família pelo constante apoio e incentivo que sempre me deu.

Por fim aos meus amigos que não estando tão envolvidos tão directamente neste trabalho sempre me apoiaram e não devem ser esquecidos.

A todos muito obrigado.

Palavras-chave

RSSI, localização relativa, MLE, *multidimensional scaling*, RF, MicaZ, rede de sensores sem fios.

Resumo

Actualmente a utilização de redes de sensores sem fios, com nós quer estáticos quer moveis, é cada vez mais apelativa. Desde simples aplicações de monitorização, como por exemplo parâmetros ambientais, até aplicações complexas de busca e salvamento, a localização dos vários nós da rede é fundamental. No caso de mobilidade na rede acresce ainda a necessidade de uma capacidade de navegação eficiente.

Dado o facto de que em muitas das aplicações de redes de sensores sem fios, como por exemplo operações de busca e salvamento em que o tempo de resposta tem de ser obrigatoriamente curto, é impossível fazer previamente o planeamento e a implementação de uma infra-estrutura, torna-se imprescindível a utilização de métodos de localização que não dependam de pontos conhecidos.

No âmbito desta dissertação são estudadas técnicas de localização e navegação relativas, baseadas simplesmente no sinal RF das comunicações sem fios. Relativamente à localização foram realizados testes com diferentes parâmetros relacionados com as comunicações. Estes são importantes devido à necessidade de estudar o impacto destes factores no cálculo da topologia da rede. O trabalho desenvolvido relativamente à navegação foi avaliado experimentalmente, com incidência na avaliação comparativa dos diversos métodos propostos, i.e., um método oblívio baseado em direcções aleatórias e outro baseado na técnica MLE - *Maximum Likelihood Estimator*. Apresentam-se nesta dissertação os respectivos resultados que permitem verificar o melhor desempenho em convergência para o objectivo usando MLE à custa de maior custo computacional. Em particular, foi possível fazer um robô móvel percorrer um trajecto entre dois faróis de RF, navegando apenas com informação de RSS.

Keywords

RSSI, relative localization, MLE, multidimensional scaling, RF, MicaZ, wireless sensor network.

Abstract

Nowadays the usage of wireless sensor networks, with either static or mobile nodes, has been an area of growing interest. From the simplest applications of monitoring, i.e. environmental parameters, to the most complex search and rescue applications, the localization of the various nodes of the network is fundamental. In the situation at which the network has mobility there is additionally a need of the ability to efficiently navigate.

Due to the fact that in many of the applications, i.e. search and rescue situations where the time of action is critical, is impossible to perform a previous planning and building of a framework, anchor free relative localization methods become indispensable.

In this dissertation several relative localization and navigation techniques, based only on the RF signal of the wireless communications, are studied. On the subject of localization, different parameters related with the communications were tested. These are significant because of the necessity of studying the impact of such factors in calculating the network topology. On the subject of navigation the resulting work was experimentally evaluated, with emphasis on the comparative evaluation of the several methods presented in this dissertation, namely a simple oblivious method based on random directions and another one based on MLE - Maximum Likelihood Estimator. The results show the superiority of MLE concerning the speed of getting to the target at the cost of extra computations. In particular, in the scope of this dissertation we have made a small autonomous robot move between to RF beacons, using RSS information, only.

Table of Contents

1 Introduction 1

1.1 The appeal for cooperating robot teams 1

1.2 Relative localization for coordination 2

1.3 RSSI based localization and navigation 3

1.4 Objectives 4

1.5 Dissertation structure 5

2 Related work 7

2.1 Relative localization 7

2.1.1 MDS-based relative localization 7

2.2 Navigation 10

2.2.1 Using MDS 10

2.2.2 Using an oblivious method 11

2.2.3 Using MLE 11

3 Experimental framework 15

3.1 The nodes 15

3.1.1 Wireless communications 16

3.2 The robot 19

3.3 Programming the robot 19

3.3.1 Obstacle sensors 20

3.3.2 Motors 21

3.4 System architecture 21

3.4.1 Executing complex computations 22

4 Experiments with the extended connectivity matrix using MDS 23

4.1 Implementation and Set up 24

4.1.1 Mote's side 24

4.1.2 Computer's side 25

4.2 Obtained Results 26

4.2.1 Tables of experiments results 33

4.3 Conclusions 35

4.3.1 Testing maximum RSSI 35

4.3.2 Testing the use of Adaptive-TDMA (Time Division Multiple Access) 35

4.3.3 Testing the use of different sample window sizes and sample selection algorithms 35

4.3.4 Final Considerations 36

5 Implementing the navigation strategy 37

5.1 Implementation description 37

5.1.1 The beacons 37

5.1.2 The robot 40

5.1.3 The computer 41

5.1.4 Condition of arrival at the beacon 47

5.2 Obtained Results 48

5.2.1 Oblivious method results 49

5.2.2 MLE method results 52

5.2.3 Final Considerations 55

6 Conclusions and future work 57

6.1 Conclusions 57

6.2 Future work 58

6.2.1 Improving the MLE algorithm 58

6.2.2 Multi-Robot experiment 58

6.2.3 Wide space experiment 59

7 Bibliography 61

List of Figures

Introduction **1**

Figure 1 - MOSRO MINI on Patrol[8] 2

Figure 2 - 'MOSROs' on Patrol in Shopping Mall[8] 2

Figure 3 - OFRO on Outdoor Patrol[8] 2

Figure 4 - Collision Warning Auto Brake [9] 3

Figure 5 – Pedestrian detection – illustration [9] 4

Related work **7**

Figure 6 – Connectivity Matrix (left); Units topology (right) 8

Figure 7 – Extended Connectivity Matrix (left); Units topology (right) 9

Figure 8 – MDS navigations using motion vectors [2] 10

Figure 9 – The Oblivious algorithm 11

Figure 10 – The MLE algorithm 12

Figure 11 – Estimator illustration using four samples 13

Experimental framework **15**

Figure 12 – MicaZ mote 16

Figure 13 – MDA300CA board 16

Figure 14 - MIB600 - Ethernet Gateway 16

Figure 15 – IEEE802.11 and IEEE802.15.4 channels [17] 17

Figure 16 – Piggyback Message 18

Figure 17 – Piggyback Byte 18

Figure 18 – Robot created to perform the experiments 20

Experiments with the extended connectivity matrix using MDS 23
Figure 19 – RSSI table example 23
Figure 20 – Node distribution 25
Figure 21 – MDS experiment 1 26
Figure 22 – MDS experiment 2 27
Figure 23 – MDS experiment 3 28
Figure 24 – MDS experiment 4 29
Figure 25 – MDS experiment 5 30
Figure 26 – MDS experiment 6 31
Figure 27 – MDS experiment 7 32
Implementing the navigation strategy 37
Figure 28 – The MicaZ beacons setup 38
Figure 29 – The robot setup 40
Figure 30 – The computer setup 41
Figure 31 – Single-node beacon (left); Multi-node beacon (right) 43
Figure 32 – Arrival Condition 48
Figure 33 – Oblivious method path- experiment 1 49
Figure 34 – Oblivious method RSSI received by node 2- experiment 1 49
Figure 35 – Oblivious method path- experiment 2 50
Figure 36 – Oblivious method RSSI received by node 2- experiment 2 50
Figure 37 – Oblivious method path- experiment 3 51
Figure 38 – Oblivious method RSSI received by node 2- experiment 3 51
Figure 39 – MLE method path- experiment 1 52
Figure 40 – MLE method RSSI received by node 2- experiment 1 52
Figure 41 – MLE method path- experiment 2 53
Figure 42 – MLE method RSSI received by node 2- experiment 2 53
Figure 43 – MLE method path- experiment 3 54
Figure 44 – MLE method RSSI received by node 2- experiment 3 54
Conclusions and future work 57
Bibliography 61

List of Tables

Introduction **1**

Related work **7**

Experimental framework **15**

Experiments with the extended connectivity matrix using MDS **23**

Table 1 – MDS node 0 33

Table 2 – MDS node 1 33

Table 3 – MDS node 2 33

Table 4 – MDS node 3 34

Table 5 – MDS node 4 34

Table 6 – MDS node 5 34

Implementing the navigation strategy **37**

Table 7 – Number of steps needed with oblivious 55

Table 8 – Number of steps needed with MLE 55

Conclusions and future work **57**

Bibliography **61**

List of Algorithms

Introduction **1**

Related work **7**

Experimental framework **15**

Experiments with the extended connectivity matrix using MDS **23**

Implementing the navigation strategy **37**

Algorithm 1 – New message processing by MicaZ 38

Algorithm 2 – Robot's moving algorithm 40

Algorithm 3 – Matlab code 42

Algorithm 4 – getNewData method 43

Algorithm 5 – moveDone method 44

Algorithm 6 – generateMove method with oblivious 45

Algorithm 7 – generateMove method with MLE 47

Conclusions and future work **57**

Bibliography **61**

Chapter 1

Introduction

The work is part of a larger framework involving the studies [1], [2], [3], [4] and involves the study, as well as the comparative experimental evaluation of RSSI-based relative localization and navigation algorithms.

This chapter begins with a small introduction to the subject of this dissertation pointing out the growing interest for teams of robots cooperating with each other; then it refers to relative localization; and concludes with RSSI-based navigation and localization.

1.1 The appeal for cooperating robot teams

It is said in [5] that “multiple-robot systems can accomplish tasks that no single robot can accomplish”, and that is why a cooperating team of mobile robots joining together to accomplish a common objective is an eye-catching possibility. Some of the applications they can be used in include surveillance, exploration, manufacturing, and large volume transportation. Adding to this, in situations where operator presence is impossible or involves too high risks, an autonomous solution is even more appealing.

A good example of this is a team of vacuum cleaner robots [6], each one of them being equipped with a wireless transceiver and communicating with each other. This enables them to distribute according to a pattern and quickly perform the task, covering a large area.

Adding to that and having the cost and feasibility in mind, the various robots may have different acting capabilities, as in a mine field. As mentioned in [7], some of the team elements may have detecting abilities and, while searching for mines in a formation, they

guarantee coverage. The other elements, a smaller number of them, have in turn sweeping abilities and are summoned after detection.

Figure 1 - MOSRO MINI on Patrol[8]

Figure 2 - 'MOSROs' on Patrol in Shopping Mall[8]

Figure 3 - OFRO on Outdoor Patrol[8]

1.2 Relative localization for coordination

In what localization is concerned there are two major options. The first one is to know the absolute localization; this means that there is the knowledge of an exact location, e.g. a car in a motorway that is at kilometer 12.2. The other is to know the relative localization; this means there is only the knowledge of a location based on the perception one has of the environment, e.g. a car in a motorway traveled 12.2km relatively to the starting point.

As to the first one, and focusing on robots, some of the used localization methods are to build a reference infrastructure or to use GPS (Global Positioning System). But, as it is mentioned in [2], building an infrastructure is expensive, and even impossible in an emergency situation, like search and rescue. The GPS, being satellite dependent and only providing coarse-grained positions, is not an option in every situation.

The second one, on the other hand, doesn't need any infrastructure and as such it can be used in any place rapidly and with smaller costs. Being independent from exterior means allows it to be used at any location and, using changes on the MANET (Mobile Adhoc Network) topology, to cover larger areas.

There are multiple possible applications to relative localization. Some of them are indoors or outdoors residential patrolling, airports, seaports, warehouses or even shopping mall patrolling, as mentioned in [8] and as it can be seen in Figure 1, Figure 2 and Figure 3. In those applications, it's possible to position robots with different capabilities, as mentioned by [1], so that an actuator robot, with a fire extinguisher, can approach a probe robot that can detect a fire and interact according to the needs. Upscale this and one can have a team of flying robots detecting and fighting forest fires.

Figure 4 - Collision Warning Auto Brake [9]

But the use of relative localization is not limited to robots. In addition, relative localization can be used by humans. As with FINDER [10] with which firefighters can follow a signal to another firefighter in trouble. Another example is monitoring people inside a hospital, where patients can be located in case of an emergency. Additionally, radar systems found on airplanes, or on submarines, and even a compass are means of relative localization.

Another area where relative localization can also have a useful application is Inter-Vehicle Communication. In this area, different categories of applications are defined in [11]. ATIS (advanced traveler information systems) where the targets, of communications containing traffic situation, congestion and regulation, can be selected according to their location; management of groups of vehicles, e.g. taxis; the messages exchanged between drivers; and safety systems based on information from the vehicles, like the driver's intention to accelerate or break. Some of the existing applications, which can be seen in [9], are adaptive cruise control, collision warning with auto brake, and even a prototype for pedestrian detection, as in Figure 4 and Figure 5.

1.3 RSSI based localization and navigation

One of the possible information, which can be used to relatively locate different objects, is the RSSI (Received Signal Strength Indicator) information. This is possible because RSSI is a distance dependent value, although coarse, and can be obtained by the messages

normally exchanged between the elements on a team of robots. This allows the creation of a concept of signal space distance that, though not “meter” exact, is proportional to the real distance and can be used to map a network using point-to-point measurement, as presented in [2],[3] and [12].

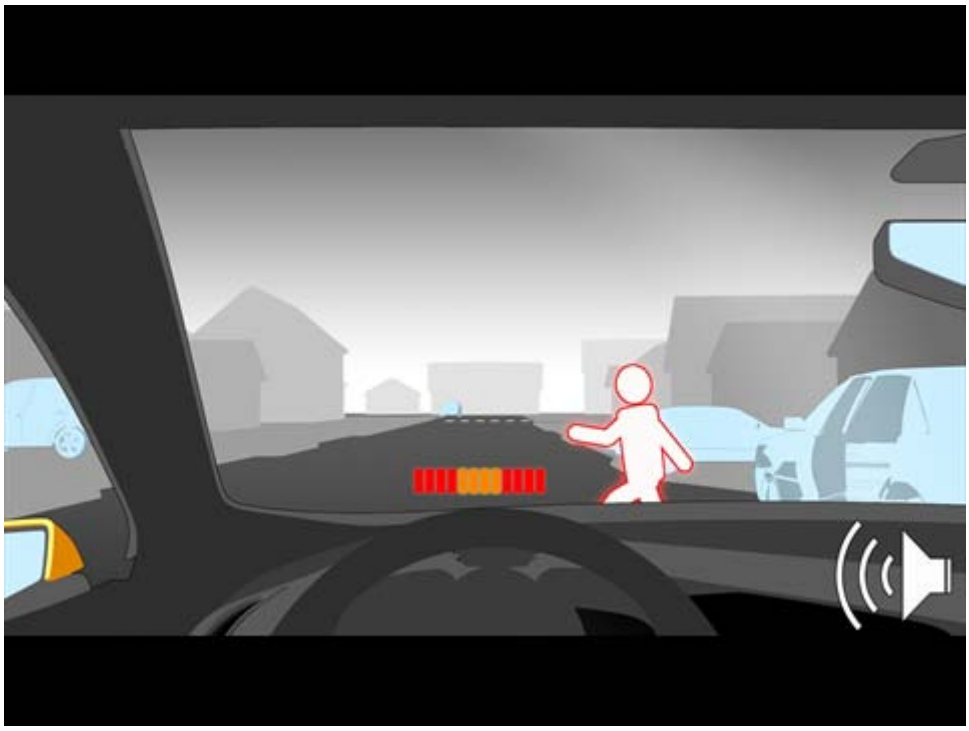


Figure 5 – Pedestrian detection – illustration [9]

This information can be used in both mobile and static robot networks to various purposes. One of them is to extrapolate the node topology of a static sensor network to optimize communications sequence, and even, with enabled auto reconfigurations, to allow optimizations on new node entries and exits. Another application in which this information can be used, this time with a team of mobile robots, is to obtain the topology of the network, [2], enabling the possibility of movement to maintain robot connectivity or to perform a given task as in [3], [4] and [1]. Finally, using a RSSI based map obtained from transceivers placed on containers in a freighter ship, can turn the difficult task of monitoring and finding a specific container into a much easier one.

1.4 Objectives

There are many different ways of getting a relative localization, among them visual recognition, distance measurement sensors using light or sound emissions and TOA (time of arrival) of messages transmitted.

The objective of this dissertation is to study, implement, and, comparatively, experiment techniques of relative localization and navigation in a multi-beacon environment, using solely the RSSI information of the RF signal of the wireless communications.

The localization itself will be carried out by making changes to different message configuration and RSSI data filtering parameters with the MDS (Multidimensional Scaling) algorithm in order to probe its capabilities. Furthermore, by guiding a robot back and forth between two beacons in an indoor environment, with obstacles, using several navigation algorithms, the navigation will be put to practice.

1.5 Dissertation structure

This dissertation is organized in six chapters. In chapter two, some of the previous work in relative localization and navigation, based on RSSI, will be discussed, starting with an algorithm to derive the topology of the network and proceeding with the description of three navigation methods.

In chapter three there is the description of the framework, i.e. nodes, robot, and computer. There, the framework will be described starting with the setup of the wireless communications, going through the various components such as the motors, and finishing on the architecture of the team.

Chapter four is about the first experiment on localization (topology derivation), using the multidimensional scaling algorithm. Tests affecting some of the wireless transmission parameters, RSSI filtering, and the MDS algorithm inputs will be presented, compared, and discussed.

Chapter five deals with the results of the experimental navigation tests evaluation. Here two methods of navigation based solely in RSSI are presented, compared, and discussed. One, the oblivious method, is a simple method of navigation while the second, the Maximum Likelihood Estimation (MLE) method, is an iterative one, being more effective but more costly in computing requirements.

Finally, chapter six presents the conclusion and future work.

Chapter 2

Related work

In this chapter, the previous work made on relative RSSI localization and navigation will be discussed. The points focused will be: first, for localization, the MDS (multidimensional scaling); then, for navigation, the MDS, an oblivious, and an iterative mode.

2.1 Relative localization

2.1.1 MDS-based relative localization

The MDS (multidimensional scaling) algorithm is a method which can be used to obtain a spatial distribution that has shown fairly good capabilities, [13] [2] [3]. This algorithm uses an n-by-n symmetrical hollow matrix, in which each element represents the distance between every two nodes. This matrix is then used to obtain a compatible configuration matrix in a p-dimensional space, for some p<n. For this work only 2 dimensions are considered.

