Sensor 9k: A testbed for designing and experimenting with WSN-based ambient intelligence applications

A. De Paola, S. Gaglio, G. Lo Re, M. Ortolani

In Pervasive and Mobile Computing, vol. 8, issue 3, 2012, pp. 448-466

SENSOR$_{9K}$: A Testbed for Designing and Experimenting with WSN-based Ambient Intelligence Applications

Alessandra De Paola, Salvatore Gaglio, Giuseppe Lo Re, Marco Ortolani

Department of Computer Engineering, University of Palermo,
Viale delle Scienze, ed. 6, 90128 Palermo, ITALY

Abstract

Ambient Intelligence systems are typically characterized by the use of pervasive equipment for monitoring and modifying the environment according to users' needs, and to globally defined constraints.

Our work describes the implementation of a testbed providing the hardware and software tools for the development and management of AmI applications based on wireless sensor and actuator networks, whose main goal is energy saving for global sustainability. A sample application is presented that addresses temperature control in a work environment, through a multi-objective fuzzy controller taking into account users' preference and energy consumption.

Keywords: Wireless Sensor and Actuator Networks, Ambient Intelligence, Probabilistic Reasoning, Energy Efficiency.

1. Introduction and Motivations

The main goal of Ambient Intelligence (AmI) is the development of systems aimed at adapting the surrounding environmental conditions so that they can match the users' needs, whether those are consciously expressed or not, while at the same time satisfying other system-driven goals, such as the minimization of global energy consumption. An implicit requirement is the use of pervasively deployed sensing and actuating devices, following the ubiquitous computing paradigm which states that technology must not

Email addresses: depaola@unipa.it (Alessandra De Paola), gaglio@unipa.it (Salvatore Gaglio), lore@unipa.it (Giuseppe Lo Re), ortolani@unipa.it (Marco Ortolani)

intrude into human lives; hence, control and monitoring devices should be deployed so as to remain invisible to the users [21, 46]. Wireless Sensor Networks (WSNs) fully meet these requirements, thanks to their intrinsic pervasiveness and low intrusiveness [4, 11, 16], and may thus represent a suitable choice for the sensory layer of AmI systems.

This work presents SENSOR$_{9K}$, a testbed for designing and experimenting with WSN-based Ambient Intelligence applications, deployed at our labs in the context of our research projects [20]. The name of the testbed is meant to emphasize its pervasiveness, as it ideally recalls the fictional *HAL 9000* AI system, whose extremities pervaded the spaceship in "2001: A Space Odyssey"; in particular, we intend to address the issue of implementing effective policies for energy saving in the context of indoor environments. Recent studies have shown that ICT technologies, and AmI systems in particular, may play a twofold role since their constituting elements are both significant consumers, and potential actors in steering a more clever overall usage of the available energy resources [25, 44]. The pervasive sensory infrastructure may be profitably used to gather information about the current energy usage, as well as the corresponding environmental conditions, into a centralized server, where artificial reasoning techniques may be implemented. Centralizing the reasoning activity preserves its consistency and unitarity [6], and allows to steer the behavior of the distributed actuators in order to bring the environment to the desired state.

In this context, our testbed aims to boost the development of AmI applications, and we argue that providing an abstraction towards the physical layer by means of a composition of core services will effectively let the AmI designer focus on higher-level issues; in this perspective, SENSOR$_{9K}$ provides a set of "building blocks" that implement basic intelligent functionalities on top of the underlying distributed sensory and actuating infrastructure.

SENSOR$_{9K}$ specifically focuses on indoor environments, where relevant environmental quantities will be monitored through pervasively deployed WSNs without adversely impacting the integrity of the pre-existing structures. Sensor nodes host the software implementing the logic of the application to be tested, as defined by the AmI designer, as well as additional testbed-specific functionalities.

