Localization and tracking using an heterogeneous sensor network

Masters' Degree Project
Stockholm, Sweden August 2008

XR-EE-RT 2008:016
Contents

1 Introduction 9
  1.1 Motivation .............................................. 9
  1.2 Problem Formulation ................................. 10
  1.3 Contributions ........................................ 10
  1.4 Outline ................................................. 11

2 Localization and tracking using an heterogeneous sensor network 12
  2.1 Overview of techniques and methods to determine location ... 12
    2.1.1 Techniques ....................................... 13
    2.1.2 Methods ......................................... 14
  2.2 Dynamical Models ..................................... 17
    2.2.1 Model 1 ........................................ 17
    2.2.2 Model 2 ........................................ 18
  2.3 Kalman Filter ......................................... 19
  2.4 Proposed Filter ..................................... 21
  2.5 Scheduling ........................................... 23
    2.5.1 Offline scheduler .............................. 24
    2.5.2 Covariance-Based scheduler ................. 30

3 Experimental set-up 39
  3.1 Hardware ............................................. 39
    3.1.1 Wireless Sensor Network Testbed ............. 39
    3.1.2 Ultrasound sensor .............................. 50
    3.1.3 Vision based system ......................... 53
    3.1.4 Mobile agent .................................. 55
    3.1.5 Fusion Center ................................ 55
  3.2 Software .............................................. 56
    3.2.1 Ultrasound system ............................ 56
    3.2.2 Vision based system ......................... 59
    3.2.3 Fusion Center ................................. 68
4 Experimental validation
   4.1 Ultrasound System ........................................ 75
      4.1.1 Straight Line ........................................ 76
      4.1.2 Localization ................................... 77
   4.2 Vision Based System .................................. 84
      4.2.1 Localization .................................... 84
   4.3 Fusion center ....................................... 88

5 Conclusions and Future Work ............................... 93

6 Appendix ................................................. 96
List of Figures

1.1 The switched sensor problem that is considered. How should one switch between two heterogeneous sensors to get a good estimate $\hat{x}$ .......................................................... 11

2.1 The trilateration method for 2D accurate measurement . . . 16
2.2 Mobile agent moving according to model 1 with gaussian white process noise with zero mean and variance of 10cm/step. Starting position at (0,0). .......................................................... 18
2.3 Mobile agent moving according to model 2 with gaussian white process noise with zero mean and variance of 1cm$^2$/step. Starting position at (0,0) with zero velocity. .............................. 19
2.4 Linear gaussian state space model. ................................. 20
2.5 The function $p_{\text{average}}^*(N)$ for two different delays $d$ using model 1. ........................................................................ 27
2.6 The function $P^*(k, N)$ for different periods $N$ when delay $d = 7$ using model 1. .......................... 28
2.7 The performance cost $V_T$ as function of period $N$ for two different delays $d$ using model 1. .............................. 28
2.8 The function $p_{\text{average}}^*(N)$ for two different delays $d$ using model 2. .......................................................... 29
2.9 The function $P^*(k, N)$ for different periods $N$ when delay $d = 7$ using model 2. .......................... 30
2.10 The performance cost $V_T$ as function of period $N$ for two different delays $d$ using model 2. .......................... 31
2.11 Tree search example for covariance-based switching with maxD=2. .......................... 32
2.12 Flow diagram of the covariance-based switching algorithm. .......................... 33
2.13 The function $p_{\text{average}}(k, maxD)$ and the usage of the high-quality sensor for the covariance-based scheduling when delay $d = 3$ and model 1. .......................... 34
2.14 The function $p_{\text{average}}(k, maxD)$ and the usage of the high-quality sensor for the covariance-based scheduling when delay $d = 7$ and model 1. .......................... 35
2.15 The function $p_{\text{average}}(k, \max D)$ and the usage of the high-quality sensor for the covariance-based scheduling when delay $d = 3$ and model 2. .......................... 36
2.16 The performance function $V^*(k, \max D)$ and the usage of the high-quality sensor for the covariance-based scheduling when delay $d = 3$ and model 2. .......................... 36
2.17 The function $p_{\text{average}}(k, \max D)$ and the usage of the high-quality sensor for the covariance-based scheduling when delay $d = 7$ and model 2 .......................... 37
2.18 The performance function $V^*(k, \max D)$ and the usage of the high-quality sensor for the covariance-based scheduling when delay $d = 7$ and model 2. .......................... 38

3.1 Overview of the experimental set-up and operation based on the wireless sensor network testbed. ...................... 40
3.2 Standard KTH wireless sensor network testbed power supply. 48
3.3 Testbed deployment in the 6th floor of the Q building in KTH. 50
3.4 Picture shows the network 1 of the testbed in the corridor of the 6th floor of the Q building at KTH. ............... 51
3.5 Picture shows the wireless node Tmote Sky in casing with Ultrasound Sensor. ........................................ 51
3.6 Ultrasound transmitter cluster top view and side view. .... 52
3.7 Logitech fusion web-camera. .................................. 54
3.8 Mobile Agent. RC electric car controlled by an human operator. .... 55
3.9 Interaction between ultrasound receiver and transmitter modules. ......................................................... 57
3.10 Transmitter flow Diagram ....................................... 58
3.11 Receiver Flow Diagram ......................................... 60
3.12 Image processing flow diagram ............................... 61
3.13 Acquired image with mobile agent. Image taken on the corridor of the 6th on the Q Building at KTH. ....... 62
3.14 Cut image view. Binary image. ............................... 64
3.15 Mobile agent detection. Star shows the detected centroid of the mobile agent. .................................... 67
3.16 Vision based localization system with non-linearities. ..... 67
3.17 Data flow and treatment including the sensors and processing unit. ....................................................... 69
3.18 Flow diagram of the algorithm implemented on the fusion center processing unit. ................................ 73
3.19 Graphical User Interface created for the user to be able to watch the position of the robot with spacial references. ..... 74
4.1 Straight line average measurements. Real distance(cm) VS Measured distance(cm) .................................................. 76
4.2 Straight line average error - Linear interpolation with 14 distance points. Error(cm) VS Real distance(cm) ..................... 77
4.3 The average error(cm) in four different position for four given receiver nodes. ....................................................... 79
4.4 Ultrasound system performance when no outlier rejection method is applied. Error and position values for real position X=50. . 80
4.5 Ultrasound system performance when an outlier rejection method is applied. Error and position values for real position X=50. . 80
4.6 Ultrasound system performance when an outlier rejection method and the model 1 estimator is applied. Error and position values for real position X=50. ................................. 81
4.7 Ultrasound system performance when an outlier rejection method and the model 2 estimator is applied. Error and position values for real position X=50. ................................. 82
4.8 Localization performance of the vision based system for a steady mobile agent at (50,200). Outlier rejection method and model 1 estimation performed. ................................. 85
4.9 Localization performance of the vision based system for a steady mobile agent at (50,200). Outlier rejection method and model 2 estimation performed for W = 0.003. ..................... 86
4.10 Localization performance of the vision based system for a steady mobile agent at (50,200). Outlier rejection method and model 2 estimation performed for W = 0.09. ..................... 87
4.11 Estimated and raw position quadratic errors for offline scheduler with N and Covariance-based scheduler for model 1 and model 2. ................................................................. 91
4.12 Tested mobile agent trajectory for optimal high-quality sensor switching N = 6. Real position, estimated position and position given by raw sensor measurements over the X coordinate when performing 29 measurements. ............................. 92

6.1 Receiver Circuit .......................................................... 97
6.2 Transmitter Circuit ...................................................... 98
6.4 System Breakdown Structure ....................................... 100
6.5 Floor plan - KTH Q 6th (SSS) ....................................... 101
List of Tables

2.1  Maximum values of process noise $W$ for a given delay $d$ for each model. .................................................. 26
2.2  Optimal sensor scheduling approach for model 1 and 2 considering different communication cost $\lambda$. .................. 38
4.1  Optimal periodic high-quality sensor switching $N^*$ of $V_E$ for model 1 and 2 considering different communication cost $\lambda$. . . 89
4.2  Optimal periodic high-quality sensor switching $N^*$ of $V_T$ for model 1 and 2 considering different communication cost $\lambda$. . . 90
Abstract

Taking resource limitations into account in the design of wireless sensor networks are important in many emerging applications. The need for minimizing the communication and power consumption of individual nodes and other units pose interesting challenges for estimation and control strategies. This document describes the design, implementation, obtained results and conclusions of a cooperative localization and tracking system based on two types of sensors. Practical investigations are made to reach the optimal sensor scheduling based on offline and covariance-based scheduler approaches. One sensor gives low quality measurements and is based on an ultrasound sensor and the other is a web-camera with high precision but with delayed results. The ultrasound sensor is connected to wireless sensor nodes which are part of the KTH Wireless Sensor Network Testbed and the web-camera is connected to a data processing unit and placed in the same area. An overview about localization techniques and solutions are presented. The design, development, and implementation of the KTH Wireless Sensor Network Testbed is also discussed. The software implemented on the system is fully detailed as well as the necessary hardware. A presentation of the filtering methods used to perform localization and tracking is put forward. Analysis and conclusions of all the different approaches used are discussed. Guidelines for future work are also proposed.
Acknowledgments

First, I would like to thank my supervisors Karl H. Johansson and João Borges de Sousa for giving me the opportunity of developing this work at KTH. I am truly grateful for all your guidance and advices that made me accomplish this thesis and also by letting me learn on my own.

Henrik Sandberg, you deserve many thanks for all your support, comments to the thesis and availability to answer my numerous doubts everyday. Without your support i would not be able to perform all this work. I also have to thank Maben Rabi for answering me all the everyday, out of office hours and weekend questions I posed you that helped me moving further.

I would like also to thank my friend, roomate and partner, Bernardo Maciel for the one life time experience of living and working together in this last year. His help and advices are greatly acknowledged.

Also I would like to thank my co-supervisors Erik Henriksson and Pan Gun Park for their unconditional support whenever i needed anything. Thanks to Magnus Lindhé for all your support developing the ultrasound system and advices. I have to thank Prof. Ana Mendonça, Patric Jensfelt and Sobhan Naderi for helping me with the image processing issues. Chithrupa Ramesh thank you very much for all the discussions we had that helped me performing a better job. I would like to thank my special colleagues PG Di Marco, Cesare Carreti and Stefan Gisler for their help, contributions and advices for this thesis.

I also have to note the tremendous help that my ERASMUS colleagues gave me during this year, my life will never be the same without you.

Tomé Costa, Tiago Nunes, Nuno Medeiros, José Melo, José Barbosa and Vitor Torres, you know that what I am now, I owe it to you also.

Last but not least, I would like to thank my parents, Henrique and Arnaldina, and my brother Luis for all your support.
Chapter 1

Introduction

1.1 Motivation

The resource limitations of the wireless sensor network can be seen as an important issue in the design of emerging applications [1], [2]. The need for minimizing the communication and power consumption of individual nodes and other units, poses interesting challenges for estimation and control strategies [3]. In this work we consider the novel networked estimation problem formulated in [8], in which two types of sensors with different resource demands are used share the same or different networks.

The problem of localization and tracking of a mobile agent using observations from two types of sensors can then be seen as a motivation for this work. The sensors communicate their data to a central node that perform the processing. The first type of sensors used are ultrasound sensors with low-quality measurements, small processing delay and a light communication cost. The second type of sensor is a camera with high-quality measurements, but large processing delay and high communication cost. One can notice now that the scheduling of both in order to have the best estimation possible of the mobile position and at the same time reduce the communication costs and power consumption pose a big challenge to be solved. So in this work are made design trade-offs between estimation performance, processing delay and communication cost for a sensor scheduling problem with heterogeneous sensors. We show how optimal sensor schedules, periodic or not, can be found by means of search over a finite set. As seen in [8], sensor selection problems have been studied extensively, e.g., [4]. In [8] the approach is novel in that it incorporate communication cost in the cost criterion together with processing delays. See [5] for another recently studied problem. The motivation for this thesis comes then from the necessity to perform the experimental
validation and further developments of the work presented on [8] and also be the need of building an experimental set-up for testing all the theories developed under the topic of Wireless Sensor Networks in KTH.

1.2 Problem Formulation

The problems studied in this thesis are:

- Perform localization of a mobile agent in indoor environments using heterogeneous sensors within a wireless sensor network.
- Perform the scheduling of heterogeneous sensors.
- Design, development and implementation of a wireless sensor network testbed.

As it can be seen, when using heterogeneous sensors, their scheduling poses interesting challenges for estimation, control and power management strategies design. Design trade-offs between estimation performance, processing delay and communication cost have to be taken into account. The purpose of this thesis is then to propose an estimator that improves the accuracy of the measurements taken by the sensors and also to develop proper scheduling models for the sensors. Is also the intention of the author to show how the solutions proposed perform in an a real experimental environment, a wireless sensor network testbed.

This situation is illustrated in Fig. 1.1.

1.3 Contributions

The main contributions of this thesis are:

- Design, development, implementation and experimental validation of a wireless sensor network testbed at KTH on a joint work with another MsC Student, Bernardo Maciel.
- Implementation and experimental validation of the paper "Estimation over heterogeneous sensor networks" of Henrik Sandberg et al , submitted for the 47th IEEE Conference on Decision and Control. [8]
- Design, development, implementation and experimental validation of an covariance-based sensor scheduler (posed as further development on [8]).
Figure 1.1: The switched sensor problem that is considered. How should one switch between two heterogeneous sensors to get a good estimate $\hat{x}$

- Design, development, implementation and experimental validation of an ultrasound based localization system in a joint work with PhD Student, Magnus Lindhéd.

- Design, development, implementation and experimental validation of a vision based localization system.

### 1.4 Outline

The outline of this report is as follows.

Chapter 2 presents the localization techniques and methods, dynamical models, filters and schedules used in order to perform the localization based on heterogeneous networks. It introduces and demonstrate with examples all the proposed tools.

Chapter 3 illustrates the experimental setup designed, developed and implemented in order to validate the system developed. This chapter presents in terms of hardware and software all the components that are part of the system. It is discussed the wireless sensor network designed and used, the ultrasound system, vision based system and the fusion center system. All the components are thoroughly described.

In Chapter 4 is shown all the experimental validation performed to the algorithms and methods developed in the previous chapters. Complete analysis with illustrative simulations are presented. The report is then concluded in Chapter 5 where all the conclusions about the experimental validation and future work needed are presented.
Chapter 2

Localization and tracking using an heterogeneous sensor network

This chapter introduces the localization techniques, the system modeling approaches, presents the estimator used and gives an overview on the scheduling approaches.

In order to perform localization two different sensors are used. One is an ultrasound sensor giving information through a wireless sensor network with no delay, no communication cost and known as being a cheap sensor. On the other hand the other sensor used, a web-camera, has a certain delay due to image processing time and data transmission and also a communication cost due to the of substantial amount of energy spent on each transmission thus is considered an expensive device. From now on the ultrasound sensor is denoted as the low quality sensor (lq) and the camera as the high quality sensor (hq). Next is discussed the modeling approaches to cope with this type of system.

2.1 Overview of techniques and methods to determine location

As seen in [45] two basic approaches for determining the location of an object can be formulated.

- **Location from landmarks.** In this approach, the location system is implemented by selecting a set of landmarks or reference points with known coordinates. As it is also referred these reference points can be
moving \(iff\) their position is always known. It is easy to notice that if one has the distance measurements for a given number \(n\) of reference points to the object \(O\) one needs to detect the location of the object \(O\) is easy to achieve just by solving a system of equations. As an example of this technique is the localization technique used in [43], [45] and also this MSc Thesis, where a WSN is used to locate an object within its coverage area.

- **Location from dead-reckoning.** From [45] dead-reckoning is the technique that determines the position of an object with respect to some starting point using the dynamics of motion of the object. An example of this type of technique is when we have an object \(O\) that starts a movement at a given point \(P\) along a direction \(\Theta\) at a constant velocity \(v\), where its position coordinates at time \(t\) are given by \((vt \cos \Theta, vt \sin \Theta)\). Dead-reckoning can then be interpreted as the method with which an object is able to detect its own position by measuring its own dynamics, without external known references or sensors. This method as the drawback of accumulating measurement errors since various embedded sensors are used. Because of this, most location systems are implemented using landmarks or a combination of both.