First of all, to use this method, the node must know the state of the network. To cope with this, in [14], it's suggested the broadcast of a connectivity matrix among the units, Figure 6. This matrix alone is a representation of the topology of the network showing the connection status (1 – connected; 0 - not connected) between every pair of units. The problem here is that this connected or not connected status doesn't give information on how far or close the units are from one another.

In order to solve this problem, in [2], it's suggested the use of a new set of data much more meaningful to populate the matrix, the distance dependent RSSI values. This is called extended connectivity matrix. This solution, Figure 7, can represent not only the

status of a connection between two nodes but also whether one unit is closer or farther from another one. These values come directly from the wireless communication interface and are used according to the following expression:

$$M^k(i,j) = \begin{cases} RSSI_{j\to i} + 60, & LQI_{Received\ \ packet} \geq LQI_{threshold} \\ 0, & LQI_{Received\ \ packet} < LQI_{threshold} \end{cases}$$

As can be seen, in the previous expression, an offset of sixty is added to the RSSI reading. This is done in order to make it positive, as required by the MDS algorithm.

As stated in [1] the versatility of the extended connectivity matrix comes from the contents that, when correctly manipulated, can serve as support to mobile autonomous robot relative localization, based on ad-hoc RF communications. In [2] are presented 2D graphics showing the spatial distribution of the nodes based on the values of the extended connectivity matrix alone, values which are looked at as a signal space distance. In order to obtain these graphics, [2, 3] firstly filtered the values by using a sliding window and a kalman filter to produce smooth results. After that, the distance was calculated from the RSSI filtered data and, finally, the MDS (multidimensional scaling) algorithm was used to perform a spatial distribution. Notice that due to several reasons the RSSI values may take different values in each direction of the link.

	0	1	2	3	4	5	
	Sending nodes						
0	0	1	1	1	1	0	Receiving node 0
1	1	0	0	0	0	0	Receiving node 1
2	1	0	0	0	0	1	Receiving node 2
3	1	0	0	0	1	1	Receiving node 3
4	1	0	0	1	0	0	Receiving node 4
5	0	0	1	1	0	0	Receiving node 5

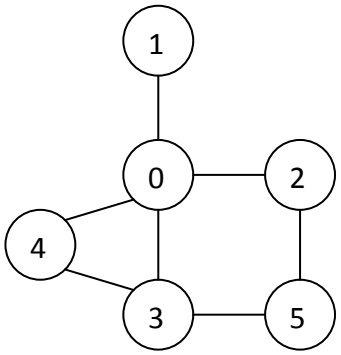


Figure 6 – Connectivity Matrix (left); Units topology (right)

Since the values change with time and some links may even disappear because of changes in the topology, which is usual in mobile robot teams, some strategy must be implemented to proceed with the maintenance of the extended connectivity matrix. The solution proposed, [2], to check if the values received are newer, is to send along with the matrix an aging-vector which creates a timeline to the received RSSI values; adding to

this, when connection from other nodes is lost the solution to update the matrix is, in each node, to check its own aging vector and, if the samples are outdated, remove the old values.

One of the drawbacks of this method is the amount of information needed to be exchanged in the network. The information in the extended connectivity matrix must be kept and broadcasted through the network. This means that for n nodes the number of values to be transmitted is n squared. But, since it is necessary, for each node, to know the topology of the entire network this information must be transmitted. So, [1] suggests to consider the matrix symmetrical so that the number of values transmitted for n nodes is only n*(n-1)/2.

	0	1	2	3	4	5	
							Sending nodes
0	0	50	50	50	20	0	RSSI values received by node 0
1	48	0	0	0	0	0	RSSI values received by node 1
2	52	0	0	0	0	60	RSSI values received by node 2
3	50	0	0	0	90	10	RSSI values received by node 3
4	25	0	0	88	0	0	RSSI values received by node 4
5	0	0	59	14	0	0	RSSI values received by node 5

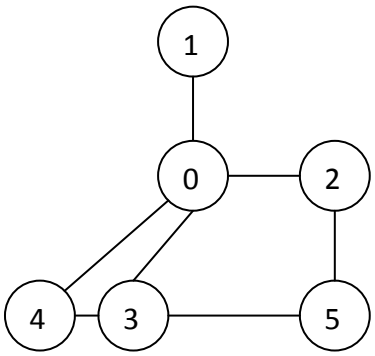


Figure 7 – Extended Connectivity Matrix (left); Units topology (right)

The issue of not fully linked networks, which are networks where one node cannot communicate directly with every other node, was explored in [2]. Due to the problems that this situation caused to the MDS algorithm, some solutions were compared. The result was that it's effective to approximate the distance between two non communicating nodes by the smaller sum of paths connecting them. What happens is that the signal space distance is calculated by using the following expression:

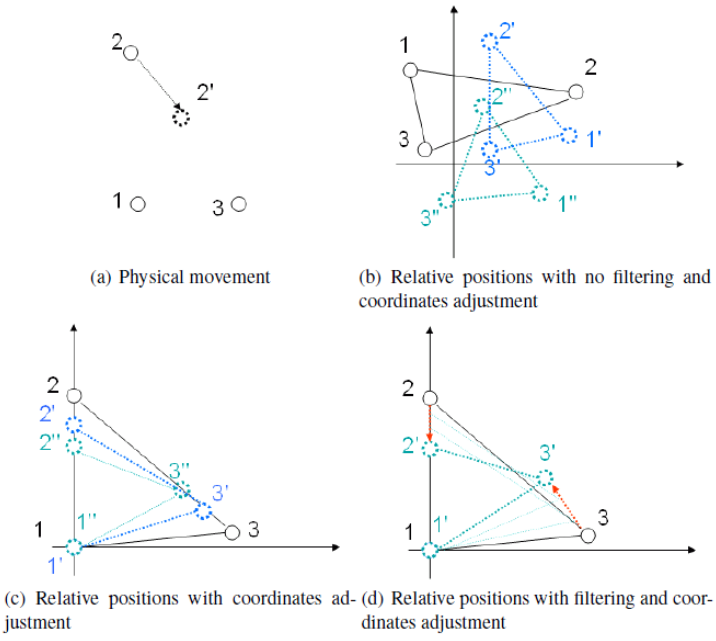
$$SignalDist(i,j) = \begin{cases} RSSI_{max} - M^k(i,j), if\ i\ is\ connected\ to\ j \\ min\left(\sum_{\forall(a,b)\in E} SignalDist\,(a,b)\right), if\ i\ is\ not\ connected\ to\ j \end{cases}$$

In the above expression, $RSSI_{max}$ represents the maximum RSSI reading that is possible to be received, E denotes a route between i and j, containing several links, and the pair (a, b) represents the extremes of a generic link in E. This solution creates a small deformation on the nodes relative positioning but this is not a problem, since signal strength space positioning is already not very accurate to physical positions.

2.2 Navigation

2.2.1 Using MDS

Resuming from MDS relative localization, on the previous topic, it is possible to go further and use the collected topology information to navigate. In [3] a method is proposed where motion vectors, which represent the movement of the network nodes, are generated, Figure 8. This solution is not as simple as it might look, at first. It must be kept in mind that a robot, initially, does not have any clue to where it is heading. So, it must make some test moves in order to deduce where it is facing, based on the topology changes visible on the motion vectors. After the robot has the notion of where it is facing, it can begin to make decisions on how to approach the objective. Due to time constraints this method wasn't tested.



(a) Physical movement

(b) Relative positions with no filtering and coordinates adjustment

(c) Relative positions with coordinates adjustment

(d) Relative positions with filtering and coordinates adjustment

Figure 8 – MDS navigations using motion vectors [2]

2.2.2 Using an oblivious method

Notice that this is quite a simple method. The algorithm basically uses the current RSSI reading and the previous one to decide on which direction to head to. This method has been used in both [3] and [4] as a possibility to be applied to a very simple robot. The decisions are merely based on three premisses: the robot approached; the robot moved away; the robot didn't either approach or move away. This algorithm is illustrated in Figure 9.

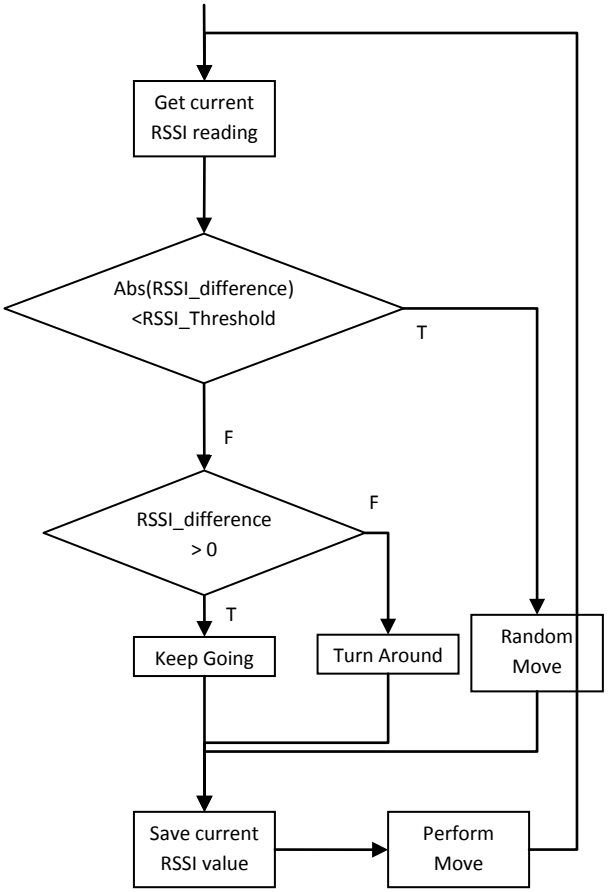


Figure 9 – The Oblivious algorithm

2.2.3 Using MLE

The last method to be discussed is MLE (maximum likelihood estimation) which, based on collected data, estimates the most likely position of the objective. This method is iterative and, as such, it requires a larger capacity in both memory and processing. For instance, since this method is based on multi-position data acquisitions to perform calculations, it is necessary to know the locations where data is acquired, relatively to each other. Therefore, a more sophisticated robot with encoders to measure movement is required.

It must be pointed out that this method has been tested in [4] both theoretically, in a Matlab simulation, and experimentally, using a moving platform. In [4] it has also been proven that by tuning some parameters this method is feasible in either noiseless or noisy environments with a faster or slower convergence velocity. For example, in a low noise environment, with a small set of data, which is basically robot positions and the respective RSSI measurements, the pursuer would quickly go to the objective. With a large set of data, however, it would take too long before even trying to approach the objective. In a high noise environment, on the other hand, a small set of data would prove insufficient and a larger number of measurements are needed to make a successful approach. For this method to step between both situations, the solution, described in [4], is to use a small initial set of data (N_{ten}), in order to start the approach quickly, and, after each approximation, add more data in a circular buffer up to a set number(N_{queue}) to reduce the error of the estimate. This algorithm is illustrated in Figure 10.

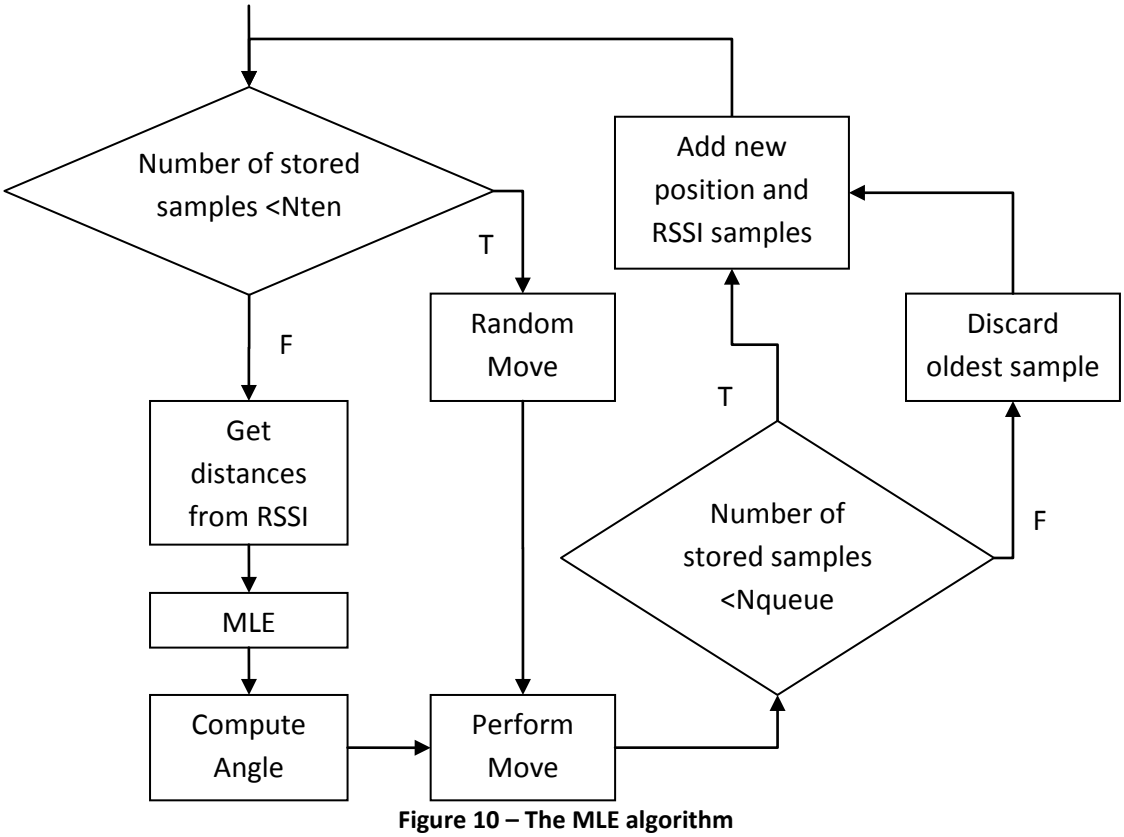


Figure 10 – The MLE algorithm

Finally, the estimator uses the least squares method to calculate the most likely position of the beacon. This method, using the sampled data held in the circular buffer, estimates the intersection point that minimizes the residual, as illustrated in Figure 11. The exact estimation method will be explained in the experimental chapter.

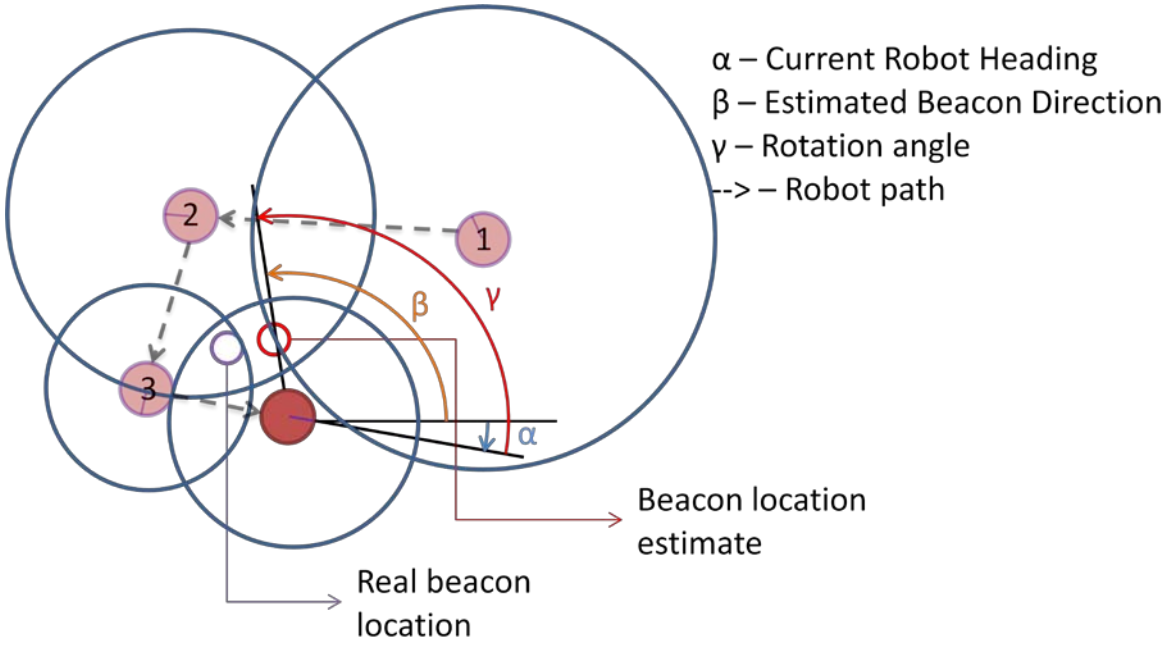


Figure 11 – Estimator illustration using four samples

Chapter 3

Experimental framework

In this chapter the framework will be discussed in what the hardware and the software used to perform the experiments are concerned.

First there is a description of the nodes used and of the wireless communication system. Further on there will be a description of the robot created to perform the experiments followed by some information about its programming. Finally the system architecture and the task distribution amongst elements will be dealt with.

3.1 The nodes

The nodes used in this work are the Crossbow's MicaZ motes (Figure 12) and, as an interface between the nodes and the robot the MDA300CA (Figure 13) expansion board [15]. The MicaZ motes are embedded systems that possess a microcontroller Atmel ATMega128L and a IEEE 802.15.4 compliant radio transceiver, CC2420 [16], which is used on the 2.4GHz band. The MIB600 - Ethernet Gateway (Figure 14) is used to both upload compiled programs to each node and function as a link between the MicaZ and the computer. The former allows communication with the computer via TCP/IP.

These nodes have been programmed in nesC, which is an extension to the C programming language made to program TinyOS – an open-source operating system made to be used in embedded wireless sensor networks. This operative system is event driven and minimizes code size trying to avoid memory constraints becoming an issue. The version of TinyOS used is 1.x.