The design of our testbed thus encompasses both hardware and software issues. Be-

sides the pervasive sensory devices, SENSOR$_{9K}$ provides a minimal set of communication and processing devices, organized into a backbone of local gateways providing access to the remote WSNs; such intermediate infrastructure is designed in a hierarchical fashion in order to accommodate scalability and fault tolerance, and its main purpose is to act as a connection interface bridging the gap between the distributed sensors and the centralized AmI server. SENSOR$_{9K}$'s core is thus represented by a middleware, partly distributed on the remote sensory devices and on the backbone gateways, and partly residing on the central AmI server, which stores the library of modules implementing basic AmI functionalities, such as user profiling, energy monitoring and multi-sensor data fusion, which we regard as the common ground for the creation of applications targeting energy saving.

In order to assess the validity of the proposed testbed, we will present a fully featured sample application, addressing temperature control in the context of a work environment, and involving conflicting goals, namely the satisfaction of the users' preferences in terms of pleasantness of the office environmental conditions, while minimizing the global energy consumption. It will be shown how SENSOR$_{9K}$ eases the development of such application; in particular a multi-objective fuzzy controller will be created by exploiting the basic SENSOR$_{9K}$'s functionalities.

The paper is organized as follows: Section 2 summarizes relevant works about software architectures for WSNs in the context of AmI. An overview of the architecture of our system is reported in Section 3, while Sections 4 and 5 contain the details about its physical and middleware layers, respectively. Finally, Section 6 describes our sample application, and Section 7 presents some experimental results.

2. Related Work

Since their introduction, Wireless Sensor Networks have steadily evolved, especially with respect to the degree of complexity of the network configuration, as summarized in [14]. The point of view has shifted from the use of one single WSN for the entire field, possibly composed by a very large number of nodes, towards a more structured approach involving several interconnected WSNs, each with a limited number of nodes, and up to a comprehensive strategy where the sensor nodes are enabled to interact with diverse

devices and applications. Such progress has consequently widened and diversified the range of issues that the different middleware systems for WSNs, presented in literature, attempted to address.

While initial efforts were mainly focused on the optimization of the resources available to the nodes, in terms of energy or computational power, later research has also addressed the functionalities for enabling interoperability among heterogeneous devices and for providing a common interface to different applications. A survey of middleware tools for WSNs must thus consider such variety in the approaches, despite the fact that they may not always be directly comparable.

A traditional categorization of WSN middleware softwares [24, 39], mostly focused on the first type of proposals of single WSN deployments, specifically distinguishes them with respect to the adopted programming model; the common goal is always the provision of an intermediate layer decoupling the node application logic from the underlying operating system and hardware. The authors of Impala [35], for instance, adopts a modular design paradigm in order to improve the applications adaptability, and to provide a simple way to keep them up-to-date; in the authors' vision, the possibility of adapting the application interface on the fly is bound to improve the performance, energy-efficiency, and reliability of the overall system. The modular approach has also led to the adoption of an agent-based programming model, as for instance in [22]; the latter work, in particular, employs mobile agents traversing several nodes, and carrying snippets of code with them; in this view, nodes are able, for instance, to host multiple applications at the same time. Another popular approach involves the use of virtual machines in order to provide the user with programming primitives in an assembly-like language, thus allowing developers to dynamically upload new "scripts" onto the network nodes; systems implementing this approach include Maté [34], and Magnet [9]. Alternatively, the entire WSN has also been viewed as a single distributed database, so that the goal becomes to transparently provide access to sensed data according to the traditional relational model; one of the most relevant works in this context is TinyDB [37].

A limitation of such classification is that it just focuses on middleware running entirely *inside* the network, thus disregarding middleware that instead runs, partly or entirely, *on top* of the sensor network level. The specific context of Ambient Intelligence, on the other

hand, has often stimulated researchers to exploit WSNs as a distributed sensory tool, and as a communication infrastructure, whereas the core of the intelligent services typically resides elsewhere, at a higher abstraction level. Our standpoint thus comprehensively considers that most of the AmI system is superimposed over the sensory equipment, and not entirely merged therein; the connection with the sensor nodes is provided by a minimal abstraction layer acting as a wrapper over the (user-defined) application logic, with the aim to hide possible future upgrades to the underlying hardware, as well as to provide a common interface towards the centralized Ambient Intelligence services.