As said before our approach will be based on the location using landmarks. Next are presented the techniques used to determine location using landmarks and also the methods available. The techniques are interpreted as the type of measurements that can be made to achieve localization of objects and methods are interpreted as the solutions available to solve the equations derived, given the measured distances and the position of the landmarks.

### 2.1.1 Techniques

As one can see if the landmark system is used, the object has to be able to know his position based on the measurements of its position to each reference point. The following methods may be used in order for the object to determine its position given a landmark-based system.

- **Distance and angle.** This is one of the most used techniques for position estimation. One of the reasons is due to the easy implementation and position computation and also because of the low price of sensors available to perform this technique (ultrasounds, microphones, etc.). Usually are measured distances or angles from the landmark to the object, using then trilateration or triangulation methods to compute
the object position. Examples of these systems are the GPS [46], the RADAR [47] and this MsC Thesis.

- **Signal signature.** In this approach, the object uses the signal strength value usually of an RF message transmitted by the landmarks in order to know its position in space. It is also possible to use the reversed way where the object transmits a RF message and the reference nodes knowing the signal strength can compute the object position. There are several projects implemented using this technique such as [50], [48] and [49].

Since one will use the distance measurement approach in order to measure the distance from the object to the landmarks it will be described next. The discussion about the different techniques the distance measurement technique uses is made next.

### Distance Measurement

There are two common techniques for measuring the distance to an object given a reference point [45]:

- **Time-of-flight.** This technique measures the time $t$ taken for some signal to travel between two points (reference point and object or vice-versa). If the speed of the signal is $v$, the distance $d$ is given by $d = v \times t$. One example of this technique is the GPS [46]. In our solution this method is used since we use the time of flight of the ultrasound signal taken from the transmitter to the receiver. In our case one knows the time of departure in the receiver based on an RF message sent at the same time as the ultrasound signal from the receiver (synchronization). Assuming that the speed of light is much greater than the speed of sounds, one can calculate the time-of-flight.

- **Time-of-Arrival.** A TDOA-based system as discussed, measures the distance between two or more given reference points, when a signal emitted arrives at those given points with a time difference. With this time difference it is possible to calculate the distance between the reference points and the object transmitting the signal knowing the speed of sound.

### 2.1.2 Methods

This sub-section discusses different methods used for determining the position of an object.
• **Triangulation.** The method of measuring the angle to a given object from at least two known reference points to determine its position is known as triangulation. See [51] for more details. The use of triangulation requires the ability of knowing the angle between object and reference points, which can be done by using microphones as sensors for example. In order to determine the object in 3D one needs to have three reference points.

• **Trilateration.** Trilateration is a method of determining the relative positions of objects using the geometry of triangles in a similar fashion as triangulation. Trilateration uses the known locations of two or more reference points, and the measured distance between the subject and each reference point to accurately and uniquely determine the relative location of a point on a 3D plane using trilateration alone, generally at least 4 reference points are needed [52]. Since this method is the one that is going to be used in this work one should discuss it further.

Assuming that one needs to achieve a 2D position in space, it is necessary to have at least 3 reference points. As explained before, the distance between each reference point and the object has to be known and is denoted by $d_i$ where $i$ is the reference point number. So for each reference point $i$ one has:

$$d_i = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2 + (z_0 - z_i)^2}$$

(2.1)

and assuming we have three reference points we can generate a equation system of three equations which we have to solve with respect to three variables $x_0, y_0$ and $z_0$. If one solves this equation the result will not be unique, having two values for the z coordinate. In our case we can assume that the robot is always in the $z_0 = 0$ plane, which makes us having three equation with only two variables giving a exact $(x_0, y_0)$ solution. One should refer that for this method to hold the three points cannot have the same $x$ or $y$ coordinate at the same time since if that occurs the solution cannot be achieve since one of the circles will not intersect the other two. In Figure 2.1 one can see the illustration of the previous method. One should notice that is not necessary to have the points placed on the axis $x$ and $y$ as illustrated in the picture. They can take any position but always with the constrained of not being all in the same line plane $x$ or $y$. 

15
The equation system was solved by the MATLAB Symbolic Math Toolbox when implemented in the ultrasound based system described in Chapter refch:setup.

- **Multilateration.** Multilateration, also known as hyperbolic positioning, is the process of locating an object by accurately computing the time difference of arrival (TDOA) of a signal emitted from the object to three or more receivers. It also refers to the case of locating a receiver by measuring the TDOA of a signal transmitted from three or more synchronised transmitters. Multilateration should not be confused with trilateration, which uses distances or absolute measurements of time-of-flight from three or more sites [53]. To determine the position of an object in a 3D space four reference points should be used.

One can summarize this Section putting forward the techniques and methods that are going to be used in this work:

- **Landmark Based System with one object (ultrasound transmitter) and 16 reference points (ultrasound receivers).**

- **Distance measurement technique performed with Time-of-flight calculation between the transmitter and receiver.** Is used an RF signal in order to perform the synchronization of the receiver node. The time-of-flight is then the time that the ultrasound signal takes from the transmitter node to the receiver node. This will suffer further discussion in Section 3.2.1.
• Trilateration method for computing the position given the Time-of-flight distance measurements.

2.2 Dynamical Models

In order to describe the dynamics of the mobile agent two different state space models are proposed for evaluation.

One has to consider that since no real robot was applied on this work, there were only designed linear models that are approximations of robots or Radio Controlled cars non-linear models seen in [43] and [44]. As a result from this the estimated position given the models will not be optimal. It was just used linear approximation models since the objective of this work was to use the same approach as in [8], which was made to cope with linear systems. For non-linear systems one had to take other considerations when designing the filter, which was not the objective of this work. Next the two different models are described.

In order to model the dynamics of the mobile agent one should first define that the sensors are used at each time step according to the scheduling defined in Figure 1.1. It were defined the sets \( T_{hq} \) and \( T_{lq} \) for each scheduling approach that is, when \( k \in T_{hq} \) the high quality sensor is used, and when \( k \in T_{lq} \) the low quality sensor is used. The way both sets are defined for the different scheduling approaches are shown in Section 2.5. Next the two different models are described.

2.2.1 Model 1

The first model assumes that the plant describe a random walk movement in terms of position, i.e., the movement of the current step varies randomly in direction from the movement of the previous one by a gaussian white process noise \( w \in \mathbb{R}^m \) with zero mean and non-zero variance. It is assumed that the plant we measure is a 1st order and linear plant,

\[
\begin{align*}
x(k+1) &= Ax(k) + Bw(k), \quad k \geq 0, \\
y_1(k) &= C_1 x(k) + v_1(k), \quad k \in T_{lq} \\
y_2(k) &= C_2 x(k-d) + v_2(k), \quad k \in T_{hq}
\end{align*}
\]  

with state vector \( x(k) \in \mathbb{R}^n \), measurements \( y_1(k), y_2(k) \in \mathbb{R}^p \) with gaussian white measurement noises \( v_1(k), v_2(k) \in \mathbb{R}^p \). The covariance of the process noise is \( \mathbb{E}w(k)w(k')^T = W\delta(k-k') \), and the covariances of the measurement noises \( \mathbb{E}v_1(k)v_1(k')^T = \Sigma\delta(k-k') \), and \( \mathbb{E}v_2(k)v_2(k')^T = \sigma\delta(k-k') \).
It is assumed that the high-quality sensor measurement $y_2(k)$ is more accurate than $y_1(k)$, i.e., $\sigma < \Sigma$, but it is delayed by $d$ samples because of an higher processing time. It is assumed that the delay of the low quality sensor can be neglected since its processing time is lower than one time step. Note that $y_1(k)$ is not defined when $k \in T_{hq}$ and $y_2(k)$ is not defined when $k \in T_{lq}$.

Note that the dimensions of the measurements $y_1(k)$ and $y_2(k)$ can have different sizes, i.e. $p_1 \neq p_2$.

Also the values taken by $A = B = C_1 = C_2 = 1$ in order to be a random walk.

Figure 2.2 shows a possible result when this model is applied on a mobile agent placed at a starting position (0,0), moving with a gaussian white process noise with zero mean and variance equal to 10. This means that the mobile agent is expected to have a velocity variance for each time step of 10cm. One can see that it shows a total random walk with no connection on the direction of movement between a previous step and the following one.

2.2.2 Model 2

The second model shows the case when instead of having a random walk in position, the mobile agent moves describing a random walk in velocity i.e. his velocity on the current step varies randomly from the velocity on the previous step but the movement direction has a small variation. It is assumed that the plant we measure is a 2nd order and linear plant,
Figure 2.3: Mobile agent moving according to model 2 with gaussian white process noise with zero mean and variance of $1cm^2/step$. Starting position at $(0,0)$ with zero velocity.

\[
\begin{pmatrix}
  x \\
  \nu
\end{pmatrix}_{k+1} = A \begin{pmatrix}
  x \\
  \nu
\end{pmatrix}_k + Bw(k), k \geq 0, \quad (2.5)
\]

\[
y_1(k) = C_1 \begin{pmatrix}
  x \\
  \nu
\end{pmatrix}_k + v_1(k), k \in T_{lq} \quad (2.6)
\]

\[
y_2(k) = C_2 \begin{pmatrix}
  x \\
  \nu
\end{pmatrix}_{k-d} + v_2(k), k \in T_{hq} \quad (2.7)
\]

, where \( A = \begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix} \), \( B = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \) and \( C_1 = C_2 = \begin{pmatrix} 1 & 0 \end{pmatrix} \), where \( h \) is the step time. The other variables have the same characteristics as the ones presented for the first model.

The result when applying this model for a mobile agent with initial position at $(0,0)$ and initial velocity equal to zero, can be seen on Figure 2.3. The variance of the process noise \( W = 1 \), which means that the acceleration varies with a zero mean and variance $1cm^2$ each step. As it was expected the direction of the movement varies slowly, but one has random variations on the acceleration of the agent.

### 2.3 Kalman Filter

In this Section will be given an introduction to Kalman filter. Assuming a Linear Gaussian state space model (LGSSM),
\[ x_{k+1} = Ax_k + Bw_k, \quad w_k \approx \text{white } \mathcal{N}(0, W_k) \quad (2.8) \]
\[ y_k = C_kx_k + v_k, \quad v_k \approx \text{white } \mathcal{N}(0, V_k) \quad (2.9) \]

and assuming that the parameters \( A_k, B_k, C_k, W_k \) and \( V_k \) are known. Assume \( x_0, v_k, w_k \) are mutually independent and also that \( x_0 \approx \mathcal{N}(\bar{x}_0, P_0) \). The aim of the Kalman filter is to compute the optimal state estimate, i.e.
\[ \mathbb{E}\{x_k|Y_k\} = \mathbb{E}\{Y_k\}, \quad \text{where } Y_k \text{ are all the observations until the current time step } k. \]
Figure 2.4 shows the block diagram of the state space model, equations (2.8) and (2.9).

Denote the Kalman filter state estimates as
\[ \hat{x}_{k|k} = \mathbb{E}\{x_k|Y_k\}, \quad (2.10) \]
\[ \hat{x}_{k+1|k} = \mathbb{E}\{x_{k+1}|Y_k\} \quad (2.11) \]
the Kalman filter covariance estimates as,
\[ P_{k|k} = \mathbb{E}\{ (x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})' \}, \quad (2.12) \]
\[ P_{k|k-1} = \mathbb{E}\{ (x_k - \hat{x}_{k|k-1})(x_k - \hat{x}_{k|k-1})' \}, \quad (2.13) \]

The Kalman filter equations in prediction form are:
\[ \hat{x}_{k+1|k} = (A_k - K_kC_k) \hat{x}_{k|k-1} + K_ky_k \quad (2.14) \]
\[ P_{k+1|k} = A_k \left[ P_{k|k-1} - P_{k|k-1}C_k^T \times [C_kP_{k|k-1}C_k^T + V_k]^{-1} C_kP_{k|k-1} \right] A_k^T + B_kW_kB_k^T, \quad P_{0|1} = P_0 \quad (2.15) \]
\[ K_k = A_kP_{k|k-1}C_k \left( C_kP_{k|k-1}C_k^T + V_k \right)^{-1} \quad (2.16) \]
Where equation (2.14) is the new state estimation, (2.15) is the Ricatti equation which calculates the error covariance matrix and (2.16) is the Kalman gain.

The Kalman filter is going to be used in order to have better estimates of the mobile agent position based on noisy measurements given by the sensors. It was chosen since it is known [12], that this filter minimizes the covariance matrix $P$ for all $k$ when the state space model is a LGSSP, which is true for model 1 and model 2.

### 2.4 Proposed Filter

In order to apply the Kalman filter to the models described in Section 2.2 one has to rewrite them to accommodate the time delay $d$.

Introducing then a new state vector $\bar{x}$ by,

$$\bar{x}(k) = \begin{pmatrix} x(k) \\ x(k-1) \\ \vdots \\ x(k-d) \end{pmatrix}$$  
(2.17)

Then the model becomes,

$$\bar{x}(k+1) = \bar{A}\bar{x}(k) + \bar{B}w(k),$$  
(2.18)

$$\bar{y}(k) = \bar{C}(k)\bar{x}(k) + \bar{v}(k),$$  
(2.19)

where,

$$\bar{A} = \begin{pmatrix} A & 0 & \ldots & 0 & 0 \\ I_n & 0 & 0 & 0 \\ 0 & I_n & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & I_n & 0 \end{pmatrix} \quad , \quad \bar{B} = \begin{pmatrix} B \\ 0 \\ 0 \end{pmatrix}$$  
(2.20)

$$\bar{C}(k) = \begin{cases} C_1 & k \in T_{lq} \\ C_2 & k \in T_{hq} \end{cases}$$  
(2.21)

$$E\bar{v}(k)\bar{v}(k+k')^T =: \bar{V}(k)\delta(k')$$  
(2.22)
\[ V(k) = \begin{cases} \sum, & k \in T_{lq} \\ \sigma, & k \in T_{hq} \end{cases} \] (2.23)

The system defined above in equations (2.19) to (2.23) is a linear time-periodic system of period N. The periodicity comes from having periodic sensing. After defining the new model one has to define the minimal possible covariance \( \bar{P}^* (k) \) (*) denotes minimal) of the estimation error that satisfies the time-varying recursive Riccati equation of the form

\[
\bar{P}^*(k+1 \mid k) = \bar{A} \left[ \bar{P}^*(k \mid k) - \bar{P}^*(k \mid k) \bar{C}(k)^T \right] \\
\times \left[ \bar{C}(k) \bar{P}^*(k \mid k) \bar{C}(k)^T + \bar{V}(k) \right]^{-1} \bar{C}(k) \bar{P}^*(k \mid k) \\
+ \bar{A}^T + \bar{B} \bar{W} \bar{B}^T \tag{2.24}
\]

where \( \bar{P}^*(k) \) is the covariance of the estimation error of the state \( \bar{x}(k) \).

The time-varying Kalman filter that achieves the optimal accuracy \( \bar{P}^*(k) \) is given by

\[ \hat{x}(k+1) = (\bar{A} - \bar{K}(k) \bar{C}(k)) \hat{x}(k) + \bar{K}(k) \bar{y}(k). \tag{2.25} \]

and

\[ \bar{K}(k) = \bar{A} \bar{P}^*(k) \bar{C}(k)^T \left( \bar{C}(k) \bar{P}^*(k) \bar{C}(k)^T + \bar{V}(k) \right)^{-1} \tag{2.26} \]

where \( \hat{x}(k+1) \) is the new state estimate and \( \bar{K}(k) \) is the Kalman gain.

