Technical Report

On a Test-bed Application for the ART-WiSe Framework

Ricardo Severino

Mário Alves

TR-061103

Version: 1.0

Date: Nov 2006

On a Test-bed Application for the ART-WiSe Framework

Ricardo SEVERINO, Mário ALVES

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail: {rars, mjf}@isep.ipp.pt

http://www.hurray.isep.ipp.pt

Abstract

This report describes the development of a Test-bed Application for the ART-WiSe Framework with the aim of providing a means of access, validate and demonstrate that architecture. The chosen application is a kind of pursuit-evasion game where a remote controlled robot, navigating through an area covered by wireless sensor network (WSN), is detected and continuously tracked by the WSN. Then a centralized control station takes the appropriate actions for a pursuit robot to chase and "capture" the intruder one.

This kind of application imposes stringent timing requirements to the underlying communication infrastructure. It also involves interesting research problems in WSNs like tracking, localization, cooperation between nodes, energy concerns and mobility. Additionally, it can be easily ported into a real-world application. Surveillance or search and rescue operations are two examples where this kind of functionality can be applied.

This is still a first approach on the test-bed application and this development effort will be continuously pushed forward until all the envisaged objectives for the Art-WiSe architecture become accomplished.

Keywords: Wireless sensor networks, real-time communications, RSSIbased localization mechanism, test-bed application, mobile robots.

List of Figures

Section 1 - Introduction

1.1 Context and Motivation

Recent advancements in wireless communications and micro-sensing embedded technologies are enabling the real deployment of Wireless Sensor Networks (WSN). WSNs consist of potentially thousands of sensor nodes with multiple sensing capabilities, such as vibration, light, temperature, magnetic and acoustic sensing. WSN nodes must also support some (limited) processing, memory and radio communication functionalities, enabling multi-hop message routing and self-healing.

Wireless Sensor Networks offer new ways to monitor our environment, continuously and almost invisibly, holding the promise of many new ubiquitous and pervasive computing applications. Examples include target tracking, intrusion detection, wildlife habitat monitoring and climate control. In fact, even though the available technology is still emerging, it has been witnessing a quick acceptance. For instance, sensor networks have already been deployed for environmental monitoring (e.g. monitoring nesting behavior of endangered birds in a remote island [1]), precision agriculture (e.g. monitoring of temperature and humidity in vineyards [2]), and military and surveillance purposes (e.g. classification and tracking of trespassers [3]), just to mention a few “real-world” examples.

While their potential benefits are clear, a number of problems must be solved in order for wireless sensor networks to gain widespread use. These problems include issues such as security, calibration and failure detection, as well as other related to the timing and reliability behavior in critical applications like target tracking, considering the limited resources of the nodes.

In order to improve the timing and reliability in wireless sensor networks, a R&D framework, called ART-WiSe (**A**rchitecture for **R**eal-**T**ime communication in **Wi**reless **Se**nsor networks) has been defined. It consists on using a two-tiered architecture where a more powerful wireless network acts as a backbone of an underlying WSN. One of the major goals in the ART-WiSe framework is to rely as far as possible on standard communication protocols rather than defining new alternatives. The main reason is to push forward solutions that match standardization efforts and commercial-off-the-shell (COTS) platforms. For that purpose, research work is being focused on the use of the recently standardized communication protocols, IEEE 802.15.4 and ZigBee, initially proposed for Low-Rate Wireless Private Area Networks (LR-WPANs).

This project addresses the design of a test-bed application, using essentially COTS, including mobile robots and wireless sensor network platforms, with the aim of providing a way to assess, validate and demonstrate the ART-WiSe architecture. This is particularly important when dealing with WSN technology since besides relying on wireless communications, which are very sensitive to the environment, the available products are very recent. These issues make it even more important to carry out experimental testing and validation of theoretical work.

1.2 Structure of this report

In Section 2 is made a brief introduction to Wireless Sensor Networks and to the ART-WiSe Framework architecture. An overview of the project, concerning the general application architecture and used technologies is presented in Section 3. Section 4 addresses the localization mechanism and Section 5 the full test-bed application architecture in more technical detail. Finally, in Section 6 is made a discussion regarding the results achieved, improvements and future work to carry on the test-bed.

Section 2 – On the ART-WiSe Framework

2.1 Overview of a WSN

A Wireless Sensor Network (WSN) is typically composed of a large set of sensor nodes with multiple sensing capabilities, such as vibration, light, temperature, magnetic and acoustic sensing, scattered in a controlled environment. This set aims the collection of specified data needed for the monitoring/control of a predefined area/region. The delivery of sensory data for process and analysis, usually to a control station (also referred as sink), is based on the collaborative work of the WSN nodes in a multi-hop fashion (Figure 1)

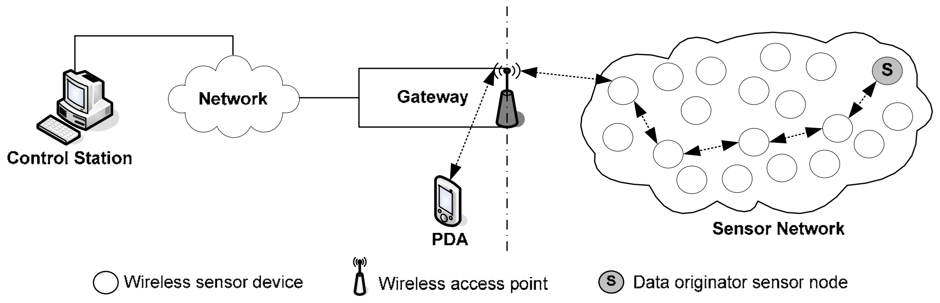


Figure 1 - Topology of a wireless sensor network

Hence, a WSN node must include some basic capabilities, namely sensing (eventually other I/O), processing (and memory) and wireless communications, acting as:

- **Data source**. Producing sensory data by interacting with the physical environment and collecting a specified data needed for control (temperature, humidity, pressure, movement…).

- **Data router**. Transmitting and relaying/routing data from one neighbor sensor node to another, towards the control station, which processes and analyses the data collected from the different sensors/nodes in the network.

2.2 Real-time Performance in WSNs

This particular form of distributed computing raises many challenges in terms of real-time communication and coordination due to the large number of constraints that must be simultaneously satisfied.

As stated in [4], WSNs interact directly with real world physical events, which may exhibit unpredictable *spatiotemporal* properties, hard to characterize with traditional methods. Moreover, when trying to achieve real-time performance, we must overcome the node's limited resources (e.g. low power, low CPU speed, limited storage capacity, bandwidth, short radio coverage). Hence, critical issues like energy efficiency and system robustness must be tackled. For example, it is not efficient to keep sensors continuously monitoring the environment, only for the benefit of a fast response, since that reduces system lifetime. Likewise, the use of computational expensive algorithms for real-time detection is not suitable for WSN applications.

2.3 The ART-WiSe Framework

The ART-WiSe (Architecture for Real-Time communications in Wireless Sensor networks) framework, aims at providing new communication architectures and mechanisms to improve the timing and reliability performance of Wireless Sensor Networks (WSNs). The ART-WiSe architecture is based on a two-tiered network structure (Figure 2) where a wireless network (Tier 2) serves as a backbone for a WSN (Tier 1).

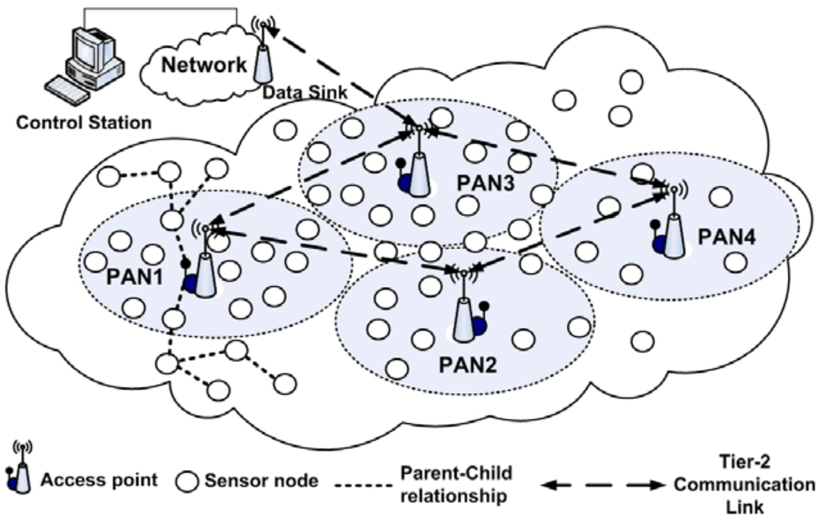


Figure 2 - Example of the ART-WiSe network topology

The ART-WiSe architecture relies (as much as possible) on standard communication protocols and commercial-off-the-shell technologies – IEEE 802.15.4/ZigBee for Tier 1 and IEEE 802.11 for Tier 2.

Tier-2 is an IEEE 802.11-compliant network acting as a backbone for the underlying sensor network. It is composed of a scalable set of special nodes called Access Points, which act as interfaces between the two tiers. Each Access Point must also act as a Personal Area Network (PAN) coordinator of the IEEE 802.15.4 Wireless PAN (WPAN) it manages. The IEEE 802.11 protocol is widely used, very mature and represents a cost-effective solution with powerful networking capabilities, high bandwidth (11-54 Mbps) and long transmission ranges (>100 m). Although the basic IEEE 802.11 does not provide any Quality of Service (QoS) guarantees, it has been shown that it performs well under lightly loaded networks in 0 and [6].

Tier-1 is an IEEE 802.15.4-compliant WSN interacting with the physical environment (e.g. to collect sensory data). The IEEE 802.15.4 protocol [7] is characterized by a low data rate (250 kbps), a short transmission range (10-30 m) and low power consumption, thus leading to limited communication capabilities. This protocol has several appealing features to fulfill different requirements of WSN applications. Tier 1 is partitioned into several independent WPANs, each of them managed by one Access Point. Each WPAN may still be structured into multiple clusters, whenever the density/location of the Access Points does not provide direct coverage for the WSN nodes.

2.4 On a Test-bed Application for the ART-WiSe framework

As previously referred, the objective of this work was to kick-off the design and implementation of a test-bed application for the Art-WiSe Framework. This field-trial will serve to access, validate and demonstrate the Art-WiSe architecture.

In our particular case, the application had to satisfy some requirements, namely the application should:

1. be as much appealing and realistic as possible, nevertheless limited to the available human and technological resources;
2. include a relevant and scalable number of WSN nodes and of static and mobile Access Points;
3. allow to assess the feasibility of the ART-WiSe architecture, based on the chosen/available technologies;
4. allow to assess the real-time behavior of the ART-WiSe architecture (tackling critical events), comparing to analytical and simulation results;
5. allow to assess the scalability of the ART-WiSe architecture (adaptable density of Access Points), enabling the comparison with “traditional” 1-tiered WSNs;

2.5 Contributions of this report

The main contributions regarding this report are the following:

- Specification of the test-bed application; this included the analysis of other field trials in the area of WSNs and the investigation of relevant and potential application domains. Several presentations and discussions within the research team leaded to the chosen “pursuit-evasion” application.
- Theoretical analysis of different localization mechanisms.
- Experimental evaluation of an IEEE 802.15.4 RSSI-based localization mechanism and implementation of this method in the test-bed application, enabling localization of the Intruder robot and positioning of the Pursuer.

Specification and development work on the following:

- Overall software architecture for the Pursuer robot to interface with the WSN and with the Control Station and to navigate in order to pursuit the Intruder.
- Overall software architecture for the remote control of the Intruder robot.
- Overall software architecture for the Control Station including the developing a virtual representation of the test-bed scenario in OpenGL and a user interface in GTK+.
- Development of the software architecture of the sensor nodes to enable the support for the localization mechanism and the overall application.

Section 3 - Project Overview

3.1 Snap-shot of the test-bed application

As previously referred, a Pursuit-Evasion application has been chosen. This kind of application imposes stringent timing requirements to the underlying communication infrastructure. It also involves interesting research problems in Wireless Sensor Networks (WSNs) like tracking, localization, cooperation between nodes, energy concerns and mobility. Additionally, it can be easily ported into a real-world application. Surveillance or search and rescue operations are two examples where this kind of functionality can be applied.

There are four entities on the application:

- **control station**; acts as a data sink, providing information related to the state of the application to the user level and performing the necessary data collection and processing necessary to the overall application;
- **Intruder robot team and Intruder Remote Control**; The Intruder robots are remotely controlled via an IEEE 802.11 link and move inside the WSN covered area. In order to control the Intruders an Intruder Remote Control was developed, capable of displaying the image from the Intruder's mounted camera and accept control input from a joystick.
- **Pursuer robot team**; with an autonomous behavior whose function is to capture the intruders based on the information provided by the Control Station and the WSN.
- **WSN**; featuring Wireless Sensor node responsible for tracking the intruders inside the deployment area and relaying that information to the Control Station.

