Collaborative Project
Small-medium-scale focused research project (STREP)

Grant Agreement n. 224168 FP7-ICT-2007-2

WIDE

Decentralized and Wireless Control of Large-Scale Systems

Starting date: 01 September 2008 Duration: 3 years

Deliverable number	**D2.2**
Title	**Simulation tool for networked control systems**
Work package	WP2 - Wide-area WSN for distributed control (RTD)
Due date	M18
Actual submission date	26/01/2010
Lead contractor for this deliverable	UNISI
Author(s)	Davide Barcelli
With the help of	Cesare Carretti, Alberto Bemporad
Revision	v1.0

Dissemination Level

PU	Public
PP	Restricted to other programme participants (including the Commission Services)
RE	Restricted to a group specified by the consortium (including the Commission Services)
→ **CO**	Confidential, only for members of the consortium (including the Commission Services)

Executive summary

This report describes the new features introduced in TrueTime for simulation of wireless control loops, and a real-time data acquisition environment in MATLAB for wireless sensors.

Contents

1 Introduction

One of the main issues identified in the WIDE project is to correctly model and simulate the behaviour of the Wireless Sensor Network (WSN). In fact, all the reliability of a wireless control system is based on the reliability and on the performance of the WSN used. For this reason, it is of paramount importance to define the network technology to use in the WSN and to be able to simulate its as realistically as possible.

The main network technology used for WSNs are ZigBee [5] and WirelessHart [2]. They both are based on the PHY level of the standard IEEE802.15.4 [9], but they define different upper levels and with different protocols. We review both of them here below.

1.1 ZigBee

ZigBee is a specification for a cost-effective, low-rate and low-power wireless communication protocol for home automation, monitoring and control. It aims at providing short-range wireless networking which is scalable, self-organizing and secure, while providing battery life up to two years and more. Although it exists since late 2004, ZigBee still has to fully prove its success, at least in the industrial domain where reliability and security are uttermost important.

ZigBee (see [10]) is a specification for the higher protocol layer, and builds upon the physical (PHY) and medium-access control (MAC) layers in the 802.15.4 specification. Mesh networking topology is supported and routing is achieved through the ad-hoc on-demand distance vector (AODV) algorithm. This means that it is the devices themselves that are responsible for route discovery, and peer-to peer-communication is possible. In a ZigBee network, all nodes shares the same channel, and frequency agility is minimal. There is no frequency hopping, and the only option is to scan for a channel with the least amount of interference at start up. There are two classes of network devices in ZigBee; Full-Function devices (FFD) and Reduced-Function devies (RFD). The former can route messages in mesh networks and act as the network coordinator, whereas the latter can only communicate with one FFD in a star network setup. In simulations only FFD devices are used.

ZigBee can operate in both beaconed and non-beaconed mode. In the beaconed mode, the nodes are to some extent synchronized and the superframe is divided into 16 slots. The slots in the frame are generally contention-based, using CSMA/CA. There is an option to use up to seven of these as dedicated slots to specific nodes to increase determinism, so-called guaranteed slot time (GTS). However, support for this is not mandatory and use of this feature might break interoperability.

1.2 WirelessHart

WirelessHART is an open-standard wireless networking technology developed by HART Communication Foundation. The protocol utilizes a time synchronized, self-organizing, and self-healing mesh architecture. It is also designed to be simple, scalable, reliable, secure, and to support existing HART technology. The protocol currently supports operation in the 2.4 GHz ISM Band using IEEE 802.15.4 standard radios.

WirelessHart (see [10]) uses only the physical layer of the IEEE 802.15.4 standard, and specifies new Data-link (including MAC), Network, Transport, and Application layers. Differently from ZigBee, which uses also the MAC of the IEEE 802.15.4 standard with a unslotted or slotted CSMA/CA, WirelessHart is a Time Division Multiple Access (TDMA) based network. All devices are time synchronized and communicates in pre-scheduled fixed length time-slots. TDMA minimizes collisions and reduces the power consumption of the devices. WirelessHART uses several mechanisms in order to successfully coexist in the shared 2.4GHz ISM band: Frequency Hopping Spread Spectrum (FHSS) allows WirelessHART to hop across the 16 channels defined in the IEEE802.15.4 standard in order to avoid

interference. Clear Channel Assessment (CCA) is an optional feature that can be performed before transmitting a message, the transmit power level is configurable, and a mechanism to disallow the use of certain channels, called Blacklisting, is available. All of these features also ensures WirelessHART does not interfere with other co-existing wireless systems that have real-time constraints.