There are properties of Kalman filter [11] that are interesting for the type of problem that is posed, such as:

- Kalman filter is a linear, discrete-time, finite dimensional system.
- Covariance \( P_{k\mid k} \) can be precomputed if matrix C is independent of the measurement. In the case of using two different sensors this does not happen since the matrix C changes if \( k \in T_{lq} \) or \( k \in T_{hq} \).
- Steady State Kalman Filter. If A,B,C,W and V are time-invariant, then under stability conditions K and \( P_k \) converge to a constant. In the case of using two sensors the matrix C could change, so only sometimes this property can be applied. If the switching is periodic, as is presented in Sections 2.5.1 and 2.5.2, the average of \( P_k \) converges to a constant.
- Amongst the class of linear estimators the Kalman filter is the minimum variance estimator.
2.5 Scheduling

In this Section is presented the scheduling problem that is necessary to solve and the two possible approaches.

Since there exists two types of sensors with different characteristics and one need to perform a trade-off between communication cost for the high-quality sensor and estimation quality, the scheduling of them has to be performed. To perform the scheduling a performance criterion in order to enable a defined switching transition is necessary.

In the following subsections will be presented two different approaches of achieving an optimal scheduling based on the trade-off mentioned above. There will be presented an offline approach and covariance-based one. Both characteristics and illustrative examples will be provided. Conclusions about the best approach will be put forward.

As estimation quality criterion was chosen the average trace of the covariance of the estimation error over the time interval \([0, k]\),

\[
p_{\text{average}}(k) := \frac{1}{k+1} \sum_{i=0}^{k} \text{trace}P(i | i)
\]  

(2.27)

The objective is then to minimize \(p_{\text{average}}(k)\), with a proper choice of the high-quality sensor usage, since it is a measure of how accurately one knows the state and takes into account the measurements performed by both sensors. Now one can define a performance criterion \(V^*(k, M)\) which is the sum of the average communication cost and the average error covariance,

\[
V_T(k, M) := \frac{M}{k} \times \lambda + p_{\text{average}}(k)
\]  

(2.28)

where \(M\) is the number of times the high-quality sensor is used on the interval \([0, k]\) and \(\lambda\) is the communication cost. One should notice that even though the cost function is denoted during this work in terms of \(M\), its value can be written in terms of \(N\) such as \(M = \frac{k+1}{N}\), where \(N\) is the high-quality sensor periodic switching.

We would like to minimize the criterion (2.28) with respect to the high-quality sensor usage i.e.,

\[
\min_{|T_{hq}|=M} V_T(k, M)
\]  

(2.29)

, for all \(k\), where \(|T_{hq}|\) is equal to the number of elements in the set \(T_{hq}\).

As explained in [8] one can compute \(p_{\text{average}}^*(k)\) by computing the Riccati equation (2.24), since \(p_{\text{average}}^*(k)\) converges to a constant \(p_{\text{average}}^*\) for large \(k\). We will usually discuss only this limiting value.
One can compute the periodic solutions $\bar{P}(k)$ and $\bar{K}(k)$ by just iterating equation (2.24) and (2.16) because of the global convergence property given in [8]. There are two case of special interest: $p_{\text{average}}^*(1)$ and $p_{\text{average}}^*(\infty)$. Both these cases collapse into time-invariant problems, and correspond to the cases when the high-quality or the low-quality sensor, respectively, is used all the time.

One may think that $p_{\text{average}}^*$ is a decreasing function of $M$. The assumption that being the high-quality sensor more accurate, and more often used ($M_{\text{high}}$), the better estimation is achieved is not correct, since there is a time delay $d$ involved when the high-quality sensor is used. This reason makes the high-quality measurements being $d$ times older than the measurements given by the low-quality sensor. If the process noise into the system is sufficiently large (W large) i.e. the trust on the system model is low, then we can get $p_{\text{average}}^*(M = 0) < p_{\text{average}}^*(M = \infty)$, which means that using the low-quality sensor all the time gives better estimates of the position.

One legitimate question to make then is that if it is useful to use the high-quality sensor due to having old measurements. This is discussed in Section 2.5.1 and 2.5.2 for both schedulers presented and is shown that scheduling sequences with $M \neq \infty$ and $M \neq 0$ are given for both scheduling approaches.

### 2.5.1 Offline scheduler

In the offline scheduling approach is assumed that the sets $T_{hq}$ and $T_{lq}$ are defined as follows:

\[
T_{hq}(N) = \{N - 1, 2N - 1, 3N - 1, \ldots \} = \{k \geq 0 \ (k + 1) \mod N = 0\},
\]

\[
T_{lq}(N) = \{1, 2, \ldots, N - 2, N, \ldots \} = \{k \geq 0 \ (k + 1) \mod N \neq 0\},
\]

where the period $N \geq 1$. That is, when $k \in T_{hq}(N)$ the high quality sensor is used, and when $k \in T_{lq}(N)$ the low quality sensor is used.

It was used the scheduler presented in [8] and it is intended to reach a $N$ periodic switching, in which the value of $N$ is pre-calculated offline. Now let us define the way the value of $N$ is obtained. As it is seen in (2.30) one can say that, at a time $k$, the optimal sensor cycle period is given by,

\[
N^*(k) = \arg \min_N \left( \frac{M}{k} \times \lambda + p_{\text{average}}^*(k, N) \right), \tag{2.30}
\]
where $p_{\text{average}}^*(k, N)$ is characterized as in Equation (2.27) but now his value is taken over a given period switching $N$ and where $M$ is the sensor usage and in the offline case is given by, $M = \lceil \frac{k+1}{N} \rceil$. It is clear that $1 \leq N^*(k) \leq k + 1$, so that (2.30) is a simple minimization problem over a finite set.

The steady-state ($k \to \infty$) optimal period $N^*$ for the sensor schedule is given by

$$N^* = \arg \min_N \left( \frac{M}{k} \times \lambda + p_{\text{average}}^*(N) \right)$$

(2.31)

$$= \arg \min_N V_T^*(N)$$

(2.32)

One can see that $N^*$ can be easily calculated with (2.32). One just need to calculate $p_{\text{average}}^*(N)$ for a delay $d$ with (2.27). After this was possible to calculate the value of $V_T^*(N)$ given by (2.28), with a proper $\lambda$ for a given interval of $N$, and then achieve the minimum of this function, which gives the $N^*$.

Next will be presented an example that illustrates the offline scheduler for both model 1 and model 2.

**Examples**

Assuming that the parameters for model 1 and 2 are the ones presented in Sections 2.2 and 2.4.

Assuming that $P^*(k) \in \mathbb{R}$. One can see that $P^*(k) \in \mathbb{R}$ since $P^*(k) = \bar{P}_{(0,0)}^*(k)$ where $\bar{P}_{(0,0)}^*(k)$ is the (1,1)-block of (2.24). As seen it is only when $k \in T_{hq}(N)$ that more information than $\bar{P}_{(0,0)}^*(k)$ is needed from $\bar{P}^*(k)$.

The values of the functions $p_{\text{average}}^*(k, N)$ and $V^*(N)$ will be given by equations (2.27) and (2.28), and $P^*(k, N) = \bar{P}_{(0,0)}^*(k, N)$.

In order to perform the offline scheduling for both models, the relationship between the process noise $W$ and the delay of the high-quality sensor $d$ was achieved. This was done in order to check, for a given interval of $d \in [3, 4, 5, 6, 7, 8]$ and $W \in [0, 10]$, where the use of high-quality measurements improves the estimation. So one had to check when $N^*(W, d) \neq \infty$. The results are shown in table 2.1. For each value of $d$ is identified the highest value $W_{\text{max}}$ in order for the high-quality measurements to be useful. The parameters for the sensors measurement noise are $\Sigma = 12$ and $\sigma = 1$, which represent the error variance for the low-quality sensor and high-quality sensor respectively. This values were achieved with tests made for each sensor and are presented on Chapter 4. Also the reason for the values of the delay from $[0, 2]$ not being taken into account is due to the fact that is impossible (as
it will be seen in Chapter 4) to have a delay lower than three for the high-quality sensor used. One can see that the value of $W_{\text{max}} = \infty$ when there is no delay, since using the high-quality sensor is the optimal solution and the maximum process noise can be any value.

As it can be seen in table 2.1 the values of $W_{\text{max}}$ decrease for increasing delay, which was expected since if the information received is older, the vehicle dynamics should vary slower. For each model the value of the process noise has different interpretation as one could see on Section 2.2. So, for first-order model 1 the process noise $W$ shows the variation in the velocity of the agent. For the second-order model 2 the process noise $W$ means shows the variation in the acceleration of the agent. So the values shown in table 2.1 cannot be directly compared. As one could see the maximum variation on the process noise for model 1 is between $[0.2, 1]$ m/s and for model 2 is between $[0.002, 0.09]$ m/s$^2$ for the referred delays. For the following examples one will just make reference to the cases of the delay being 3s and 7s. This choice was made since the lowest processing time needed to take a picture, analyse it and achieve the robot position, which is the delay $d$, is 3s as it will be seen in Chapter 4. The choice to analyse 7s comes to have a comparison from a higher delay value $d$ and evaluate how the system copes with it.

### Example: Model 1

For the tests made with model 1 the parameters were adjusted as follows:

- Process noise $W = 0.2$ which is the highest for the higher delay. It is assumed to be better to use a lower process noise to cope with the worst case. This value as explained before means the variance on the velocity of the agent.

- Measurement noises $\Sigma = 12$ and $\sigma = 1$. The accuracy of the high-quality sensor is 12 times greater that the low-quality one.

<table>
<thead>
<tr>
<th>$d$</th>
<th>$W_{\text{max}}$ Model 1</th>
<th>$W_{\text{max}}$ Model 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1.0</td>
<td>0.09</td>
</tr>
<tr>
<td>4</td>
<td>0.6</td>
<td>0.03</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>0.01</td>
</tr>
<tr>
<td>6</td>
<td>0.3</td>
<td>0.006</td>
</tr>
<tr>
<td>7</td>
<td>0.2</td>
<td>0.003</td>
</tr>
<tr>
<td>8</td>
<td>0.1</td>
<td>0.002</td>
</tr>
</tbody>
</table>

Table 2.1: Maximum values of process noise $W$ for a given delay $d$ for each model.
• $\hat{P}^*(0) = 0$ if we know exactly the position of the object on the beginning of the test and a high value for the estimator on the first iteration to believe more on the first measurement taken. In the case of the example is used $\hat{P}^*(0) = 0$ since is assumed that the position of the agent is not known.

Next is shown in Figure 2.5 the function $p^\ast_{\text{average}}(N)$. As it can be seen is not at all the case that decreasing $N$ always yields a more accurate average estimate. It is seen for a delay $d = 3$ the optimal switching period $N^* = 1$, but for a $d = 7$ the optimal switching period is now $N^* = 7$. One should remember that this function just evaluates the performance in terms of estimation accuracy not taking into account any communication cost $\lambda$. It is also seen that all the curves converge to the same value $p^\ast_{\text{average}}(M = 0)$. This is because the high-quality sensor is not used at all when $N = \infty$, and the low-quality sensor has no delay.

How the covariance $P^*(k, N)$, $N = 1, 7, \infty$, evolve over time for delay $d = 7$ is shown on Figure 2.6. As expected $P^*(k, N)$ converges to periodic trajectories. As it can be seen also its better to use $N = 7$ for delay $d = 7$ because the covariance value $P^*(k, N)$ takes lower values. As one can expect for $d = 3$ when $k$ evolves, the value of $P^*(k, N)$ is lower when $N = 1$.

In Figure 2.7 we find the optimal sensor cycle period $N^\ast$. Now is shown the value of $V^\ast(M)$ for a communication cost $\lambda = 0.3$. This value was chosen in order to show that even if the accuracy of the average estimate is better for $N = 1$ when $d = 3$, for this given communication cost, it is better to
use a periodic switching with $N = 2$ instead, in order to have an optimal performance cost $V^*$. Also one can notice that now for a delay $d = 7$ it is better to not use the high-quality sensor at all even if the estimation accuracy is improved when using $N = 7$. For other results see [8].

**Example: Model 2**

For the tests made with model 2 the parameters were adjusted as follows:
• Process noise $W = 0.003$ which is the highest for the higher delay. It is assumed to be better to use a lower process noise to cope with the worst case. This value as explained before means the variance on the acceleration of the agent.

• Measurement noises $\Sigma = 12$ and $\sigma = 1$. The accuracy of the high-quality sensor is 12 times greater that the low-quality one.

$\bar{P}^*(0) = 0$ if we know exactly the position of the object on the beginning of the test and an high value for the estimator on the first iteration to believe more on the first measurement taken. In the case of the example is used $\bar{P}^*(0) = 0$ since is assumed that the position of the agent is not known.

Next is shown in Figure 2.8 the function $p_{\text{average}}^*(N)$ for two different delays $d$ using model 2.

Figure 2.8: The function $p_{\text{average}}^*(N)$ for two different delays $d$ using model 2.

• $\bar{P}^*(0) = 0$ if we know exactly the position of the object on the beginning of the test and a high value for the estimator on the first iteration to believe more on the first measurement taken. In the case of the example is used $\bar{P}^*(0) = 0$ since is assumed that the position of the agent is not known.

Next is shown in Figure 2.8 the function $p_{\text{average}}^*$. Again, as was shown in the example for the first model, is not at all the case that decreasing $N$ always yields a more accurate average estimate. It is seen for a delay $d = 3$ the optimal switching period $N^* = 1$, but for a $d = 7$ the optimal switching period is now $N^* = 5$. One should remember that this function just evaluates the performance in terms of estimation accuracy not taking into account any communication cost $\lambda$. It is also seen that all the curves converge to the same value $p_{\text{average}}^*(\infty)$. This is because the high-quality sensor is not used at all when $N = \infty$, and the low-quality sensor has no delay.

How the covariance $P^*(k, N)$, $N = 1, 5, \infty$, evolve over time for delay $d = 7$ is shown on Figure 2.9. As expected $P^*(k, N)$ converges to periodic
Figure 2.9: The function $P^\ast(k, N)$ for different periods $N$ when delay $d = 7$ using model 2.

trajectories. As it can be seen also its better to use $N = 5$ for delay $d = 7$ because the covariance value $P^\ast(k, N)$ takes lower values. As one can expect for $d = 3$ when $k$ evolves, the value of $P^\ast(k, N)$ is lower when $N = 1$.

In Figure 2.10 we find the optimal sensor cycle period $N^\ast$. Now is shown the value of $V^\ast(N)$ for a communication cost $\lambda = 0.3$. This value was chosen in order to show that even if the accuracy of the average estimate is better for $N = 1$ when $d = 3$, for this given communication cost, it is better to use a periodic switching with $N = 2$ instead, in order to have an optimal performance cost $V^\ast$. Also one can notice that now for a delay $d = 7$ it is better to not use the high-quality sensor at all even if the estimation accuracy is improved when using $N = 5$. For other results see [8].

One can expect then in the experimental validation that when using the offline scheduler the accuracy of the estimated position be higher when using the 1st model instead of the second for the cases of period $N = \infty$ and $N = 2$ for the parameters explained before. For $N = 1$, i.e, when the high-quality sensor is used all the time one can say that the accuracy when using the second model is improved. All this conclusions are based on the values given by $p_{\text{average}}^\ast$ for these different $N$ values when comparing both models.

2.5.2 Covariance-Based scheduler

The first approach discussed was the offline scheduler were the sensor switching was assumed to be periodic. In this Section, we instead let the sensor
switching be based on how much increase in accuracy one can get from using a particular sensor at that time instance. We call this covariance-based scheduler or covariance-based switching as presented in [8].

The method is based on knowing the value of the iteration of the Riccati equation (2.24) at max\(D\) steps ahead for all possible switching combinations. In order to perform this one no longer can use the model based on the periodicity \(N\) as defined in Section 2.5.1. Now is defined a switch schedule \(s(k)\). The objective of this method is to find first the value of a function \(f\) where \(f = \min \bar{P}_{0,0}^*(k + maxD)\). So one does a tree search of the lowest value of \(\bar{P}_{0,0}^*\) at the tree depth equal to \(maxD\) at a given instant \(k + maxD\). Figure 2.11 illustrates this.