The objective of the application is to detect, localize, track and pursuit the Intrude, until the Pursuer robot, aided by the WSN abilities to find the Intruder, gets close enough to it. Figure 3 illustrates an example scenario.

Currently, the intruder and pursuer teams include just one robot each but the complexity of the application will tend to grow in the medium-term. Commercial-off-the-shelf mobile robots platforms (WifiBot [13]]) are being used for this purpose.

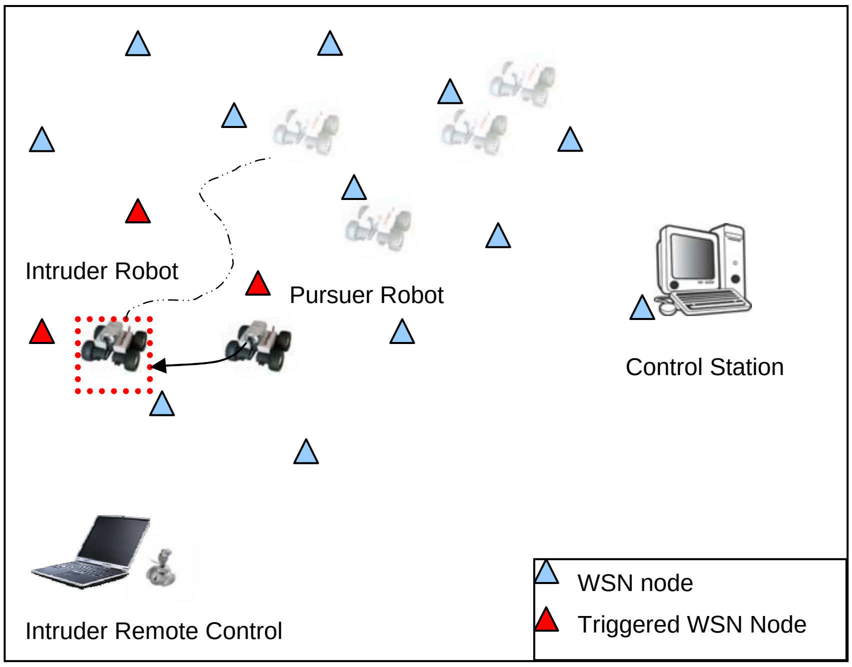


Figure 3 - Snapshot of the Test-bed application

The four entities previously mentioned are represented in Figure 3. An Intruder robot is driven through the WSN remote controlled by an operator. Some of the WSN nodes are triggered by the presence of the Intruder and this information is relayed to the Control Station. The Control Station then computes the Intruder's location and informs the Pursuer robot that will immediately initiate the pursuit by moving towards the last known position of the Intruder. This process will be repeated until the Pursuer is close enough to the Intruder.

3.2 On the used technology

3.2.1 WSN nodes

MICAz motes (Figure 4) from Crossbow [8] have been used to deploy the WSN. They feature an ATMEL ATmega128L 8-bit microcontroller with 128 KB of in-system programmable memory. This low-power microcontroller features an advanced RISC architecture with 133 instructions; most of them with a single clock cycle execution time. Its operation is fully static and can offer an up to 8 MIPS throughput when running at 8 MHz.

Figure 4 - MICAz mote

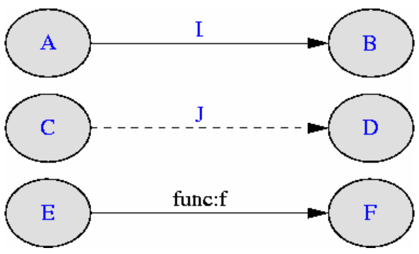
It also features:

- 128 KB of Program memory (in-system reprogrammable flash);
- 4 KB of EEPROM;
- 4 KB of Data memory (internal SRAM);

Besides these memory capabilities, the ATmega128L enables to address up to 64 KB optional external memory space.

These nodes run TinyOS [9] which is an open-source event-driven operating system designed for wireless sensor network nodes that have limited resources. The operating system files are written in NesC [10]. As stated in [11], this language is an extension to C designed to embody the structuring concepts and execution model of TinyOS.

As stated in [12] there are two types of files in TinyOS: components and interfaces. The components can be configuration or modules. The modules implement one or more interfaces; the configuration wires other components together. An application is a combination of several components linked or “wired” together. Figure 5 shows the graphical arrangement of this component “wiring”. The interaction between components is provided by the interfaces. For a component to call the commands in an interface it must implement the events of that interface.



A requires interface **I**, **B** provides **I**, and **A** and **B** are wired together.

C and **D** both require or both provide **J**. The direction of the arrow indicates that the original wiring is "**C** = **D**".

E requires function **f**, and **F** provides function **f**.

Figure 5 - Graphical arrangement of the components and their wiring

TinyOS's component library includes network protocols, distributed services, sensor drivers, and data acquisition tools, each of them can be used as-is or be further refined for a custom application.

TinyOS's event-driven execution model enables fine-grained power management yet allowing the scheduling flexibility required by the unpredictable nature of wireless communication and physical world interfaces.

3.2.2 Mobile Robots

The mobile robotic platform used in the test-bed application is the WifiBot [13]. The system architecture is build around a double bus Ethernet-I²C and a CPU that acts as a bridge between the two. This same CPU works as an IEEE 802.11 access point, enabling wireless access to the Ethernet bus from the outside. Figure 6 depicts the internal architecture of the robot.

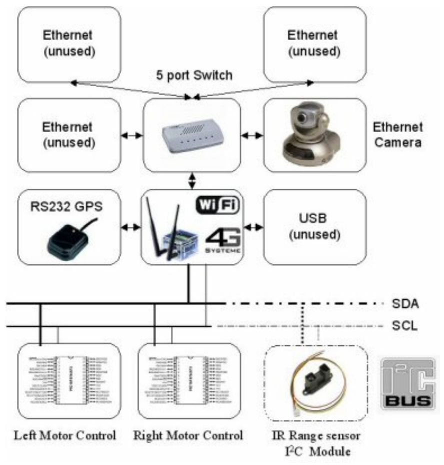


Figure 6 - Internal architecture of the WifiBot

In general the embedded LAN is used for peripherals of a certain importance such the IP camera while the I²C bus is useful for connecting more simple modules based on microcontrollers. To finish, the robot features one RS232 port which can be bridged to upper levels as well. This makes possible to add to the robot new modules based on simple microcontrollers.

The embedded CPU is a 4G Access Cube [14]. This is a new hardware platform dedicated to Wireless LAN Mesh Routing, developed by 4G Systems. Some of its interesting features are:

- ✓ 400 MHz MIPS processor AMD Alchemy Au1500
- ✓ 64 MB RAM
- ✓ 32 MB Flash
- ✓ 100 Mbps Ethernet
- ✓ Power Over Ethernet Standard IEEE 802.3af
- ✓ USB host/USB device

- ✓ Scope for installing up to 8 MiniPCI devices via four dual adapters. The robot has space for one MiniPCI.
- ✓ WLAN card with RP-SMA connection
- ✓ Dimensions: 7 x 5 x 7 cm
- ✓ Power rating: 4 W

The Access Cube runs the nylon Linux distribution as operating system. It features WirelessLan, routing, MeshRouting, autoconfiguration, IpSec and VPN all in a compact design to fit on the 32 MB flash. It is completely licensed has Open Source and is based on OpenEmbedded [15] which is a tool for building embedded devices. The datasheet of the robot is showed in Annex 2.

As it should be expected in an embedded system, the platform lacked a compiler. So, in order to compile an application for a different architecture from the one we were working on, like MIPS and x86, a toolchain had to be built. The toolchain consists of a number of components. The main one is the compiler itself gcc, which can be native to the host or a cross-compiler. This is supported by binutils, a set of tools for manipulating binaries and by the C-library glibc.

Since in the future it could be necessary to install new developed software packages in the robot, or even compile a special crafted linux distribution for it, the most reasonable option to build the toolchain was to install the OpenEmbedded environment, since all the nylon packages are available under it. OpenEmbedded was designed to be able to handle different hardware architectures, support multiple releases for those architectures, and utilize tools for speeding up the process of recreating the base after changes have been made.

Bitbake is the tool that reads the OpenEmbedded metadata and does all the work. It is responsible for:

- ✓ Building the compiler and cross-compiler versions specified, as well as configuration tools.
- ✓ Fetching sources from the internet.
- ✓ Configuring, compile and deploy, create packages including the C library.
- ✓ Compiling for several architectures in parallel, just by duplicating the build directories.
- ✓ Supporting several package formats: ‘.rpm’, ‘.ipk’, ‘.deb’.
- ✓ Cross-compiling single packages.

Section 4 – On the Localization Mechanism

4.1 Choosing the localization system

A problem faced throughout the design of the application was how to achieve localization inside the Wireless Sensor Network (WSN), both for the Pursuer positioning and Intruder detection. One possibility would be to rely on odometry readings from the robots for determining position. Nevertheless, this option has the disadvantage of accumulating errors over time, so common on dead reckoning. The *systematic* and *non-systematic* errors from this positioning mechanism plus the inaccessibility to the raw data from the wheel encoders of the robots, since only speed values can be obtained from the I2C bus, would eventually lead to positioning problems over time. So, the need for an absolute positioning system was demanding. By using the WSN for this task, in addition to solving the above issue, it was possible to add more stress into the network, necessary for later performance assessment of the ART-WiSe architecture. It also allowed us to go even further, by learning more on how much we could rely on a deployed WSN for obtaining positioning.

There are many proposals on this subject using different kinds of range measurements, like Time of Arrival (ToA) or Radio Signal Strength (RSS). For instance, the Cricket [16] indoor localization system uses ToA obtained by combining ultrasound and Radio Frequency (RF), whereas MoteTrack [17] uses RSS measurements to provide location signatures for each node in the network. As stated in [18], the majority of existing location discovery approaches consist of two basic phases: distance (or angle) estimation and distance (or angle) combining. The most popular methods for estimating the distance between two nodes are:

- Received Signal Strength (RSS) techniques measure the power of the signal at the receiver. Based on the known transmit power, the respective propagation loss can be calculated. Theoretical or empirical models are used to translate this loss into a distance estimate. This method has been used mainly for RF signals.

- Time based methods (ToA,TDoA) record the time-of-arrival (ToA) or time-difference-of-arrival (TDoA).The propagation time can be directly translated into distance, based on the known signal propagation speed. These methods can be applied to many different signals, such as RF, acoustic, infrared and ultrasound.

- Angle-of-Arrival (AoA) systems estimate the angle at which signals are received and use simple geometric relationships to calculate node positions.

A more detailed discussion of these methods can be found in [19]. For the combining phase, the most popular alternatives are:

- Hyperbolic tri-lateration, which is the most basic and intuitive method, locates a node by calculating the intersection of 3 circles.
- Triangulation is used when the direction of the node instead of the distance is estimated, as in Angle of Arrival (AoA) systems. The node positions are calculated by using the trigonometry laws of sines and cosines.

We have opted by the RSS range measurement method since it would not involve special hardware design and it could be easily implemented on the MICAz mote by using the CC2420 [20] Radio Signal Strength Indicator (RSSI) function.

Since we are estimating distances, the chosen method ought to be Lateration. However, this method imposes some practical problems. First, it is a computationally expensive method with a high number of floating point operations. Since it is probable that later on some of the localization computation will be made by the sensor nodes, this presents an issue given that the WSN lacks high computational skills. Second, it is highly sensitive to distance measurements errors. The RSSI measurements present some error, particularly when the nodes are deployed in an indoor environment, due to the multiple unpredictable interferences. Hence the lateration algorithm will not output a result in cases where it is not possible to find an intersection point (of the three circles).

A much simpler method is presented by Savvides et al. [21] as part of the N-hop multilateration approach. The main idea is to construct a bounding box for each anchor using its position and distance estimate, and then to determine the intersection of these boxes. The position of the node is set to the centre of the intersection box.

Figure 7 illustrates the Min–max method for a node with distance estimates to three anchors. Note that the estimated position by Min–max is close to the true position computed through Lateration (i.e., the intersection of the three circles).