For message routing two different mechanism are provided: *Graph routing* and *Source routing*. Graph routing uses pre-determined paths to route a message from a source to a destination device. To use path redundancy, a graph route consists of several different paths between the source and destination devices. Source routing uses ad-hoc created coutes for the messages without providing any path diversity (used mainly in network diagnostics).

1.3 Simulation environment

To simulate the behavior of both ZigBee and WirelessHart networks, following the suggestions emerged by the WIDE first review meeting of October 2009, we have chosen the *TrueTime* [4] MATLAB/Simulink [3] environment. Other network simulator were taken into consideration in fall 2009, such as the *PiccSim* simulator, but the choice of TrueTime was made because of its reliability, its easy-to-use environment, and last but not least the precision of produced simulations.

The TrueTime environment already offers a ZigBee network system. A WirelessHart extension was made by two master students of UNISI in collaboration with ABB [7, 8]. The work done by UNISI in Taks 2.4 “Simulation support for networked control system design”, which is reported in this deliverable, was to take the existing work as a starting point, and to develop a simple interface to TrueTime to make really easy to generate simple (and less simple) simulations using ZigBee or WirelessHart. The nodes chosen in the WIDE project are made by *ESENZA* [1] and use a WirelessHart-like network system. For this reason, to have a simulation as much coherent as possible with the actual E-Senza nodes that will be used in the pilot test case, a lot of emphasis was put on the use of the WirelessHart network kernel in TrueTime.

2 TrueTime

2.1 Objectives

The TrueTime toolbox allows one to simulate a wireless-based closed-loop system defined by the user. The details that can be simulated include accounting for several facets of the wireless part (e.g., signal propagation, interference, noise, device programming, etc). At the same time, however, some quick startup features are needed to simplify the setup of a standard wireless simulation, therefore allowing the user to focus on control issues.

2.2 Extension motivations

The toolbox needs accuracy and simplicity of use at the same time. This is achieved by extending the existing TrueTime MATLAB toolbox, exploiting its capability of dealing with process scheduling issues, to include *automatic model and code generation* capabilities. Clearly, although dealing directly with TrueTime blocks and files permits to control in detail the very basic features one is simulating, a lot of programming effort is required in order to set every feature up. The added automatic generation capability asks the user to provide some very basic informations, such as the number of sensors and actuators, and generates all needed files (processes and inits) and a Simulink model with all nodes automatically. Then, the user can complete the model filling in the dynamics of the controlled process and the controller. In such a way, an accurate standard simulation environment is quickly set up, which is also a starting point for the user who wants to go deeper into wireless/scheduling issues.

2.3 Automatic model creation

Two MATLAB scripts are available to allow the user choosing the protocol to work with. This step is needed to tackle the large differences between the two protocols and mostly their implementation in TrueTime, that require different features of the toolbox.

2.3.1 Zigbee

The ZigBee implementation does not allow multi-hop and does not require slotting. Therefore, two nodes are introduced in the simulation diagram for each wireless measurement link: a transmitter node that acquires the measurement from the sensor and sends it to a receiver node that is connected to the controller (actuation links easily follow).