So for \(maxD = 2\) one has to look at four different values of \(\bar{P}_{0,0}^*\) for \(k = k + maxD\). After being known the minimum value for \(\bar{P}_{0,0}^*\) at \(k = k + maxD\) one can know the sensor switching path that has to be taken from instant \(k\) to \(k + maxD\). If there are equal minimum \(\bar{P}_{0,0}^*\) values for \(k = k + maxD\) one should choose the minimum cost path. Here one just uses the value of \(P_{0,0}(k)\) to calculate \(p_{average}(k)\) following the same reasons as the ones followed in Section 2.5.1. So taking the same assumptions as in Section 2.5.1, one will have a communication cost \(\lambda\) whenever the high-quality sensor is used. In the end, one will have all the paths with distinct costs and then one chooses the minimum. This algorithm is run at each \(k = (num \ast (maxD)) + 1\), being \(num = 0\) for \(k = 1\), and at each time a new search is made the \(num\) value is increased by one. So, for instance if \(maxD = 2\), then we look ahead
at \( k = 1, 3, 5, (...) \). The chosen path is then stored in the set \( T_{hq} \) and \( T_{lq} \) and then the action taken for sensor switching is based on these sets. It is assumed that the first sensor to be used is the low-quality sensor. Figure 2.12 shows the flow diagram that illustrates the discussed algorithm.

Now let us then define the switch schedule \( s(k) \) as

\[
s(k) = \begin{cases} 1, & T_{lq} \\ 2, & T_{hq} \end{cases}
\]  

and the following \( \bar{C} \) and \( \bar{D} \) matrices are used in the Riccati equation 2.24.

\[
\bar{C}(k) = \begin{cases} \begin{bmatrix} C_1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & \ldots & 0 & C_2 \end{bmatrix}, & s(k) = 1 \\ \Sigma, & s(k) = 2 \end{cases}
\]

\[
\bar{V}(k) = \begin{cases} \Sigma, & s(k) = 1 \\ \sigma, & s(k) = 2 \end{cases}
\]

For a given initial covariance \( P^*(0) \), the schedule \( s(k) \) can then be computed on-line.

**Examples**

Next one will show with an example for each model what is the behavior of the covariance-based scheduler comparing to the offline one. It will be run
Figure 2.12: Flow diagram of the covariance-based switching algorithm.

the covariance-based scheduler for maxD ∈ [1, 10] for the same conditions assumed for the examples of the offline scheduler and a brief discussion will be made. Considerations about the number of times the high-quality sensor and the difference on the p\textsuperscript{*\text{average}} for both schedulers will me made also.

In order to cope with the trade-off between communication cost and estimation quality one need to have a cost function as was discussed in Section 2.5.1. The function p\textsubscript{average}(k) has the same interpretation given in (2.27) which is the average trace of the covariance matrix P(k). As assumed in the examples of Section 2.5.1 one will just use the value of P\textsubscript{0,0}(k) to calculate p\textsubscript{average}(k).

If one wanted to be precise the computation time taken for higher maxD should be also taken into account. But until maxD = 10 it was seen that the computation time for each tree search is less then 0.1s so it can be neglected.

One should refer again that the parameters used were achieved in Section 2.5.1. It is reasonable to think that there are values of maxD, that for higher relation W/d then the ones establish in Section 2.5.1, gives scheduling sequences that the webcam is used to perform localization. The achievement of the boundary for W/d was not calculated for this report but can be seen
Figure 2.13: The function $p_{\text{average}}(k, maxD)$ and the usage of the high-quality sensor for the covariance-based scheduling when delay $d = 3$ and model 1.

as future work.

**Example: Model 1**

In this first example one will deal with the model one designed in Section 2.2. For this model one will compare the performance in terms of $p_{\text{average}}(k)$ and cost function $V_T$ for the given process noise $W$ and delay $d$ that was considered in the examples for the offline scheduler. Figure 2.13 shows the function $p_{\text{average}}(k)$ for different values of $maxD$ when delay $d = 3$ for a process noise $W = 0.2$. So in this case it is not considered any communication cost. At each $maxD$ is shown the number of times the high-quality sensor is used. There were made 200 iterations when calculating $P(k)$, so $k_{\text{max}} = 200$. As one can see in Figure 2.13 the covariance-based scheduler performs better for almost all values of $maxD$ in terms of estimation quality. Also one can see that the number of times the high-quality sensor is used is lower then the offline case ($N^* = 1$ means $M=200$). So one can conclude that even adding any communication cost the performance criterion $V_T$ will always be smaller for the covariance-based case.

In Figure 2.14 is shown the case where the delay $d = 7$. As it was seen in 2.5.1 the optimal switching periodicity $N^* = 7$. It is shown that the function $p_{\text{average}}(k)$ always takes smaller values in the offline case. As it can be seen also since the optimal periodicity $N^* = 7$ the value of $M = 28$, so is impossible to get a better performance $V_T^*$ using the covariance-based
Figure 2.14: The function $p_{\text{average}}(k; maxD)$ and the usage of the high-quality sensor for the covariance-based scheduling when delay $d = 7$ and model 1.

scheduler given this parameters of $W$ and $d$. Also one should notice that using $maxD = 1$ or $maxD = 5$ gives us the same $p^*_{\text{average}}$ and same number of high-quality sensor usage.

**Example: Model 2**

Now one will describe the comparison between the offline and covariance-based schedulers when applied to model 2 describe in Section 2.2. As for the first example one will take the same parameters described in Section 2.5.1.

The first example taken will be for $d = 3$ and process noise $W = 0.003$ and the results are shown in Figure 2.15. As one can see it is not possible to achieve with the covariance-based scheduler, better estimations then the ones given by the offline scheduler. Also one can note that the lowest value of the function $p_{\text{average}}(k)$ is for $N = 1$ for the offline case. Since the number of times the sensor is used on the covariance-based case is always lower then $k_{\text{max}}$ one can already foreseen that when calculation the performance criterion with a certain communication cost it will be better to use the covariance-based scheduler instead of the offline one for delay $d = 3$. This is shown in Figure 2.16 where it can be seen that for $maxD = 2$, i.e., doing a 2 depth tree search one can get better estimates using the web-camera then not using it.

The value used for the communication cost $\lambda = 0.3$. Figure 2.18 illustrates the function $p_{\text{average}}(k)$ when the delay $d = 7$ and the process noise $W = 0.003$.  

35
Figure 2.15: The function $p_{\text{average}}(k, \text{maxD})$ and the usage of the high-quality sensor for the covariance-based scheduling when delay $d = 3$ and model 2.

Figure 2.16: The performance function $V^*(k, \text{maxD})$ and the usage of the high-quality sensor for the covariance-based scheduling when delay $d = 3$ and model 2.
Figure 2.17: The function $p_{\text{average}}(k, \text{maxD})$ and the usage of the high-quality sensor for the covariance-based scheduling when delay $d = 7$ and model 2

As it can be seen for $N^* = 5$ in the offline case the value of times that the high-quality sensor is used is $M = 40$, since $k_{\text{max}} = 200$. Also as one can see the estimation quality is higher when used the offline scheduler. But as seen in this Section in the other examples, since for $\text{maxD} = 10$ the value of $M = 38$, one can find a given value for the communication cost $\lambda$ that makes the performance criterion $V^*_T(k, \text{maxD})$ lower when using the covariance-based scheduler. It can be seen that for $\lambda = 0.002$ this situation occurs. This is shown in Figure 2.18. Here one can conclude that for the first model is not possible to achieve better performances when using the covariance-based scheduler in comparison with the offline one when delay $d = 7$. On the other hand using both models when delay $d = 3$ is possible to improve the system performance by using an covariance-based version instead of an offline one. Also for the second model this is true for a high delay $d = 7$.

Table 2.2 shows the comparison between both scheduling approaches when using the different models and different communication costs. It is seen that the covariance-based approach is better then the offline one for both models when the communication cost was set to be $\lambda = 0.3$ since the covariance-based scheduler gave scheduling sequences with lower high-quality sensor usage. It was also seen that in terms of estimation accuracy $p_{\text{average}}$, for the optimal scheduling sequence when using model 1 and model 2, model 1 was the one that gave lower $p_{\text{average}}$ values, i.e. more accurate estimations.
Figure 2.18: The performance function $V^*(k, \text{maxD})$ and the usage of the high-quality sensor for the covariance-based scheduling when delay $d = 7$ and model 2.

<table>
<thead>
<tr>
<th>Model 1</th>
<th>Model 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W=0.2$</td>
<td>$W=0.003$</td>
</tr>
<tr>
<td>$\lambda = 0$</td>
<td>Offline</td>
</tr>
<tr>
<td>$\lambda = 0.3$</td>
<td>Covariance-based</td>
</tr>
</tbody>
</table>

Table 2.2: Optimal sensor scheduling approach for model 1 and 2 considering different communication cost $\lambda$. 
Chapter 3

Experimental set-up

In this chapter is discussed all the implementations made in order to perform the experimental validation of all the localization algorithms and tools described in the previous Sections. The chapter is divided in two Sections. The first Section is the hardware description and finally the software. In particular is described the designed and developed wireless sensor network testbed, the ultrasound localization system, the vision based system and the mobile agent used. Figure 3.1 gives an overview of the operation of the network and all the hardware involved.

3.1 Hardware

3.1.1 Wireless Sensor Network Testbed

In this chapter is presented the work developed towards the design, develop and implementation of a Wireless Sensor Network Testbed. First is given an overview about WSNs and WSNs testbeds. Is presented the WSN state of the art followed by the design approach taken, its implementation and necessary validation.

Background

In this Section is presented an overview of WSNs and WSNs testbeds. One will focus on the state of the art of WSN testbeds all show their features, architectures, communication characteristics and the hardware and software used.
Wireless Sensor Networks

Nowadays all the systems used to perform a broad type of tasks such as producing lines, airplanes, cars, electric plants, satellites, health-care, a feedback control is applied [14]. Since the early 1930’s we have watched developments on engineering in order to do a better control of processes. This theories started to be called as ”classical control” being the control made by hardware and continuous time. In the middle 1950’s was introduced the ”digital control” where computers were the ones responsible for closing the loop (discrete control). After the 1980’s due to communication link improvements a new ”networked control” was used, where a group of computational units are used in a cooperative and decentralized fashion to perform a certain task. In the beginning of the 21st century, novel control theories were introduced in order to cope with a new technology, the wireless, originating ”wireless control” [15]. The main characteristic of these systems is that the link between sensors and controllers and controllers and actuators is made through wireless. In [17] is presented the development path of WSN since 1994 where
DARPA funded research on “Low power wireless integrated microsensor”, followed by an announcement in 2003 from MIT saying that the WSN is one of the 10 technologies that will have the highest influence on the future. We are currently in 2008 and from all the research being made in this area (IEEE Signal Processing, Robotics, Communications) one can see that a lot of work has to be done. As the forecast presented in [14] shows, the number of wireless sensors and embedded devices in the world will of more then 1 Trillion fitting the idea that we can connect everything using wireless networks. These networks can be applied to solve or/and ease tasks that are nowadays performed by cable solutions and are also creating novel applications. The applications can be as seen in [14], [17], [18] and are for example:

- Wireless mining ventilation control.
- Wireless control of flotation process (Industrial monitoring and process control).
- Vehicle fuel efficiency with networked sensing (automotive).
- Disaster relief support using mobile sensors.
- Surveillance with networked autonomous vehicles.
- Environmental monitoring (Terrestrial and aquatic monitoring).
- Habitat monitoring.
- Security and Defence of airports, stations, buildings, etc.
- Military - Information exchange, sniper detection, mine detection, etc.
- Domotics.
- Health care.

As advantages of the application of wireless networks one can easily see that the cost (wiring and installation work), flexibility (less physical design limitation, more mobile equipment, faster commissioning and reconfiguration) and reliability (no cable wear and tear and no connector failure) are reduced or completely removed. One can point out also disadvantages to this type of networks which are the current challenges of all the research community. Is shown in [14] that security, reliability, lack of knowledge, cost, lack of commercialized solutions, low and slow data transmissions are the main drawback of this technology at the moment. A lot of this features are due to
the characteristics of the wireless communications which are the less computational capability, low energy storage, large variations in connectivity, low bandwidth, delays and packet losses and the not well developed communication theory at the moment [15]. Also presented in [15] and [16] are given some solutions used in order to use WSN for performing control which are:

- Communication protocol suitable for control ([19], [20] and [21]).
- Control application that compensates for communication imperfections ([22], [23] and [24]).
- Cross-layer solution with integrated design of application and communication layers.

One can obviously see that in wireless control the communication and control are always together, and so solutions in both areas have to be searched in order to improve the WSNs when employed for this type of tasks.

In order for the research community to test their control algorithms, power management solutions and protocols on WSNs there were built WSNs testbeds. These testbeds are composed by an amount of wireless nodes which are deployed indoors or outdoors according to the needs of the test one wants to perform.

A wireless sensor network testbed was developed in KTH and its design, development and implementation is discussed in the following Sections.

State of the art

Figure 6.3 shows some WSN testbeds recently developed and successfully deployed and tested, in several locations and with different features and settings. The references shown are websites and [26]. Various possibilities can be seen regarding the design of the testbeds.

Features

Most testbeds share the possibility of

- remote programming and debugging
- monitoring and real-time interaction
- logging
- network administration and management
This subset of features allows running experiments with data collection and control over events. Additionally, batch mode, scheduling, quotas and support for multiple, simultaneous users are also common and interesting characteristics. Scalability is also a concern, although the importance given to it varies: it is extremely important for the SenseNeT developers [33] but not as much for the MoteLab [32].

**Architecture and Communication**

Typically, testbed architecture consists of a control station connected to one or more gateways, which provide the interface with the motes. Users connect to the testbed (possibly to all its levels) via the control station. Depending on the links used between the various levels, the testbed can have a more or less hierarchical setup. The choices of network devices and channels for communication also allow different levels of flexibility (see [35] for an example of a flexible testbed).

Starting at the top, the communication between users and the control station is normally done through an existing LAN and, possibly, through the Internet. This allows users to access the testbed by simply using a web browser to connect to the server set up in the control station.

The connection of the control station to the gateways is done with various technologies. If motes are acting as base stations, then USB (serial) is used [34, 36]. A particular and interesting case is that of the Deployment Support Network by ETH Zurich [27], where Bluetooth is used (via BTnodes [37]). Ethernet is used especially with gateways with some processing power and when such a backbone is already installed (see, for instance, [32]). If mobile nodes are used (e.g., robots) then 802.11 is used when gateways (e.g., Stargates) have such possibility [30, 31].

Motes connect to gateways depending on which type (and how many) of the last is deployed and on which tasks are expected to be performed over the back-channel. If a mote acting as a base station is used, then radio is the channel employed [33, 36]. Otherwise, USB is usually utilised [30, 35].

**Hardware and Software**

**Wireless nodes**

The motes usually chosen are Tmote Sky or Mica2/Z. There are other options, but they are not normally considered unless the testbed is required to support more than one kind of motes.

Motes almost invariably run TinyOS. For network-wide programming, Deluge [38] is commonly used.
Gateways

Gateways are often Tmote Connect, Crossbow MIB-600 or Stargates (running Linux). Sometimes, expansion cards are used to have other possibilities for the communication links, e.g., 802.11. This is common when robots - being the Acroname Garcia a common choice - carry motes, acting as mobile nodes.

Control station

Depending on the functionality required, the control station can be better or worse equipped. Linux is widely used. Web servers use Apache or others; databases are built with MySQL; scripts in various languages are used as a management or experimentation tool.

Design

In this Section is presented the steps performed to reach the design that fit the KTH S3 research team requirements. System Engineering tools were used.

System Breakdown Structure

The System Breakdown Structure (SBS) is a diagram built to organize, divide and set a hierarchy for the processes and products that comprise the system architecture. The SBS is then used to help project development and organisation of the team tasks. For the construction of this diagram, the IEEE standard presented on [26] was used. The SBS is part of the Systems Engineering Management Plan (SEMP), also described on that reference. The diagram presented on Figure 6.4 shows the SBS. The top level presents the identification of the system to be created: “Development of a WSN Testbed”. The second level corresponds to the products of this system that are the technological part of the system i.e. the WSN Testbed itself (in grey), and the systems engineering processes that will take place during the system life cycle. The third and last level shows all the components (sub-products) of the WSN Testbed. Under each product or sub-product all the tasks and topics related to them are detailed.