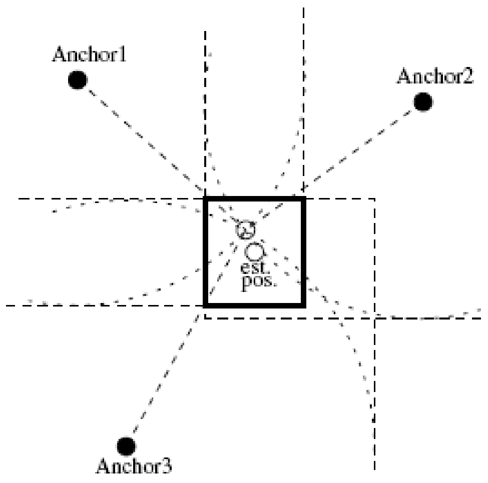


Figure 7 - Estimated Position by the Min-max algorithm and by Lateration

The bounding box of anchor *a* is created by adding and subtracting the estimated distance (d_a) from the anchor position (x_a, y_a) (1):

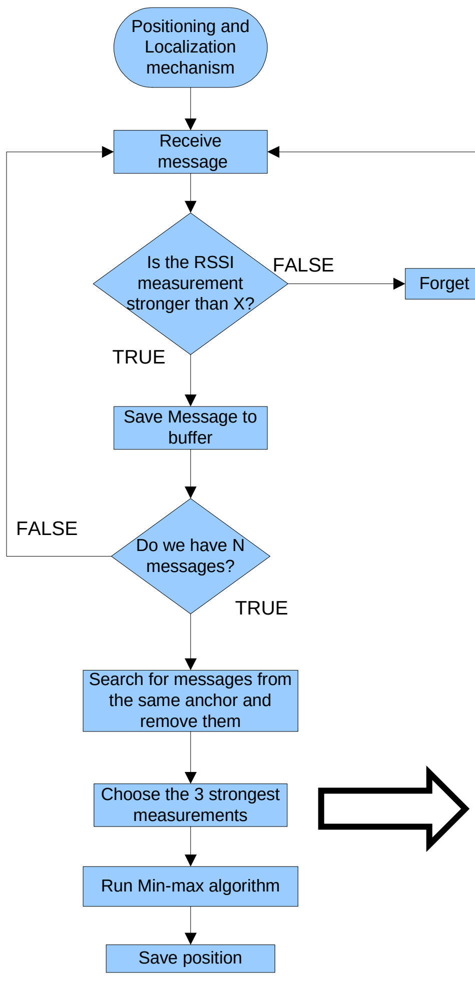
$$[x_a - d_a, y_a - d_a] \times [x_a + d_a, y_a + d_a] \tag{1}$$

The intersection of the bounding boxes is computed by taking the maximum of all coordinate minimums and the minimum of all maximums (2):

$$[\max(x_i - d_i), \max(y_i - d_i)] \times [\min(x_i + d_i), \min(y_i + d_i)] \tag{2}$$

The final position is set to the average of both corner coordinates. As for Lateration, the final position should only be accepted if the residue is small.

Figure 8 shows the implementation of the Min-max algorithm.


Positioning and Localization mechanism
Receive message
Is the RSSI measurement stronger than X?
FALSE
Forget message
TRUE
Save Message to buffer
FALSE
Do we have N messages?
TRUE
Search for messages from the same anchor and remove them
Choose the 3 strongest measurements
Run Min-max algorithm
Save position

```
        // Xcoord and ycoord are the coordinates. AnchorXdist are the
    distances to the anchors

        xsum[0]=xcoordA+anchorAdist; ysum[0]=ycoordA+anchorAdist;
        xsum[1]=xcoordB+anchorBdist; ysum[1]=ycoordB+anchorBdist;
        xsum[2]=xcoordC+anchorCdist; ysum[2]=ycoordC+anchorCdist;

        xsub[0]=xcoordA-anchorAdist; ysub[0]=ycoordA-anchorAdist;
        xsub[1]=xcoordB-anchorBdist; ysub[1]=ycoordB-anchorBdist;
        xsub[2]=xcoordC-anchorCdist; ysub[2]=ycoordC-anchorCdist;

        xmin=xsum[0];
        ymin=ysum[0];
        xmax=xsub[0];
        ymax=ysub[0];
```

Figure 8 - Min-max implementation

As stated in [22] when noise is introduced in the range measurements, the two algorithms (Min-max and Lateration) show different behavior. Figure 9 shows the sensitivity of Lateration and Min-max when standard deviation percentage was varied from 0 to 0.25.

Lateration outperforms Min-max for precise distance estimates, but Min-max takes over for large standard deviations. Min-max is rather insensitive to bias, because stretching the bounding boxes has little effect on the position of the center.

For precise distance estimates and a small bias factor Lateration outperforms Min–max, but the bottom graph (Figure 9) shows that Min–max is probably the preferred technique when the standard deviation rises above 10%.

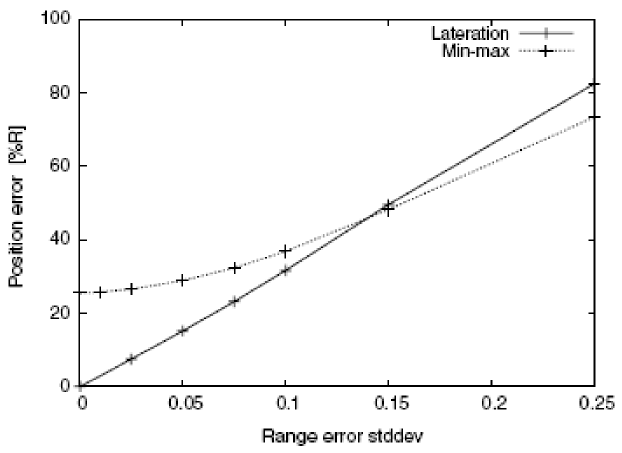


Figure 9 – Sensitivity to deviations on the range measurements by the Min-max and Lateration algorithms

On the experimental tests carried out and showed on 4.2, standard deviation values exceeded 10% validating the option for this algorithm.

4.2 Achieving localization

In order to achieve localization a table that could relate Distance and RSSI was built. This table allows the conversion of the RSSI measurements into distance for later use by the Min-max algorithm, on the Control Station and on the Pursuer Robot.

Four motes were placed around the Pursuer Robot setup to send broadcasts at certain power levels. Another mote was placed on top of the Pursuer connected to a MIB510 interface board [23]. This board enabled the interface through the serial port with a laptop running a serial port logging software. The purpose of this mote was to gather the broadcast messages and specially the received RSSI values and relay them to the serial port.

Figure 10 shows the setup of the experiment.

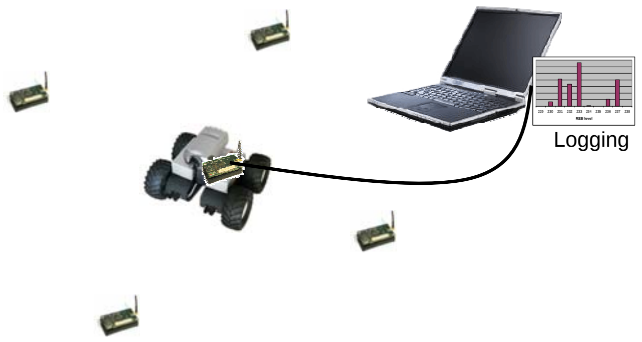


Figure 10 - Experiment setup

Experiments were made with power levels 3, 4, 5 and 6 of the MICAz mote, varying from -25 dBm to -15 dBm respectively.

The following graph (Figure 11) shows the plot of Distance vs RSSI for three of the tested power levels. A good linearity is important since it allows a more accurate translation from RSSI to Distance. As showed the results for the TX power level 5 present the best results. For the lower power level (power level 3), some linearity was found for a small distance of 90 cm. For the next two power levels better linearity was found. Yet, power level 4 only presented accurate results until a maximum of 120 cm away from the transmitter. This would prevent a lower granularity for the deployed sensor network. With power level 5, more accurate results were achieved at higher distances. This allowed a granularity of 180 cm which was enough for the constricted indoor environment where the test-bed was to be deployed.

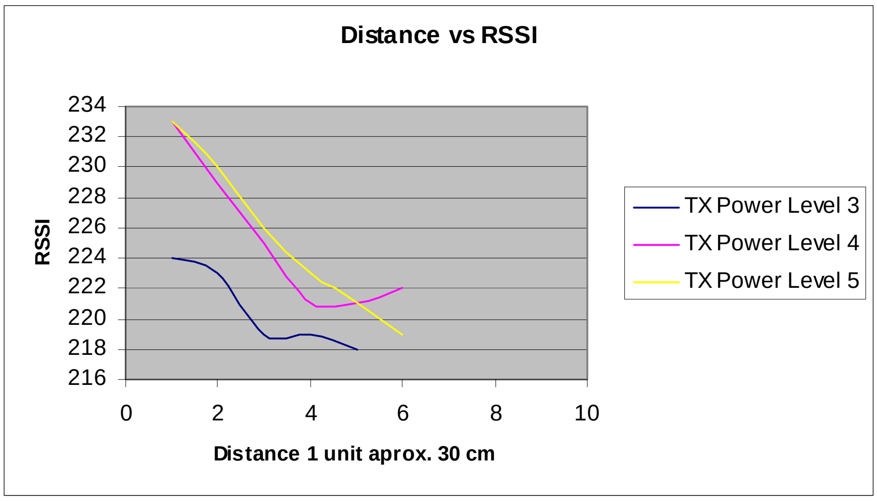
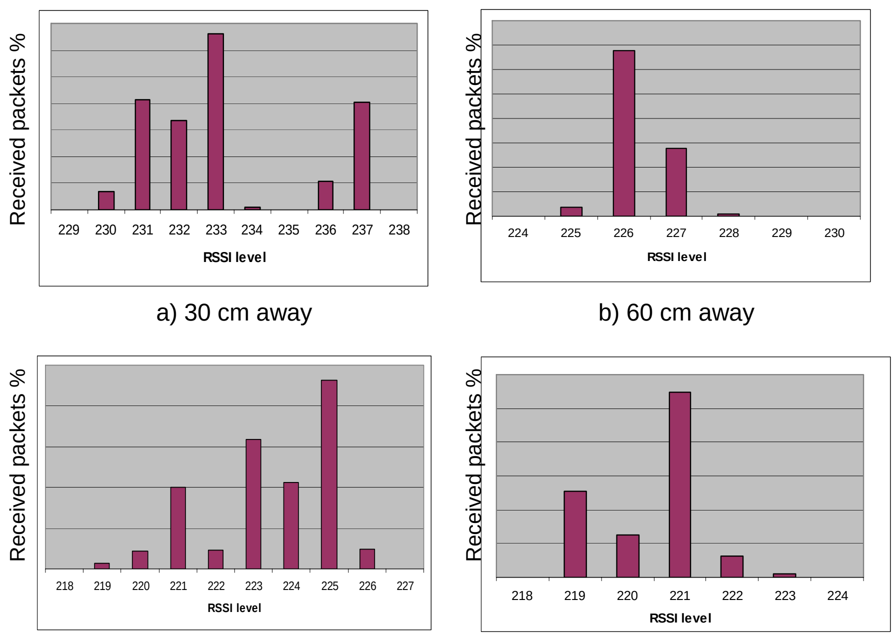


Figure 11 - Distance vs RSSI obtained at different TX Power Levels

Notice the placement of the four motes around the robot in Figure 10. By placing the motes in that shape it was possible to test the effect that different antenna orientations had on the RSSI values received by the mote on the robot and to find the spread of RSSI values that were likely to be found at a given distance even with different antenna orientations.

Histograms were built with the received values from the four motes at different distances for power level 5. Some of them are presented below (Figure 12).



a) 30 cm away b) 60 cm away

c) 90 cm away d) 130 cm away

Figure 12 - Histograms for the RSSI at different distances

The histograms in Figure 12 show the RSSI levels received at different distances by the four motes surrounding the robot. These results provided us with knowledge of which RSSI values were expected to be found at different distances.

Based on this experimental information was possible to build the table showed on Figure 13 that relates the RSSI values with the distance to the anchor.

```
if (rssi >= 235)                            range = 0.15;
else if (rssi < 235 && rssi >= 227)         range = 0.45;
else if (rssi < 227 && rssi >= 226 )        range = 0.75;
else if (rssi < 226 && rssi >= 224 )        range = 1.05;
else if (rssi < 223 && rssi >= 220)         range = 1.35;
else if (rssi < 220 && rssi >= 219)         range = 1.50;
else if (rssi <= 219)                       range = 1.65;
```

Figure 13 - RSSI to Distance Table

The algorithm presented in Figure 13 was implemented both in the Pursuer robot and in the Control Station to enable the translation from RSSI level to distance.

Because multiple RSSI values were found for the same distance in certain cases, most of the times due to the mote's antenna orientation, the algorithm was built by establishing a correspondence between discrete range levels and the spread of RSSI values encountered for that same range.