Figure 12 – MicaZ mote

Figure 13 – MDA300CA board

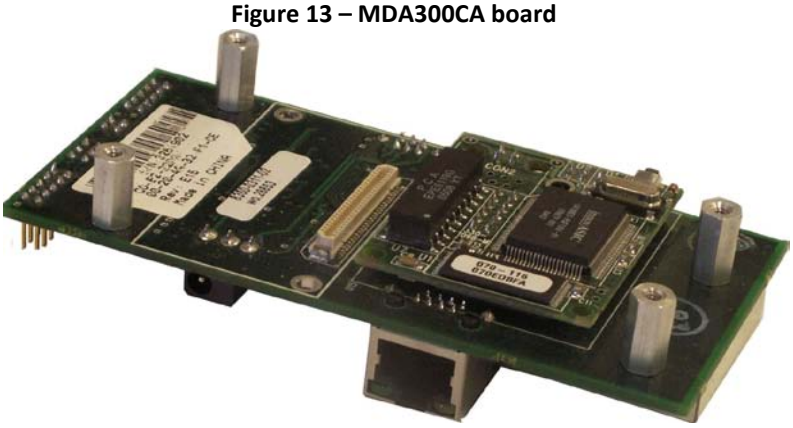


Figure 14 - MIB600 - Ethernet Gateway

3.1.1 Wireless communications

All the experiments revolve around RSSI based relative localization. Consequently the nodes used must possess the means to do so.

As mentioned before, the MicaZ motes communicate with each other using a CC2420 transceiver. They do this through an omnidirectional antenna and the information is always broadcasted so that every node within range receives information relative to all nodes.

Adding to this the CC2420 chip can give two important message parameters to the localization. The first parameter is the LQI (link quality indication), which allows knowing whether the sending node is well or barely within reach. This allows rejection of values sent from too far away, which could be fallacious. According to observations in [2] a good link presents a LQI value typically above 100. The second parameter made available by

the CC2420 is the important RSSI. The RSSI value can be obtained, according to [16], from a register in the transceiver which makes available a value between -60 and 40, corresponding to an RSSI between -100dBm and 0dBm. So to make it positive, and ease the transmission, an offset of 60 is added setting this value between 0 and 100.

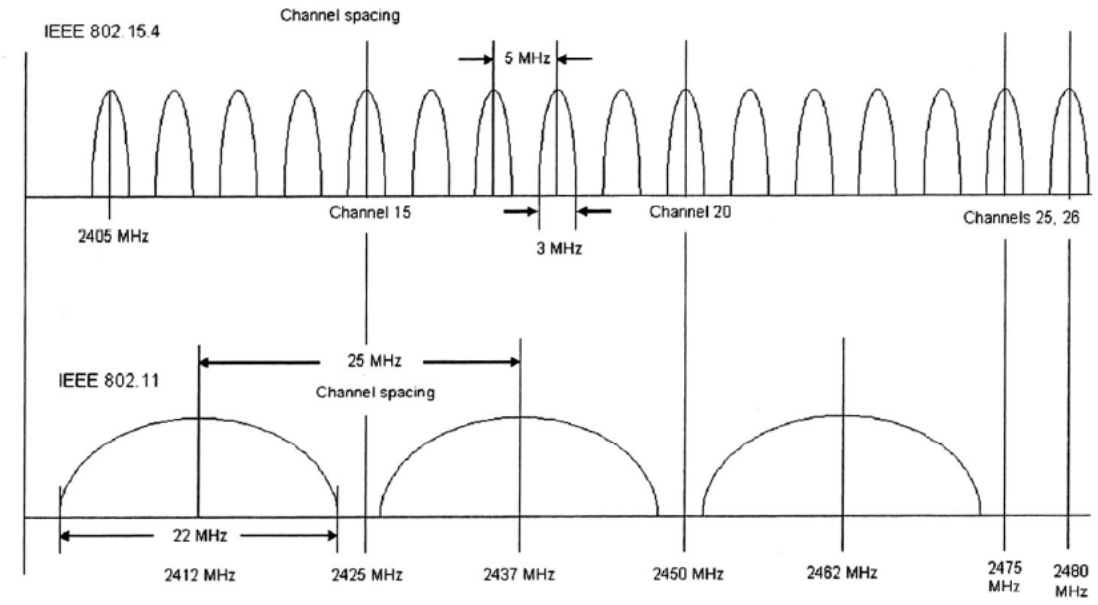


Figure 15 – IEEE802.11 and IEEE802.15.4 channels [17]

The nodes communicate using the wireless medium, obviously. This medium is heavily loaded with communications, especially in the 2.4GHz band. This happens because the IEEE 802.15.4 protocol shares the frequency spectrum with the widely used IEEE 802.11, which almost all computers, and increasingly numbers of cell phones and PDAs use. One way to prevent this is using a channel that doesn't overlap with the IEEE 802.11 protocol. This channel, as it can be seen in Figure 15, is channel 26.

At this point it must be again stressed that the amount of information to be transmitted is an important issue for the communications. For instance, the (extensive) extended connectivity matrix, the requested move to the robot (one per moving robot) and the performed move by the robot (one per moving robot), as illustrated in Figure 16 in which a piggybacked message was defined so as to transmit only the necessary information. In the piggyback byte, Figure 17, the bits inform the receiver of the contents of the message enabling it to recover that information. This allows not only transmitting multiple and variable information in the same message but also transmitting everything in one transmission optimizing the bandwidth and reducing the number of possible collisions.

Robot ID	Piggyback Byte	Extended Connectivity Matrix + Aging Vectors	Requested Moves	Performed Moves
1 Byte	1 Byte	((Number of nodes)^2+ Number of nodes) Bytes	((2+2+2)*Number of moving nodes) Bytes	((2+2+2)*Number of moving nodes) Bytes

Figure 16 – Piggyback Message

Once a message is received, the information in it must be processed. First, the receiving node gets the LQI of the message. If this value is above the threshold, the RSSI is saved. Otherwise, it will be considered 0. Secondly, the receiving node goes to the data contained on the message getting the extended connectivity matrix and the aging vector. Having this information it compares the age of the local information, saved in a local aging vector, with the age of the received information then replacing it where the first is older. Thirdly, the node checks the requested and performed moves, these contain an aging element, a forward step amount, and a rotation step amount. This information is intended to the moving nodes and, as such, it is forwarded until it reaches them. If the requested moves received are newer than the local ones, they are replaced. The same applies to the performed moves. Finally, the node, in case it has receiving either requested or performed moves checks, for each moving node, which of these needs to be sent. For example, if all the requested moves are more recent than all the performed moves, then the information on the performed moves has been spread and reached the move coordinator, which has issued a new order of movement to every moving node. As such, the information on performed moves is obsolete.

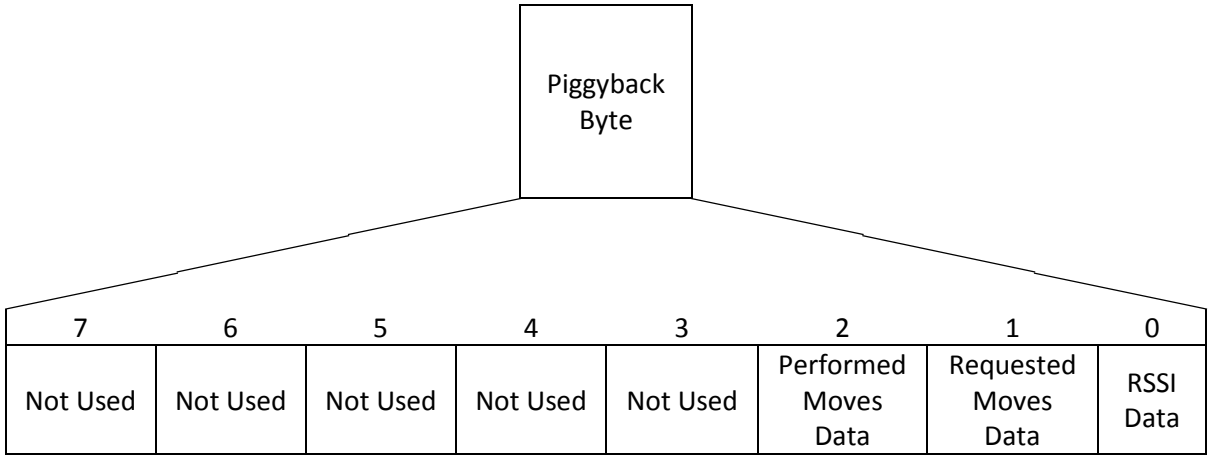


Figure 17 – Piggyback Byte

Finally, on this matter, the order and times of transmissions. To avoid collisions, which can delay and interfere with the spreading of information, an Adaptive-TDMA algorithm (Time Division Multiple Access) [18] has been used. This algorithm considers the communications failures that may easily happen with wireless communications in moving robots. To deal with it a period of communication is initially set, for instance 500ms, this time frame is then divided by the number of nodes. If they are 5, it means every node transmits every 100ms. This algorithm sets the time of the next transmission to 500ms after the last one. But, if another node is late to transmit then the remaining nodes readjust their transmitting times in order to minimize the impact of this delay and continue to avoid collisions.

3.2 The robot

In order to enable the execution of some of the tests, which require a mobile platform, a robot was created to be controlled by a MicaZ mote, as shown in Figure 18.

The robot developed for this work has two wheels attached to motors mounted on each side, both equipped with a quadrature encoder. These encoders are in turn connected to two 32bit quadrature counters LS7366 [19], which possess a spi interface used to communicate with the expansion board (MDA300CA). The motors are controlled by two PWM (pulse width modulation) applied to a current driver L293E [20].

Adding to this, a set of three sensors GP2D12 [21] was attached to the robot in order to detect and avoid obstacles, making it possible to execute the experiment in an indoor environment. As it can be seen in Figure 18, the sensors are placed outwards and forwards. The decision to place the sensors in such manner was made, based on the study presented on [22], because the width of the robot is quite large – to avoid hitting obstacles with the wheels the small area covered by the sensors had to be pointed outwards. Although the robot will get some large blind angles, which might be regarded as a disadvantage, this should not be a problem throughout the experiments.

3.3 Programming the robot

Since the robot is controlled by a MicaZ mote, it is also programmed in nesC and uses TinyOS as well. The open source community has made available software that allows programming the expansion board MDA300CA and the MicaZ motes.

The MDA300CA board has multiple ADC's available, so they allow the use of obstacle avoidance algorithms from the measure of analog obstacle sensors. Adding to this the digital I/O, that are only a few and four were needed to control the motors, made it

imperative the use of serial interfaced multi-slave quadrature counters to count the motors' shaft turns.

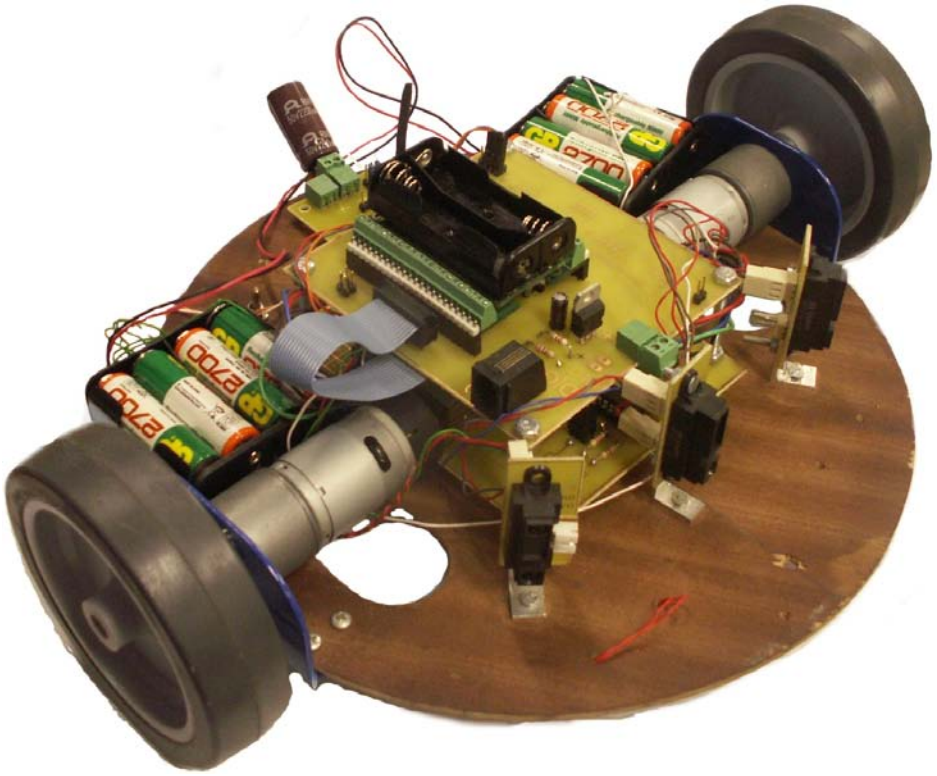


Figure 18 – Robot created to perform the experiments

3.3.1 Obstacle sensors

As mentioned above, the robot has three obstacle sensors. One is pointed forwards, one outwards to the left and the final one outwards to the right. These sensors are connected to the MDA300CA board ADC's 0, 1 and 2.

The sensors put out larger voltage values than those that the ADC's can read but, since the maximum read voltage is enough, and the safety limits mentioned in the datasheet are met, Vdd+0.5V, this is not an issue, so nothing was done to prevent it.

According to the device datasheet [21], the sensors put out a new value every 50ms but due to time constraints this is done every 100ms. The read value is then saved in a global variable and used to decide whether to stop or not to stop. That is, if the robot is going forward, it stops so that it doesn't hit the obstacle; and if the robot is rotating, it continues until it clears all obstacles.

3.3.2 Motors

The motors used in the robot are EMG30, which, as mentioned above, have built in quadrature encoders.

In order to control the velocity of the motors, a current driver L293E was used and two digital outputs per motor were the controllers. One output was meant to set the direction of movement and the other to generate a PWM to control the speed at which the robot moves.

Moreover, since some of the experiments require the knowledge of movement, the two quadrature counters feed each one a LS7366 quadrature counter, which was used as a 16bit counter monitoring the movement of each wheel. A spi communication system using five digital outputs was developed to be used with the MDA300CA board. The chip enable selects one chip or the other (this signal feeds one chip directly and the other through an inverter); the MISO (master input slave output) receives data; the MOSI (master output slave input) sends data; the clock signal; and an interruption line to signal the end of the movement.

Since the robot only closes the loop with the counters when the movement ends, and same values of PWM on different motors have different results, a simple algorithm was developed to try and match the speed of the wheels. Basically, in the end of each movement, if one wheel traveled too slowly, the respective speed is increased; if the maximum speed has been reached, the other wheel is slowed down. Finally, since the movement is never stopped immediately when the order is given, the robot doesn't travel the distance it was ordered to. To avoid this, the robot reads the difference and adjusts an offset, notice that this is not a key feature since what is important is to know how much it traveled rather than the exact distance it was ordered to.

3.4 System architecture

It is worth stressing that the set of nodes in the experiment has a low capacity with respect to processing. So, in order to both collect and easily analyze data, and to be able to perform complex processing, one of the nodes is connected to a computer. This connection, as previously referred to, is made by TCP/IP and the computer is the "brain" of all the navigation and localization on the system.

3.4.1 Executing complex computations

First of all, as stated above, the complex computations are made by a computer. This computer uses the java programming language to execute all the code and, in order to analyze data graphically, it runs on the Matlab virtual machine, which allows easy access to the data and respective graphic register.

The nodes of the network have relatively small intelligence, i.e. they only collect/spread information and, if it is a mobile node it moves were ordered to. This is made possible by making all the spread information reach the computer. One of the nodes is connected to the computer, via the MIB600 board, and every time it receives a message it sends the information through the Ethernet port. The computer then filters the information and makes all the necessary calculations in order to decide the next step to take. Finally, the information about the step is sent, again through the Ethernet port, to the node, which then spreads it throughout the network.

Chapter 4

Experiments with the extended connectivity matrix using MDS

One possible method to compute relative positions in signal strength space, using the extended connectivity matrix, is the MDS (Multidimensional Scaling) algorithm, which, by using a hollow symmetric matrix, generates one compatible space distribution. This algorithm transfers a known n-by-n matrix of dissimilarities to n points of a p-dimensional Euclidean space so that the pairwise distances between points are compatible with the dissimilarities matrix. In this work only the two first dimensions are considered.

The fact that the matrix must be symmetric creates a problem: due to communication interference, slightly different transmission power in different nodes, etc., the extended connectivity matrix is not symmetric, as shown in Figure 19.

	0	1	2	3	4	5	Sending nodes
0	0	35	22	31	31	33	RSSI values received by node 0
1	38	0	35	24	52	34	RSSI values received by node 1
2	23	35	0	29	23	42	RSSI values received by node 2
3	0	23	29	0	19	36	RSSI values received by node 3
4	31	51	24	21	0	27	RSSI values received by node 4
5	31	33	43	37	26	0	RSSI values received by node 5

Figure 19 – RSSI table example

That being said, in order to create and feed a symmetric distance matrix (distance = RSSImax – RSSI) to the MDS algorithm, we have the following options:

- Use the top triangle of the matrix;
- Use the bottom triangle of the matrix;
- Use the mean between the top and the bottom triangles of the matrix (e.g.: mean between the value 0 received from 1 and 1 received from 0);
- Use the maximum value between the top and the bottom triangles of the matrix (e.g.: maximum value between the value 0 received from 1 and 1 received from 0);
- Use the minimum value between the top and the bottom triangles of the matrix (e.g.: minimum value between the value 0 received from 1 and 1 received from 0).

Adding to this, a study of the impact various factors on the data used in the MDS algorithm is important information, such as maximum RSSI considered, communications period and synchronization, and data sampling and selection. Since there's none, to our best knowledge, in this chapter experiments concerning these issues will be displayed, compared and discussed.

4.1 Implementation and Set up

In order to analyze the previous possibilities, the following was set up.