Requirements

This Section presents the requirements observed for the WSN Testbed. The fields of interest of the KTH S3 research team is first discussed and then the
requirements will be shown.

**WSN Testbed fields of action and demonstrations**

We are now going to describe the fields and the respective specific demonstrations that are going to be performed on the testbed.

- Multi-robot Systems
  - Localisation
  - Coordination
- Networking
  - Routing
  - Power Control
  - Radio Propagation Models/Measurements
- Control/Estimation over Wireless Networks
  - Consensus Filters
  - Control over wireless link
  - Tracking (of objects/persons)
- Security

**WSN Testbed Requirements**

The requirements pointed next are based on the state of the art analysis and the fields of interest listed in the previous Sections. So, in order to be considered a state of the art testbed, it should be capable of:

- Remote network-wide programming and debugging
- Event and (collected) data logging and trend monitoring
- Real-time interaction and batch experiments
- Network management and administration - link mapping, nodes status
- Support for multiple users with scheduling and quotas
- Easy to use - web based user interface
• Expandability - more and/or different sensors (including cameras)
• Scalability
• Source of power supply control and fault injection
• Actuation and mobile nodes - temperature, humidity and light controllers; robots; doors and windows actuators
• Generate different environments - meet Swedish industry/government needs

Additionally, the network should cover the corridors and possibly a few office rooms. Cost should also be minimised as well as human effort.

One also needed to have an external power supply that should meet the following requirements:

• Provide external power supply to the motes.
• Possibility to use batteries and external supply at the same time and to switch between them if desired.
• Easy to deploy in any place inside a building.
• Possibility to have different structural configurations and different number of motes.
• Reduced price.

Implementation

In this Section is discussed the solutions provided to implement the testbed, the power supply and the wireless programming.

Solutions

We divided our approach in two parts. One consists in the solution to be adopted at the moment, called basic, and the other is the complete solution. The basic solution is the one that will fulfill some requirements pointed out in Section 3.1.1. This solution will not fulfill the requirements where actuators are required. Moreover, some testbed capabilities will not be available since servers will not be put to work in this first part.
Basic Solution

The basic solution includes only a portion of the complete testbed, both in the fulfillment of requirements and in the amount of motes deployed. This solution was implemented to be used on this MsC thesis project. The proposed approach is as follows.

- Set up a group of motes in boxes (properly adapted) on the ceiling of the KTH S3 corridors (Network 1 - 16 motes) and MsC Students/Visitors room (Network 2 - 8 motes).
  Use the floor plan (see Appendix 6) to determine where to deploy the motes; look out for energy consumption (reduce transmission power) and possible traffic problems

- Configure two computers (running Linux) to act as the central computers i.e. where the mote acting as base station is connected. There were configured two computers in order to have two different networks working. This has to deal with the fact the two different projects had to be developed on the testbed.

- Have ultrasound sensors connected to motes and a mobile agent in order to perform localization on the testbed. The mobile agent should be an electric RC car since is a cheap solution to start. This is only available on network 1.

- Work with Deluge so that wireless, network-wide programming can be done

- Implement a sleeping mode for the motes i.e. complete or selective testbed power down functionality.

- Create radio messages for debugging (use ready-made software for logging).

In order to implement the required VCC power supply of 5V the following approach was taken. The developed solution was to use the USB connector of the Tmote Sky. Since the Tmote is already prepared to be supplied power through this connector (5V) and at the same time by batteries, without causing any harm to the mote, this was considered the optimal solution for our problem. In order to perform this task, a 5V VCC adapter is connected to a 1 meters long cable, which in each 1 meters as a female (type A) USB connector derivation, which then is connected to the Tmote supplying the 5V. Tests were made in order to analyse the integrity of the Tmote.
In each External Supply Network solution there has to be:

- $l$ m length Cable.
- $l/d$ USB female (Type A) connectors.
- $l/d-1$ Derivation connectors.
- One VCC adapter with output current capable of supply $l/d$ motes consumption. Each Tmote has a consumption of 23mA. Considering that sensors and actuators can be plugged in the tmote there is a need to have the right adapter to supply the Network. The cable adapter plug is capable to have different types of adapters.

The length $l$ and connector step $d$, will depend on the application. It was made a standard configuration of:

- 10m length cable ($l=10$).
- 1m connector step ($d=1$).

With this solution one can see that the user has the freedom to choose the desired configuration shape (circle, square, rectangle, etc. on 2D or 3D) and the number of motes for each application.

In Figure 3.2 is presented the schematic for a standard External Supply Network.
Complete Solution

This complete solution presented in this Section is based on the analysis of requirements presented on Section 3.1.1. The characteristics addressed in the basic solution should be maintained entirely, except for the increase in number of devices. Nevertheless, several more features are required than those, which increases the complexity of the system.

The architecture should be roughly the same as the one presented in the basic solution. Some motes should act as base stations, connected to computers via USB. These computers should act as control stations and be connected through the private Ethernet network (alone and/or with 802.11, if desired) already present in the floor.

The number of base stations and computers should be determined through an analysis of the floor plan, as with the basic solution. However, one computer should be configured to run as a web, database and management server; this shall be set as the central computer, where full control over the system is permanent.

The following actions should be possible to be performed through the web - using the user interface that is being developed - or locally.

- Experiment planning, scheduling and interaction
- Data storage and analysis
- Run the user interface which is currently being developed.
- Allow access to motes along with debugging and programming
- Network control, supervision and management

In terms of actuators the testbed should have

- Robots - 3 or more, due to localization needs; possibly Acroname Garcia. They are the preferred choice among the state of art testbeds
- Mote Cameras - CITRIC mote camera, see [39].
- Temperature, humidity and light actuators
- Doors and windows actuators for security issues and environmental state changes
Deployment

In Figure 3.3 one can see the current deployment of the testbed. There exist two different networks in order to develop different work. Network 1 is used to perform localization and network 2 is being used to test detection system algorithms and . The experimental validation presented in Chapter 4 was implemented on network 1. One can also see that there were installed three cameras, two control stations to control and program each network and a mobile agent. The characteristics of the cameras, mobile agent and the ultrasound sensors are presented in Section 4. Figure 3.4 and 3.5 shows the deployed sensor nodes on the corridor of the 6th floor of the Q building in KTH and the the wireless node Tmote Sky in casing with Ultrasound Sensor.

3.1.2 Ultrasound sensor

The ultrasound sensing system is based on having a cluster of ultrasound transmitter microphones that emits ultrasonic signals, an ultrasound receiver microphone which receives the signals sent by the other and two wireless nodes that interact with them. In order to interact with the sensor one had to develop a circuit for each in order to perform the signal conditioning.

The transmitter design structure is seen in Figure 3.6. It is assumed by testing that the ultrasound microphone span is a cone with an angle of 60°. There were placed six microphones separated by 60° (Figure 3.6, top view) covering all the area around the transmitter cluster. Also the microphones are placed with 30° displacement angle between the ground plane and each microphone centroid line as seen in Figure 3.6(side view) being then able to have a 3D coverage area of an half sphere. So with this design one can cover
Figure 3.4: Picture shows the network 1 of the testbed in the corridor of the 6th floor of the Q building at KTH.

Figure 3.5: Picture shows the wireless node Tmote Sky in casing with Ultrasound Sensor.
all the area around the transmitter.

For one microphone placed in a vertical position if the ceiling height is 2m one can say that the coverage area of just that microphone is a circle that can be calculated by $A = \pi \times r^2$ where $r$ is the radius of the cone described by the microphone span angle in the ceiling. One can know $r$ by calculating $r = \tan 30 \times 2$ where 30 is half of the span angle. Performing the calculus one knows that the coverage area is of about $5m^2$.

The transmitter circuit has to be able to modulate a signal for the input of the microphone of 40KHz which is the working frequency of the ultrasonic microphones. Also since the ultrasonic signal is going to be sent through seven microphones, the signal amplitude has to be high. The value chosen was of 12V. This voltage is supplied by a 12V and 2.2 Ah Battery. The circuit has as an input a signal from the wireless node which has the function of enabling the transmission of the ultrasound wave from the microphone. The wave is enabled when the input signal is low (0V) and disable when high (3-5V). The designed and developed circuit is presented on Figure 6.2.

The ultrasound receiver circuit works as follow. The ultrasonic signal received by the receiver microphone generates an electric signal which is then filtered and amplified. After this, one needs to evaluate if the electric signal is due to an ultrasonic signal or just some random noise. So, knowing a priori what is the usual shape of that signal, one needed to have a threshold value to perform the signal identification. When first developed this system one tried to use the wireless node ADC in order to check when the signal measured is higher then this threshold value. Problems arise due to the high variation of
the readings. After this was tried to design an hardware comparison using an comparator based on operational amplifiers. This shown to give low varying signal with low noise so this solution was adopted. The threshold value was set to be 1.8V. One achieved this value by testing the system. Once the signal suffers the comparison the generated signal is now a binary signal consisting of an high value (5V) or low value (0V). This signal is then given as an input to an interruption input port of the wireless node in order to keep track on the arrival of an ultrasonic signal. The circuit of the ultrasound receiver is shown on Figure 6.

After describing the transmitter and receiver circuitry one will show the characteristics of the wireless nodes connected to each one.

The transmitter node has to be capable of generating a binary signal to the transmitter circuit. This is done by clearing the output pin connect to the transmitter circuit. The pin selected was the pin 5 of the wireless node (see [13] for details). Also the transmitter wireless node has to be able to send a message at the same time as the electric signal. This is explained in Section 3.2.

The receiver node has just to enable interruption on the port selected, wish was pin7 of the expansion connector (Port GIO0, see [13] for details). Also it has to be able to receive the message from the Tx node in order to perform the calculation of the Time-of-flight. The Tx and Rx procedure is fully detailed in section 3.2.

### 3.1.3 Vision based system

The Vision Based System is composed by two modules. The web camera (Sensor) and the Processing Unit. They are both presented next. The video acquisition, frame grabbing and posterior image analysis has to be made in a dedicated processing unit rather then with the fusion center processing unit since this role of tasks block the computer processing and so it would not let the system ask for a ultrasound value while computing a position value for the respective image.

#### Web-camera

The camera chosen for this system was the web-camera, Logitech Quickcam Fusion with the following characteristics:

- Gross sensor resolution of 1.3 Mega pixels
- CMOS Optical Sensor Type
Figure 3.7: Logitech fusion web-camera.

- Color
- 1280x960 of image size
- USB Interface and powered through USB

It was chosen a web-camera since was the cheapest solution that allow us to have a simple and quick connection to the processing unit and enable the image processing with a good image quality. Figure 3.7 shows the web-camera used.

**Processing Unit**

The Processing unit has to be able of:

- Interface with the web camera through USB.
- Acquire video, grab frame and process frame to achieve world coordinates of the position of the object).
- Send the position of the robot using UDP communication protocol on MATLAB.

The processing unit is a laptop, equipped with an IEEE 802.11g wireless card in order to communicate with the fusion center. One could use also the wireless sensor network to process the data exchange. For this one needed
Figure 3.8: Mobile Agent. RC electric car controlled by a human operator.

to set the base station node to listen to position measurements given by the
vision system processing unit and also to have a dedicated node connected to
the vision system processing unit. Due to construction simplicity one kept
the IEEE 802.11g wireless solution instead. As a future development one
could develop a solution only based on the WSNs IEEE 802.15.4 protocol.

3.1.4 Mobile agent

The mobile agent used was an electric Radio Controlled car. The car was
then equipped with one Tmote Sky node, the Transmitter circuitry and one
12V, 2.2 Ah battery. The car radio controller has a range of approximately
20m. It has the function of simulate an autonomous robot moving and send-
ing ultrasound pulses and RF messages through the wireless node, on the
corridors where the testbed is deployed. The control is done by a human
operator. In the future the car will be changed for an autonomous robot.
Figure 3.8 shows the mobile agent.

3.1.5 Fusion Center

The fusion center is composed by one processing unit which interfaces with
the sensor network through a wireless node connected to it (base station
node) and interfaces with the vision based system processing unit through a
wireless IEEE 802.11g connection. The processing unit is a laptop, equipped
with an IEEE 802.11g wireless card in order to communicate with the vision
system and USB ports enabling the wireless node connection.
3.2 Software

In this Section one describes all the operational procedures and algorithms implemented for each module of the localization system. It is described the ultrasound system, the vision based and then the fusion center.

3.2.1 Ultrasound system

The procedure followed to develop the ultrasound system can be divided into two Sections, Transmitter (Tx) and Receiver (Rx). It will be presented the implementation in each Section. The general procedure is based on sending a message through the Tx node to the receiver node and at the same time a ultrasound signal. Considering that the message is instantaneously received by the Rx node (speed of light), the ultrasound wave emitted by the Tx node is then captured by the Rx microphone which triggers an interruption on the receiver node. The time difference of arrival between the two signals (RF Message and Ultrasound) is then used to calculate the distance between the two nodes. The time value is then sent to the based station node which after filtering the data (Kalman Filter) gives the estimated position (See Section 3.2.3). In order to perform localization, three receiver nodes would be needed. The algorithm is based on the fact that the speed of sound is approximately constant and equal to 340.29 m/s, so multiplying the time difference of arrival by the speed of sound, the distance between the nodes is achieved. As we can see in [10] the speed of sound can be better approximated if the temperature and humidity are known. For now this fact was ignored since the values obtained are less than 3% of the measured distance(See Section 4.1).

Figure 3.9 presents the model explaining the global procedure of the ultrasound system.

Transmitter

The transmitter node has to be capable of:

- Generate and send a message (with a tag showing that it is the ultrasound message) and at the same time clear the pin 5 on the expansion connector in order to generate an ultrasound signal in the ultrasound circuit. This is set to be done periodically in order to help in the performance tests evaluation on the receiver node. One can see that power constraints were placed in the mobile node this would no happen. Then is assumed that there are no power constraints in the mobile agent.
Figure 3.9: Interaction between ultrasound receiver and transmitter modules.

The tinyOS functions used by the transmitter are:

- **Timers** - Millisecond resolution - Pin set and clear.
- **Message** - Message Sending
- **Counter** - Microsecond resolution - Evaluate the timer accuracy
- **Printf** - Debugging

The transmitter operation can be seen in the flow diagram in Figure 3.10.

An important feature of the ultrasound Tx is that the periodicity of the message and ultrasound transmission can be lowered to 100ms. For values under this the receiver starts receiving inaccurate values of the time difference. The period value used for all the tests was 250ms, in which the ultrasound is emitting for 100ms and waiting 150ms for sending again.

**Receiver**

The receiver node has to be capable of:

- Receive the message sent by the transmitter node and decode it. Start the counter.
- Have a comparator that compares the voltage in the ultrasound channel with 1.8V. Output is 1 if higher then 1.8V, 0 if lower.
Figure 3.10: Transmitter flow Diagram

- Enable interrupt on GIO0 port in the MSP430.
- Generate interruption and tag the arrival time of the ultrasound.
- Calculate the time difference of arrival between message and ultrasound.
- Broadcast a message with the calculated value.

It was chosen a value of 1.80V for the threshold value because with the tests made, when a ultrasound pulse arrives the signal is always higher then this value. This value can always be changed in order to improve accuracy.
on the detection point, since it establish the margin between a Ultrasound pulse reception.

The tinyOS functions used by the receiver are:

- **Timers** - Millisecond resolution - Accuracy and performance tests.
- **Message** - Message Reception.
- **Counter** - Microsecond resolution (32KHz) - Calculate the Time difference of arrival.
- **Interrupt** - Tag the time when ultrasound pulse received.
- **Printf** - Debugging.

The Receiver flow diagram is presented on Figure 3.11.

As one can see that since the counter clock is 32KHz the minimum distance the node can solve is around 1cm assuming the velocity of sound of 340.29m/s.

### 3.2.2 Vision based system

Here is presented the algorithms and tools used to on the vision based system.

As seen before the processing unit has to be able to:

- Interface with the web camera through USB.
- Acquire video, grab frame and process frame to achieve world coordinates of the position of the object.
- Send the position of the robot using UDP communication on MATLAB.