4.3 Implementation of the localization system

Two different approaches were implemented, based on the same localization mechanism. The first results in a positioning mechanism for the Pursuer robot, while the second works as the localization method of the Intruder.

Figure 14 shows the two implementations of the localization mechanism used on the application.

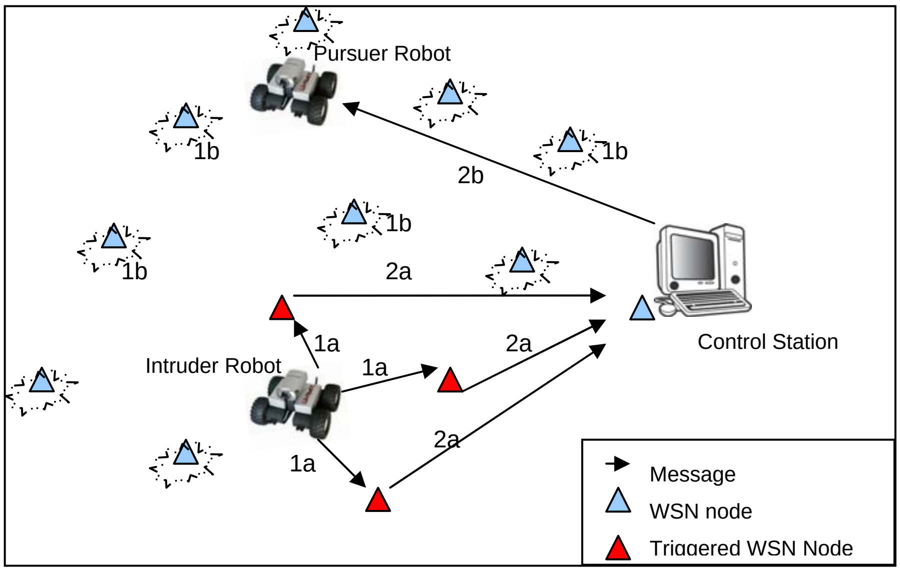


Figure 14 - Representation of the two implementations for the localization

For the Pursuer positioning, all the WSN nodes periodically broadcast a RSSI message which contains the node (x,y) coordinates and address (1b). These messages are received by the mote on the Pursuer robot and are then processed in the robot. For the Intruder localization, the robot initiates the process by announcing his presence sending a broadcast message similar to the latter but without coordinates (1a). This message is then received by the WSN and relayed to the Control Station in an “Intruder Alert message” (2a). The Control Station then processes these messages and instructs the Pursuer where to go (2b).

The intruder detection mechanism and the following mission dispatch to the Pursuer Robot are covered in more detail in the diagram of Figure 15.

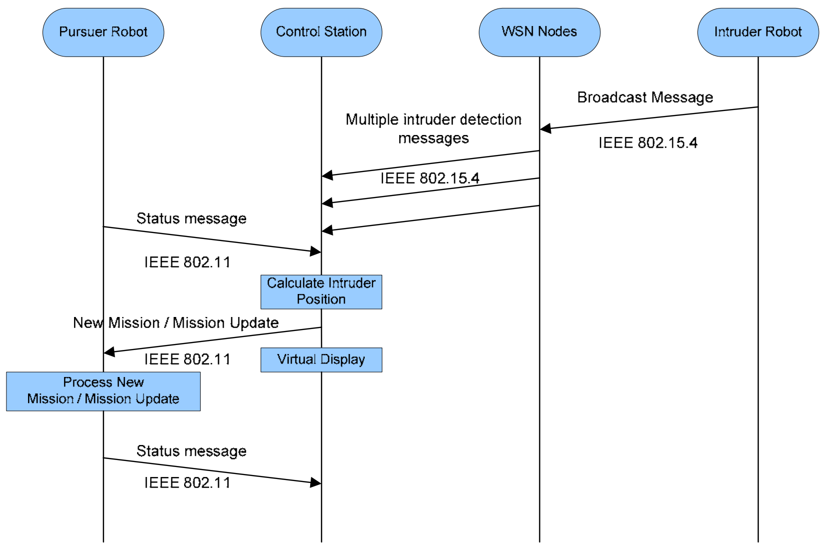


Figure 15 - Diagram of the intruder detection and localization mechanism

This process is repeated several times until the Pursuer Robot reaches the Intruder. A node placed on top of the Intruder robot broadcasts messages at a pre-programmed transmission power and timing rate. The WSN nodes that receive that Intruder message, save the received RSSI and build a new message with that recorded value and their coordinates and send it to the control station. The Control Station is expected to receive multiple messages of this kind from different nodes. As soon as a sufficient number of messages is received the Intruder, position is calculated based on the same algorithm used for the Pursuer Positioning service. This position is then relayed to the Pursuer robot. A virtual display of both the Pursuer and Intruder robots is built based on status messages from the Pursuer Robot and on the Intruder Localization mechanism respectably, as presented in 5.4.

The localization mechanism presented a maximum error of approximately 70 cm. We did not expect better results for the localization mechanism, as stated in [24], there are many sources of RSSI variability like:

Transmitter variability: Different transmitters behave differently even when they are configured exactly in the same way. In practice, this means that when a transmitter is configured to send packets at a power level of d dBm then the transmitter will send these packets at a power level that is very close to d dBm but not necessarily exactly equal to d dBm. This can alter the received signal strength indication and thus it can lead to inaccurate distance estimation.

Receiver variability: The sensitivity of the receivers across different radio chips is different. In practice, this means that the RSSI value recorded at different receivers can be different even when all the other parameters that affect the received signal strength are kept constant.

Antenna orientation: Each antenna has its own radiation pattern that is not uniform. In practice, this means that the RSSI value recorded at the receiver for a given pair of communicating nodes and for a given distance between them varies as the pairwise antenna orientations of the transmitter and the receiver are changed.

Multi-path fading and shadowing in the RF channel: in indoor environments, the transmitter signal gets reflected after hitting on walls or other objects. Both the original signal, as well as the reflected signal reach the receiver almost at the same time since they both travel at the speed of light. As a result, the receiver is not able to distinguish the two signals and it measures the received signal strength for both of them.

Battery condition can influence both the received RSSI as well as the transmission signal strength. When using old batteries, despite the nodes are configured in the same way, different RSSI readings are observed.

All the above reasons may cause error on the RSSI measurements eventually leading to a wrong position computation.

From the tests it was found that the standard deviation on the range measurements exceeded 10%. Indeed, the minimum standard deviation observed was 13.4% for the distance of 60 cm but values like 25% were obtained for other distances. Hence, the choice of the Min-max algorithm for the computation of the position did not impose significant error comparing to the lateration algorithm and may in fact improve the outcome since it is not so sensitive to distance measurement errors.

Section 5 - On the Test-bed Application Design

5.1 On the Sensor Nodes

The test-bed application is based on the sensor nodes programming. They are the entity responsible for allowing intruder detection, and for providing the positioning for the pursuer robot. The WSN consist of 20 MICAz motes deployed in a grid topology in an indoor environment. Figure 16 shows a view of a region of the WSN deployment area where four sensor nodes are showed.

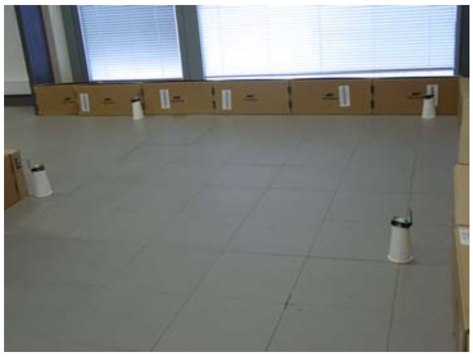


Figure 16 - Test-bed WSN sensor nodes

In order to support the localization service for the application, two message frame types were built. The first is the RSSI message periodically broadcasted by every mote (

Figure 17 a)). This frame format is used by the WSN nodes and by the Intruder mote to send periodic messages for RSSI identification by the receiver. The difference between the two cases is the non-filling of the Coordinates field by the Intruder mote and the different Message ID field. The Intruder Alarm Message (Figure 17 b)) is used by the WSN nodes to relay the measured Intruder message RSSI together with the coordinates of the WSN node to the Control Station.

a) RSSI broadcast Message

8 bits	*8 bits*	*32 bits*
Message ID	*Source Address*	*Coordinates*

b) Intruder Alarm Message

8 bits	*8 bits*	*32 bits*	*8 bits*
Message ID	*Source Address*	*Coordinates*	*RSSI Value*

Message ID = 11 – RSSI Broadcast Message

Message ID = 22 – Intruder Alarm Message

Figure 17 – Message types in the WSN

The flowchart below (Figure 18) shows the operation of the WSN nodes. Notice that each node has different coordinates pre-programmed.

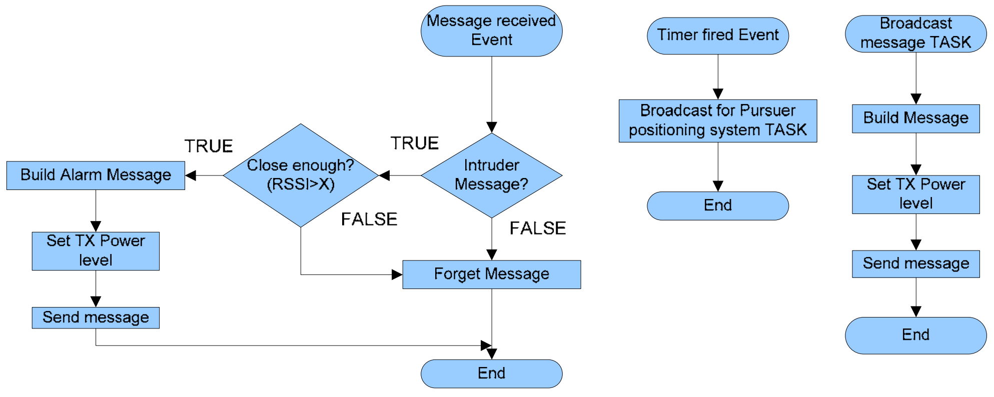


Figure 18 - WSN node workings

The Pursuer Robot WSN Interface and the Base Station WSN Interface behave in a similar manner. Both are responsible for filtering the message id field from the received frame, and processing the respective message, relaying the data through the serial port.

The difference lies on the extra RS232-to-TTL converter hardware that is placed on the Pursuer Robot to interface the mesh cube hardware. The schematic for this circuit was developed within this project and is showed in Annex 3 as well as the correspondent board layout.

5.2 On the Pursuer Robot

To enable the chase of the Intruder a Pursuer robot was developed. It accepts the missions from the Control Station and it features autonomous behavior. A picture of the Pursuer robot is presented in Figure 19.

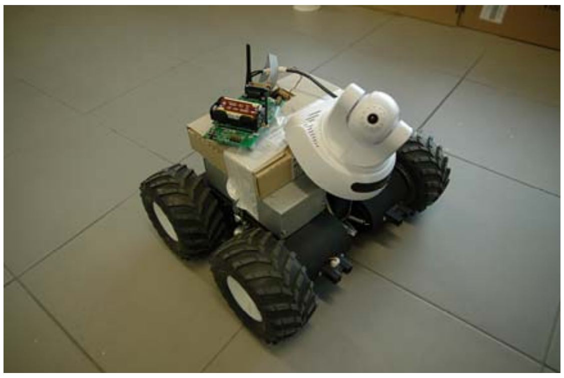


Figure 19 - The Pursuer robot

The block diagram in Figure 20 depicts the Pursuer robot architecture and the connections with the other test-bed entities. The test-bed entities which interact with the Pursuer are the WSN for positioning, the Control Station for mission dispatching and other eventual pursuer robots.