4.1.1 Mote's side

On this side, six crossbow's MicaZ motes, which were placed according with the diagram presented in Figure 20, send a periodic beacon with the RSSI table they have and the aging vector mentioned in chapter 2. When the other motes receive this beacon, they update their RSSI table accordingly and save the information the CC2420 chip provides about the transmission (Link Quality Indicator and Received Signal Strength Indicator) for future input of their own information in the table.

All the received transmissions with a LQI inferior to a threshold, in our case 100, are disregarded; so, neither LQI nor RSSI are saved (according to CC2420 datasheet, RSSI is above -60 and below 40, so, in order to make it positive an offset of 60 is added).

Also periodically, the values on the RSSI table are first cleaned up (if too old), and new values (previously saved) are included in the table.

This information is dispatched by one node connected via MIB600 board to a PC .

Due to hardware issues, an adaption of the Adaptive-TDMA algorithm has been implemented. This adaption makes the nodes synchronize with the preceding in the order

of communication. If this synchronization is not possible, then the node will transmit one transmission cycle after it last transmitted.

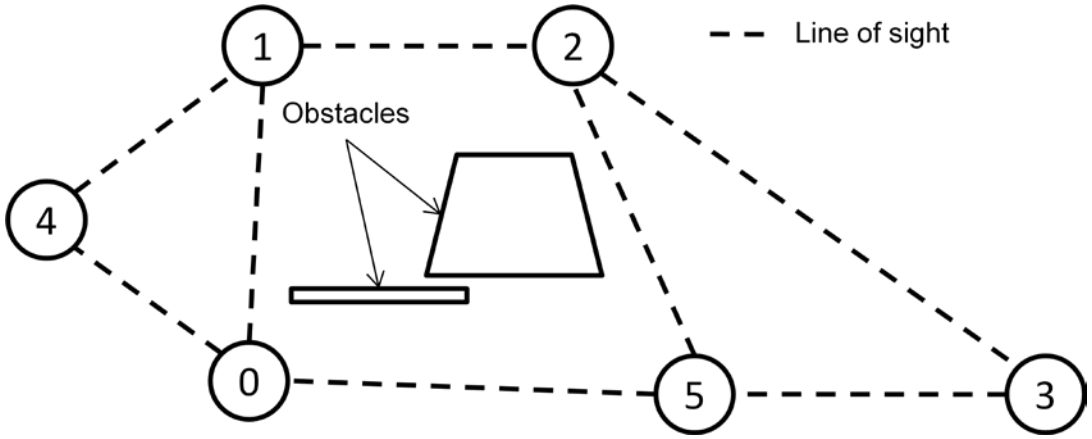


Figure 20 – Node distribution

4.1.2 Computer's side

On the computer we have MATLAB running Java code.

This program receives information (the RSSI table) from the MIB600 board via TCP/IP, and writes it on a sampling table which holds the information of some of previous tables (sampling window). I.e.:

$$RSSIwindow(t-n) = RSSIwindow\big(t-(n-1)\big), n = 1..SAMPLING\_WINDOW$$

$$RSSIwindow(t) = RSSIreading$$

Then, with this information a mean is calculated for each element in the sampling table, creating the RSSI sample. I.e.:

$$RSSIsample_t = mean(RSSIsample(t), RSSIsample(t-1), \dots)$$

This calculated data is then put through a kalman filter in order to further soften the data.

Finally, the data is used to compute the Signal Strength Space positioning with Classical MDS algorithm.

Also, in order to obtain a more perceptible view of the various outputs the nodes are moved so that node 0 is in the origin of the referential and rotated so that node 1 is in the vertical axis. If necessary, the nodes are flipped around the vertical axis so that node 2 is on the right.

4.2 Obtained Results

On the following figures, containing the experiments' outputs, the dots represent the MDS positions estimation result, and the ellipses are centered on the average of these estimations, its size representing the standard deviation, in each run (one color per run). In each run a different sub-matrix was used, top bottom, mean, maximum, and minimum, as mentioned above. The Max RSSI value it's a user-defined parameter and it represents the maximum RSSI which is expected to be received. Note that the transmission power was set to -10dBm.

Experiment 1

Parameters of the experiment:

- Max RSSI value: 250
- Adaptive-TDMA: Yes
- Message transmitting cycle: 500ms
- Number of samples in sampling window: 3
- Sample rejection: No

Figure 21 represents the result of this experiment.

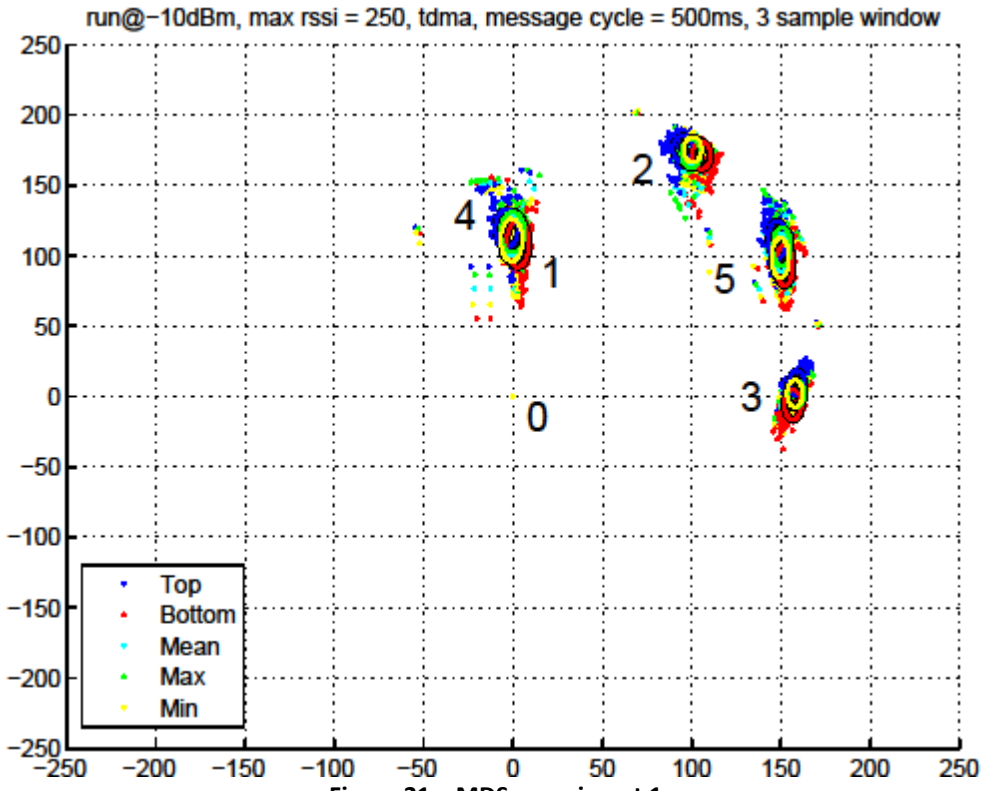


Figure 21 – MDS experiment 1

Experiment 2

Parameters of the experiment:

- Max RSSI value: 100
- Adaptive-TDMA: Yes
- Message transmitting cycle: 500ms
- Number of samples in sampling window: 3
- Sample rejection: No

Figure 22 represents the result of this experiment.

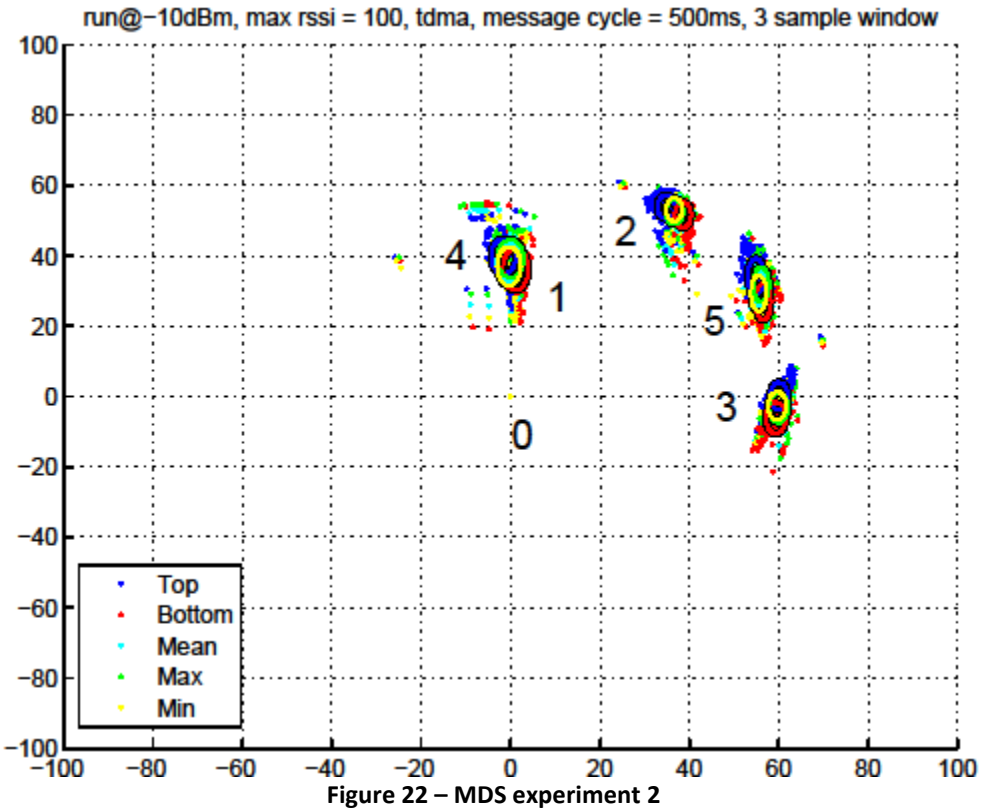


Figure 22 – MDS experiment 2

Experiment 3

Parameters of the experiment:

- Max RSSI value: 100
- Adaptive-TDMA: No
- Message transmitting cycle: 500ms
- Number of samples in sampling window: 3
- Sample rejection: No

Figure 23 represents the result of this experiment.

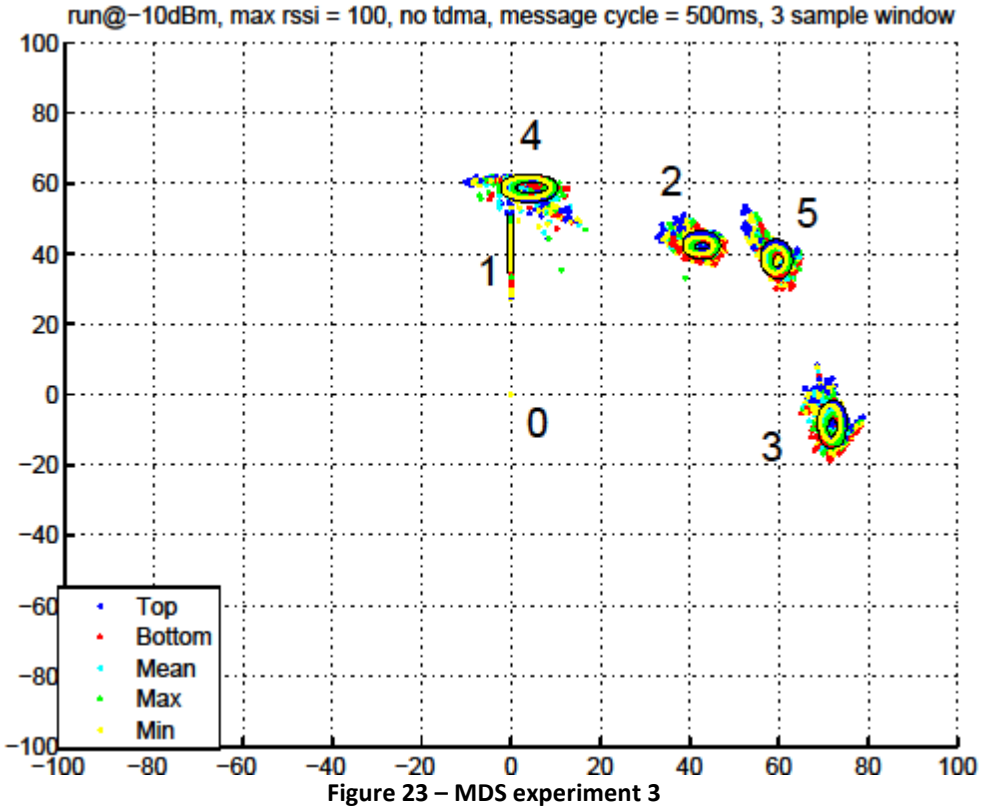


Figure 23 – MDS experiment 3

Experiment 4

Parameters of the experiment:

- Max RSSI value: 100
- Adaptive-TDMA: Yes
- Message transmitting cycle: 100ms
- Number of samples in sampling window: 3
- Sample rejection: No

Figure 24 represents the result of this experiment.

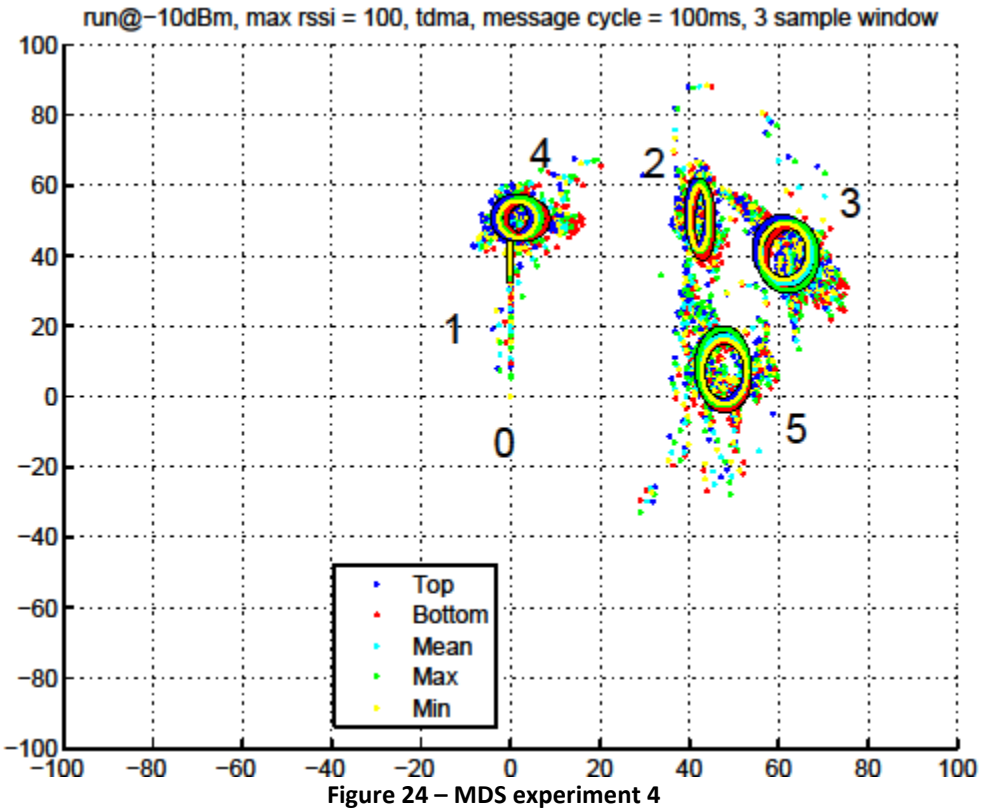


Figure 24 – MDS experiment 4

Experiment 5

Parameters of the experiment:

- Max RSSI value: 100
- Adaptive-TDMA: No
- Message transmitting cycle: 100ms
- Number of samples in sampling window: 3
- Sample rejection: No

Figure 25 represents the result of this experiment.

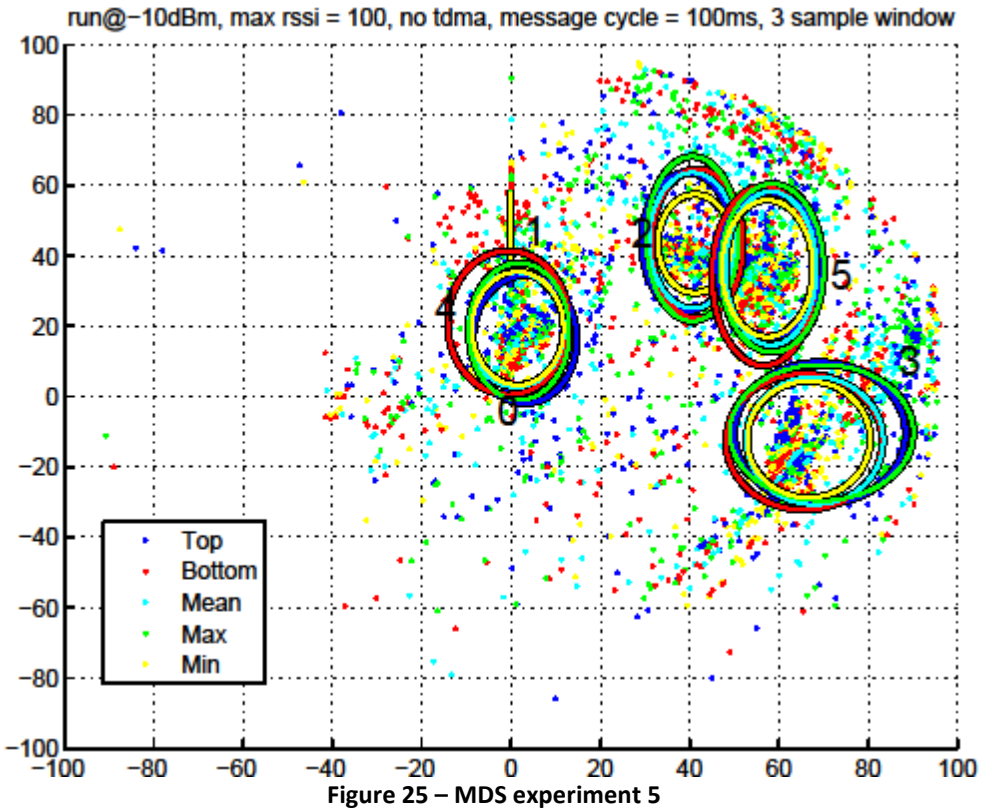


Figure 25 – MDS experiment 5

Experiment 6

Parameters of the experiment:

- Max RSSI value: 100
- Adaptive-TDMA: Yes
- Message transmitting cycle: 100ms
- Number of samples in sampling window: 5
- Sample rejection: Yes

Figure 26 represents the result of this experiment.