**Image Processing**

The video acquisition, frame grabbing and image processing were all primarily done with the MATLAB Image Processing Toolbox, because it is the easiest and most straightforward software tool to use. All the MATLAB functions were chosen depending on several quality tests made during the implementation of the algorithm. As one could see while running the tests, all the processing time taken was in the worst cases from 8s to 11s which was too much given the system requirements. In order to reduce the time spent
on video acquisition and frame grabbing a new approach based on OPENCV (see [42] for further details) was implemented. It was seen that the time spent for analysing the image and getting the object the position values was always 1s using the OPENCV solution.

In order to improve the robustness of the image processing algorithm one had to achieve a way to clean from the image the parts where the robot cannot be found. For the first time the system was run a white rectangle was marked on the floor, limiting the area where the robot can be found which we call the clear path. The image is then stored as cutimage.jpg and is then used to eliminate all the surrounding of the clear path. One can easily see that if the position of the camera is changed one has to deal with this step.
again. This procedure is seen as the camera calibration step and its need is explained next. It was performed in order to reduce the computational time of the image processing algorithm by reducing the detection area. Another feature of the algorithm is that the object necessarily has to be round white circle with a diameter larger than 5cm in order to be detected. It was placed a white circle with a diameter of 20cm in order to cover all the top of the mobile agent.

The flow diagram of the image processing is presented next as well as the description of each function used.

![Image Processing Flow Diagram](image.png)

Figure 3.12: Image processing flow diagram

- Image Acquisition
The image acquisition is only made if the threshold of the gray scale image is above a certain value. This value was set to be 0.4 because we saw that above this value the contrast of the image is sufficient in order to perform the localization of the lines and the object in the image. For this the MATLAB function \textit{graythresh} was used.

In Figure 3.13 one can see an example of an acquired image with the mobile agent.

There are two types of ways of acquiring the image with depending on the scheduling scheme used:

- Periodic Triggering - signal enabling the picture acquisition from the fusion center is periodic.
- Async Triggering - signal enabling the picture acquisition from the fusion center is not periodic.

The image acquisition and storage is made by using the OPENCV library functions. It was created a C++ code which acquires, shows and store an image in a \textit{jpeg} format file. The code is presented next and it explains how the image acquisition, display and storage is processed.
With the tests made using MATLAB function `cputime` function was possible to evaluate that the delay between calling the function and have an updated image available was of 0.75s. This program has to be called on the Linux terminal before the MATLAB image processing algorithm is called.

```cpp
// Initializes capturing video
CvCapture* capture = cvCaptureFromCAM( CV_CAP_ANY );
 (...)

// Create a window in which the captured images will be presented
cvNamedWindow( "mywindow", CV_WINDOW_AUTOSIZE );

// Show the image captured from the camera in the window and repeat
while( 1 ) {
    // Get one frame
    IplImage* frame = cvQueryFrame( capture );
    (...)

    //Store image under filename 'VideoFrame.jpg'
    sprintf(filename,"VideoFrame.jpg");

    //Show image on the window
    cvShowImage( "mywindow", frame );

    // Release the capture device housekeeping
    cvReleaseCapture( &capture );
    cvDestroyWindow( "mywindow" );

• Pre-Processing - Filtering and image resizing; Edge detection; First the RGB image is converted to grayscale format using the function `rgb2gray`. Then the precision class is change to double using the function `im2double`. After this 5 pixels around the border of the frame are removed in order to help on the segmentation task. With this approach the image borders are not detected as lines. We used the `unsharp masking` filter with the `fspecial` function followed by a edge detection with `prewitt` filter with level 0.02. After this step if it is the
first time the system is used \( (k = 1) \) or if the camera has changed position, then one has to detect the lines of the white rectangle drawn on the floor as explained before. This is done by the step \textit{Segmentation}. If is not the first time the system is used and there are no changes made to the camera position one can go for the \textit{Apply cut image mask} step.

- \textbf{Segmentation} - Get the border of the rectangle where the robot is inserted; Functions \textit{HoughTransform, houghlines} and \textit{houghpeaks} are used.

- \textbf{Cutting} - Cut image; Everything outside of the rectangle has to be cut in order to help on the robot detection; It is created and stored a binary image with pixel value 1 inside the rectangle and 0 outside the rectangle. The result of this step can be seen in Figure 3.14.

- \textbf{Apply cut image mask} - In this step the cut image is multiplied by the smooth image (output of \textit{Pre-processing}) and another edge filtered is applied (\textit{canny} filter with 0.08 level). The area outside the rectangle is then removed i.e. all the pixels of that have value 0 now.

- \textbf{Robot image coordinates} - Calculation of the robot coordinates;
  
  - Calculate the object boundaries on the image using \textit{bwboundaries()};

Figure 3.14: Cut image view. Binary image.
– Labeling bounded objects;
– Filtering the image in order to remove the noise using erode and dilate filters;
– Labeling the areas of interest;
– Eliminate the areas which are not circles; Excentricity higher then 0.8;
– Calculate the centroid of the circular labeled objects;

• Position transform and storage of M. This step is only performed for the first step \( k = 1 \), otherwise for \( k \neq 0 \) the next step is performed. In this step the following is done:

– Calculate 4 points of interest on the image; Corners of the rectangle; Since in the image the area is no longer a rectangle (the size of the top rectangle line is much smaller then the size of the lower rectangle line) the calculation of the corner is not so trivial. A filter was applied in order to find the pixels where the pattern from of (3.1) was found. This procedure is called matching. When detecting the matching points, one looks for the points that have the lowest and highest \( i \) coordinate (left and right top corner respectively). To know the lowest line corners was straight forward and we only had to look to the points that the distance to the lower corners of the image was smaller.

\[
\text{pattern} = \begin{pmatrix}
0 & 0 & 0 \\
1 & 1 & 1 \\
0 & 0 & 0
\end{pmatrix}
\quad (3.1)
\]

– Transform image coordinates to world coordinates; This is made using the following coordinates transformation:

\[
\begin{pmatrix}
w_i \\
w_j \\
w
\end{pmatrix} = \begin{pmatrix}
m_{11} & m_{12} & m_{13} & m_{14} \\
m_{21} & m_{22} & m_{23} & m_{24} \\
m_{31} & m_{32} & m_{33} & m_{34}
\end{pmatrix} \times \begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
\quad (3.2)
\]

where \( i \) and \( j \) are the \( x \) and \( y \) pixel coordinates of the object, \( m_{ij} \) are the values of the transformation matrix \( M \), \( w \) is the focal lenses scale factor and \( X,Y \) and \( Z \) are the world coordinates of the object. One can notice that the values of \( m_{i3} = 0 \) since \( Z = 0 \)(the object is in the ground plane). Then is easy to discover that the
value $m_{34}=1$. Now we have that the matrix $M$ has 8 unknown variables and so one needs 4 points $(i_k,j_k)$ in the pixel coordinates to solve the equation system with 8 unknown variables $m_{ij}$. One needs 4 points since each point allow us to have 2 equations since $w$ can be written in a function of only $X$ and $Y$ and then is substituted on the equations generated by the two lines above. This can be interpreted as:

$$\begin{align*}
  w_{ik} &= m_{11}X + m_{12}Y + m_{14} \\
  w_{jk} &= m_{21}X + m_{22}Y + m_{24} \tag{3.3} \\
  w &= m_{31}X + m_{32}Y + 1
\end{align*}$$

So now one has to solve the 8 equations given by the points $(i_k,j_k), k \in [0,4]$ to get the values $m_{ij}$.

In the end the matrix $M$ is stored in order to be used for the next steps. This is done in order to reduce the computational time.

- **Robot real coordinates**

  After having the matrix $M$ calculated is easy to get the values for $X$ and $Y$ for a given centroid value $(i,j)$ since we only have two unknown variables with two equations. One has to substitute $w$ in the two first equations of (3.3) for $w = m_{31}X + m_{32}Y + 1$.

  In Figure 3.13 one can see the final result of the image processing algorithm made to the image presented in Figure 3.15. Next is described the vision based processing unit.

Note that it is assumed that the camera has a top view and not lateral view as in the implementation. This was done in order to reduce the complexity of the position achievement. If this simplification was not made the coordinate transformation would be non-linear as it is shown in Figure 3.16 where the web-camera span-angle is $\theta$ and its pixel resolution is denoted by $\Delta \alpha$. As one can see the world resolution $\Delta X_i$ to pixel resolution $\Delta \alpha$, would change according to the distance of the mobile agent to the camera, given by a non-linear function, so $\Delta X_1 \neq \Delta X_2$. This problem is not dealt in this work, we assume that $\Delta X_1 = \Delta X_2$.

**Processing unit**

The way to send data from the vision based(VB) processing unit to the base station(Fusion center(FC) processing unit) is done by UDP protocol in MATLAB using the wireless standard IEEE 802.11g. The base station sends
Figure 3.15: Mobile agent detection. Star shows the detected centroid of the mobile agent.

Figure 3.16: Vision based localization system with non-linearities.

a message over wireless with value ’1’ if it wants to have information about the mobile agent position from the vision system. The vision system waits to get this value, when this occur, the image processing algorithm is called
and the position value is returned to the base station also using the same protocol over the wireless link. The code of the algorithm implemented is presented next.

- VB Processing unit side

```matlab
VB=udp('IP OF CS','LocalPort', 15005, 'RemotePort', 15004); %define port properties
fopen(VP); %open port

%Wait for action enabling from Base Station
%Always reading until get a 1 as input
%send by Base Station (enabling signal)

while reception~=1
    Value=fscanf(VP,'%c',1) %read from the port
end

%Call algorithm and returns X and Y
Image_Processing_Algorithm
reception=0;
A=[X;Y]

%Return position value from ethernet port to base station
fprintf(u,A); % write in the port
```

### 3.2.3 Fusion Center

The fusion center is composed by one processing unit which interfaces with the sensor network through a wireless node connected to it (base station node) and interfaces with the vision based system processing unit through a wireless connection as it was seen in Section 3.1. Next is presented the operation of the base station node, the ultrasound reader, the vision based reader and the fusion center position estimator implemented. The schematic with all the components involved and data flow and treatment is presented in Figure 3.17.

**Base Station Node**

The base station node is responsible for receiving the messages from each receiver node on the wireless network and then send their values, with the respective node ID, through the serial port to the fusion center system processing unit. The base station only sends the values through the port when
it receives a message from four nodes in order for the time in which the localization is made is the same for all. At this node is also made a pre-selection of nodes which should give their time measurement. The nodes which gives time difference values higher then $14690\mu s \ (\approx 5m)$ are discarded. Since the highest distance between transmitter node and receiver given all the clusters of four motes (combination of all the 16 motes used) is $\approx 2.60m$, in order to cope with possible message collisions between closer nodes, one has chosen $5m$ value as a threshold. This value can be adjusted according to the placement of the nodes in the network. There are four values received but only three nodes are required because the localization is made in 2D since we assume that the mobile node is at $Z=0$. The reason for taking values from four motes is that with this one can apply an algorithm in order to put the most far away node and the others in sleeping mode because one knows that it will not be required to perform localization. This enables less power consumption on the network. Also this is needed if the three nodes are in the same plane X or Y one have to discard their readings since the solution for the trilateration method cannot be achieved if this happens, as described in Chapter 2. The way to deal with this fact is described next.

The flow of the algorithm implemented in the node is:

1. Receive message $k$ from node $n$.
2. If value of node $n \geq 14690$, then discard value. Otherwise keep the value and increase $k$.
3. If the values are received by three nodes located in the same plane X or Y discard the value of the node that has the highest time difference value and wait for another reading.
4. If $k = 4$, send the four values trough the serial port. If $k < 4$ go to step 1.
The message sent has the format \( d_{k1} = \Delta t_{k1}, d_{k2} = \Delta t_{k1}, d_{k3} = \Delta t_{k3}, d_{k4} = \Delta t_{k4} \).

One should notice also that if the nodes are too close sometimes there could be a measurement sequence where the received distance can be given by 4 nodes in the same X or Y plane. In order to cope with this one can increase the number of distance values transmitted to the fusion center from the base station node.

**Ultrasound reader**

The method described in Chapter 2 used for calculating the position of the object is the trilateration method. All the different methods for determining position are described in Section 2.1.2. We are going to calculate the position in 2D as discussed in the previous Section.

For each distance received the procedure of the ultrasound reader algorithm implemented on the fusion center processing unit is as follows:

1. Sum a characteristic offset value for each node to the distance value. The offset value was given by tests done to each node and getting the respective average of the error, which was shown to have a low variance (See Chapter 4).

2. Calculate X and Y based on the values given by three of the fours nodes applying the trilateration method. Since the node that is more far away from the target is the one that should have more error, a comparison between the four values are made and the highest is discarded. As discussed before this enables a better power management of the network.

**Vision based reader**

As discussed in Section 3.2.2 was implemented a MATLAB program to perform the request of a position measurement given by the camera and to receive this value. This program has to be able to trigger the algorithm developed on the vision system processing unit as described in Section 3.2.2. This is done by sending a message through the wireless link, using the UDP protocol, with value=1. The time between the request and reception at the base station node is estimated to be of 0.5s maximum. When the image processing is concluded on the vision system processing unit the value with the position of the mobile agent is sent back to the fusion center. Then the vision based reader MATLAB code implemented on the fusion center
processing unit is responsible to store the position value to be used for the mobile agent position estimation according to the defined scheduler.

The code developed is as follows:

```matlab
FC=udp('IP of VB','LocalPort', 15001, 'RemotePort', 15000);
%define port properties
fopen(FC); % open port

%Take new measurement
fprintf(FC,'1') %write 1 on port

%Wait for measurement from port
X=fscanf(FC,'%c')%read port and save as string vector X

%Since the values are stored on a string one has to read it.
%Each character is in the ASCII form so one has to pass it to integer.
for i=1:length(X)
    X(1,i)=X(1,i)-48; %passing to integer
end

%Now one has to give to each digit its original position value.
%For example, if X=1000, the digit one has to be multiplied by 1000 \(10^3\) and then sum zeros multiplied by power of 10 for the rest digits according to their position.