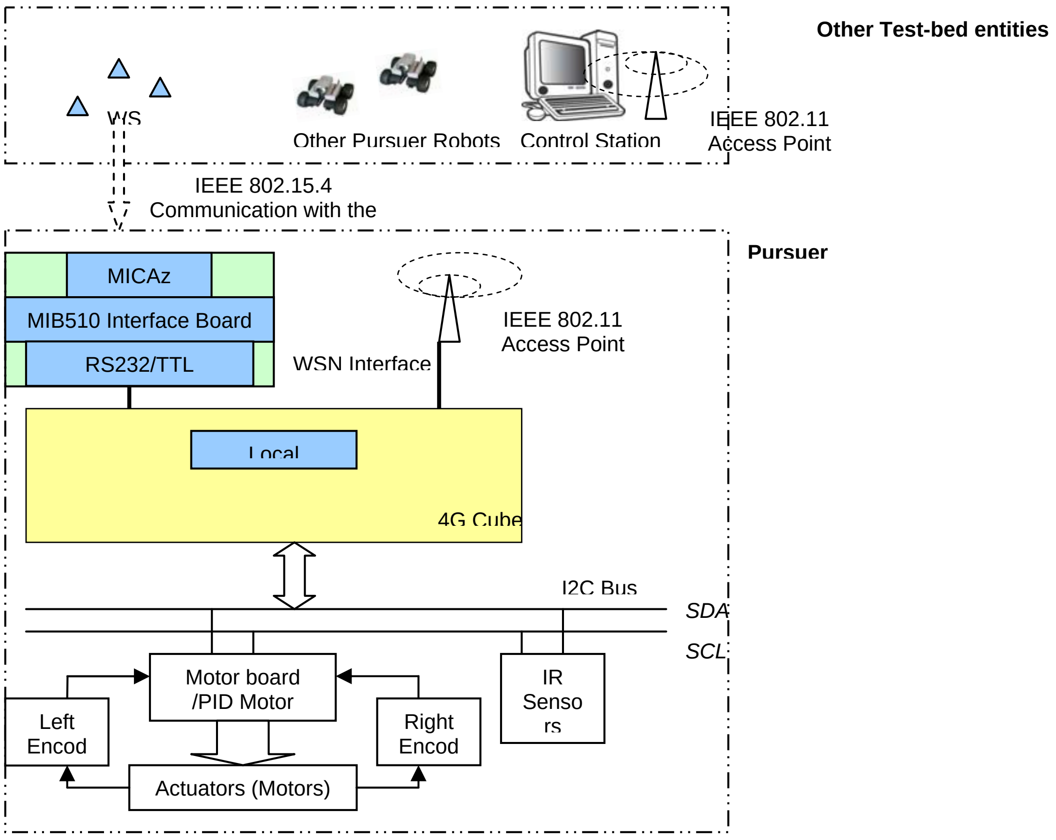


Figure 20 - Pursuer architecture and connections with other test-bed application entities

The Local Control block that appears in Figure 20 encompasses two fundamental software modules: Communications and Navigation. The first is responsible for maintaining all the necessary communications while the second deals with all the navigation issues to get the robot from one place to another.

5.2.1 Communications

The communications module of the Pursuer includes all the interface mechanisms linking Pursuer - Control Station and Pursuer – WSN, allowing the robot to run the WSN Positioning System and maintain communication with the Control Station, informing it of his current status. This communication is made in a broadcast fashion using the UDP/IP Protocol, since we envisage adding more Pursuer Robots in a near future and those robots must know the position of the other team members at all time.

Communications made between the Pursuer and the Control Station use the following message structure (Figure 21):

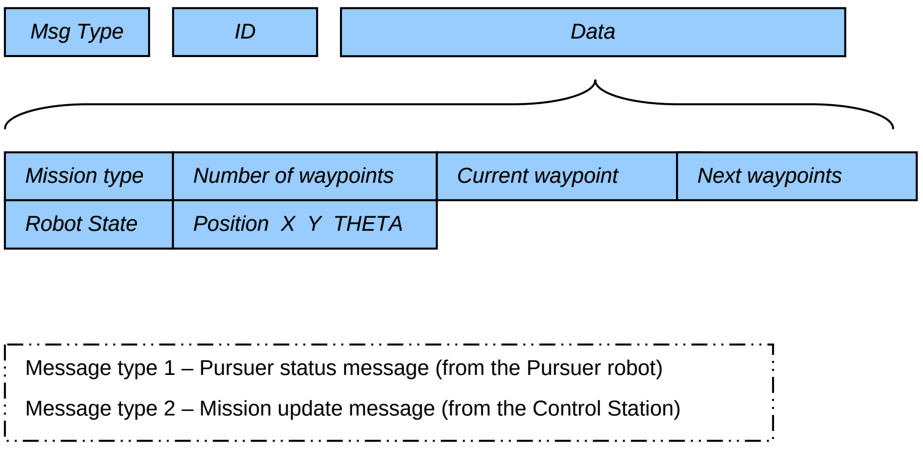

Msg Type
ID
Data
Mission type
Number of waypoints
Current waypoint
Next waypoints
Robot State
Position X Y THETA
Message type 1 – Pursuer status message (from the Pursuer robot)
Message type 2 – Mission update message (from the Control Station)

Figure 21 – Message structure for communications between Pursuer and Control Station.

The Message Type field identifies the contents of the message. Next, the ID field identifies the source of the message. The next blocks of data give multiple information about the mission the robot is currently running: Mission type, Number of Waypoints on the mission, the Current Waypoint and the coordinates of the next ones.

The last groups of data provide information about the current robot Position (X, Y, THETA), and of his internal state.

This same data structure is used for the messages coming from the Control Station. However, these come with a different Message Type field and just some of the other fields are used.

The following example (Figure 22) shows how the Control Station instructs the Pursuer with a new mission message:

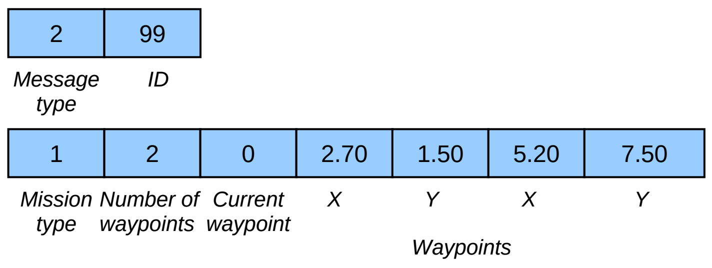


Figure 22 - New Mission message for the Pursuer

The WSN Positioning System works by reading the beacon messages from the WSN motes and running the Min-max algorithm on these readings. These beacon messages use the frame type specified at Figure 17 b) and are received by the WSN interface of the Pursuer robot. Messages are then processed and relayed for processing by the Robot according to the diagram in Figure 23.

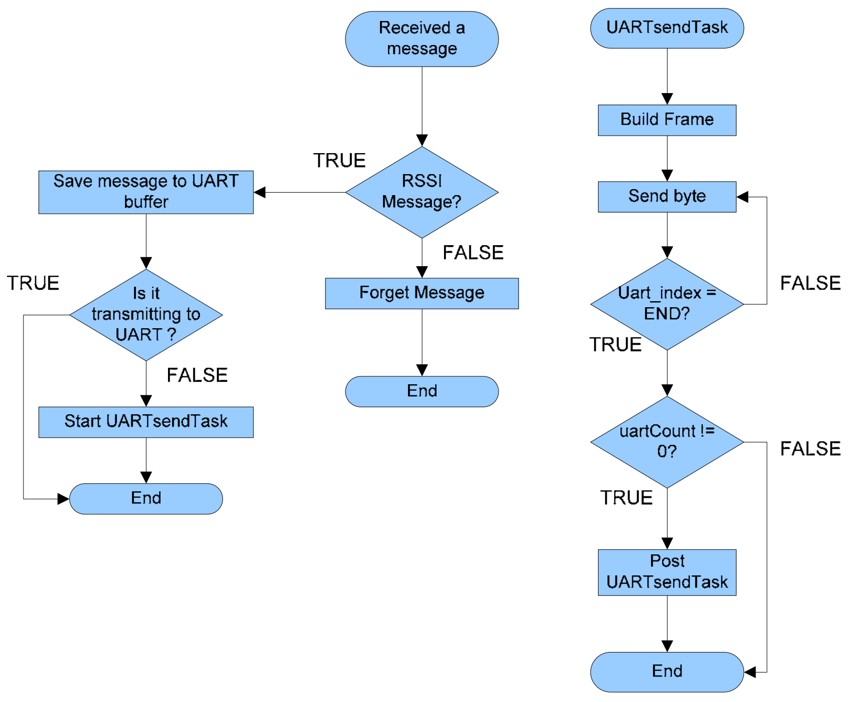


Figure 23 - Diagram of the interface with the WSN

5.2.2 Navigation Module

The navigation module encompasses the following tasks:

- obstacle avoidance;
- mission control;

The obstacle avoidance uses the two IR sensors for obstacle detection providing a detection range from 20 to 150 cm. The avoidance is merely reactive. The reading of the IR sensors values is made by interfacing the I2C bus.

The mission control software is responsible for executing the mission dispatched from the Control Station and for updating the current mission status continuously. Figure 24 shows the navigation algorithm used in the robot.

```
while(1) {
        if (OBSTACLE) {
                Avoid_obstacle();     // use reactive behavior to avoid the obstacle
        }
        if (!OBSTACLE) {
                if (STATE == DONE_AVOIDING_OBSTACLE)
                        {
                                Start_walking();
                                STATE = WALK_SAFE_DISTANCE;
                        }
                if (STATE == WALK_SAFE_DISTANCE)
                        {
                                If ( Distance > S1 ) {
                                        STOP();
                                        Update_position_WSN;
                                        STATE = NAVIGATING;
                        }

                                        // Start/Continue mission execution
                if (MISSION_UPDATE)     /* If there was a mission update from the
                                        control station in the meanwhile */
                        {
                                Update_waypoints();
                                Adjust_heading();
                        }
                if (STATE == NAVIGATING)
                        {
                                if ( POSITION == WAYPOINT) {
                                        MISSION_UPDATE = TRUE;
                                        }
                                else {  // we're not there yet…
                                        Start_walking();
                                        if (Distance > S2) {
                                                Update_position_WSN;
                                                Adjust_heading();
                                                }
                                }
                        }
        }
}
```

Figure 24 - Navigation algorithm

It starts by doing the obstacle avoidance. If the path is clear then it proceeds by navigating towards the destination waypoint. This is done by adjusting the heading in the direction of the waypoint and by moving a pre-programmed distance in a straight line. Position is then calculated through the WSN positioning mechanism and the heading is found and adjusted accordingly. The process is repeated until the destination waypoint is reached.

5.3 On the Intruder Robot

The intruder robot architecture is simpler than the one of the pursuer. It does not have to include the WSN interface and the robot is remotely controlled by a human operator (Figure 25). The WSN locates him via a MICAz mote mounted on board whose function is to periodically send a broadcast message at a pre-programmed power level. It can be considered as if the robot was announcing his position. By doing this, we did not have to be concerned with the WSN sensing capabilities by choosing the best type of sensors for this task, which was not our primary goal.

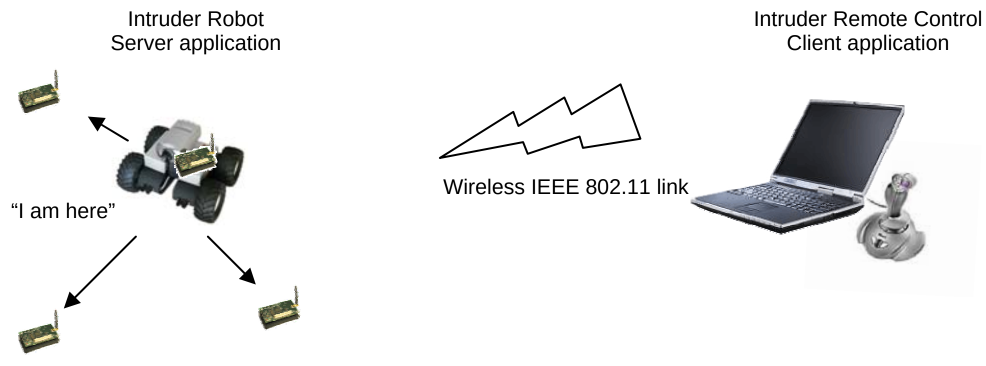


Figure 25 - Intruder communications with the remote control

Figure 26 shows the Intruder robot.

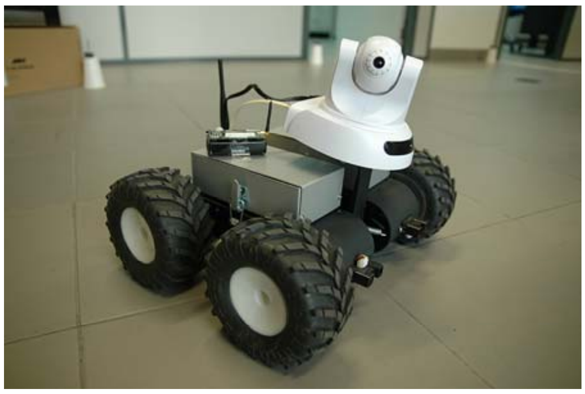


Figure 26- Intruder robot carrying a mote

The Intruder runs a server application that listens for incoming connections on his TCP/IP 15000 port for enabling the remote control of the robot (Figure 25). This connection is established wirelessly, through an IEEE 802.11 link, from a remote machine running the client application and with joystick input.