A profitable comparison with other related approaches must thus analyze the functional aspects, besides the architectural paradigm; more specifically, a meaningful distinction for the classification of middleware in this perspective needs to consider the respective roles of the application logic at the sensor node level, and the higher-level application logic, that characterizes the behavior of the whole system, apart from the sensor network itself. While most middleware implementations discussed in literature are in fact an integral part of the sensor node software, the authors of [14] highlight the importance of both aspects, as functionalities implemented at the sensor node level may be used to provide novel services, whereas higher-level abstractions are useful to generalize the nodes' behavior so that it matches the upper layer requirements. GSN [1], for instance, uses some abstractions to communicate with the software running on the sensor network gateways, which consequently do not depend on the use of some custom software of their own, although of course software drivers specific for each application have to be implemented. An advance toward the extension of the middleware beyond the WSN layer is presented in [5], where the authors focus on multi-sensor data fusion in order to dynamically select the sensory services matching some application-specific Quality of Service (QoS) requirements.

Other works more explicitly separate intelligence from sensing; in [33], for instance, the authors propose systems for healthcare, especially targeted to the monitoring of chronic illness, and for assistance to the elderly. Both works employ WSNs as the supporting infrastructure for biometrical data collection toward a central server; in this view, sensor nodes are thus simply required to transfer data packets through multiple hops without any distributed processing intervention on them. WSNs are also at the

basis of the work presented in [3], although its goal is specifically the collection of information about user presence in the monitored premises; collected data are aggregated in order to compute predictions on the occupants' behavior.

Complete testbeds for experimenting with AmI applications in the context of smart environments are also reported in literature, as widely surveyed in [16, 18]; each of them proposes an ad-hoc approach to some specific scenario, although it is possible to abstract some common functionalities that may be considered as the basic necessary tools for building the overall intelligent behavior. The MavHome [17] has been designed following the agent-oriented approach, and is composed by a set of agents able to communicate through a hierarchical interconnection schema for control and information flow. The main research connected to this project concerns the prediction of users' activity exploiting past collected motion and lighting information, with the aim of performing an automating environmental control. The Aware Home [31] is a living laboratory for researching how ubiquitous computing can support everyday home life for elderly people. The project focus is devoted to sensing users' interaction with the surrounding environment. This project includes systems for human position tracking through various hardware, such as ultrasonic, video, and floor sensors.

Finally, the iDorm research [28] considers the scenario of a student bed-sitting room that provides the normal furniture arrangement found in a typical student study/bedroom environment, including bed, work desk and wardrobe, and allows the simulation of different activities like sleeping, working, and entertaining. The whole environment is purposely constructed in order to implement the designed testbed, given that a great number of wires and networked devices are hidden above the ceiling and behind the walls. The Gator Tech Smart House research [27] targets instead a single-family home, permeated by a wide set of sensors and actuators that can be automatically integrated through a generic middleware, that allows application programmers to assemble provided services in order to achieve their own goals, named programmable pervasive spaces. Both these works present flexible and expandable architectures thanks to the use of a middleware layer that gives a homogeneous representation of heterogeneous physical devices. However, in order to build an environment with the intended functionalities a preliminary design of the physical deployment must be carefully planned in advance, possibly requir-

Table 1 Comparison of the proposed testbed with other middleware for WSNs.

	SENSOR$_{9K}$	MidFusion	Impala	Agilla	Maté	Magnet	TinyDB	GSN
Support for an abstraction sensor language	Yes	No	No	No	No	No	No	Yes
Scalability	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Heterogeneity management	Yes	Yes	No	No	No	Yes	No	Yes
Energy awareness	Yes	No	Yes	No	Yes	Yes	Yes	No
Provides intelligent functionalities	Yes	Yes	No	No	No	No	No	No
Transparency to network topology changes	Yes	Yes	Yes	Yes	No	Yes	No	No
Sensor network management	No	Yes	Yes	No	No	No	Yes	Yes
Support for diverse applications	Yes	Yes	No	Yes	No	Yes	Yes	Yes
QoS awareness	No	Yes	No	No	No	No	No	No
Independence from knowledge about exact sensors	Yes	Yes	No	No	No	No	No	No
Support for information fusion	Yes	Yes	No	No	No	No	No	No
Dynamic sensor discovery	No	Yes	No	No	No	No	No	Yes
Service selection	No	Yes	No	No	No	No	No	No

ing heavy modifications to some pre-existent premises, because of the non negligible level of intrusiveness of some specialized devices, such as for instance the *smart floor*.