%Store position for X
posX(l,1)= X(1,1)*10^(X(1,12))+X(1,3)*10^(X(1,12)-1)+X(1,4)*10^(X(1,12)-2)+X(1,5)*10^(X(1,12)-3)+X(1,6)*10^(X(1,12)-4)+X(1,7)*10^(X(1,12)-5)
%store position for Y
posY(l,1)= X(1,14)*10^(X(1,25))+X(1,16)*10^(X(1,25)-1)+X(1,17)*10^(X(1,25)-2)+X(1,18)*10^(X(1,25)-3)+X(1,19)*10^(X(1,25)-4)+X(1,20)*10^(X(1,25)-5)
```

One should notice that in order to test the system for higher communication delays then the ones presented before it was implemented a function in the base station processing unit to delay the usage of the position received by the vision based reader for the established value. Normally the vision based reading is received in the base station processing unit 2s after the request. As one will see in the Experimental Validation Section the time delay used was \(d = 3\) so one needed to delay the reading usage for about 1s. Only when the 3s have passed after asking a position the values received are used.
Fusion center position estimator

As it was described in the previous sub-sections one have access to the position given by the two sensors through different networks. One is the ultrasound communicating with the fusion center by a wireless sensor network (IEEE 802.15.4) and the other the web-camera communicating with the fusion center through a wireless communication (IEEE 802.11g). Both values are now available at the fusion center as it was described before. Now one had to implement the estimator presented in Section 2.4 taking also into account the scheduler, offline or covariance-based, one wants to implement (see Section 2.5 for details). So, a code was formulated according to the flow diagram presented on Figure 3.18.

As one can see the algorithm is based on five steps.

1. Initialization - Initialization of the Kalman filter and other required variables.

2. Define scheduler - In this step one has to choose if the offline or covariance-based scheduler is used. If the offline scheduler is used one has to choose the switching periodicity \( N \) of the vision based system (see Section 2.5 for details). Is assumed that when the scheduler is selected the model has also to be defined. One can choose between model 1 and model 2.

3. Perform a measurement according to the scheduler defined for the current instant \( k \). Store the value of the position \((X,Y)\). This is achieved by calling the ultrasound or vision based reader algorithms previously described.

4. Apply the outlier rejection method. This method works as a filter to discard bad measurements. It has the characteristic of discarding the measurements made of \((X,Y)\) when the traveled distance between two steps \( k \) is higher then 3\( \text{meter} \). One can expect that the worst (highest distance traveled without reading) is when the vision based system is used. As it was seen in Section 3.2.2 the delay of the Vision based system is 3s. According to table 2.1 one can see that the highest process noise for model 1 is 1\( m/s \) and model 2, 0.09\( m/s^2 \). One can assume that the velocity and acceleration of the mobile agent when using model 1 and model 2 cannot be higher then these values expected. As one can easily see establishing the maximum velocity for the mobile agent on 1\( m/s \) the maximum traveled distance on each coordinate is 0.3\( m \) in 3s. So 3\( \text{meters} \) can be considered an appropriate threshold to classify a bad position measurement when the web-camera is used. When the
web-camera is not used one should put the threshold to be 1.5\textit{meters} since the ultrasound system delay is of 1\textit{m/s}, so a bad measurement can be considered if the displacement is above 1.5\textit{meters}.

In the case of a bad measurement the value of $X$ and $Y$ is obtained by assuming that the position of the mobile agent has not changed. So, \( X_k = X_{k-1} \) if $X_k$ is considered an outlier. The same is done for $Y$. One can also see that other methods could be applied to solve some sensor malfunction. One can imagine that using a movement predictor to estimate the position this method could be improved. This can be seen as a future improvement to this work.
One should also refer that if the first movement is a bad measurement (no real value) the position is set to (0,0).

5. The final step would be to estimate the position at instant $k$ using the Kalman filter proposed on Section 2.4. In the end we have an estimated position $(\hat{X}, \hat{Y})$, which is assumed to be more accurate than the ones given by the sensors as was seen in Section 2.5.1 and 2.5.2.

Another feature created for the fusion center was a Graphical User Interface that enables the user of the system to see the 2D spacial positioning of the robot on the defined area. The GUI can be seen in Figure 3.19. As one can see there are active nodes defined with green circles and non-active nodes defined with red circles. They are active if the ultrasound transmitter is positioned within their range. It is assumed that the camera can always visualize the robot if it is within the central area as it is seen in Figure 3.13.

In the GUI is also provided the direction of movement performed by the mobile agent in order for the user to predict if the one has delay in the measurements what will be the next position of the agent.
Chapter 4

Experimental validation

In this chapter are presented the proposed experimental validations for the approaches described. First is shown the behavior and characteristic of the ultrasound system and the vision based system when used separately. There are made tests on localization when the estimator is used and then comparisons to the situation when it is not used are made. After this first step one needed to validate the fusion center system where tracking situations are put forward for different schedules used. Considerations on the results achieved are made for each Section.

4.1 Ultrasound System

Tests were made in order to evaluate the performance of both transmitter and receiver nodes. On each node, hardware and software tests were performed. All the software implemented in each node worked well.

There were made three type of tests in order to estimate the accuracy of the system in three types of situations:

- Straight line - Test the accuracy of node and the maximum distance between Rx and Tx node, when they were placed in front of each other. The distance goes from 100cm to 1500cm with 100cm increments. Fixed Tx node.

- Localization - Test the accuracy on the localization of the Tx node given 4 receiver fixed cluster of nodes placed on the ceiling. Fixed Tx node.
Figure 4.1: Straight line average measurements. Real distance(cm) VS Measured distance(cm)

- Tracking - Track the movement of the Tx node within the area given by 12 Receiver nodes placed on the ceiling. Only the nodes in the middle section were used.

There were only used 12 ultrasound receivers in order to cover only the middle corridor illustrated in Figure 3.19.

4.1.1 Straight Line

This test was performed in one of the corridors of the 6th floor of the Q building. There was a clean line of sight between the Rx and Tx but there were objects on the sides of the corridor. As said before the Tx and Rx where placed in front of each other. In Figure 4.1 one can see the average of 400 measurements of the distance for each real distance. We can see that the distance measured by the node depends linearly on the distance as was expected. But as it is seen there is a clear offset value that needs to be corrected.

In figure 4.2 is presented the linear interpolation of the error taking into consideration the distances from 100cm to 1400cm. These error values given for each distance were a first approach to get the offset values discussed in Section 3.2.1. In the next sub-section will be explained why these values do not fit for that purpose.
This test was used also to verify the maximum distance between the Rx and Tx node and it was seen that after 1200cm the deviation of the values is very high and so the accuracy is low.

### 4.1.2 Localization

In order to check the accuracy of the ultrasound system an overall test to the localization algorithm was made. For this the Tx node was placed in various places within the area defined by the 12 Rx nodes. For the Tx placement 100 position values were measured and analysed. First the ultrasound system was tested without any outlier filtering, estimation and offset calibration applied to the distance measured by each node. It was seen that the performance of the system would require those methods to improve it. As was discussed before in the straight line test the offset value as to be taken into account otherwise the error will be quite high. To confirm if the offset value calculated in the previous Section was right, the Tx node was placed in four different positions and values were taken from 4 different receiver nodes. We were able to see that the errors are different on each node but this difference is not higher then 10cm which is shown on Figure 4.3. The explanations for this fact are:

- Communication errors - Variable delays on the message transmission
from the Tx node.

- Interferences on the ultrasound signal due to objects in the surroundings.

- Non-perfect circuits - even though the circuits developed were implemented on Plastic Circuit Boards (PCBs) one always have small differences on each circuit when they are built. Also there are interferences, resistances and capacitance errors, etc. on the circuit that can affect the comparison triggering.

- Interruption errors - There could be time differences on the triggering of interruptions on each mote due to circuit inequality (Tmote Sky).

The average of the errors for each node was chosen to be the offset value to sum to the measurement made at each time instance. This value can also be calculated by the linear interpolation of the errors given several tests on each point. So each node has its own offset value, but they do not differ more then $2 - 5\text{cm}$ from one to another. This can be seen as the calibration phase of the network. Better solutions to reach the optimal offset value were not required since the final error with this approach was within the requirements, but it can be seen as an improvement for future work. One discarded the offset value calculated by the linear interpolation given in Figure 4.2 since with the new localization tests performed one saw a $10\text{cm}$ difference compared to the previous ones.

After setting up the offset value and having more accurate results for the distance measurements taken by each node one can now focus on the position. With the distance measurements one can now apply the trilateration method to calculate the position $(X, Y)$ given three distance measurements.

After calculating the position is applied the outlier rejection method. This method, as explained in Section 3.2.1, is intended to reject all the wrong position measurements made by the ultrasound sensors. Let us remind that a measurement is considered wrong when the variation between one position at time $k - 1$ and $k$ is higher then $1.5\text{meter}$.

Next are presented in Figure 4.4 and 4.5 the performance of the system for a given location of the mobile agent at $X = 50\text{cm}$ and $Y = 200$, when:

1. The outlier rejection is not used.

2. The outlier rejection is used.
For simplicity reasons only the $X$ coordinate is evaluated.

So as one can see in Figure 4.4, when the outlier rejection is not used the system cannot eliminate esporadic errors that occur sometimes on the system due to receiving bad information from the nodes. As one can also see for the first measurement there was an error on the reading (measurement $X = \infty$) which without the outlier rejection will influence the system performance since it is not removed.

In Figure 4.5 the outlier rejection is applied. One can see that the erroneous measurement occurred in the middle of the measurements has been removed. Also for the first measurement the value $X = \infty$ is now substituted by $X = 0$ which does not have a great influence in the overall performance of the system since the mean error is $0.05\text{cm}$. One should notice that since the minimum resolution of the system is $1\text{cm}$ the mean error value of $0.05\text{cm}$ cannot be taken into account.

As discussed in Section 3.2.1, after being applied the outlier rejection method the estimator is used in order to improve the accuracy of the measurement.

Next are presented in Figure 4.6 and 4.7 the cases when the estimator for model 1 and model 2, respectively are applied to the system. Notice that the parameters of the estimator for both models are:
Figure 4.4: Ultrasound system performance when no outlier rejection method is applied. Error and position values for real position X=50.

Figure 4.5: Ultrasound system performance when an outlier rejection method is applied. Error and position values for real position X=50.

- Estimator model 1 - Process noise $W = 0.1$. 
Estimator model 2 - Process noise $W = 0.003$ and $W = 0.09$.

The measurement errors were kept the same as in Section 2.5.

One only tried, as in Section 2.5, to use the lowest process noise which is for the highest delay $d = 7$. Since the performance was high for the first model, as one will see next, there was not tested the cases where the process noise $W$ is higher. It is intended to show the performance of the system in the worst case scenario which occurs if the camera would be used and the delay would be $d = 7$ (highest constrained situation to the overall system). Also the covariance matrix $P(k)$, as described in 2.5, is not initialized as a matrix of zeros but as a matrix of ones with dimension $\mathbb{R}^{d+1 \times d+1}$ for the model 1 and also a matrix of ones but with dimension $\mathbb{R}^{2d+2 \times 2d+2}$. One can notice that being the $P(k)$ matrices of this size that a choice to use a separate estimator implementation for each variable $X$ and $Y$ was made. So each coordinate is estimated in parallel.

One can see in Figure 4.6 that due to the fact of the bad measurement occurred at $k = 1$ the first estimated position is $X = 0, Y = 0$ and so the estimated coordinate $X$ takes at least 10 measurements to get on a 10cm error boundary of the real measured value. One should also notice that the sharp displacements on the $X$ position are attenuated with the filter which is a good
characteristic for the overall performance. One could expect better results from this filter if the bad measurement at $k = 1$ have not occurred. Even though there exists the referred “settling” time and the average is 1.4cm.

In Figure 4.7 one can see the system performance when the estimator of model 2 is applied. In this case the performance decreases and the result is more influenced in the same way of the estimator of model 1 for the case of the bad measurement at $k = 1$. It takes now 30 steps to recover to the 10cm error boundary of the real value. One can also notice that the sharp displacements on the position are attenuated in the same way as the estimator of model 1. One can notice that this happens due to the fact that the filter model does not expect quick variations in position, but in velocity. The estimator follows quickly the measurement from $k = 1$ to $k = 2$ and overestimates it because it thinks it will continue moving in that direction but then it takes a long time to get back to the actual position. One was already expecting this as it was shown in the example in Section 2.2. Even though these errors occur the average error is 6cm which is lower then the 10cm error discussed before. It was also tested the case when the process noise is $W = 0.09$. For this case one checked that since the model expects a higher variation in velocity from one step to the other the ”settling” time will be lower and so the values will be inside the 10cm error boundary more
quickly at $k = 10$.

Also one could notice that the maximum position error when the outlier rejection method is used is $23cm$. When the estimation is also used, in the case of model 1 the maximum error is $8cm$ (not counting the values until the step $k = 10$ since the result is based on the bad measurement at $k = 1$. For the $2^{nd}$ model estimator the maximum error is $6cm$ when $W = 0.003$, also without counting the errors when $k < 30$ for the same reasons pointed before. For the case where the process noise is $W = 0.09$ using the $2^{nd}$ model, the error was of $7cm$. One can say that the ultrasound system as an error less then $8cm$. More tests where made in different positions and it was seen that for the localization problem the ultrasound solution gives accurate results.

As a summary one can say that with the results shown, in order to have the best estimation possible of the position when performing localization with the ultrasound system one should use the following techniques:

1. Sum an offset value characteristic of each particular node.

2. Apply the outlier rejection method. Chose the threshold in accordance to the problem in question. In our case was chosen the threshold $= 150$ when using only the ultrasound sensor.

3. In our case was also better to use the estimator of model 2 after using the outlier rejection method. This is due to the fact that it gives the lowest maximum error, and assuming that in further developments the values before $k < 30$ are not considered.

Also it has to be seen that if the bad measurement did not occurred for $k = 1$ the performance of the estimator would be increased and give much better results then the ones presented here. As a solution for this and other cases is that the system should be turned on and one should not take in consideration the values given before step 30., which means 30s of calibration time. After this the estimator for both system can be used helping to give better performance to the system.

One should notice also that the estimator shown here was the one without any prediction adjustment since it was seen that for localization purposes and low process noises used the difference was not significant to be shown here. One should also mention that using the prediction adjustment the error for model 1 one the same but not for model 2. In the case of model 2 the error was slightly lower when using the prediction adjustment but one has chosen not to show it here.
4.2 Vision Based System

In this Section one can find the analyses of the tests made to the vision based system when performing localization.

First one should remark the characteristics of this system discussed in Section 3.2.2.

It is known that the system takes a new image each 0.75s using the OPENCV C++ function and takes normally less then 1s to analyse the acquired image. Also one should account with a 0.5s minimum time for position transmission and availability so one can say that the processing time delay is always less then 3s which was one of the values for the delay $d$ taken into account during the scheduling solutions evaluation presented in Section 2.5. One can notice that in fact the delay time is in the minimum case 1.5s but one established a delay $d = 3s$ in order to cope with eventual higher transmission times.

Also one should refer that sometimes during the tests problems occurred with the acquired image quality due to the fast time one takes a picture even though is 0.75s. During the tests were observed that the failure probability was only of 5% (5 failures in 100 measurements).

Another error that sometimes occur is that the mobile agent goes outside the defined calibration rectangle and so the detected image will have a partial circle detected which is not always considered as the mobile agent due to the constrains applied in the Robot image coordinates step as shown in Figure 3.14 in Section 3.2.2. The outlier rejection method shown in the previous section will tackle these errors, but now the threshold was set to be of 3m as was explained in Section 3.2.3.

4.2.1 Localization

Here one will discuss the performance of the vision based system when acquiring data of a fixed mobile agent. It is shown the same path described in the previous Section but with a different simulation. Both models were tested for this case.

In Figure 4.8 one can see the case where the localization using the vision based system is performed using the model 1 estimator. The parameters were set as:

- Process noise $W = 0.2$.
- Delay $d = 3$. 
Figure 4.8: Localization performance of the vision based system for a steady mobile agent at (50,200). Outlier rejection method and model 1 estimation performed.

- Initial covariance matrix with dimension $\mathbb{R}^{\text{delay}+1 \times \text{delay}+1}$ with ones in all the positions.

It is noticeable that the performance of the vision based system in order to detect the mobile agent is very high, with errors less then 1 cm for each coordinate. It is shown also that almost all the position values are place on a 2 cm$^2$ frame for both methods used. The number of steps for this to occur is lower then 10. There are only two situations that require particular attention. As it is noticed there is one jump on a reading made by the web-camera in which the Y coordinate is displaced by 3 cm up. This case is not tackled by the outlier rejection method since the displacement threshold detector was set to 1 m. But as one can notice this error is perfectly filtered by the estimator. Also using the vision based system there is another important estimator characteristic which is the effect of the delay. Since the delay is set to 3 s, only when that the time step is higher then this value the web-camera measurements are taken into account. So, due to this fact the position estimation value takes almost 10 steps in order to reach the 2 cm$^2$ frame discussed before.