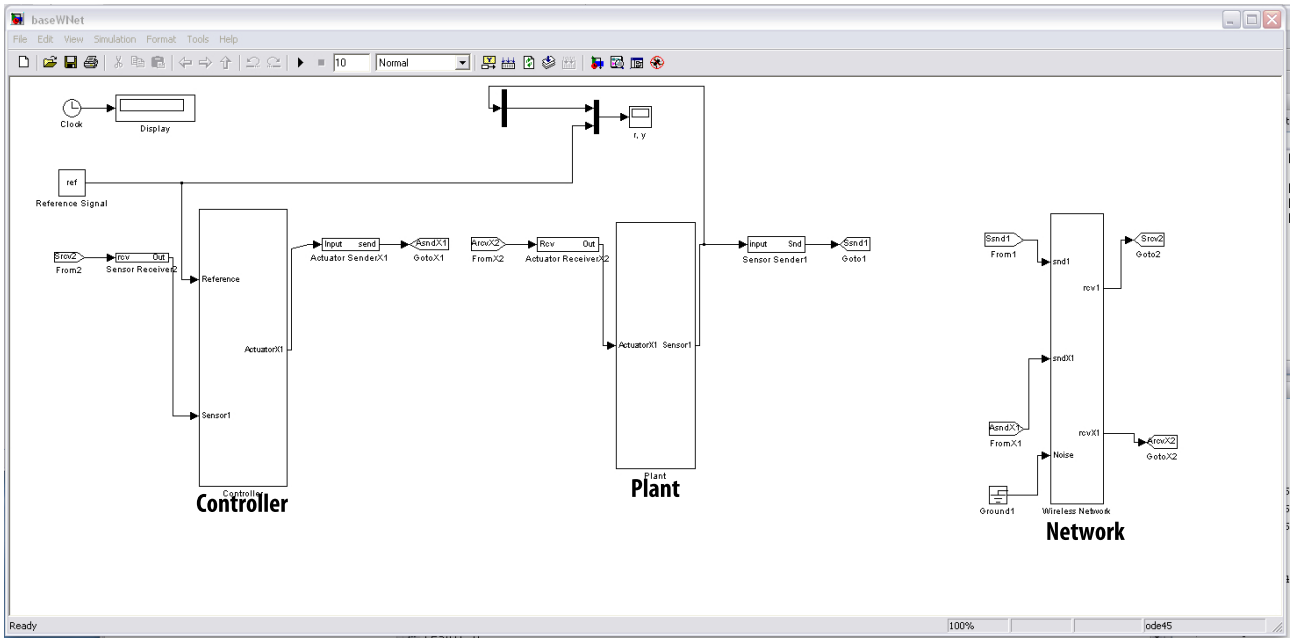

Figure 1: The basic scheme for ZigBee simulations, which is used by the automatic generation functions to create the user-defined final configuration

Figure 1 shows the basic general scheme of the TrueTime wireless network for control. The main blocks are the plant and the controller, that are completely free to be defined by the user and have a coherent number of inputs and outputs with the number of sensors and actuator in the wireless network. The network block is the wireless simulator core and permits the definition of the physical layer configuration, such as Data Rate and Transmit Power.

Figure 2 shows and example of automatically generated diagram with 4 sensors and 3 actuators. Note that the blocks are positioned in a way that the user can easily comprehend and modify them.

2.3.2 WirelessHART

The WirelessHART protocol is based on slotting and its implementation in TrueTime by UNISI students C. Snickars and M. De Blasi [7, 8] allows for static routing, permitting a predefined number of hops between nodes. Thus, this scheme needs to be extended, as the routing have to be decided and the time slots to be assigned, which implies that for each node in an indirect path one needs to specify exactly when to receive a packet and when to copy it. Such a system cannot be implemented in a way as simple as in the ZigBee case, being it very application dependent. Thus, the toolbox