Figure 27 shows a flowchart of the server application software, running in the Intruder Robot. This software consists of two threads running in parallel. The main thread listens to the data that comes from the TCP/IP connection established by the remote client, and sets the speed of the right and left wheels accordingly. The other thread behaves as a software watchdog. This thread is a part of some safety concerns we had with the intruder software design. Since the robot operates remotely through a wireless connection and sometimes may be driven merely based on the robot camera images, accidents can easily happen. This is particularly acute when the robot is driven at maximum speed (about 1.6 m/s). Additionally, wireless communications may experience communication failures. This can also happen if the robot’s battery power level drops bellow a certain threshold, preventing it from maintaining communications with the Intruder Remote Control. Eventually, the last value in the I2C bus, correspondent to the motor speed sent by the remote operator would be sustained, which could result in damage to the robot.

To tackle these problems the following solutions were developed:

- Watchdog thread – This thread is a part of the server application running on the robot. It is a basic implementation of a software watchdog. In general, there is a shared variable between this one and the main thread, which is continuously incremented by the watchdog thread and reset by the main thread while there is data in the communication socket. If the value of the variable exceeds the pre-programmed value the main thread does not reset the variable. The robot is stopped by the watchdog thread and the main thread will be closed as well as the connection with the remote station. Then, the main thread will restart and wait for another connection.

➢ Obstacle Detection – Because of operator error, it is possible the robot will be driven into an obstacle. By taking advantage of the two IR sensors on the robot, the main thread on the server application stops the robot if an obstacle is sensed at close range (approximately 20 cm). Then, only commands that drive the robot away from the obstacle are accepted.

➢ Joystick enable button – To prevent the accidental dispatch of commands to the robot, the input of the joystick is only accepted if the operator is holding the FIRE button.

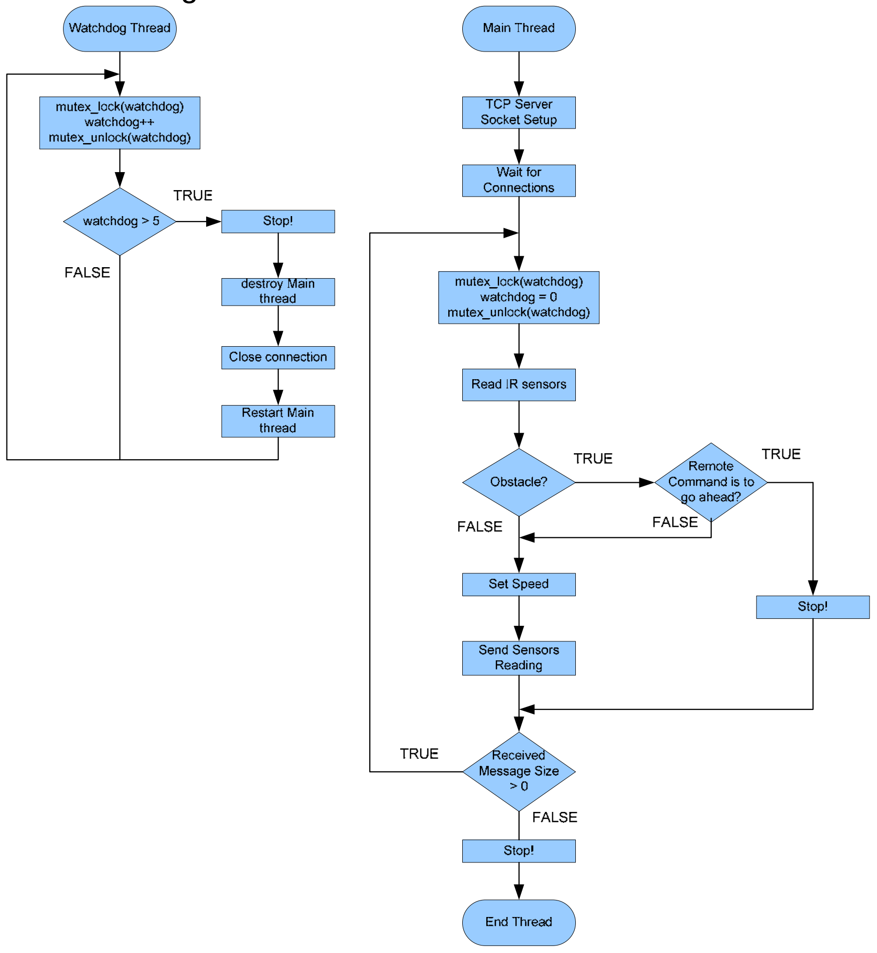


Figure 27 - Diagram of the server application software in the Intruder Robot

Figure 28 depicts the client application software used by an operator to remotely control the Intruder Robot. The main program launches two threads, one responsible for continuously sending the control actions to the robot and therefore maintaining the connection with it, and the other one responsible for receiving data from the robot (IR sensors, speed…). Simultaneously, the main program reads the joystick input and processes it. It allows the remote control of both the motion of the robot, setting the speed of the right and left wheels, and of the IP camera in the robot through the CGI interface of the camera.

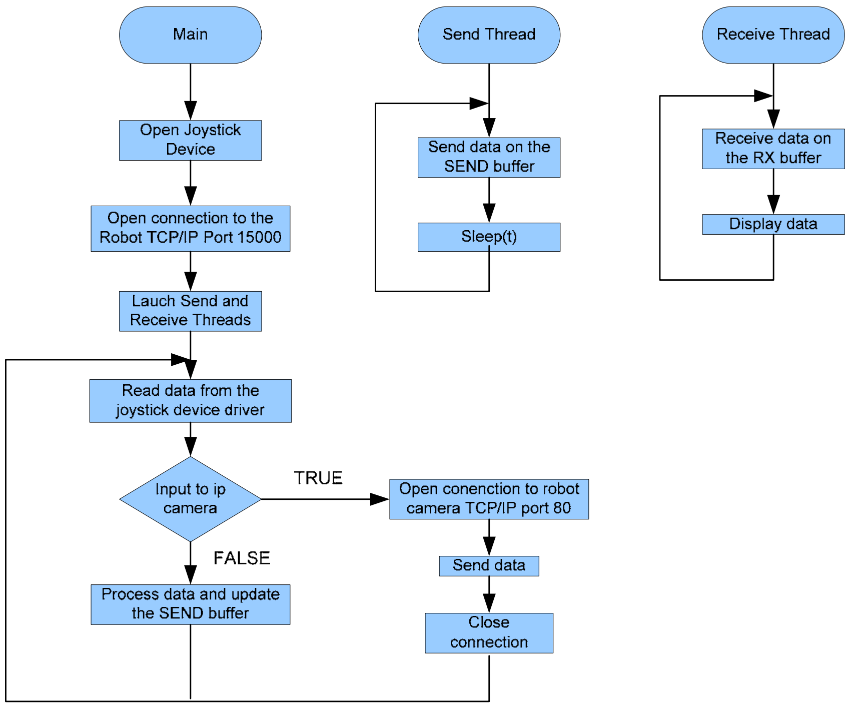

Main
Open Joystick Device
Open connection to the Robot TCP/IP Port 15000
Lauch Send and Receive Threads
Read data from the joystick device driver
Input to ip camera
TRUE
FALSE
Open conenction to robot camera TCP/IP port 80
Send data
Close connection
Process data and update the SEND buffer
Send Thread
Send data on the SEND buffer
Sleep(t)
Receive Thread
Receive data on the RX buffer
Display data

Figure 28 - Diagram of the client software application for the Intruder (Intruder Remote Control)

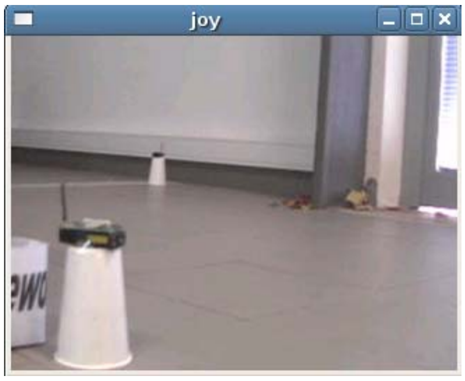


Figure 29 - View from the IP camera on the Intruder

The IP camera image is displayed in a GTK window (Figure 29) allowing the remote operation of the intruder.

5.4 On the Control Station

The control station runs an application on a Personal Computer running Linux, with a user interface made on GTK+. It has communications capabilities for both IEEE 802.11 and IEEE 802.15.4, the first for communications with the pursuer robot, and the second for receiving messages from the WSN concerning intruder detection. The position is then calculated and displayed in a 3D virtual representation of the test-bed set, built with OPENGL.

The control station instructs the Pursuer robot where to go (Intruder location), and updates this information constantly (Figure 15). This information also includes the waypoints the robot needs to reach in order to take the short and most effective path to the Intruder location. This algorithm is static. Several areas are defined statically as well as the correspondent waypoints to follow in order to reach the destination area when it is not possible to move in a straight line Figure 30 shows the four waypoints used by the Pursuer in the test-bed layout.

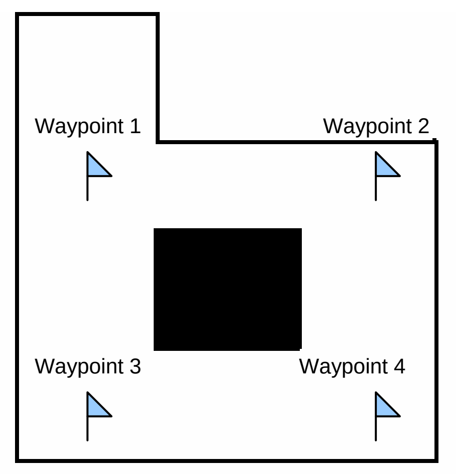


Figure 30 - Pursuer waypoints on the test-bed layout

The Control Station also displays assorted information related to the pursuer robot status: position, heading, mission status, waypoints. The diagram on Figure 31 depicts the internal architecture of the Control Station application.

The main thread is responsible for configuring the OPENGL and the GTK environments. It also launches the Communications Thread. This thread takes care of all communications to and from the Control Station. It listens for Status Messages from the Pursuer, Intruder Alert Messages from the WSN and images from the IP Camera on the Pursuer.

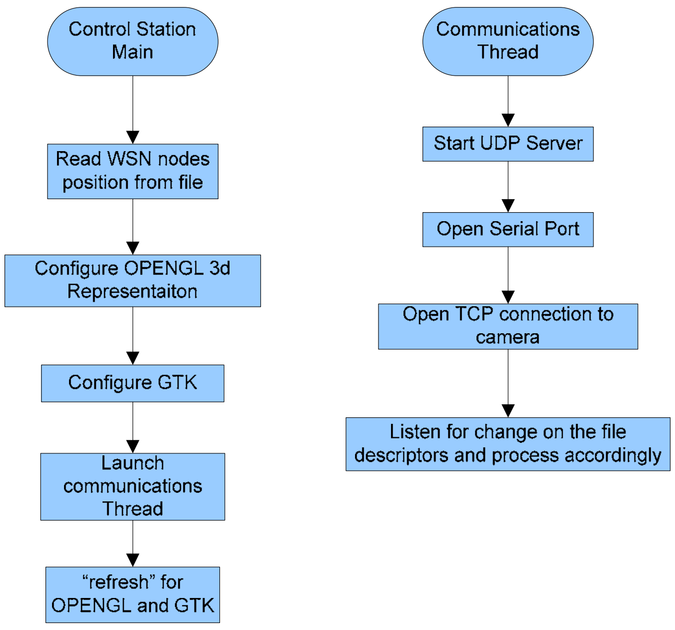
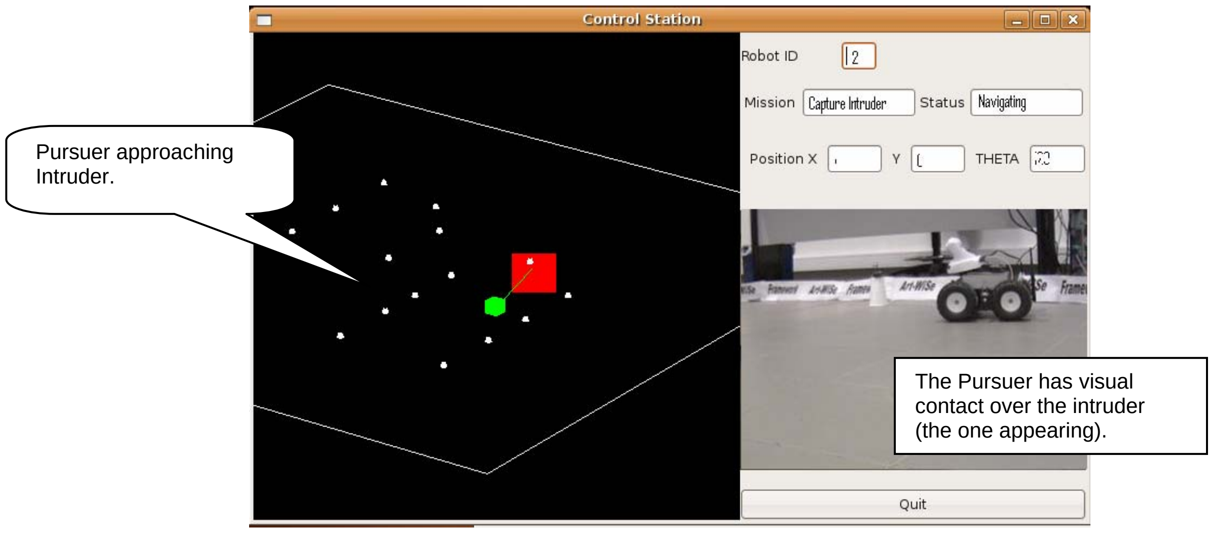

Control Station Main
Read WSN nodes position from file
Configure OPENGL 3d Representaiton
Configure GTK
Launch communications Thread
“refresh” for OPENGL and GTK
Communications Thread
Start UDP Server
Open Serial Port
Open TCP connection to camera
Listen for change on the file descriptors and process accordingly

Figure 31 - Diagram of the software on the Control Station

Figure 32 shows two screenshots of the GUI (Graphic User Interface) of the Control Station showing the Intruder capture. On the left hand side of the window (the black part), we can see the OPENGL representation of the WSN. The Pursuer is represented by a green cube and the position of the Intruder by a red square. The white cones represent the sensor nodes. In the right pane, some information about the Pursuer current status is displayed as well as the view from the IP camera of the Pursuer.