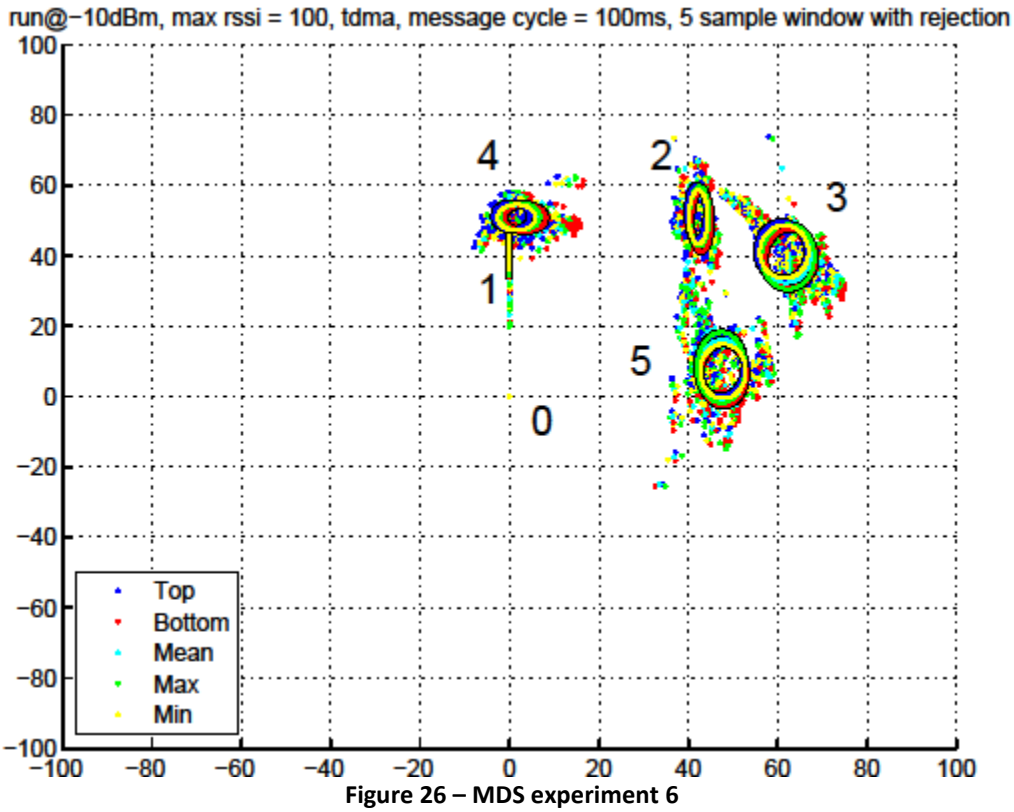


Figure 26 – MDS experiment 6

Experiment 7

Parameters of the experiment:

- Max RSSI value: 100
- Adaptive-TDMA: No
- Message transmitting cycle: 100ms
- Number of samples in sampling window: 5
- Sample rejection: Yes

Figure 27 represents the result of this experiment.

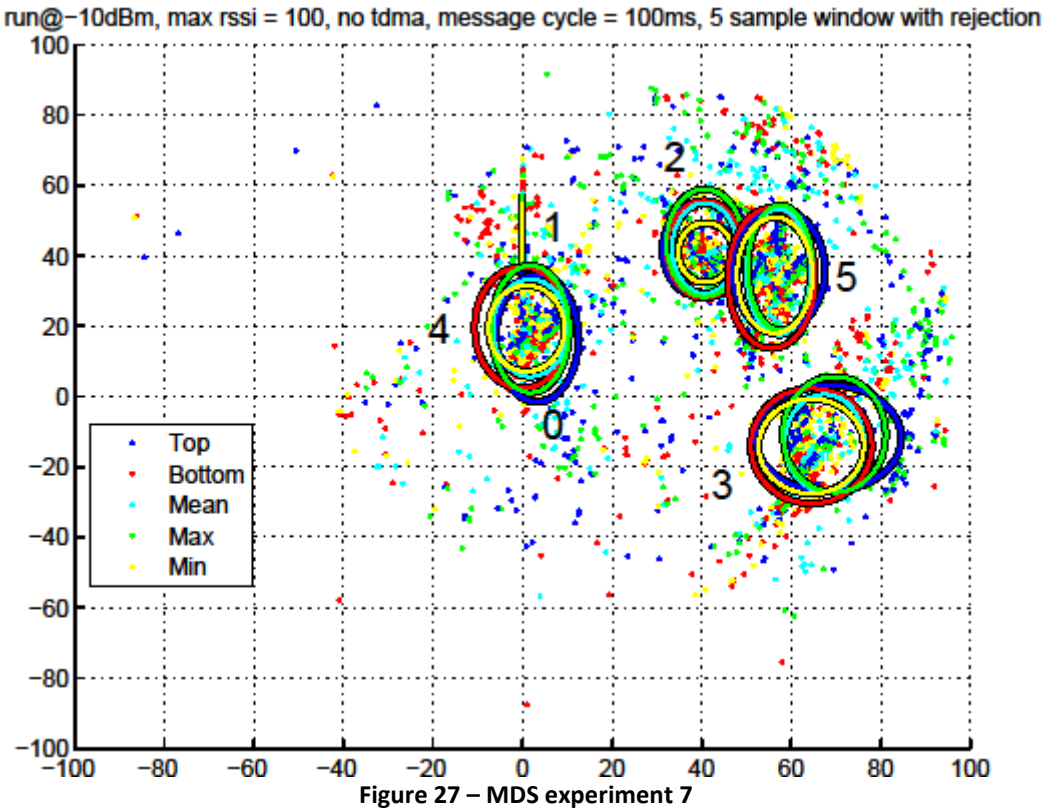


Figure 27 – MDS experiment 7

4.2.1 Tables of experiments results

Node 0

Table 1 – MDS node 0

	Mean Values (x, y)					Standard Deviation Values (x, y)				
Matrix Part	Top	Bottom	Mean	Max	Min	Top	Bottom	Mean	Max	Min
Experiment 1	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000
Experiment 2	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000
Experiment 3	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000
Experiment 4	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000
Experiment 5	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000
Experiment 6	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000
Experiment 7	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000	0.000, 0.000

Node 1

Table 2 – MDS node 1

	Mean Values (x, y)					Standard Deviation Values (x, y)				
Matrix Part	Top	Bottom	Mean	Max	Min	Top	Bottom	Mean	Max	Min
Experiment 1	0.000, 116.852	0.000, 109.876	0.000, 113.675	0.000, 114.321	0.000, 112.776	0.000, 7.369	0.000, 7.868	0.000, 7.277	0.000, 8.499	0.000, 6.585
Experiment 2	0.000, 36.961	0.000, 35.267	0.000, 36.104	0.000, 36.534	0.000, 35.799	0.000, 2.581	0.000, 2.581	0.000, 2.708	0.000, 2.902	0.000, 2.329
Experiment 3	-0.000, 44.714	-0.000, 40.941	-0.000, 42.901	-0.000, 44.034	-0.000, 41.567	0.000, 6.684	0.000, 6.503	0.000, 6.187	0.000, 6.553	0.000, 6.660
Experiment 4	0.000, 41.547	0.000, 39.108	-0.000, 40.260	0.000, 40.226	-0.000, 40.533	0.000, 6.546	0.000, 6.932	0.000, 6.687	0.000, 7.317	0.000, 6.132
Experiment 5	-0.000, 44.461	-0.000, 46.331	-0.000, 46.432	-0.000, 45.693	-0.000, 46.697	0.000, 13.089	0.000, 11.543	0.000, 11.330	0.000, 12.659	0.000, 10.968
Experiment 6	-0.000, 41.953	0.000, 39.374	-0.000, 40.560	-0.000, 40.507	-0.000, 40.840	0.000, 5.497	0.000, 5.987	0.000, 5.691	0.000, 6.260	0.000, 5.170
Experiment 7	0.000, 45.094	-0.000, 45.703	-0.000, 46.996	-0.000, 46.600	-0.000, 46.885	0.000, 11.840	0.000, 9.521	0.000, 8.849	0.000, 10.580	0.000, 8.096

Node 2

Table 3 – MDS node 2

	Mean Values (x, y)					Standard Deviation Values (x, y)				
Matrix Part	Top	Bottom	Mean	Max	Min	Top	Bottom	Mean	Max	Min
Experiment 1	95.089, 175.918	105.452, 171.218	100.217, 173.975	99.776, 172.490	100.555, 175.399	5.629, 9.440	5.126, 9.619	5.157, 9.339	5.505, 9.934	4.954, 9.001
Experiment 2	34.860, 53.845	38.583, 51.562	36.701, 52.759	36.716, 52.465	36.668, 53.132	2.029, 3.343	1.814, 3.595	1.830, 3.538	1.961, 3.570	1.801, 3.299
Experiment 3	42.613, 43.011	42.872, 41.561	42.783, 42.278	43.242, 42.230	42.200, 42.260	3.492, 2.602	2.793, 2.375	2.912, 2.305	2.986, 2.260	3.134, 2.505
Experiment 4	41.839, 52.034	42.907, 49.507	42.374, 50.633	42.473, 50.969	42.236, 50.725	2.202, 9.222	2.266, 10.002	2.198, 9.554	2.331, 9.936	2.151, 9.081
Experiment 5	39.441, 43.943	41.753, 43.408	40.680, 44.346	40.640, 44.767	41.186, 43.401	10.169, 19.732	10.270, 21.213	9.659, 19.314	10.335, 23.399	8.248, 14.027
Experiment 6	41.902, 52.307	42.917, 49.748	42.383, 50.939	42.523, 51.150	42.248, 50.952	2.130, 7.787	2.192, 8.585	2.156, 8.204	2.229, 8.563	2.117, 7.688
Experiment 7	39.813, 41.718	40.589, 41.796	40.707, 42.099	40.723, 43.601	40.886, 40.884	8.551, 13.449	8.451, 13.868	7.268, 12.864	8.608, 15.307	6.056, 8.462

Node 3

Table 4 – MDS node 3

	Mean Values (x, y)					Standard Deviation Values (x, y)				
Matrix Part	Top	Bottom	Mean	Max	Min	Top	Bottom	Mean	Max	Min
Experiment 1	158.930, 9.565	156.871, -6.073	158.102, 2.045	158.389, 2.072	157.845, 2.176	4.410, 9.049	4.821, 10.523	4.549, 9.541	4.724, 9.680	4.406, 9.694
Experiment 2	60.088, 0.050	59.372, -6.104	59.816, -2.980	59.986, -3.122	59.653, -2.709	1.956, 3.967	2.261, 4.461	2.054, 4.147	2.181, 4.320	2.004, 4.020
Experiment 3	72.268, -8.004	71.662, -9.968	71.979, -8.968	72.357, -9.792	71.613, -8.165	2.346, 5.344	2.192, 4.331	2.197, 4.488	2.081, 4.279	2.296, 5.056
Experiment 4	60.759, 41.762	62.741, 39.428	61.771, 40.678	62.136, 40.016	61.353, 41.175	5.966, 8.997	5.686, 8.889	5.731, 8.840	6.366, 9.910	5.432, 8.003
Experiment 5	69.242, -10.504	65.676, -12.918	68.061, -12.614	69.585, -10.297	66.669, -12.230	19.150, 20.271	17.509, 19.350	15.348, 18.184	20.549, 19.389	13.755, 16.572
Experiment 6	60.938, 41.351	62.894, 39.266	61.873, 40.520	62.342, 39.680	61.477, 41.031	5.611, 8.228	5.610, 8.326	5.612, 8.181	5.882, 9.178	5.455, 7.542
Experiment 7	68.283, -11.816	64.494, -14.079	67.470, -13.211	69.657, -10.815	64.510, -14.131	16.302, 15.142	13.505, 16.388	8.854, 13.825	11.488, 16.504	11.759, 13.440

Node 4

Table 5 – MDS node 4

	Mean Values (x, y)					Standard Deviation Values (x, y)				
Matrix Part	Top	Bottom	Mean	Max	Min	Top	Bottom	Mean	Max	Min
Experiment 1	-3.701, 119.261	2.601, 108.350	-0.315, 114.023	-0.085, 116.728	-0.814, 111.176	6.328, 13.555	6.399, 16.247	6.283, 14.606	6.765, 14.470	6.086, 15.207
Experiment 2	-1.557, 39.848	1.094, 36.152	-0.226, 38.234	-0.017, 39.190	-0.355, 37.034	2.813, 4.969	2.845, 6.011	2.848, 5.470	2.976, 5.352	2.692, 5.522
Experiment 3	4.203, 58.375	4.496, 59.006	4.465, 58.797	5.037, 58.741	3.631, 58.764	5.387, 2.943	4.714, 2.687	4.677, 2.695	4.750, 2.743	5.291, 2.955
Experiment 4	0.459, 50.838	3.497, 50.373	1.962, 50.563	2.412, 50.792	1.512, 50.402	4.173, 5.564	4.706, 5.163	4.392, 5.344	4.600, 5.663	4.221, 5.151
Experiment 5	3.108, 16.169	-0.551, 20.964	1.160, 18.952	1.809, 18.440	1.523, 19.280	11.600, 18.173	13.630, 20.438	10.752, 16.307	11.265, 19.456	10.264, 16.104
Experiment 6	0.497, 51.158	3.529, 50.718	1.924, 50.870	2.428, 51.026	1.573, 50.847	4.066, 3.812	4.619, 3.800	4.251, 3.696	4.497, 3.950	4.229, 3.705
Experiment 7	3.276, 16.483	-0.112, 19.516	1.810, 19.429	1.740, 19.153	0.928, 19.345	9.207, 18.100	10.568, 17.280	8.443, 13.852	8.744, 18.081	8.514, 12.221

Node 5

Table 6 – MDS node 5

	Mean Values (x, y)					Standard Deviation Values (x, y)				
Matrix Part	Top	Bottom	Mean	Max	Min	Top	Bottom	Mean	Max	Min
Experiment 1	147.999, 111.239	152.073, 93.247	150.260, 102.481	150.617, 106.293	149.721, 98.406	5.187, 13.685	4.896, 14.516	4.902, 13.744	5.090, 13.811	4.846, 14.270
Experiment 2	54.930, 34.104	56.539, 26.837	55.822, 30.636	56.128, 31.810	55.529, 29.325	1.881, 5.066	1.766, 5.139	1.770, 4.933	1.818, 4.963	1.757, 5.101
Experiment 3	59.394, 39.378	59.865, 37.128	59.706, 38.180	60.119, 38.247	59.160, 38.323	2.754, 3.836	2.148, 3.384	2.285, 3.353	2.267, 3.403	2.511, 3.724
Experiment 4	47.395, 8.802	47.997, 6.019	47.689, 7.225	47.540, 8.170	47.891, 6.957	5.449, 10.115	5.298, 9.788	5.347, 9.987	5.606, 10.971	5.056, 9.155
Experiment 5	57.887, 37.436	56.636, 34.243	58.265, 36.427	58.480, 36.479	57.464, 36.363	11.775, 20.548	11.542, 25.529	10.239, 21.819	11.570, 23.888	10.200, 19.467
Experiment 6	47.245, 8.883	47.936, 6.159	47.610, 7.309	47.394, 8.155	47.805, 7.135	5.276, 8.897	5.119, 8.786	5.215, 8.798	5.365, 10.061	5.071, 7.724
Experiment 7	57.167, 35.897	55.645, 33.589	57.176, 35.446	57.595, 36.504	56.364, 34.408	10.360, 17.279	9.792, 19.971	7.148, 17.875	7.262, 17.806	8.451, 16.483

4.3 Conclusions

4.3.1 Testing maximum RSSI

On this matter, the maximum RSSI value, which is the base of the transformation from RSSI to signal space distance (distance = RSSImax – RSSI), is tested with a big value and, based on the CC2420 datasheet, with a more realistic approach.

The graphic disposition of the nodes is clearly alike in experiments one and two, but, the standard deviation (STD) is larger in the first. However, this is not significant since the ratio is about the same.

This is the expected outcome, since the values of RSSI and the filtering was the same, so the distribution should be the same as well.

4.3.2 Testing the use of Adaptive-TDMA (Time Division Multiple Access)

On this matter, Adaptive-TDMA is turned on and off and the results are compared. Furthermore, two values of message transmitting cycle are compared.

As can be easily seen, from experiments two and three vs. four and five, the message cycle 500ms produces the best results. This is usually due to the mean occupancy. But, since the messages transmitted are relatively short (6*6 byte matrix and 6*1 byte aging vector plus message header and tail), at a transmission rate of 250Kbps the time of 6 messages is in the order of 10-20 millisecond, much smaller than the window of 100ms. So, in this case, the results were not fully expected. At any rate, the larger transmitting window produces the best results.

As far as the Adaptive-TDMA influence in the results is concerned, it can be seen that in the 500ms case the difference is not even close to significant. But, if the 100ms case is considered, the Adaptive-TDMA experiment produces results with a much smaller STD than the no Adaptive-TDMA. Once again, we can infer that the organization of the communications produces less noise on the environment creating better RSSI readings.

Notice that with a 500ms window the results come with less scattering but the system dynamic is slower than with the 100ms.

4.3.3 Testing the use of different sample window sizes and sample selection algorithms

On this matter, two different cases have been studied: a sampling window of 3, where all non-zero samples are taken into account, and a sample window of 5, where some

samples are rejected. The rejection rule is: If only one non-zero value exists, it is used; if two exist, the smaller is rejected; if more than two exist, the highest and the lowest are rejected and the others are taken into account.

In experiments six and seven, these rules were put to the test. First, by using Adaptive-TDMA and, in the latter, not using Adaptive-TDMA. In the first, and comparing it with experiment four, the results were very improved, making these values usable in the algorithm. On the other hand, comparing experiments five and seven the scattering still exists in approximately the same scale.