Unlike the previously mentioned frameworks, SENSOR$_{9K}$ explores a topic currently at the forefront of research, i.e. energy efficiency in the context of pervasive systems, with the aim to obtain a globally sustainable development. This goal is achieved by providing the developer with comprehensive support, ranging from physical level to application level issues, and covering all the necessary functionalities for enabling energy-aware AmI systems.

Table 1 qualitatively compares some of the previously mentioned middleware proposals with SENSOR$_{9K}$, in an analogous fashion as in [5].

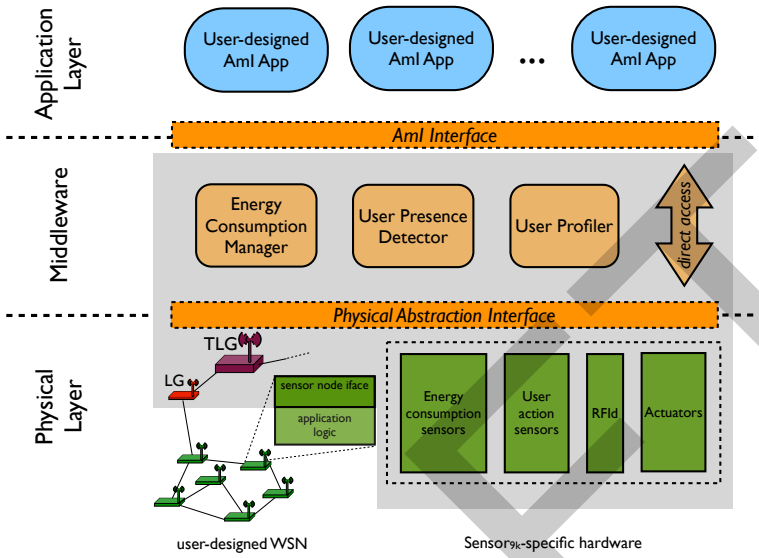


Figure 1 The logical 3-tier architecture.

3. The Testbed Architecture

SENSOR$_{9K}$ is aimed at simplifying the construction of complete AmI applications by providing basic hardware and software tools that can be composed and extended in order to build an intelligent comprehensive entity for controlling the environment.

The architecture is logically designed according to a 3-tier model, as depicted in Figure 1: the *physical* layer is composed by all the sensory and actuation devices, including those necessary to implement the testbed's functionalities, and possibly those required by the end user's application; the physical abstraction interface, in particular, will take care of exporting higher-level abstractions identifying the basic monitored units (e.g. each office room) besides dealing with basic connectivity issues among gateways, and will group together all the functionalities related to message relaying, monitoring and control of the testbed infrastructure health, and reconfiguration due to changes in the underlying physical infrastructure. On top of it, the *middleware* defines a toolset of basic AmI functionalities in the form of building blocks for implementing intelligent services over the available hardware; finally, the actual AmI applications created by the final developer, although strictly speaking not a part of the testbed, will be hosted at