It is also shown that the position values when using the outlier rejection method and the estimator have lower variation comparing when the estimator is not used, as was expected.
Figure 4.9: Localization performance of the vision based system for a steady mobile agent at (50,200). Outlier rejection method and model 2 estimation performed for $W = 0.003$.

Now in Figure 4.9 one can see the system performance when using the model 2 estimator. The parameters chosen for the model were,

- Process noise $W = 0.003$
- Delay $d = 3$
- Initial covariance matrix with dimension $\mathbb{R}^{2d+2\times2d+2}$ with ones in all the positions.

It was chosen to show not the overview of the position but the characteristic of the X coordinate over the measurements. As it is seen the performance of the system decreases significantly due to the delay observed. Now one needs almost 20 steps in order to achieve the $2cm^2$ frame, which is the double of the previous case. This is explained by the fact that the estimator is not expecting such high variations in the movement direction. As one can see even if this occurs, the mean error is only $7cm$, but when comparing to the performance of the first model it is truly worst. The case when the process noise is $W = 0.09$ was also studied and the evaluation of its performance can be seen in Figure 4.10. One can see that for higher process noises the
Figure 4.10: Localization performance of the vision based system for a steady mobile agent at (50,200). Outlier rejection method and model 2 estimation performed for $W = 0.09$.

settling time decreases and as a consequence also the mean error. This is then a good improvement when using model 2.

For the localization problem one can say that the model 1 estimator has a better performance than the one given by the second model. This has to do with the fact that the second model has a worst performance with so high variations in movement direction due to the delay involved, which is the case presented here since the readings will be zero until $k = 3$ and after that it has a jump to the first reading performed by the web-camera.

One should notice also that the estimator shown here was the one without any prediction adjustment since it was seen that for localization purposes and low process noises used the difference was not significant to be shown here. One should also mention that using the prediction adjustment the error for model 1 one the same but not for model 2. In the case of model 2 the error was slightly lower when using the prediction adjustment but one has chosen not to show it here.
4.3 Fusion center

As discussed in previous Sections, according to the sensors variance, the sensor measurement noises used for the validation for the models proposed were set for the ultrasound-based system as $\sigma = 12$ and for the vision-based system as $\Sigma = 1$.

For the experimental validation a new performance criterion based on the position error is put forward and analysis are made comparing the results achieved by using the estimator or just using raw measurements from the sensors. Both scheduling approaches and models are validated.

The tracking test was based on the situation where the mobile agent performs a movement with variations only in one coordinate (straight line with fixed $Y$ and varying $X$). The traveled distance was of approximately $4.5m$ and was performed in $12s$. The trajectory was tried to be performed with a constant speed but as one can know, using a RC car as the mobile agent has the drawback that one cannot know its current velocity, have an exactly constant speed or either performing an exact straight path. Since the values given by the camera are very accurate and taken at precise time one can trust that the velocity can be approximately given by its position values when calculating $v = \frac{x_k - x_{k-1}}{\Delta t}$. For the mobile agent start up (first $2s$) the velocity was approximately constant and equal to $11cm/s$, followed by an approximately constant velocity of $50cm/s$. When the car achieves the final position it deaccelerates for a constant velocity of $11cm/s$ as for the start up but now this is done over $3s$.

According to the velocities involved in the test was established that the process noises used for model 1 and model 2 were $W = 0.5$ and $W = 0.05$. As explained in Section 2.2 the process noise in model 1 is the velocity of the mobile agent, so is expected that in each step it will move $0.5m$ with random direction. For model 2 since one is not expecting high variations in velocity during the test, the acceleration of the mobile agent was set to be $0.05m/s^2$. One expects to have low variations in the direction of movement but random variations, according to the process noise, on the acceleration.

One should notice also that after the system was turned on, a $17s$ waiting time before starting the mobile agent movement, in order to remove the initial estimation overshoot since the starting position is unknown.

Tests were made with model 1 and model 2 for different periodic switching cycles $N$ and for the covariance-based optimal switching sequence. There were taken 29 samples which are equivalent to $29s$ since the sampling period was set to be $1s$. For each model one did the analysis over the position mean quadratic error taking into account the high-quality sensor communi-
Table 4.1: Optimal periodic high-quality sensor switching $N^*$ of $V_E$ for model 1 and 2 considering different communication cost $\lambda$.

\[
\begin{array}{cc}
\lambda & \text{Model 1}_W=1.8 & \text{Model 2}_W=0.05 \\
0 & \text{Offline: } N^*= 9, 27 & \text{Offline: } N^*= 17, \infty \\
2000 & \text{Offline: } N^*= 14, 27, 28 & \text{Offline: } N^*= 17, \infty \\
\end{array}
\]

This new performance criterion is an approximation of (2.28) since now instead of the theoretical estimation accuracy is analysed the experimental estimation accuracy. Has one can expect if the communication cost $\lambda \to \infty$ the optimal high-quality sensor switching will be $N = \infty$, since is too expansive to use the vision-based system. Also if $\lambda = 0$ only the mean quadratic error is taken into account since no cost is attached to the vision-based system usage.

First the estimation performance for the adopted process noises $W = 0.5$ and $W = 0.05$ for model 1 and model 2 respectively was tested. A comparison based on the performance criterion $V_E(M)$ between the estimated position and the position given by raw measurements with just outlier-rejection filtering is made. The communication cost was set to $\lambda = 0$ since just the position accuracy wanted to be evaluated. It was seen that for model 1, with process noise $W = 0.5$, the estimator did not improve the position accuracy comparing to the raw measurements. The process noise $W$ was then tuned for $W = 1.8$. The results are shown in Figure 4.11, comparing both models.

In Figure 4.11 is also shown the covariance-based scheduler errors using model 1 and model 2. The optimal scheduling sequence obtained when performing this approach was that for model 1 and $W = 1.8$ the high-quality sensor should never be used ($M = 0$), and for model 2 was that using $maxD = 4$, a sequence composed by a $N = 2$, $N = 3$, $N = 5$ and then a periodical $N = 2$ scheduling should be used. With this the high-quality sensor usage is of $M = 12$. It is seen that this approach does not give better results than using an offline scheduler. The optimal high-quality sensor periodic switching $N$ using both models and different communication costs $\lambda$ is seen in Table 4.1.

In Figure 4.12 the mobile agent tracking example is shown. Both estimated and raw position measurements are compared to the real mobile agent
The optimal high-quality sensor switching considered is $N = 6$. This value was chosen since it was seen as a good example to demonstrate the tracking improvement when using the estimator. One can see that model 2 performs better when comparing to model 1 as was expected from Section 2.2, since it fits well in the type of path performed. It is noticeable that model 2 requires a set-up period due to having an high settling period with fixed mobile agent position which is not a requirement of model 1.

It was also made the theoretical validation of the system with the mobile agent and sensor models parameters previously referred, based on the performance criterion $V_T$ from (2.28). Here only the estimation accuracy is taken into account. Table 4.2 shows the optimal scheduling sequence and approach for different communication costs $\lambda$ and process noise $W$ using model 1 and model 2. Note that the communication cost $\lambda$ is not the same since the estimation accuracy criterion $p_{\text{average}}$ does not have the same order of the quadratic mean error. Analysing and comparing the theoretical results and the experimental results for the same process noises, one can see that the covariance-based scheduler has higher cost performance which makes it better then the offline scheduler to be used on both models, which does not happen in the experimental validation, where the offline scheduler is the best. When just analysing the accuracy for model 2, which was the one that gave better results when experimentally validated, the estimation accuracy is higher for an offline $N = 2$ scheduling, where the web-camera is used each two steps, while the position accuracy is higher when the web-camera is used fewer times ($N = 17$) or even not used ($N = \infty$). For model 1 is also seen that the theoretical result does not match the experimental but is more close then the one achieved for model 2. Also was seen that the model 1 gave lower $p_{\text{average}}$ values for the optimal scheduling sequence achieved when comparing to model 2, making the estimator with less errors when using it.

As a summary one can see that the comparison between experimental and theoretical validation shows that the position accuracy does not match the estimation accuracy. Also one should refer that in terms of estimation accuracy model 1 was the one that gave better results, but when experimentally validated, model 2 is better on tracking the mobile agent.

| $\lambda$ | Model 1 $|W=1.8$ | Model 2 $|W=0.05$ |
|-----------|----------------|----------------|
| 0         | Offline: $N^* = \infty$ | Offline: $N^* = 2$ |
| 0.3       | Covariance-based: Sequence of $N = 2$, $N = 3$, $N = 5$ and a periodical $N = 2$ | Covariance-based: $N^* = \infty$ |

Table 4.2: Optimal periodic high-quality sensor switching $N^*$ of $V_T$ for model 1 and 2 considering different communication cost $\lambda$. 

position.
Figure 4.11: Estimated and raw position quadratic errors for offline scheduler with $N$ and Covariance-based scheduler for model 1 and model 2.

(a) Model 1 with process noise $W = 1.8$.

(b) Model 2 with process noise $W = 0.05$. 
(a) Model 1 with process noise \( W = 1.8 \).

(b) Model 2 with process noise \( W = 0.05 \).

Figure 4.12: Tested mobile agent trajectory for optimal high-quality sensor switching \( N = 6 \). Real position, estimated position and position given by raw sensor measurements over the \( X \) coordinate when performing 29 measurements.
Chapter 5

Conclusions and Future Work

During this work was seen how to design, develop and implement a localization and tracking system.

We showed how to develop a wireless sensor network testbed with a systems engineering approach.

All the concepts presented in Chapters 2 and 3 were validated and was shown that they can be applied in real system to perform localization and tracking in a wireless sensor network environment.

One saw also the development of two localization systems based on a ultrasound sensor connected to a wireless sensor network and a vision based system with a web-camera connected through wireless to a processing unit. It was seen that the maximum error while using the ultrasound system to perform localization was of 8cm when using all the filtering approaches. One should also consider that sometimes this could not hold due to interferences on the environment and possibly network congestions. As it was possible to see the errors are not so low while performing tracking. One should try to improve the robustness of the ultrasound system while performing tracking since it was seen that due to the high quantity of motes being used the measurement quality decreases.

One should refer here that were not made any considerations about the packet losses and delays in the wireless sensor network while using the ultrasound system. As a further work these characteristics should be carefully described and analysed in order to be able to characterize bad performances from the ultrasound system and relate them to these parameters.

The vision based system was seen to give measurements with high accuracy with almost no error. One should also try as a future work to develop a faster image processing algorithm based in OPENCV only and lower communication delays in order to reduce the total delay from 3s to 1s. It will remove the interest in the problem formulated in this work but will give a
accurate networked solution to be used for further work developed in KTH.

When using the vision based system the delay presented turned to be a
difficult problem to solve. One knows from the analysis made that for a given
model, the process noise $W$ should not be higher then a given value for a
given delay. Trying to cope with this delay showed that sometimes is better
even if the estimation accuracy is higher while using the camera more often,
to use more times the ultrasound sensor. This has also to do with the fact
that the ultrasound sensor measurements are not so inaccurate as they are
modeled to be.

As a further step one should also think about putting the vision based
system not communication through the 802.11g wireless protocol but through
the same wireless sensor network. This is possible to be performed using a
wireless node connected to the vision based processing unit and by enabling
the base station node to receive those measurements also. Also one should
try to test the system using the wireless camera nodes provided in [39]. This
upgrade in the network should pose interesting problems since the packet
losses and delays in the network will increase.

There were developed two models in order to describe the dynamics of
the mobile agent. From the two models proposed, model 2 was the one that
showed better performances in terms of tracking and localization accuracy
having lower position errors, even though model 1 was the one that had best
estimation performance when theoretically validated under the same condi-
tions. One can say that for the type of movements and velocities analysed
model 2 describe with good accuracy the real dynamics of the mobile agent.
Even though tracking with low errors was achieved for different types of
trajectories model 2 will not fit so good as for straight line movements. A
dynamical model that truly copes with the real mobile agent dynamics which
are not linear, but non-linear, should be developed. One can also foreseen
that for the new system the Kalman filter will not guaranty that the esti-
mated solution achieved will be optimal, since this is only seen for the linear
cases.

Another suggestion for further work is also that one should think about
developing a model that predicts the movement of the robot and helps the
estimator to give more accurate results.

A difficulty posed in this work was how to cope with the velocity and
acceleration requirements given by the estimator for both proposed models.
Here one saw that in a future work one should have a Robot with capability
of measuring, displaying and performing the movement under the estimator
requirements.

In terms of scheduling was seen that an offline solution gave more ac-
curate tracking estimations then the covariance-based one. One should not
forget that for the delay analysed the covariance-based has higher performance when a trade-off between estimation quality and communication cost is made, but for the problem presented here, when experimentaly validated, the covariance-based scheduler had worst performances.

As a motivation driven by the non-linear characteristics of the vision-based system one should develop a purely online scheduling solution, where the model parameters change depending on the location of the agent on the room. Also another case can be when the temperature, humidity, light, interferences, etc. can influence the model parameters as well.

As an interesting application for this system could be to try to use it with more ultrasound sensors and more cameras covering more areas. One can see that an interesting problem could be to cooperatively schedule 2 or more cameras covering two or more different areas while scheduling also the ultrasound sensors. With this and having an estimator based on the movement prediction one could turn on cameras based on the knowledge that the agent will be in the area covered by that given camera in some $t$ seconds. This pose interesting power management, estimation and modeling design problems.

Also the case where more then one mobile agent is used should be considered. This will put forward interesting problems to be solved for the ultrasound system and also vision based system.
Chapter 6

Appendix
Figure 6.1: Receiver Circuit
Figure 6.2: Transmitter Circuit
<table>
<thead>
<tr>
<th>Name</th>
<th>Location</th>
<th>Platforms</th>
<th>Architecture</th>
<th>Software</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carina Lab</td>
<td>Edgar Mine</td>
<td>Remote programming, monitoring</td>
<td>Node is central Ethernet backchannel</td>
<td>TinyOS</td>
<td><a href="http://www.wisc.edu/~moldovan/docs/macro.pdf">http://www.wisc.edu/~moldovan/docs/macro.pdf</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PDAQ over Ethernet (PAE)</td>
<td>USB camera</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tiny Sky + Tiny Connect</td>
<td>Deluge</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ETH Zurich</td>
<td>Switzerland</td>
<td>ETNode-based support for different mote models</td>
<td>ETNode connected to ETNodes</td>
<td>TinyOS</td>
<td><a href="http://www.industry.ethz.ch/Projects/JavaETNode">http://www.industry.ethz.ch/Projects/JavaETNode</a></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Server communication through Bluetooth with mote and Internet with users</td>
<td>MySQL, SQL Server, PERM, TCP, IP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IBM Zurich</td>
<td>Switzerland</td>
<td>Focus on WSN communication/performance testing</td>
<td>Sensor network + mote + Internet</td>
<td>Linux</td>
<td>Fungar/6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Gateway connects via Enterprise Network to clients (uses publish/subscribe model, MQTT protocol)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Johns Hopkins</td>
<td>USA</td>
<td>Configuration and various users</td>
<td>Similar to TelLab</td>
<td>TinyOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tiny Sky + Tiny Connect</td>
<td>MySQL, Ruby on Rails/AJAX</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(=2.02)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Deluge T2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kanai</td>
<td>Ohio, SU</td>
<td>Experiment scheduling</td>
<td>Similar to TelLab</td>
<td>TinyOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tiny Sky</td>
<td>MySQL, Ruby on Rails/AJAX</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(=2.02)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Deluge T2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MobileEmulab</td>
<td>Utah</td>
<td>Management of multiple hardware resources</td>
<td>Mobile mote via 802.11 NICs</td>
<td>TinyOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Linux</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Cameras, Zeiss</td>
<td>Johnson/06</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MoniSense</td>
<td>University of Porto</td>
<td>Focus on support for the WSN lifecycle and application mobility (easy deployment of plug-and-play)</td>
<td>Mobile mote connected to 802.15 NICs</td>
<td>TinyOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Linux</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Cameras, Zeiss</td>
<td>Johnson/06</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MITE Lab</td>
<td>Harvard</td>
<td>Interfaces with scheduling, programming, automatic logging and administration (quasi)</td>
<td>Mobile mote connected to central server via Ethernet and Internet</td>
<td>TinyOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Linux + Apache/MySQL/PHP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not necessarily TinyOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Warner-Aiken/05</td>
<td></td>
</tr>
<tr>
<td>MobileNeXt</td>
<td>UC Berkeley</td>
<td>Debugging</td>
<td>Wireless mote (P2P)</td>
<td>TinyOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Deluge</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network Emulab</td>
<td>UCBerkeley</td>
<td>Emulating real-world</td>
<td>Wireless mote (P2P)</td>
<td>TinyOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Deluge</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SensorNet (Userlabs)</td>
<td>UC Berkeley</td>
<td>Focus on energy cost and bandwidth</td>
<td>Wireless mote (P2P)</td>
<td>TinyOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Deluge</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tuxedo</td>
<td>University of California</td>
<td>Support for one type of mote (flat or hierarchical)</td>
<td>Wireless mote (P2P)</td>
<td>TinyOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Deluge</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.3: Wireless Sensor Network Testbeds - a survey
Figure 6.4: System Breakdown Structure
Figure 6.5: Floor plan - KTH Q 6th (SSS)
Bibliography