4.3.4 Final Considerations

In the end, neither of the different approaches, each using different parts of the extended connectivity matrix, shows an improvement to another. This means that in order to use MDS the extended connectivity matrix can be considered symmetrical and, as such, the number of values transmitted can be reduced.

On the other hand, the factors tested (Max RSSI considered, communications cycle duration and synchronization, and data sampling and selection) are very significant to the results. After these experiments, it is possible to conclude that when accurate results are required, a large window of transmission is a good option. Also, when fast dynamics are required, a smaller window, a synchronizing algorithm, a sample window, and rejection algorithm should be used in order to reduce scattering.

Chapter 5

Implementing the navigation strategy

To perform the navigation tests with the robot two methods were tested. The first one is the oblivious method that, as already described, is a very simple method that only needs to know the current and the previous RSSI reading. The second one is the MLE (maximum likelihood estimation) method, that, as previously described, is an iterative method that needs to know the steps taken by the robot and, in addition, a larger set of RSSI readings than the oblivious one. In spite of different implementations, the algorithm is very similar, being the only difference the move decision made.

In this chapter, the setup, the implementation, and the experiment are explained and the experimental results are shown and commented.

5.1 Implementation description

This implementation has three important elements which will be described below – the beacons, the robot and the computer.

5.1.1 The beacons

To begin with, crossbow's MicaZ motes, Figure 28, were placed as two beacons. Each beacon is actually a set of 3 nodes separated by 5cm. This was done as to reduce the impact of RSSI noise, as suggested by [4]. These nodes, which have one antenna each, are properly synchronized and their purpose is to emulate a single node with three antennas. The synchronization is made in the following way. Each beacon has a master node, that triggers the beacon transmission, and two slaves. It's just the masters of each beacon that synchronize between them using the Adaptive-TDMA method. The slaves are synchronized in each beacon by the respective master, transmitting after a short interval.

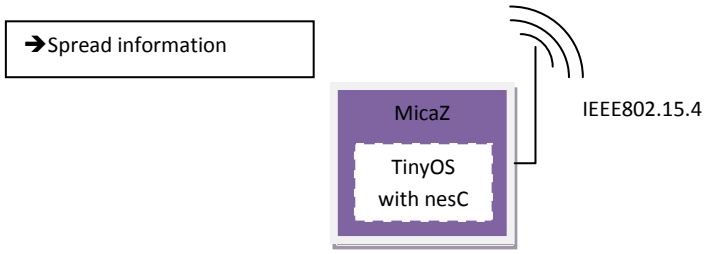


Figure 28 – The MicaZ beacons setup

In each node, a periodic beacon containing the RSSI table, an aging vector and, if needed, requested moves and performed moves, is sent with a period of 500ms. When the other motes receive this beacon, they update their RSSI table accordingly and save the information the CC2420 chip provides about the transmission (Link Quality Indicator and Received Signal Strength Indicator) for future input of their own information in the table. Additionally the received transmissions with a LQI inferior to a threshold, in our case 100, are disregarded; so, RSSI is not saved (according to CC2420 datasheet, RSSI is above -60 and below 40, so, in order to make it positive, for a simpler transmission, an offset of 60 is added). This is shown in Algorithm 1.

Also periodically, the values on the RSSI table are first cleaned up (if too old), and new values (previously saved) are included in the table.

Finally once the node connected to the computer receives a message, it dispatches the information to the computer via the MIB600 board.

The following experiments were all done with the transmission power set to -19.17dBm.

Algorithm 1 – New message processing by MicaZ

```
/* TDMA */
if(source is the master of this group)
        set_send_time;
else
        if(source is another master and  i am a master)
                resync;
        endif
endif


/* Get RSSI data from message */
if(receivedLQI>LQI_THRESHOLD)
```

```
        RSSI=getRSSI;
else
        RSSI=0;
endif
LQI=receivedLQI;

/* Get RSSI table */
if(piggyback has RSSI table)
        get_data_from_message;
        for i=other_nodes
                if(received_table_age[i] is newer)
                        replace_local_table[i];
                        replace_local_age[i];
                endif
        endfor
endif

/* Get requested moves */
if(piggyback has requested moves)
        if(received_requested_move[i].step is newer)
                get_requested_move[i];
                if(is for me) start_moving;
        endif
endif

/* Get perdformed moves */
if(piggyback has performed moves)
        if(received_performed_move[i].step is newer)
                get_performed_move[i];
        endif
endif

/* Check and set send moves status */
if(performed_moves_steps >= requested_moves_steps)
        send_only_performed_move;
else
        send_only_requested_move;
endif

/* Send data to computer */
if(connected to computer)
        send_data_to_computer;
endif
```

5.1.2 The robot

To begin with, since these experiments include a moving robot controlled by a MicaZ mote, as seen in Figure 29, the above explanation is still valid. The only difference resides in the fact that only one node, and not a set of nodes, is present on the robot.

Algorithm 2 – Robot's moving algorithm

```
/* Move Robot */
if(robotState is IDLE)
        robotState = ROTATING;
        reset_counters;
        rotate;
else
        if(robotState is ROTATING)
                go_forward;
        else
                robotState = IDLE;
        endif
endif
```

Adding to the previous, there is the task of putting the robot in motion. This task has been simplified by making the moves very simple: rotate and move forward, Algorithm 2; but it still has a problem associated the existence of obstacles in the way of the robot. To cope with this nuisance the obstacle sensors were embedded in the robot and their usage is also very simple, as described below.

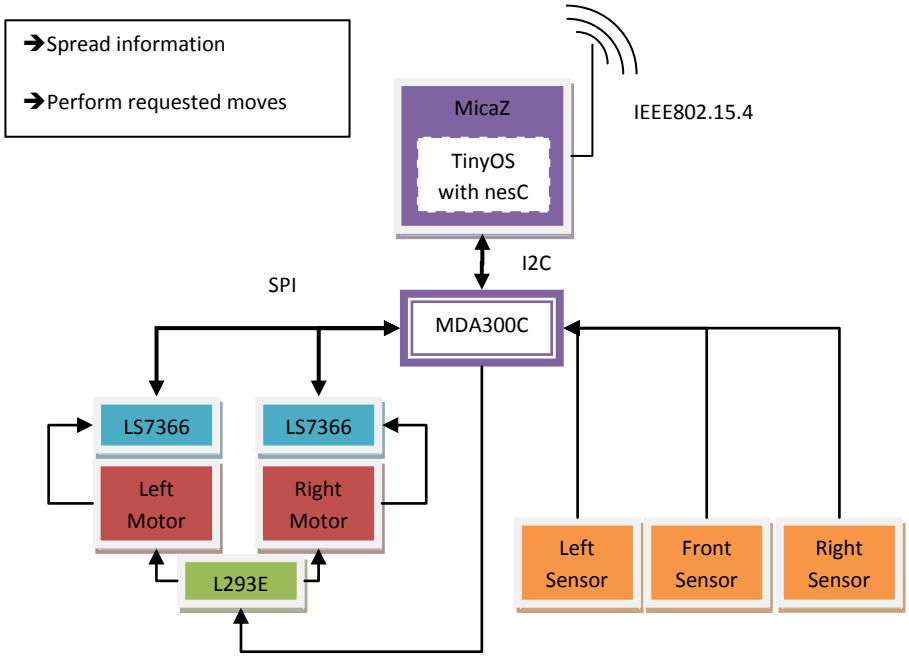


Figure 29 – The robot setup

Initially, the robot receives a message containing the orders issued by the computer (step sequence number, forward distance, and spin angle) and, if the sequence number on the message is greater than the previous one it follows those instructions (see Algorithm 1). First, the robot starts to rotate and when the desired angle has been reached it stops. If an obstacle is in front of the robot the forward movement will be impossible. So, in order to avoid this situation, the robot, continues to rotate until the way is clear. After that, the forward move is performed. In this case, the solution is even simpler. If an obstacle is too close, it stops.

Once this is done the robot didn't follow exactly the instructions issued by the computer. So, in order to have a correct assessment of the robots' movement on the computer, a message containing the values of spin angle and forward distance is sent back to the computer through the network.

5.1.3 The computer

The computer setup is as shown in Figure 30 and its functions go from filter the RSSI data to control the progress of the robot throughout the experiments. The program runs in Matlab, Algorithm 3, and calls Java methods.

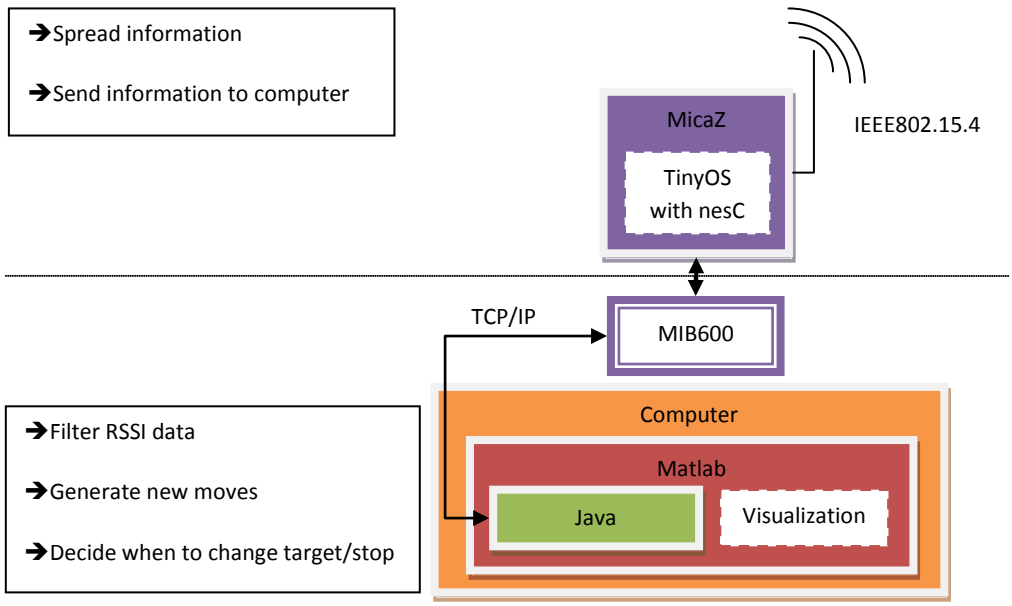


Figure 30 – The computer setup

RSSI data Processing

Note that up to this point the emulation of a three antenna node is not concluded since the received information is still from 7 different nodes. So, in order to finalize this emulation, the mean of the several non-zero RSSI values, that integrate a set of nodes in the extended connectivity matrix, is calculated like it is shown in Figure 31.

Finally, to conclude the RSSI data acquisition, Algorithm 4, each time this program receives the extended connectivity matrix, from the MIB600 board via TCP/IP, it writes it on a sampling table, which holds the information of three previous matrixes (sampling window). Then, this information is used to calculate a mean for each element in the sampling table creating the RSSI sample.

Algorithm 3 – Matlab code

```
RunStep=1;
currentBeacon=0;
generateMove(RunStep, currentBeacon);
while(RunStep<500)
   requested_move_done=getNewData(); % Includes Filtering
   if(trequested_move_done)
      RunStep=RunStep+1
      moveDone(currentBeacon);
      generateMove(RunStep, currentBeacon);

          {...get log...}

      if(RSSI_reading>35)
         counter=counter+1

         if(counter==3)
            if(theend)
               break;
            endif

            counter=0;

            currentBeacon=currentBeacon+1

            currentBeacon=mod(currentBeacon,2);
            if(currentBeacon == last_beacon)
               theend=true;
            endif
         endif
      elseif(RSSI(movingCluster+1,currentBeacon+1,RunStep)<30)
         counter=0;
      endif
   endif
endwhile
```

Movement Planning

Adding to the extended connectivity matrix, the information about the performed moves also arrives to the computer. This information, if newer than the previous one, is saved and a set of operations is triggered.

	0	1	2	3	4	5	6
0	0	35	22	31	31	33	25
1	38	0	35	24	52	34	24
2	23	35	0	29	23	42	44
3	0	23	29	0	19	36	12
4	31	51	24	21	0	27	11
5	31	33	43	37	26	0	23
6	26	24	43	13	11	23	0

	0	1	2
0	0	33,2(2)	31
1	33,125	0	15.6(6)
2	31	15.6(6)	0

Figure 31 – Single-node beacon (left); Multi-node beacon (right)

First, since this means the previous move is done, the information on the new position and angle is calculated and saved (note that this information is only important in the MLE method as the oblivious does not need to know the position or angle). The current RSSI data is also saved associated with the position, Algorithm 5.

Once this is done, there is finally time to perform the new move. This will be done in different manners, depending on the method in use.

Algorithm 4 – getNewData method

```
Outputs: newPerformedMoveArrived

data=readDataFromMIB();
if(piggyback has RSSI table)
        receivedRSSI=data.RSSI;
endif
```

```
/* No requested moves return to computer*/

/* Get perdformed moves */
if(piggyback has performed moves)
        performedMoves=data.performedMoves;
endif

RSSI = calculate_multi_node_beacon_RSSI(receivedRSSI);
Samples[SampleCounter] = RSSI;

SampleCounter++;
SampleCounter%=SAMPLE_SIZE; //SAMPLE_SIZE=3
SampleRSSI=mean(Samples);

if(new_performed_move)
        return true;
else
        return false;
endif
```

Movement with the oblivious method

In the beginning of the run, the robot goes to a random direction. If the difference between the RSSI reading in this new location and the reading in the previous location proves to be positively greater than a set threshold, then the robot proceeds in the same direction, since it means the robot is approaching the beacon. If, on the other hand, the value is negatively greater than the threshold, the robot turns around, since the beacon is further. Finally, if the threshold is not met, the robot rotates randomly and proceeds. Once the computer decides what to do, it sends the information to the robot, which will perform the step. This is shown in Algorithm 6.

Algorithm 5 – moveDone method

```
Inputs: currentDestination

        calculate_new_angle_from_performed move;
        calculate_new_position from performed move;
        writeLog();
```

Movement with the MLE method

To begin with, the MLE method, Algorithm 7, needs some information to start the iterations, which will, eventually, lead to the objective. This is done by making a small number of random tentative moves, N_{ten}, in order to acquire data to feed to the

algorithm. After this data is available, the iterations start, and the pursuer starts to approach the beacon.

Algorithm 6 – generateMove method with oblivious

```
Inputs: RunStep
        currentDestination

if (abs(RSSI-RSSI_old)<RSSI_THRESHOLD)
        Random_Move;
else
        if(RSSI-RSSI_old>0.0)
                Keep_Going;
        else
                Turn_Around;
        endif
endif
RSSI_old= RSSI;
while(angle>180.0)
        angle-=360.0;
endwhile
while(angle<-180.0)
        angle+=360.0;
endwhile
send_move_request();
```

As mentioned before this is not enough in high noise environments and as such, after each step is taken, more data is collected filling a queue until a maximum size, N_{queue} . This queue is used as a circular buffer, in which newer information replaces the oldest one. This allows the collection of more data while already approaching the objective, and, in a low noise environment, the quick approach to the objective.

The experiments in [4] suggest the use of $N_{ten} = 4$ and $N_{queue} = 12$ so that, in a low noise environment, after four steps, the pursuer is already chasing the objective; in a high noise environment, the pursuer collects data up to twelve steps and is still able to reach the objective.

Note that, since the computer has all the data regarding the RSSI readings from all the nodes, once the robot changes target, the data fed into the MLE algorithm is data previously collected. So, instead of beginning with zero entries in the queue, it begins with the most recent entries already collected, up to a maximum of twelve.

This data, which is composed by positions and RSSI readings, is used during the iterations to estimate the position of the beacon. The first thing to is do, is to transform the RSSI

reading in signal space distances and then feed the positions and these calculated distances to the MLE algorithm. This algorithm uses the system with n equations that describe the distance between two points:

$$\begin{cases} (\bar{x}_{beacon} - x_1)^2 + (\bar{y}_{beacon} - y_1)^2 = {d_1}^2 \\ \vdots \\ (\bar{x}_{beacon} - x_n)^2 + (\bar{y}_{beacon} - y_n)^2 = {d_n}^2 \end{cases}$$

Then, the n-th equation is subtracted from the others, resulting in the n-1 equation system:

$$\begin{cases} 2 * \bar{x}_{beacon} * (x_n - x_1) + 2 * \bar{y}_{beacon} * (y_n - y_1) = \\ = {d_1}^2 - {x_1}^2 + {x_n}^2 - {y_1}^2 + {y_n}^2 \\ \vdots \\ 2 * \bar{x}_{beacon} * (x_n - x_{n-1}) + 2 * \bar{y}_{beacon} * (y_n - y_{n-1}) = \\ = {d_{n-1}}^2 - {x_{n-1}}^2 + {x_n}^2 - {y_{n-1}}^2 + {y_n}^2 \end{cases}$$

This allows writing:

$$A\bar{x} = b$$

$$where,$$

$$A = \begin{bmatrix} 2 * (x_n - x_1) & 2 * (y_n - y_1) \\ \vdots & \vdots \\ 2 * (x_n - x_{n-1}) & 2 * (y_n - y_{n-1}) \end{bmatrix}$$

$$\bar{x} = \begin{bmatrix} \bar{x}_{beacon} \\ \bar{y}_{beacon} \end{bmatrix}$$

$$b = \begin{bmatrix} {d_1}^2 - {x_1}^2 + {x_n}^2 - {y_1}^2 + {y_n}^2 \\ \vdots \\ {d_{n-1}}^2 - {x_{n-1}}^2 + {x_n}^2 - {y_{n-1}}^2 + {y_n}^2 \end{bmatrix}$$

The next step is to solve the system in order to get the estimate beacon position:

$$A\bar{x} = b \Leftrightarrow A^T A\bar{x} = A^T b \Leftrightarrow \bar{x} = (A^T A)^{-1} A^T b$$

Note that this is actually the least squares method, which minimizes the residual of the beacon position estimate.