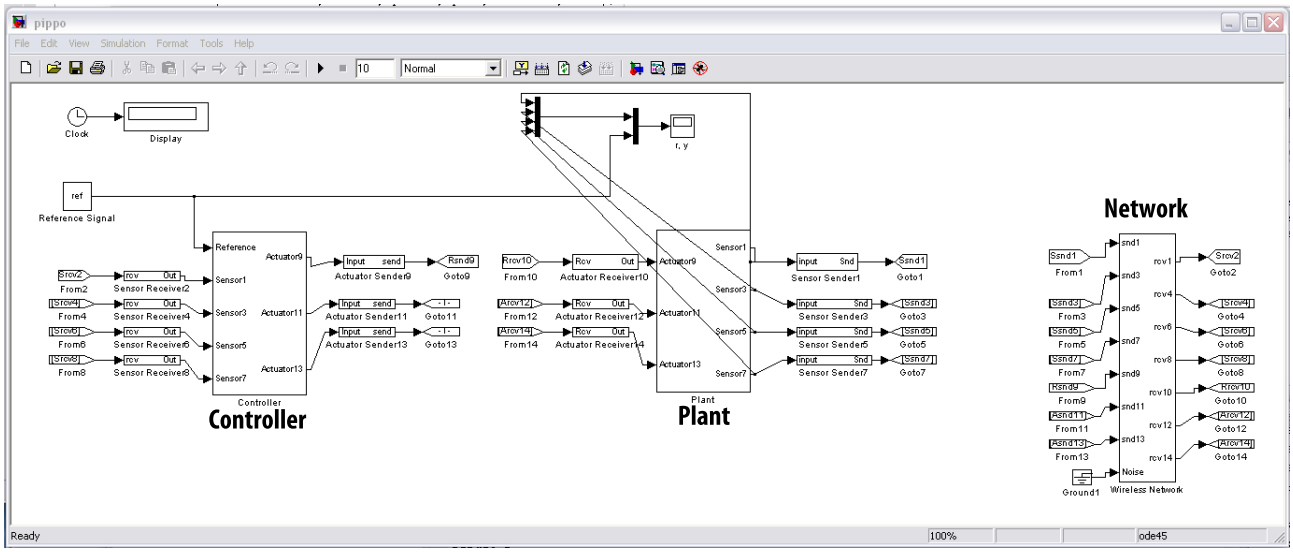

Figure 2: Example for ZigBee network of an automatically created configuration with 4 sensors and 3 actuators

provide a static scheme, that however is easily scalable, with a partial description of the architecture in a realistic framework. Its application in a subset of the railcar simulation benchmark [6] is currently under investigation. The considered subset consists of a heater and four temperature sensors in a T-like configuration. In the first instants each of the terminal sensors send their measurements to the central node, that acts like a gateway, and then the packet, enriched with the measurement of the central sensor, is copied to the heater node, which complete the feedback packet.

Finally the user can open the plant block (where there are as many inputs as actuators and outputs as sensors) and the controller block (where there are as many inputs as sensors and outputs as actuators), feeding in the additional Simulink block that complete the wireless control simulation model.

2.4 Automatic code generation

The generation script described earlier creates the Simulink model and all the required files to simulate it. TrueTime uses an initialization file to create the task to be executed during the simulation, and optionally the so called mailboxes for intra-node communication. The initialization file also specifies task priorities, offsets with respect to an (assumed ideal) common clock, and, in case of periodic tasks, the period. The code file specifies the functionalities to be realized in each segment (Simulink instants), in particular the sensor read / actuator set or packtet transmission / reception.

With regards to the example depicted in Figure 2, to have an idea of the complexity of the proposed simulation setup, more than 80 files are needed for simulation, to cover initialization tasks and other functions. This is due to the fact that every wireless connection is realized by using two wireless nodes, and each of them needs about 5 files. Therefore, it is apparent to the reader the importance of the automatic code generation feature introduced by WIDE, that really helps the user to save time.

For WirelessHART, nodes acting as a gateway are peculiar, in the sense that both the init and code file must exploit the knowledge of the routing table and slotting configuration in order to avoid collisions. The previous subsection describes the example routing table and slotting configuration, relative implementation is briefly discussed in the sequel to clarify inherit issues. Packets from terminal nodes have to be first communicated to the gateway, via a dedicated mailbox (to be defined in the gateway init file and implemented in the code one), and then retransmitted to the actuator, via an other mailbox. It follows that a dedicated mailbox is required for all hops of all packets, leading to a complex