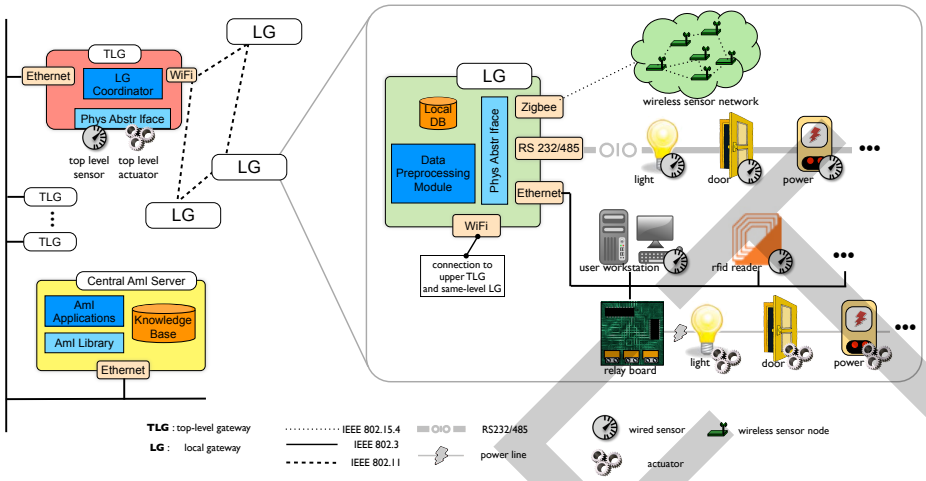


Figure 2 The hierarchical gateways system. The Central AmI Server represents the point of access for AmI applications; Top Level Gateways (TLGs) are connected to it, each managing a different environment, while fine-grained monitoring is guaranteed by Local Gateways (LGs).

the *application* layer. The shadowed part of Figure 1 shows the logical boundaries of SENSOR$_{9K}$'s own components, as opposed to the user-provided ones.

In the scenario of workplace monitoring, the basic monitored unit would be a single office room; however, the architecture of our testbed has been designed in order to be scalable with respect both to the number of monitored premises and to the potentially diverse employed technologies. According to this perspective, the logical architecture described has been designed as a partly centralized and partly distributed system; more specifically, high-level functionalities are implemented in a central AmI server, whereas the functionalities for managing low-level data gathering and command injection are pushed forward into the distributed nodes pervading the environment, as shown by the deployment diagram of Figure 2. In particular, the entire middleware is distributed over several components: part of it, namely the AmI modules and their interface with the applications, lies in the central AmI server, whereas most of the underlying services are provided by the remote gateways, and finally a tiny middleware layer is superimposed to the remote sensor nodes in order to have them respond to SENSOR$_{9K}$'s commands.

Each of the remote networks deployed into a specific room includes both wireless and

wired sensor nodes and actuators. SENSOR$_{9K}$ provides access to the room via a dedicated gateway node that implements the bridge between the physical devices, and the system itself. Such a component, named Local Gateway (LG) in the rightmost side of Figure 2, simplifies the connection among different network technologies and provides the higher layers with a homogenous representation of data originated by the heterogenous sensory technologies.

The remote LGs may be installed on lower-performance computing devices, such as microserver nodes connected to the wireless sensor and actuator network. Those devices are more powerful than low-end wireless nodes, and may provide temporary storage trough their local DBs and an increasingly refined preliminary processing for data, before forwarding them to the remote processing units.

LGs are connected to each other in order to create a communication backbone in the controlled premises; this LG Network is coordinated by a Top-Level Gateway (TLG) which plays the role of collector of the information coming from all of the LGs. The TLG also supplies the programming interfaces towards some devices, related to general sensing and actuating functions, not specific for a single premises. For example, the TLG provides the interface for the sensors and actuators used to perform the access control for a unique floor. If the system architecture needs to be implemented for an entire building, individual rooms will be managed by a dedicated LG; one TLG will take care of coordinating them and of dealing with floor-wide functionalities; finally, the TLGs for different floors will all report to the Central AmI Server which will possess a comprehensive view over the physical system.

AmI applications access the functionalities of our testbed only through the Central AmI Server; the intelligent software components of the system are not directly coupled to the hardware, thus making the development of ad-hoc applications both simpler and more generalizable. Figure 1 also shows the specific middleware modules provided by SENSOR$_{9K}$ for implementing basic functionalities, such as monitoring the energy consumption of remote appliances and actuators, analyzing sensory data to infer the user's presence in a given area, and finally analyzing the users' interaction with the actuators in order to build a profile of their preferences. The outcome of such modules will provide the common ground over which general higher-level AmI applications may be developed.