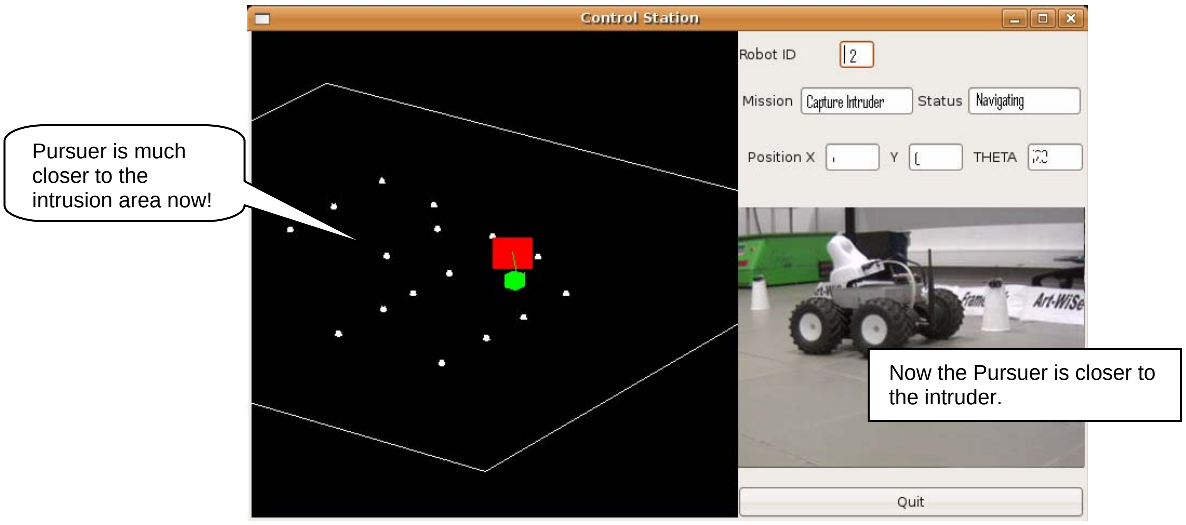

Control Station
Quit
Pursuer approaching Intruder.
The Pursuer has visual contact over the intruder (the one appearing).

Robot ID 2

Mission Capture Intruder Status Navigating

Position X [illegible] Y [illegible] THETA [illegible]

a)


Pursuer is much closer to the intrusion area now!
Now the Pursuer is closer to the intruder.
Quit

Control Station

Robot ID: 2

Mission: Capture Intruder

Status: Navigating

Position X

Y

THETA

b)

Figure 32 - The Control Station GUI screenshots

The previous screenshots enable the user to see what the Pursuer “sees” when chasing a target and the virtual representation of the test-bed scenario while the pursuit is running, all of this through the GUI in the Control Station.

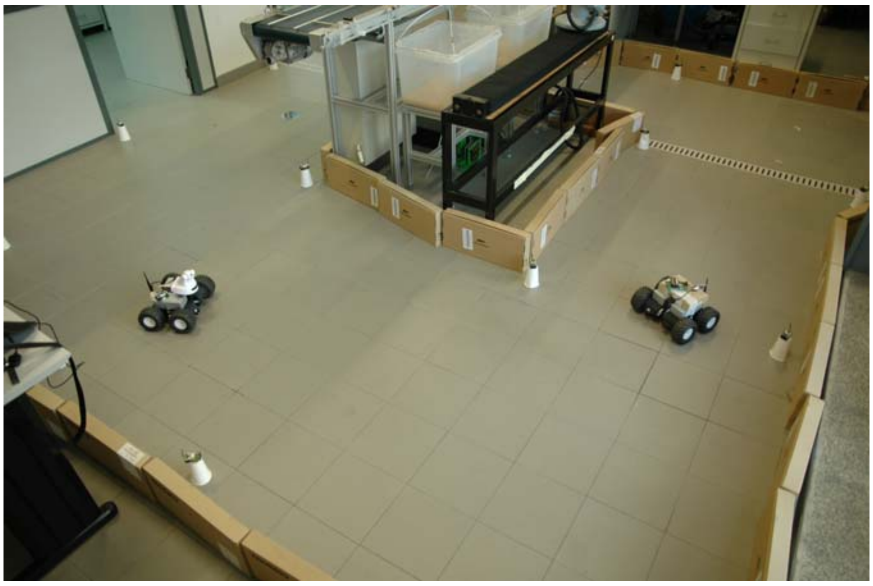


Figure 33 - Current test-bed deployment

Figure 33 presents a view of the current test-bed deployment showing a pursuit in progress.

Section 6 - Lessons learned and future work

This project was about starting up a test-bed application for the ART-WiSe framework – an architecture for real-time communications in WSNs.

In the scope of this work a vast amount of different technologies were embraced: robotics, wireless communications, wireless sensor networks and linux programming, are some examples. Additionally, the specification of the test-bed application was only achieved after a long analysis of other field trials in the WSN area and a lot of investigation of relevant and potential application domains. The development of the localization mechanism for the application also involved the analysis of different localization methods, while searching for the most adequate for this particular application, and a lot of testing while tuning the system.

The chosen application has proven to be an interesting way of demonstrating the ART-WiSe architecture in the medium-term. Nevertheless, some improvements regarding the application design were identified and eventually will be done later on. One of them would be to compute the localization of the intruder in the sensor network itself instead of doing this in the Control Station. This will eventually allow a faster identification of the intrusion area by increasing the performance of the application.

Still regarding the localization matter, despite the success of the implementation of the positioning system for the Pursuer robot, some improvements are also envisaged on this matter. We believe that by using an extended Kalman filter it will be possible to achieve better results on the positioning, particularly when fusing odometry information with the positioning system. It is also expected that a smaller granularity for the WSN can be achieved by obtaining a better behavior in an outdoor deployment since there will be almost no obstacles for message transmission by the positioning system, reducing the multi-path fading and shadowing effect on the range measurements.

The inclusion of the IEEE 802.15.4/ZigBee protocol stack under development in the WSN nodes is a major step in order to completely fulfill the test-bed objectives since it will allow the actual assessment of the ART-WiSe architecture. This assessment is also depending on the inclusion of access points capable of bridging the Tier 1 with the Tier 2, as well as the development of some kind of database for logging important performance data. As soon as the task of including the IEEE 802.15.4/ZigBee implementation becomes completed some other new improvements can be made to the test-bed application. For example, by adding a simple cluster-tree topology network to the implementation it will be possible to provide a more efficient mechanism to the localization of the intruders. It will also be possible to use routing tables and therefore to support multi-hop communication (which is one of the requirements for Tier 1).

Along this project, many new issues kept emerging. Some of them were related to the cooperation of wireless sensor networks with mobile entities like mobile robots. This application can be looked at as an example of how a WSN can take advantage of mobile robots, in this case by helping to secure the deployment area. Besides this obvious application, one can envisage other functions like the introduction of mobility to the network. For instance, some events may need a greater coverage by the network. Mobile robots could be used to grant that, if configured as access points (i.e. using the envisaged ART-WiSe architecture), by moving them to the desired region.

The employment of mobile robots can be particularly useful when targeting harsh environments or very large areas where it is not feasible to have people carrying on such tasks.

This WSN/mobile robots cooperation may also be a way of improving the limited capabilities of the robots. In this work, a simple method for positioning a mobile robot through RSSI measurements was proposed. This may be one of many services the sensor network can grant to a mobile robot. Another way of improving the mobile robot capabilities is for the robot to take advantage of the WSN sensing abilities. Despite no real sensors were used in the deployed network, it is possible to easily equip the nodes with sensors of different kind (e.g. temperature, noise, moisture). This will provide a higher coverage of the area reducing the amount of equipment we would eventually need to acquire in order to achieve the same results with a mobile robot. Herewith, multiple kinds of events can be triggered by the WSN for later inspection by the robots.

Like it was previously mentioned, the test-bed development effort is a work in progress and it will be carried on until all the envisaged objectives for the Art-WiSe architecture become accomplished and maybe in the future, as a support for other researches on WSNs.

References

[1] R. Szewczyk, A. Mainwaring, J. Polastre, and D. Culler. “An analysis of a large scale habitat monitoring application”. In Proceedings of the Second ACM conference on Embedded Networked Sensor Systems (SenSys), 2004.

[2] J. Burrell, T. Brooke, and R. Beckwith. “Vineyard computing: Sensor networks in agricultural production”. IEEE Pervasive computing, 3:38–45, 2003.

[3] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y-R. Choi, T. Herman, S. S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: “A wireless sensor network for target detection, classification, and tracking”. Computer Networks (Elsevier), 46(5):605–634, 2004.

[4] J. A. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou, "Real-Time Communication and Coordination in Embedded Sensor Networks," Proceedings of the IEEE, vol. 91, pp. 1002-1022, 2003.

[5] L. Zheng, A. Dadej, and S. Gordon, "Fairness of IEEE 802.11 Distributed Coordination Function for Multimedia Applications," in WITSP'03, Coolangatta (Australia), 2003.

[6] V. Bharghavan, "Performance Evaluation of Algorithms for Wireless Medium Access," in ICPDS'98, Durham, North Carolina (USA), 1998.

[7] IEEE 802.15.4 Standard Part 15.4: “Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)”, IEEE Standard for Information Technology, IEEE-SA Standards Board, 2003.

[8] Crossbow Technologies INC. http://www.xbow.com

[9] TinyOS http://www.tinyos.net

[10] David Gay, Phil Levis, Rob von Behren, Matt Welsh, Eric Brewer, and David Culler, “The nesC language: A holistic approach to network embedded systems”, in PLDI’03.

[11] Gay D., Levis P., Culler D, Brewer E., nesC 1.1 Language Reference Manual, May 2003

[12] Emmanuel LOMBA, André CUNHA, “MICAz motes simple data communications”, IPP-HURRAY! Technical Report, 2006

[13] WifiBot, http://www.robosoft.fr/wifibot.html

[14] Mesh Cube, 4G Systems GmbH
http://www.meshcube.org/index_e.html

[15] http://www.openembedded.org/

[16] N.B. Priyantha, A. Chakraborty and H. Balakrishnan, “The Cricket Location-Support System”, ACM Sigmobile (Mobicom), 2000.

[17] Konrad Lorincz and Matt Welsh, “MoteTrack: A Robust, Decentralized Approach to RF-Based Location Tracking”, in Proc. of the Int. Workshop on Location and Context-Awareness (LoCA 2005) at Pervasive 2005, May 2005.

[18] Andreas Savvides, Chih-Chieh Han, Mani B. Strivastava, “Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors”, in 7th annual international conference on Mobile computing and networking, Rome, Italy, pp. 166-179, 2001.

[19] J. Gibson, “The Mobile Communications Handbook”, IEEE Press 1999.

[20] Chipcon, SmartRF CC2420 Datasheet (rev 1.3), 2005. http://www.chipcon.com

[21] A. Savvides, H. Park, M.Srivastava, “The bits and flops of the N-hop multilateration primitive for node localization problems”, in First ACM International Workshop on Wireless Sensor Networks and Application (WSNA), Atlanta, GA, 2002, pp. 112-121.