Finally, once the beacon estimate position is calculated, the computer can calculate how much the robot needs to rotate. This is done by transforming the Cartesian coordinates (x, y) into polar coordinates (distance, angle) and, finally, by subtracting to this calculated angle the angle the robot is currently pointing to. This new requested move is then sent back to the robot.

Algorithm 7 – generateMove method with MLE

```
Inputs: RunStep
        currentDestination

if(Nsamples<=Nten)
        random_move();
else
        positions=getPositions(Nsamples);
        distances=getDistances(Nsamples);
        Beacon = MLE(Nsamples,positions, distances);
        angle = Math.atan2(y_Beacon-y_RobotPosition, x_Beacon-x_RobotPosition);
        angle = angle-currentDirection;
        while(angle>180.0)
                angle-=360.0;
        endwhile
        while(angle<-180.0)
                angle+=360.0;
        endwhile
endif
send_move_request();
```

5.1.4 Condition of arrival at the beacon

The final issue to be taken into account by the computer is the arrival at the beacon. The strategy to consider a valid arrival at the beacon was based on observation of the behavior of the RSSI with the distance. The conclusions from observation were that near the beacon the RSSI values would easily be above thirty five. This value was then made into a threshold so that, when the RSSI rises above it, the robot is in the vicinity of the beacon. A problem still persists at this point. The interferences allow such a high reading

to be received far away from the beacon. So, one reading above the threshold is not enough to validate the arrival. Three readings above that threshold were then made compulsory so that the program considers a successful arrival. But, adding to this, one last issue remains since the robot can move away from the beacon or even go to an area that has destructive interference. Therefore, in order to finally settle this, a second lower threshold, with the value of thirty, was created so that the count does not reset while the RSSI doesn't drop below that value. This is illustrated in Figure 32 and Algorithm 3.

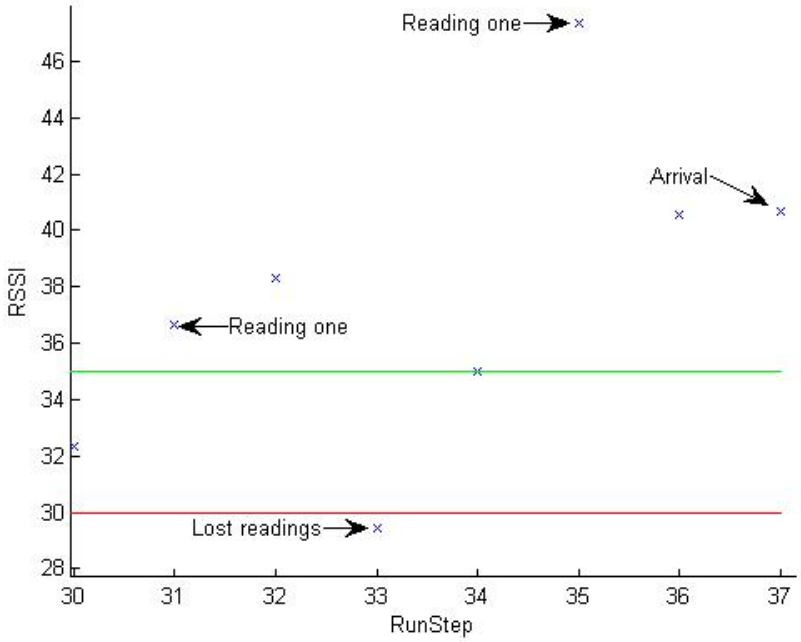


Figure 32 – Arrival Condition

5.2 Obtained Results

Here, the results of the navigation experiments will be shown. The objective of this experiment is to test the capability of navigation in a multi-beacon environment using the two methods described before, oblivious and MLE. The setup, as mentioned above, is two beacons, one robot and one computer. The task the robot has to perform is to go from the starting point (0, 0), to beacon zero (0, 300), then go to beacon one (45, 75) and, finally, return to beacon zero (0, 300). All this based only on the received RSSI readings, i.e. no encoder readings will be used to return to beacon zero.

All the following experiments were made in the same conditions and repeated several times, both with the oblivious and MLE methods. Bellow, three samples of each are presented and discussed.