4. The Physical Layer

The main sensory infrastructure of SENSOR$_{9K}$ is represented by WSNs; moreover, our testbed has been designed in order to be easily customized with additional sensors, thanks to the adoption of a standard abstraction layer. Such layer has been designed to comply with a reduced version of the specifications provided by the OpenGIS sensor model language [12], which indicates models and arrangement rules for device interfaces in order to obtain the maximum degree of interoperability among different technologies. We loosely adhere to their specifications for data types and for describing sensor characteristics, and adapt them to our case by including actuators. In particular, SENSOR$_{9K}$ provides a library of low-level common functionalities, analogous in some aspects to the SWECommons of [12], in that it supplies basic software tools for data management, with additional functionalities for actively modifying the data gathering process in terms of sensing rate or precision. We also defined a protocol for the communications between the LG and the sensor/actuator nodes, as well as from the LG upward to the higher levels of the system hierarchy. The protocol allows for the injection of commands into the network and for sensory data retrieval. Typical commands include switching nodes on and off, triggering the sensing, varying the sensing rate, and even more radically changing nodes' behavior, e.g by having them aggregate data before transmitting them. Analogous commands exist for the actuators, ranging from simpler ones for on/off actuators, to more elaborate ones for the air conditioning, or light dimming systems. The interfaces of the LG towards the various connected elements rely on physically diverse media which may be wired or wireless, and the presence of the mentioned protocol allows the data management and communication software operation on board of the LG to be unaware of data formats and communication modalities specific to each of those technologies. Figure 3 shows a partial view of the DB schema contained in a LG, and showing the internal representation of the sensory and actuator devices relative to office rooms, according to the physical abstraction interfaces exported by each node.

We assume that the choice of the sensors and of the sensor nodes platform for environmental monitoring is determined by developer of the AmI application under testing; however, our testbed shall use supplementary sensors for specific testing purposes, so as to provide AmI applications with additional information in order to allow them to

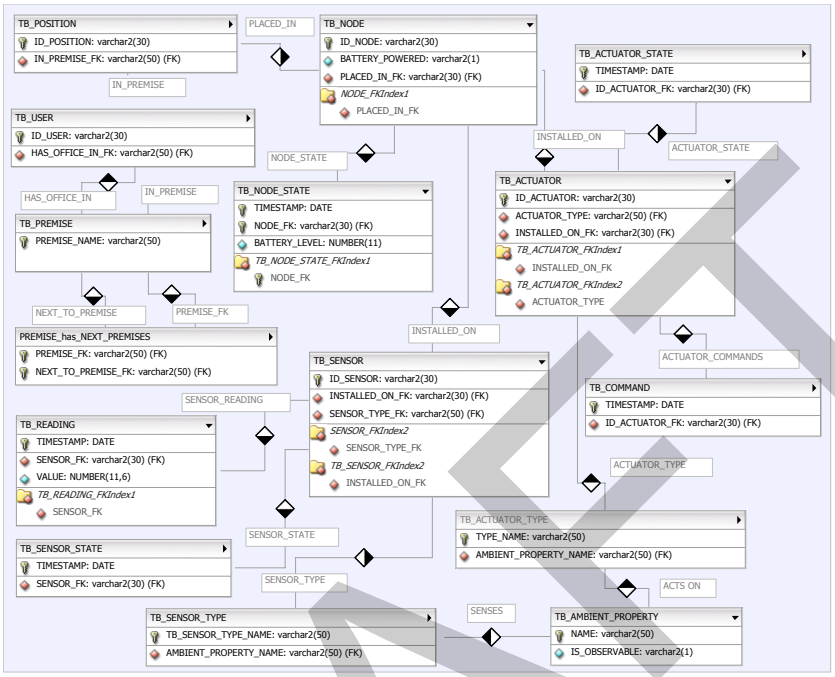


Figure 3 Part of the DB-schema showing the internal representation of the sensory and actuator devices for an indoor environment.