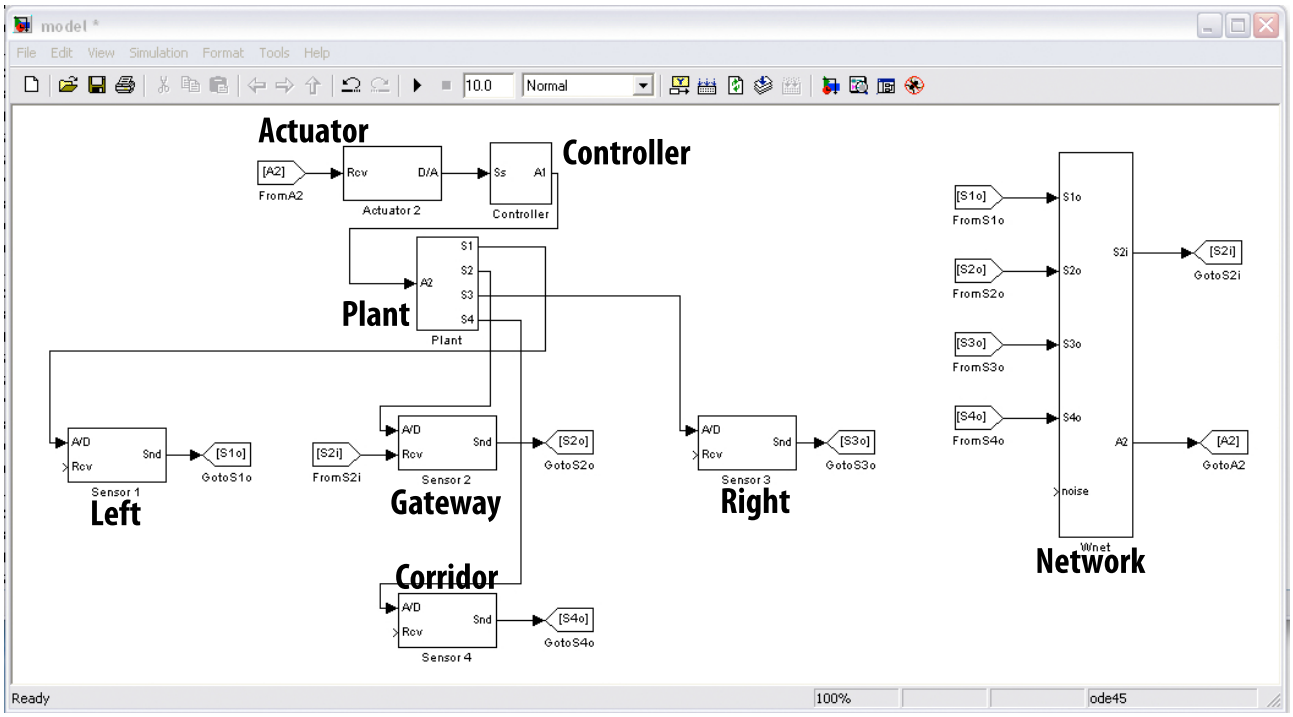

Figure 3: Base section of the temperature example proposed in [6]. The gateway collected data from corridor, left and right sensors and send to the actuator where the controller calculate and provide the action for the plant.

situation which is hard to obtain via automatic generation.

3 E-Senza wireless device acquisition

ESENZA is the wireless sensor provider partner in the WIDE project, thus an interface which can communicate from and to E-Senza nodes is required for using such devices within the MATLAB environment, for analysis, identification and possibly closed-loop control.

The basic network configuration consists of a gateway node, that is connected to a computer and powered via USB, whose goal is twofold: first, receive messages from other nodes and copy them to the USB port; second, copy messages from the computer to the remote devices. The wireless part of the pah is ruled by the proprietary E-Senza protocol, that is a WirelessHART-like, while the data transmitted via USB follows AT Command Protocol SenzaNet. Such scheme allows one to use the USB as a serial port for data I/O, in an asynchronous way, according the protocol specs.

The number of sensors is rather arbitrary, as the SenzaNet is can reconfigure itself and manage multi-hop routing table dynamically.

The E-Senza node I/O interface consists of analog and digital ports that can be set both as input or output. The analog configuration allows for voltage or current driven devices in 0-2 V or 0-10 V and 0-20 mA or 4-20 mA ranges respectively. Each node support up to 4 channels, with restriction on the possible configurations.

The AT+ Command protocol frame includes the command code and a command string in which the sending/receiving node ID is always specified. The remaining command string contains the time stamp and the number of channels, followed by each channel number and coded value. Digital channels are represented in bits in the standard way, while analog values are represented in 12 bits (the first byte and first half of second byte) with most significant bit, namely 16, represent the