5.2.1 Oblivious method results

Experiment 1

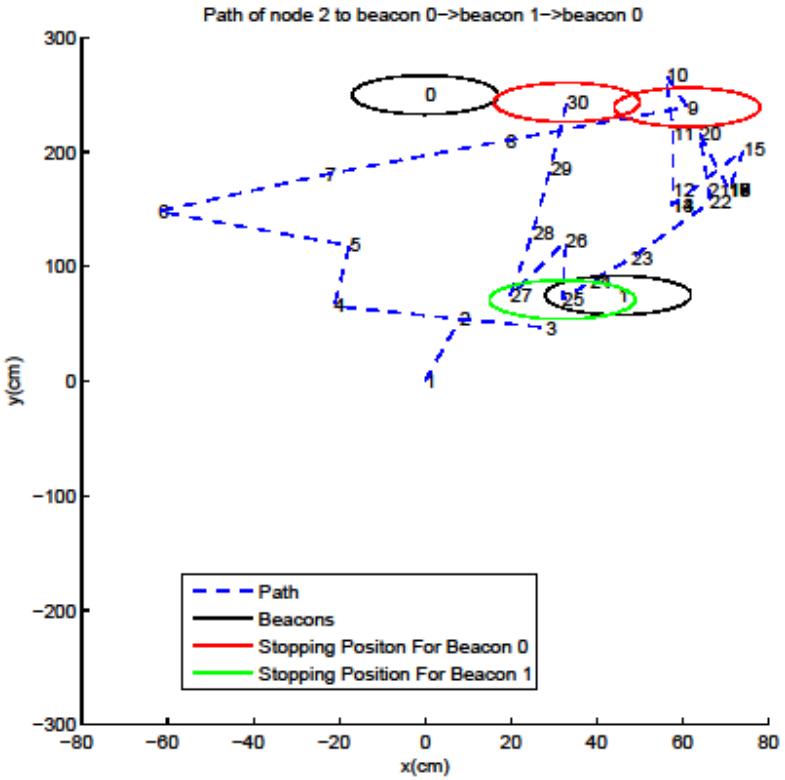


Figure 33 – Oblivious method path- experiment 1

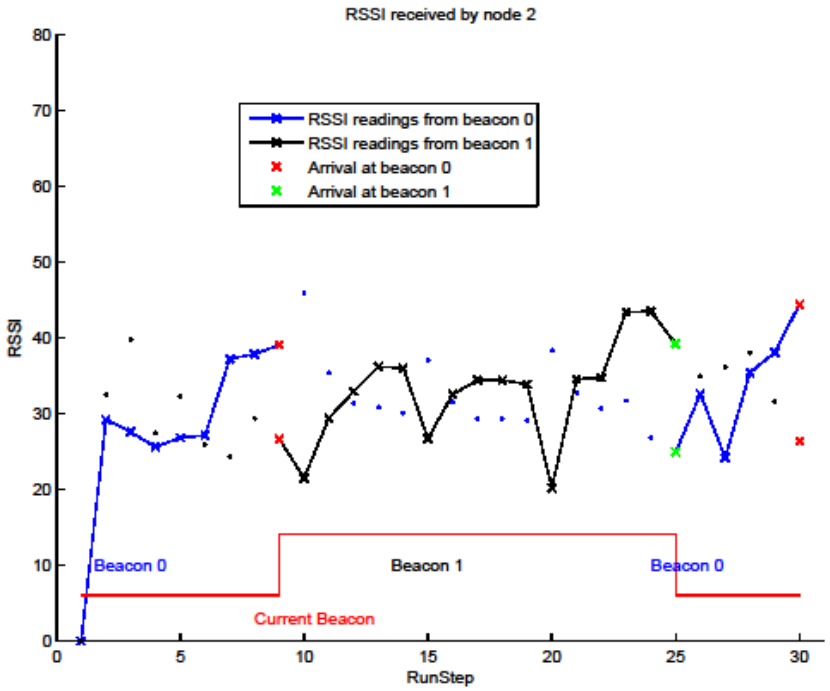


Figure 34 – Oblivious method RSSI received by node 2- experiment 1

Experiment 2

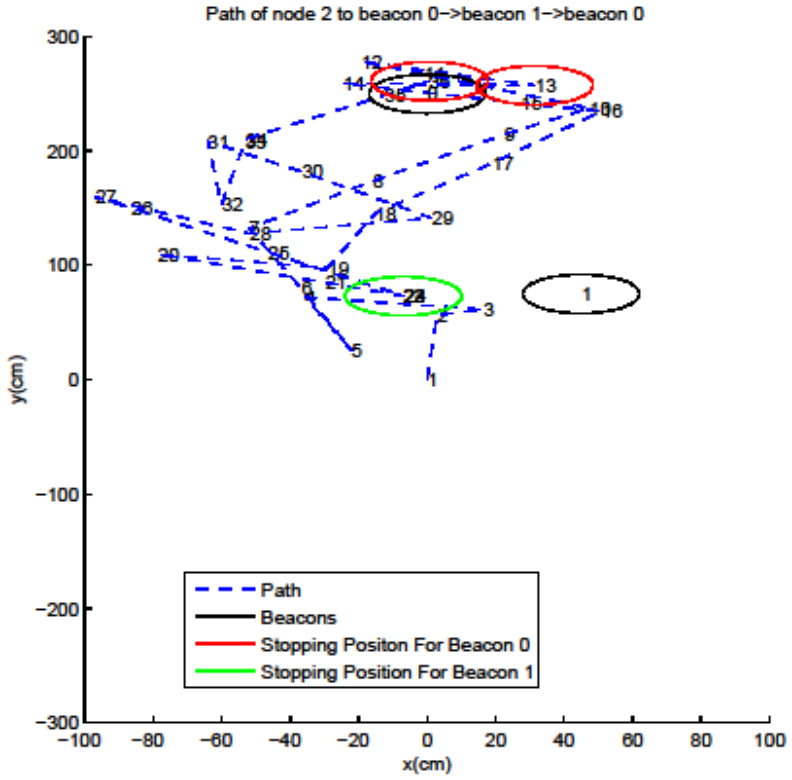


Figure 35 – Oblivious method path- experiment 2

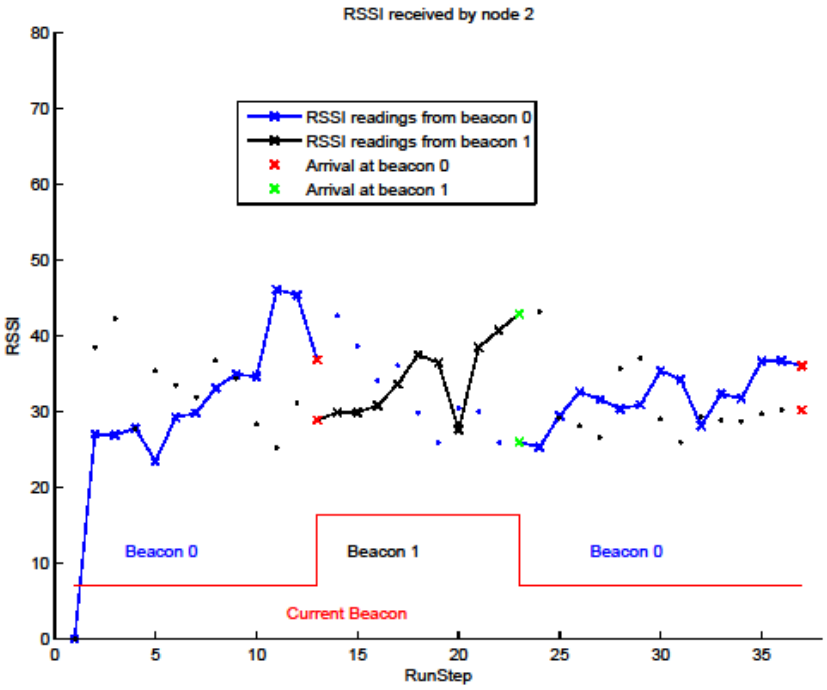


Figure 36 – Oblivious method RSSI received by node 2- experiment 2

Experiment 3

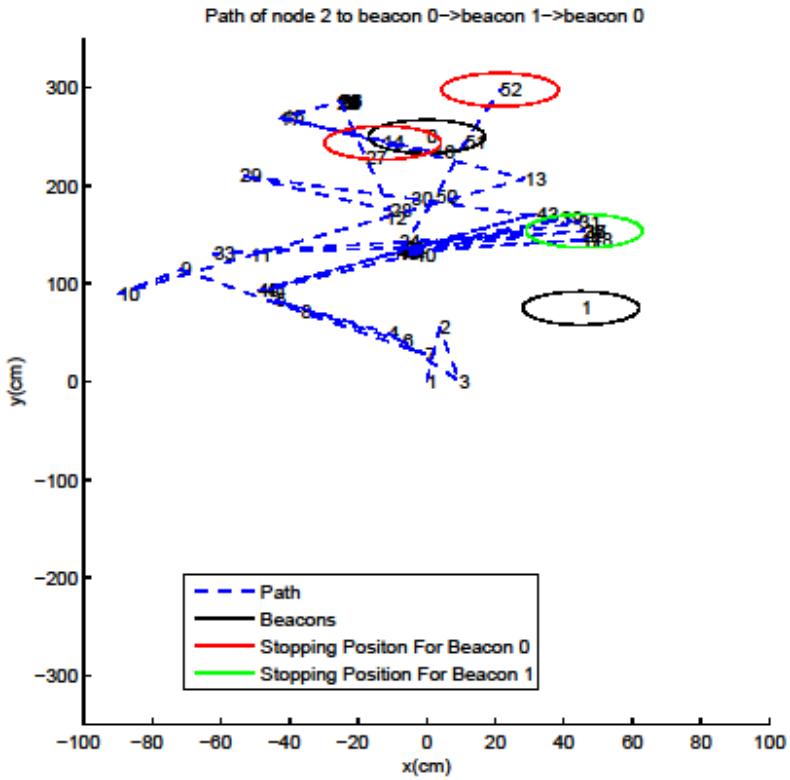


Figure 37 – Oblivious method path- experiment 3

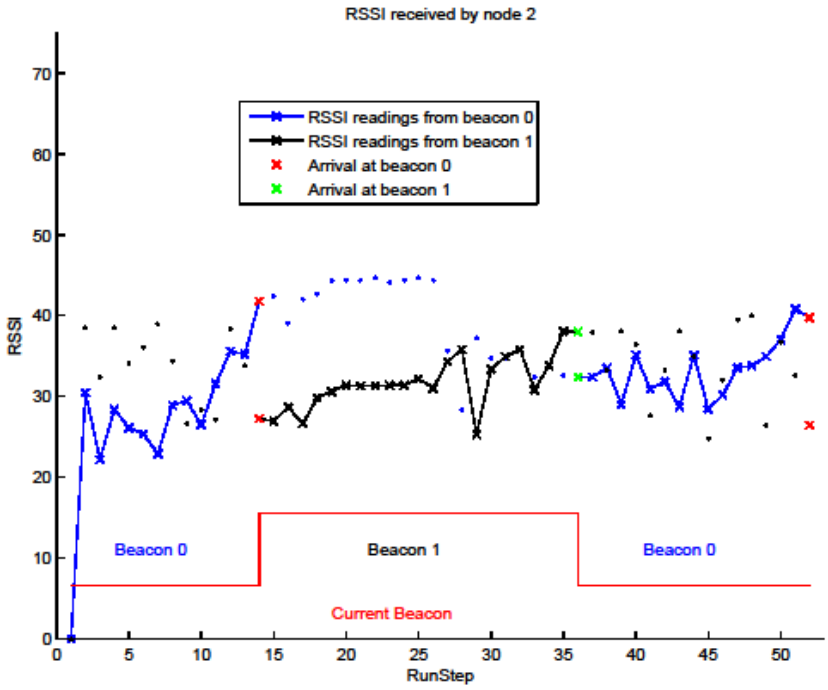


Figure 38 – Oblivious method RSSI received by node 2- experiment 3

5.2.2 MLE method results

Experiment 1

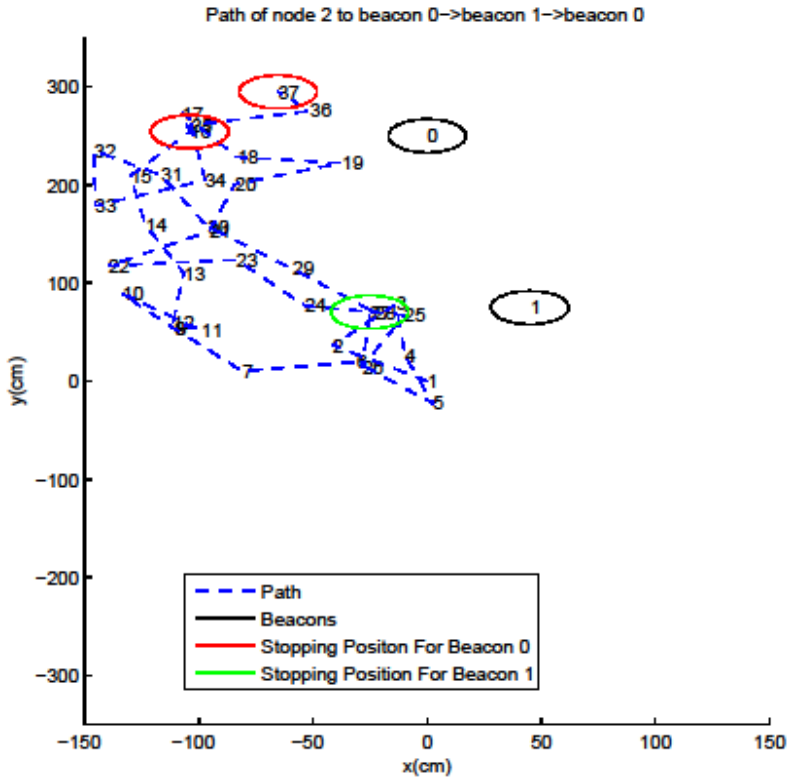


Figure 39 – MLE method path- experiment 1

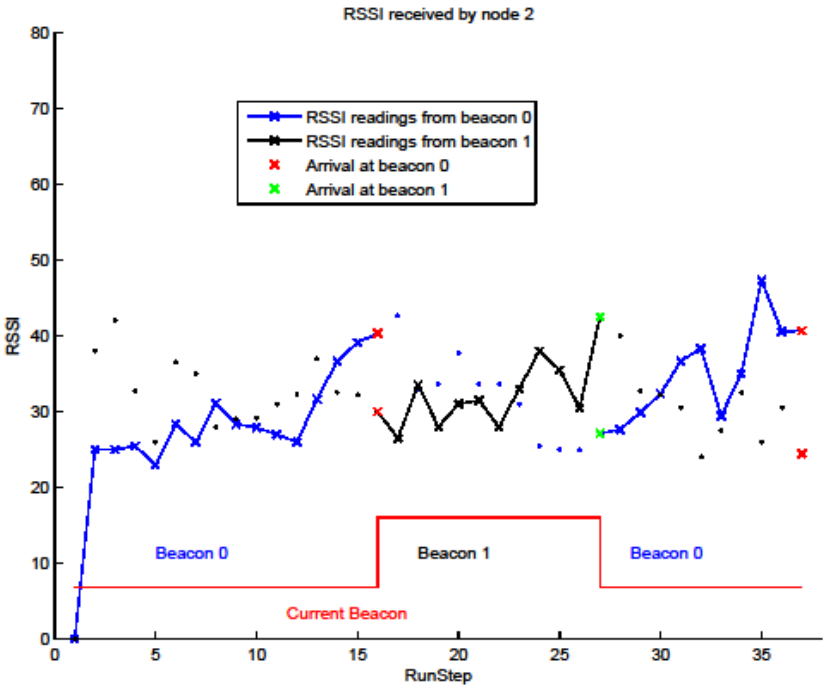


Figure 40 – MLE method RSSI received by node 2- experiment 1

Experiment 2

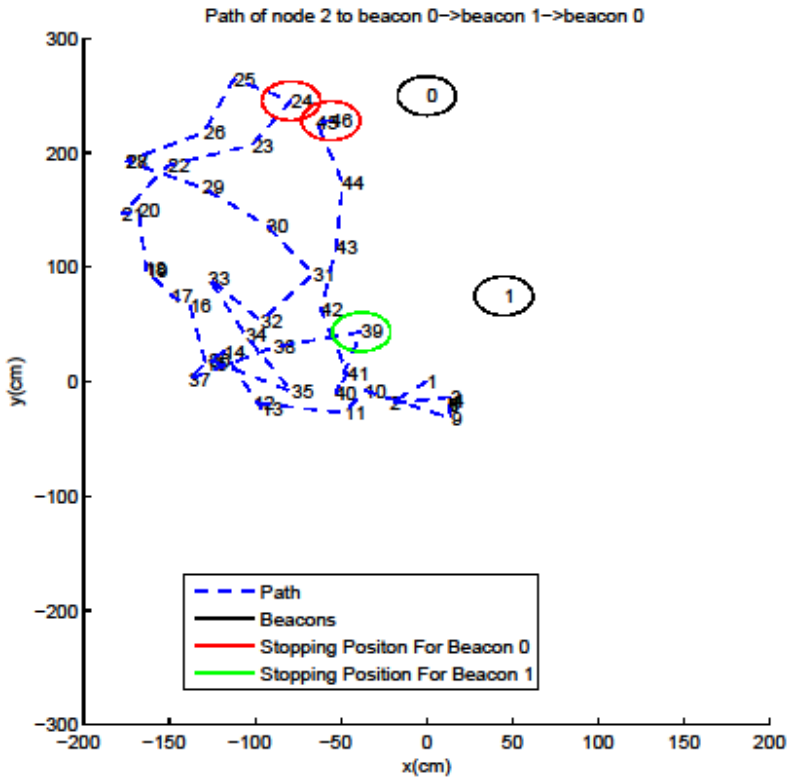


Figure 41 – MLE method path- experiment 2

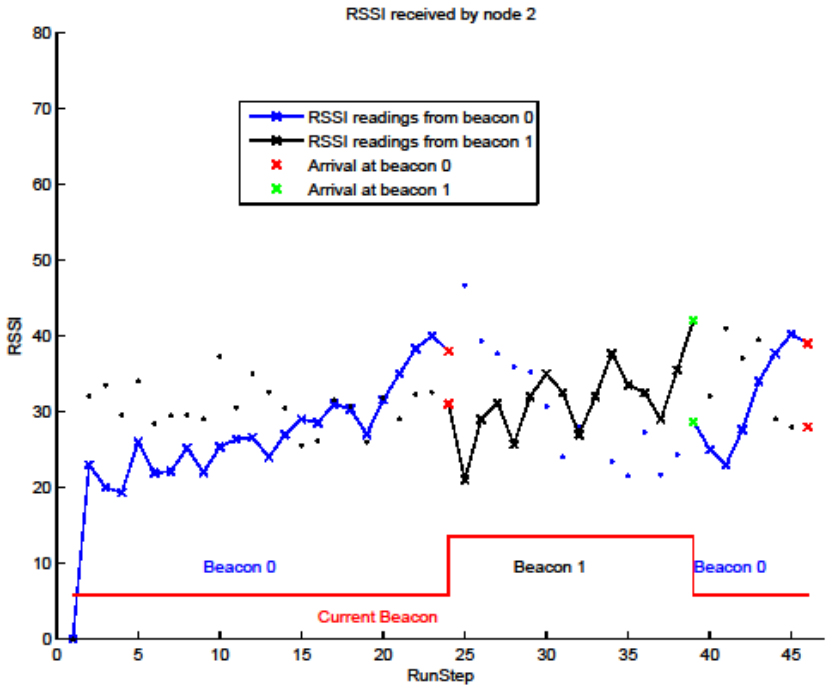


Figure 42 – MLE method RSSI received by node 2- experiment 2

Experiment 3

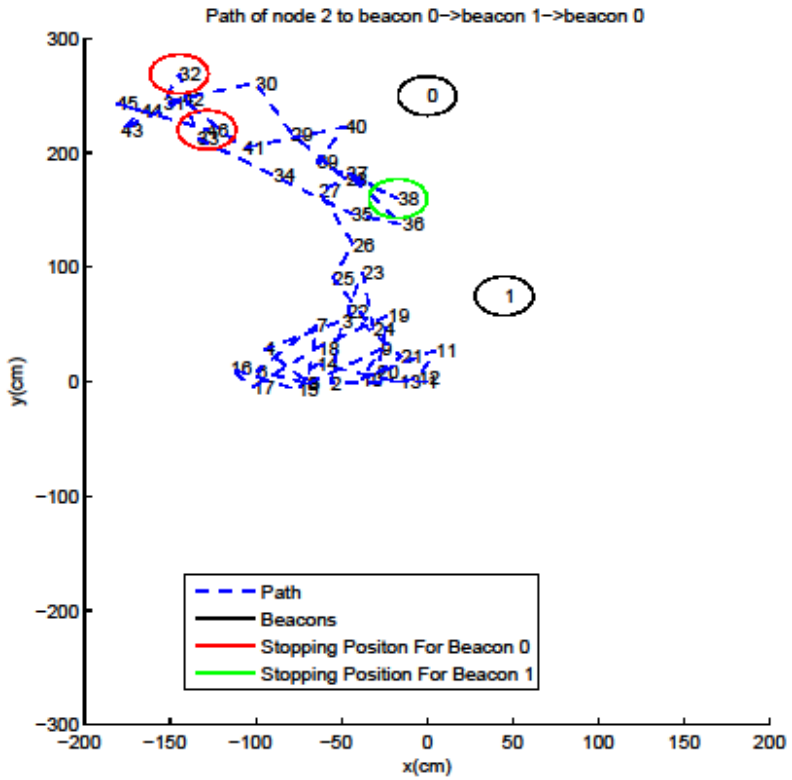


Figure 43 – MLE method path- experiment 3

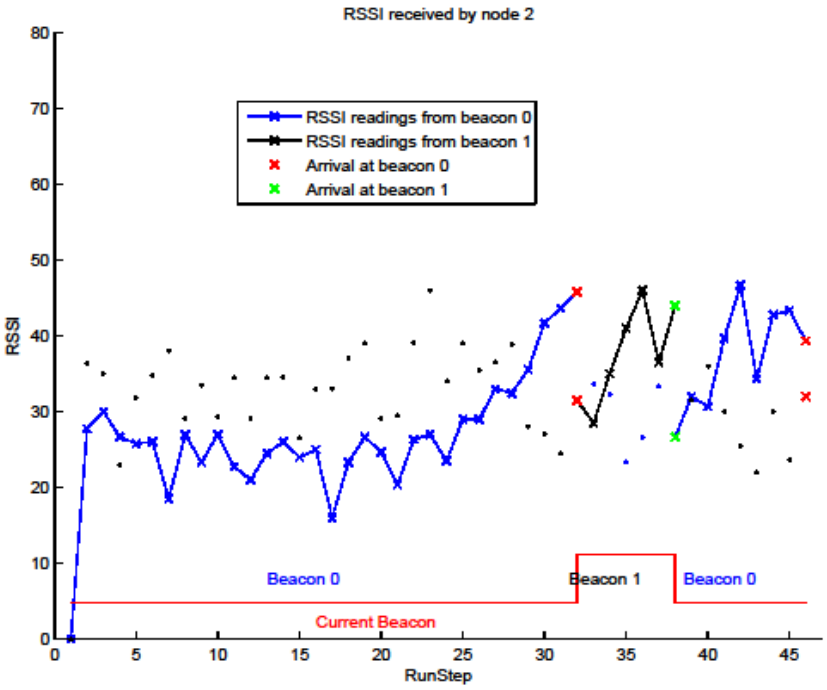


Figure 44 – MLE method RSSI received by node 2- experiment 3

5.2.3 Final Considerations

By a quick analysis of the above graphics above, it becomes clear clear that the number of steps needed to make the first approach, using the MLE method, is much larger than in the second and third approaches. This is easily explained. While in the first approach there is a lack of values, on the queue of data fed to the MLE, in the following approaches there are 12 values available to feed the algorithm, which makes a much more precise estimation possible. Another point of interest is the constant rise of the RSSI values with this method, which shows how effective the algorithm is.

On the oblivious method, on the other hand, there are rises and falls in the readings and the results are much more inconstant. While with the MLE, the first approach is slow and the subsequent are faster, with the oblivious method sometimes they are faster and sometimes they are slower. This is not at all unexpected due to the random nature of the oblivious algorithm.

Table 7 – Number of steps needed with oblivious

Number of steps needed	Experiment 1	Experiment 2	Experiment 3
Beacon 0	8	12	13
Beacon 1	16	10	22
Beacon 0	5	14	16

Table 8 – Number of steps needed with MLE

Number of steps needed	Experiment 1	Experiment 2	Experiment 3
Beacon 0	15	23	31
Beacon 1	11	15	6
Beacon 0	10	7	8

Finally in experiment 3, with the MLE, a lot of going back and forward is visible. Although this seems contradictory to the algorithm, by observing the experiment, it is possible to see that that was caused by the existence of obstacles which did not allow the robot to move where it wanted to and, therefore, collect RSSI values with a larger difference. This shows a possible weak point of the MLE algorithm – the big dependence on a good relationship between the RSSI with the distance – which makes the robot, if trapped in a location where the readings are very similar, to take a while or not be able to proceed. Although a similar pattern exists in experiment 3, with the oblivious method, this was not created by the obstacles but by the natural randomness of this method.

A possible solution for the MLE problem mentioned above is to check the positions the robot was at, and where it is. Based on that, and on the beacon estimates, is possible to make the robot move somewhere he hasn't been in recent time so that it can collect more and different information. Also interesting would be to perform these experiments

in an obstacle free environment. This would avoid interferences caused by the obstacles in the algorithms, either helpful or detrimental.

Chapter 6

Conclusions and future work

6.1 Conclusions

The objective of this dissertation was to study, implement and test several relative localization and navigation techniques on a multi-beacon environment, based only on the RF signal of wireless communications.

On the subject of localization, the MDS algorithm was tested with different transmission synchronization, RSSI filtering, and bandwidth parameters and the results were given comparative evaluation. This test, to the best of our knowledge, had never been done and confirmed that the tuning of these parameters has a significant toll on the performance of this algorithm. Other tests were conducted on the MDS algorithm and on these the intent was to check if using the different parts of the extended connectivity matrix, e.g. top triangle or bottom triangle, would make any improvements or, on the downside, make the behavior worse. But, unlike the parameters adjustment, the difference on the results these tests produced proved to be too little to be considered either a benefit or a drawback.

On the subject of navigation, the resulting work was experimentally evaluated, with emphasis on the comparative evaluation of the oblivious method and the MLE method. In the performed experiments it was perceived that neither method is quicker than the other, which was not expected, since the iterative method was thought to be faster to complete the experiment than the oblivious method. However, this was clearly due to the relatively large number of initial steps taken by the MLE method. After the initial steps, needed to acquire a good notion of the target direction, MLE was much faster, with

relatively little deviations, directed to the position where the beacon was. The oblivious method, on the other hand, shows a constant and higher tendency to deviate from the beacon due to its randomness. Finally, the MLE´s big dependence on a good relationship between the RSSI and the distance was exposed as a weak point when using this method.

6.2 Future work

Some ideas, either by lack of means of lack of time, were not experimented on. Thus, they will be dealt with in this section.

6.2.1 Improving the MLE algorithm

The MLE algorithm used only moves according to the beacon estimate position. But, since the robot already has encoders, it is possible to go beyond this.

The suggestion left here is to use the already equipped platform of the robot to map the positions the robot has visited before. Using this information, together with the beacon estimate, makes it possible to plan the movement, which can avoid the collection of data in the same points over and over. This will not only help avoid the dependence issued in the experiments, which can have detrimental effects on the algorithm, but, possibly, will also reduce the time the robot takes to go to the beacon, since the samples will be taken with a larger distance from one another.

6.2.2 Multi-Robot experiment

Due to hardware constraints, in the work developed it was only possible to build a robot. So, the second suggestion is that it would be interesting to compare the results of a robot approaching a beacon with a robot approaching a robot.

These tests are appealing because a larger queue of data to feed the MLE reduces the error of the estimate. Nevertheless, if the target moves, the system will be slower to react to that change. So, the feasibility of applying MLE in a noisy environment with a moving target is not as straightforward as it is with a static beacon. On the other hand, a reactive method, like the oblivious, should react very much in the same way either with static or moving nodes.

6.2.3 Wide space experiment

It has been mentioned before that the obstacles interfere with the movement of the robots. This happens in both ways, either helping the robot reach the target as well as making it harder. The point is, to be able to really test the efficiency of the algorithms and compare them, without any help or hinder from the obstacles, it would be necessary to perform the experiments with the algorithms in a large area. This could bring precious extra information that could help to further differentiate the performance between the algorithms.

Bibliography

[1] F. F. d. N. D. Carramate, "Localização Relativa de Robôs Móveis Baseada no RSS de Comunicações RF," 2008.

[2] H. Li, L. Almeida, Z. Whang, and Y. Sun, "Relative Positions Within Small Teams of Mobile Units," 2007.

[3] H. Li, L. Almeida, F. Carramate, Z. Whang, and Y. Sun, "Connectivity-Aware Motion Control among Autonomous Mobile Units," 2008.

[4] H. Li, L. Almeida, F. Carramate, Z. Wang, and Y. Sun, "Using Low-Power Radios for Mobile Robots Navigation," in *FET 2009 - 8th IFAC Conference on Fieldbuses and Networks in industrial and embedded systems*, 2009.

[5] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, "Cooperative Mobile Robotics: Antecedents and Directions," 1997.

[6] S. Baek, S. Ahn, and S.-Y. Oh, "Fast Localization Algorithm for The Cleaning Robot By Using Self-Organization Map," *International Symposium on Computational Intelligence in Robotics and Automation,* pp. 19-24, 2007.

[7] H. Li, L. Almeida, Z. Whang, and Y. Sun, "Relative Positions Within Small Teams of Mobile Units," 2007.

[8] "QUADRATEC Limited," in *http://www.quadratec-ltd.co.uk/Security_Surveillance_systems.htm*, 14-10-2009.

[9] V. Cars, "https://www.media.volvocars.com/global/enhanced/en-gb/Home/Welcome.aspx," 16-10-2009.

[10] "Finder system for fire-fighter location," in *http://www.i-a-i.com/view.asp?aid=105*, 15-10-2009.

[11] M. Aoki and H. Fujii, "Inter-Vehicle Communication: Technical Issues on Vehicle Control Application," in *IEEE Communications Magazine*, 1996, pp. 90-93.

[12] Y. Shang, W. Ruml, T. Zhang, and M. P. J. Fromherz, "Location From Mere Connectivity," in *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc)*, 2003, pp. 201-212.

[13] X. Ji and H. Zha, "Sensor Positioning in Wireless Ad-hoc Sensor Networks Using Multidimensional Scaling," 2004.

[14] T. Fracchinetti, G. Buttazzo, and L. Almeida, "Dynamic Resource Reservation and Connectivity Tracking to Support Real-Time Communication among Units," in *EURASIP Journal on Wireless Communications and Networking*, May 2005, pp. 712-730.

[15] Crossbow, "http://www.xbow.com/," 22-10-2009.

[16] T. Instruments, "C2420 - 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver," 2007.

[17] Crossbow, "Avoiding RF Interference Between WiFi and Zigbee."

[18] F. Santos, G. Currente, L. Almeida, N. Lau, and L. S. Lopes, "Self-configuration of an Adaptive TDMA wireless communication protocol for teams of mobile robots," 2007.
[19] I. LSI Computer Systems, "LS7366 - 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver," 2009.
[20] S.-T. MICROELECTRONICS, "L293E - Push-pull four channel drivers," 1993.
[21] S. Corporation, "GP2D12 Optoelectronic Device," 2005.
[22] M. Ruas and J. L. Azevedo, "Robot Voyager II – Reactividade e eficiência." vol. 4: REVISTA DO DETUA, September 2005.