[22] Koen Langendoen, Niels Reijers, “Distributed localization in wireless sensor networks: a quantitive comparison” in The International Journal of Computer and Telecommunications Networking, Special issue: Wireless sensor networks, pages 499--518, November 2003.

[23] MIB510 Interface Board
http://xbow.com/Products/productsdetails.aspx?sid=79

[24] Dimitrios Lymberopoulos, Quentin Lindsey, Andreas Savvides, “An Empirical Analysis of Radio Signal Strength Variability in IEEE 802.15.4 Networks using Monopole Antenas”, Yale University ENALAB TR 050501.

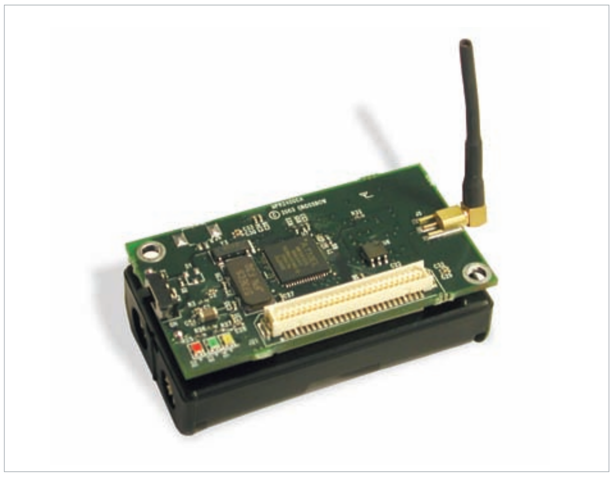
[25] Cory Sharp, Shawn Schaffert, Alec Woo, Naveen Sastry, Chris Karlof, Shankar Sastry, David Culler, “Design and Implementation of a sensor Network System for Vehicle Tracking and Autonomous Interception”, In Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN), 2005 UC-Berkeley.

Annex 1 – MICAz Datasheet

Crossbow

MICAz

WIRELESS MEASUREMENT SYSTEM



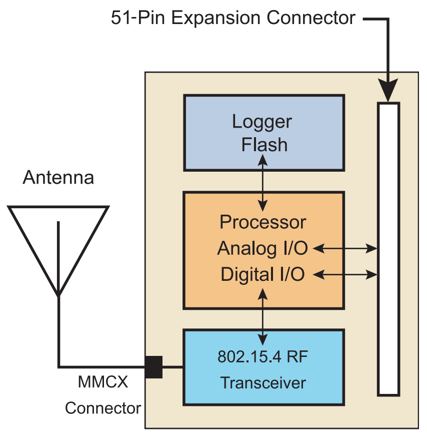
- IEEE 802.15.4, Tiny, Wireless Measurement System
- Designed Specifically for Deeply Embedded Sensor Networks
- 250 kbps, High Data Rate Radio
- Wireless Communications with Every Node as Router Capability
- Expansion Connector for Light, Temperature, RH, Barometric Pressure, Acceleration/Seismic, Acoustic, Magnetic and other Crossbow Sensor Boards

Applications

- Indoor Building Monitoring and Security
- Acoustic, Video, Vibration and Other High Speed Sensor Data
- Large Scale Sensor Networks (1000+ Points)
- ZigBee Compliant Systems and Sensors

ZigBee™ Alliance

A member of the ZigBee Alliance



MPR2400CA Block Diagram

MICAz

The MICAz is a 2.4 GHz, IEEE 802.15.4 compliant, Mote module used for enabling low-power, wireless, sensor networks. The MICAz Mote features several new capabilities that enhance the overall functionality of Crossbow's MICA family of wireless sensor networking products. These features include:

- IEEE 802.15.4/ZigBee compliant RF transceiver
- 2.4 to 2.4835 GHz, a globally compatible ISM band
- Direct sequence spread spectrum radio which is resistant to RF interference and provides inherent data security
- 250 kbps data rate
- Runs TinyOS 1.1.7 and higher, including Crossbow's reliable mesh networking stack software modules
- Plug and play with all of Crossbow's sensor boards, data acquisition boards, gateways, and software

TinyOS is a small, open-source, energy-efficient, software operating system developed by UC Berkeley which supports large scale, self-configuring sensor networks. The source code software development tools are publicly available at: http://webs.cs.berkeley.edu/tos

Processor & Radio Platform (MPR2400CA)

Using TinyOS, a single processor board can be configured to run your sensor application/processing and the mesh networking radio stack simultaneously. The MICAz (MPR2400CA) IEEE 802.15.4 radio offers both high speed (250 kbps) and hardware security (AES-128). The MICAz 51-pin expansion connector supports Analog Inputs, Digital I/O, I2C, SPI and UART interfaces. These interfaces make it easy to connect to a wide variety of external peripherals.

Sensor Boards

Crossbow offers a variety of sensor and data acquisition boards for the MICAz Mote. All of these boards connect to the MICAz via the standard 51-pin expansion connector. Custom sensor and data acquisition boards are also available. Please contact Crossbow for additional information.

Base Stations

A base station allows the aggregation of sensor network data onto a PC or other computer platform. Any MICAz Mote can function as a base station by

Document Part Number: 6020-0060-03 Rev A

Processor/Radio Board	MPR2400CA	Remarks
Processor Performance		
Program Flash Memory	128K bytes	
Measurement (Serial) Flash	512K bytes	> 100,000 Measurements
Configuration EEPROM	4K bytes	
Serial Communications	UART	0-3V transmission levels
Analog to Digital Converter	10 bit ADC	8 channel, 0-3V input
Other Interfaces	Digital I/O,I2C,SPI	
Current Draw	8 mA	Active mode
	< 15 µA	Sleep mode
RF Transceiver		
Frequency band[1]	2400 MHz to 2483.5 MHz	ISM band, programmable in 1 MHz steps
Transmit (TX) data rate	250 kbps	
RF power	-24 dBm to 0 dBm	
Receive Sensitivity	-90 dBm (min), -94 dBm (typ)	
Adjacent channel rejection	47 dB	+ 5 MHz channel spacing
	38 dB	- 5 MHz channel spacing
Outdoor Range	75 m to 100 m	1/2 wave dipole antenna, LOS
Indoor Range	20 m to 30 m	1/2 wave dipole antenna
Current Draw	19.7 mA	Receive mode
	11 mA	TX, -10 dBm
	14 mA	TX, -5 dBm
	17.4 mA	TX, 0 dBm
	20 µA	Idle mode, voltage regular on
	1 µA	Sleep mode, voltage regulator off
Electromechanical		
Battery	2X AA batteries	Attached pack
External Power	2.7 V - 3.3 V	Molex connector provided
User Interface	3 LEDs	Red, green and yellow
Size (in)	2.25 x 1.25 x 0.25	Excluding battery pack
(mm)	58 x 32 x 7	Excluding battery pack
Weight (oz)	0.7	Excluding batteries
(grams)	18	Excluding batteries
Expansion Connector	51-pin	All major I/O signals

Notes

[1]5 MHz steps for compliance with IEEE 802.15.4/D18-2003.
Specifications subject to change without notice

MIB600CA Mote Interface Board

MIB520CA Mote Interface Board

plugging the MPR2400CA Processor/Radio Board into an MIB510CA/ MIB520CA serial/USB interface board. The MIB510CA provides an RS-232 serial interface while the MIB520 provides a USB interface for both programming and data communications. Crossbow also offers a stand-alone gateway solution, the MIB600CA for TCP/IP-based Ethernet networks.

Ordering Information

Model	Description
MOTE-KIT2400CB	2.4 GHz MICAz Developer's Kit (8x MPR2400CA, 4x MTS310CA, 3x MTS300CA, 1x MDA300CA, 1x MIB600CA, 1x MIB510CA)
MPR2400CA	2.4 GHz Processor/Radio Board

Document Part Number: 6020-0060-03 Rev A

Annex 2 – WifiBot Datasheet

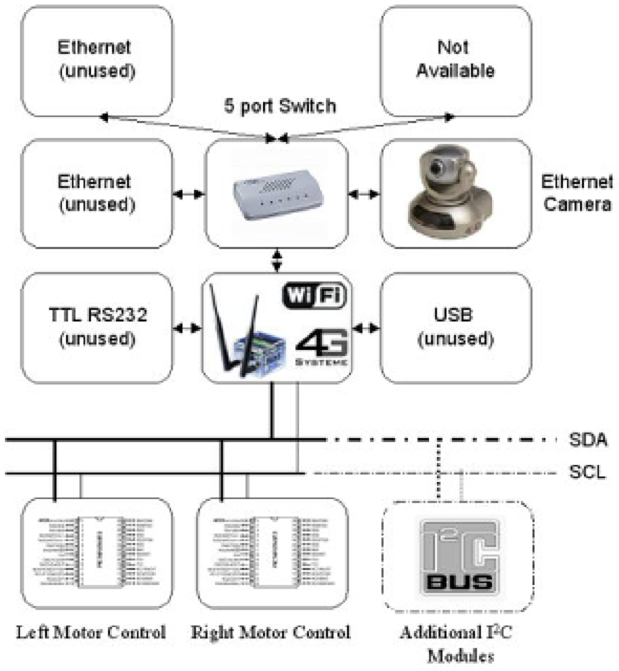
- *High mobility 4x4 platform*
- *Modular and open architecture*
- *Fully programmable under Linux*
- *Embedded LAN and mesh WI-FI networking*

WiFiBoT 4G

Multi-purpose robot

WiFiBoT 4G is a robot which is characterized above all by a great flexibility allowing it to be used in multiple environments and situations. Its mechanical design and its four wheel drive allow this robot to evolve over irregular surfaces or even small obstacles. Its small dimensions and its low weight make it easily transportable and perfect to explore narrow places.

As a system, WiFiBoT 4G is opened to all kind of uses and applications. The robot offers an entire world of expansion possibilities at different levels. It features an embedded 400MHz AMD calculator under Linux and a large choice of interfaces like embedded Ethernet, RS232, I²C, USB, as well as standard and mesh WiFi !



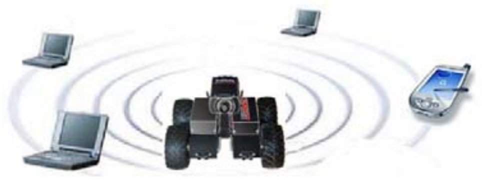
www.wifibot.com

Wi-fibot 4G

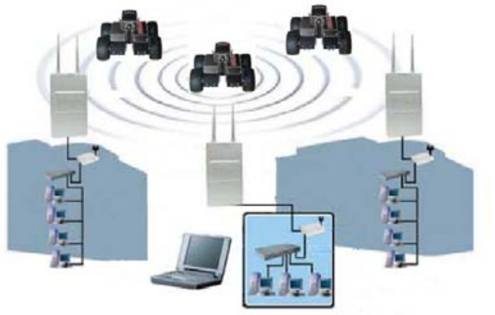
Included Features

Calculator:	MIPS CPU AMD Au1500 400 MHz RAM 64 MB Flash Storage 32 MB
Interfaces:	4x Ethernet 10/100 BaseT 1x USB (NC) 1x I²C bus 1x RS232 port (Debug)
WiFi:	WiFi 802.11b Seamless Roaming Mesh Network (OLSR) WLAN AP, Bridge, Client Bridge / Router 1x Antenna 5dBi
Sensors:	1x Pan-Tilt IP camera 2x IR range sensors 4x 300 CPR codewheel Battery level
Speed Control:	Independent PID for each motor
Motors:	4x motors 7.2V 50:1 8.87Kg/cm 120 rpm
Dimensions:	Length : 28 cm Width : 30 cm Height : 20 cm Weight : 4.5Kg
Batteries:	9.6V NiMh 9500 mAH 2h autonomy Easy replacement Charger included

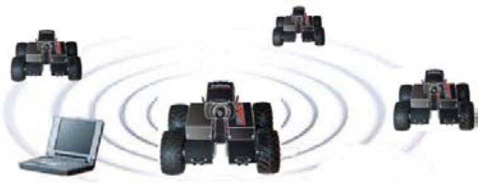
Mode of Operation	**4G Cube Configuration**
Stand Alone	AP or Ad-hoc Bridge
Infrastructure	Bridge or Mesh
Swarm	Ad-hoc Bridge or Mesh



Stand alone

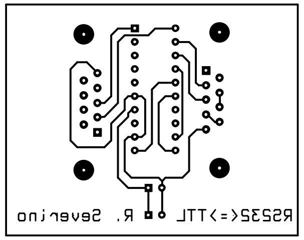


Infrastructure



Swarm

Annex 3 – RS232 to TTL Converter PCB layout



Title:		
Size: A3	Date: 2 nov 2006	Rev:
KiCad E.D.A. PCBNEW (2006-06-26)		Sheet: 1/1