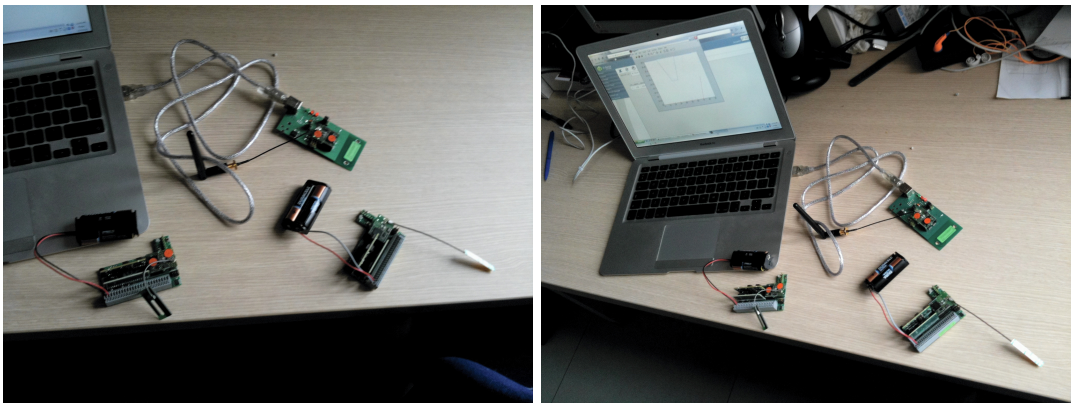

Figure 4: The E-Senza gateway node connected to a computer via USB and two battery powered E-Senza PT130 without enclosure.

sing. Following bits are 2 powers decreasing from 6 to -4, while negative values are expressed in complement to two.

For the twofold propose of setting up a ready to use environment for real-time experiments and allow real network experiments in the WIDE toolbox, a Matlab interface to E-Senza node was developed. The interface uses the Matlab Instruments Toolbox to achieve communication to a COM port, under Windows OS environment, and let the user specify for each node its E-Senza sensors/acutators configuration and ID. For driving actuators, t is easy to set up a controller that sends commands by a single line code. Together with the interface, a built-in monitor application is included for quickly plotting sensors measurements.

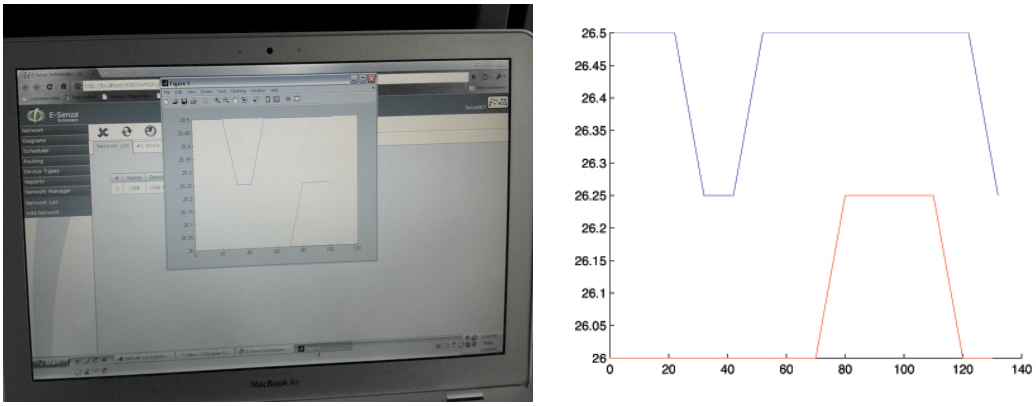

Figure 5: The acquisition procedure running and plotting the two temperature measurements acquired in real time

References

[1] E-senza technologies english home page. `http://www.e-senza.de/en/`.

[2] Hart communication fondation home page. `http://www.hartcomm.org/`.

[3] Matlab and simulink home page. `http://www.mathworks.com/`.

[4] Truetime simulator home page. `http://www.control.lth.se/truetime/`.

[5] Zigbee alliance home page. `http://www.zigbee.org/`.

[6] D. Barcelli and A. Bemporad. Decentralized model predictive control of dynamically-coupled linear systems: Tracking under packet loss. In *1st IFAC Workshop on Estimation and Control of Networked Systems*, pages 204–209, Venice, Italy, 2009.

[7] M. De Biasi, C. Snickars, K. Landernas, and A. Isaksson. Simulation of process control with wirelesshart networks subject to clock drift. In *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, pages 1355–1360, 28 2008-Aug. 1 2008.

[8] M. De Biasi, C. Snickars, K. Landernas, and A.J. Isaksson. Simulation of process control with wirelesshart networks subject to packet losses. In *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, pages 548–553, Aug. 2008.

[9] IEEE. *IEEE802.15.4 2006 Standard*. IEEE Computer Society, IEEE 3 Park Avenue New York, NY 10016-5997,USA, 2006 edition, september 2006 edition. *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*.

[10] T. Lennvall, S. Svensson, and F. Hekland. A comparison of wirelesshart and zigbee for industrial applications. In *Factory Communication Systems, 2008. WFCS 2008. IEEE International Workshop on*, pages 85–88, May 2008.