implement proper policies of energy saving. For this reason, SENSOR$_{9K}$ provides the tools for monitoring the electric energy usage of the building, for inferring the users' presence, and monitoring the interaction between users and actuators. Energy consumption may be monitored with varying resolution, and it is possible to observe the energy usage of entire premises, or specific devices. To monitor the global energy consumption of a specific room, we installed a multifunctional power analyzer; it was connected to the monophase line powering the room under observation and allowed us to collect information about voltage, current, and active and reactive power. Individual monitoring of specific devices is carried on by specialized "energy sensor nodes" that measure energy usage of any device connected to the power outlet of the sensor.

Besides mere measurement of instantaneous consumption, effective implementation of high-level policies for energy saving would also benefit from additional information. For instance, estimating the presence of specific users in the controlled areas might be interpreted as a trigger for the system to operate on the actuators only when actually

needed. Our approach is to avoid deploying additional dedicated sensors, and rather exploit pre-existing ones; however, we do not assume that any of them, separately considered, is able to provide a sufficiently precise estimate of the user's presence, so we choose to merge multiple sensory information, coming from diverse sensors, through a data fusion process.

Monitoring users' access to the premises managed by the AmI system may provide rough indications on the presence of users in specific areas, and a suitable technology for implementing simple access control is represented by RFIds [42]. In the proposed testbed, the RFId readers have been coupled with sensor nodes installed close to the main entrance and to each office door, while RFId tags have been embedded into ID badges for the department personnel, so as to completely replace traditional keys.

Also the sensory infrastructure deployed for the AmI application purposes may be exploited beyond its "natural" purpose; in particular, the WSN may cooperate to the estimate of the user's presence in a given room; namely we extend the system with additional sensor nodes carried by the users, and we use their interaction with the deployed WSN infrastructure to provide naive localization. Portable nodes are not equipped with any specific sensor, as they are only used to communicate with fixed nodes and to estimate their relative distances to them. The basic idea consists in estimating the distance between a mobile node, carried by the user, and the other environmental nodes placed at know locations, that may act as "beacons"; a simple trilateration algorithm would theoretically be sufficient to provide an approximation of the position of the mobile node. This idea is in fact not new in the field of WSN research [32, 43], and one of the easiest way to provide a rough estimate of distance is through RSSI measurements; however early research has already pointed out that this kind of signal is very noisy and that the relative measurement is highly error-prone especially because of the unpredictable signal attenuation model in unknown environments. Several works have thus been presented aimed at refining the distance estimate, either by using different, more reliable measures altogether, or via more advanced postprocessing [2, 45]. Our approach is fundamentally different in that we do not intend to provide precise localization of the mobile nodes, but we limit our system to determine the closest beacon, or in other words the macro-area through which the mobile node is currently moving. Whenever the system needs to find

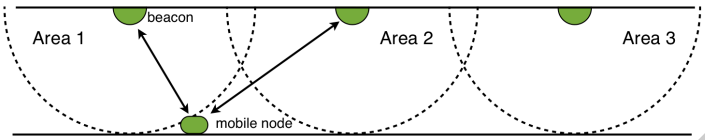


Figure 4 Example of coarse-grained localization: the mobile node is assigned to "Area1" by estimating the closest beacon.

out the position of one of the mobile nodes, it injects a query into the part of the WSN that forms the localization subsystem; beacon nodes collect the RSSI signals and send those measurements back to the querying unit that may easily identify which beacon is closer to the mobile node; the interaction is depicted in Figure 4. This may be regarded as a coarse-grained localization, and its precision of course depends on the number and density of beacon nodes; in our context, though, we do not need to refine the estimate beyond a certain threshold as this piece of information will be used in conjunction to other ones, and will only be an additional uncertain input for the upper-level decision system.

In order to observe the interactions of the users with the actuators, and thus to provide the system with some indication about the users' preferences, SENSOR$_{9K}$ includes a set of ad-hoc sensors. The simpler actuators provided by our testbed are remotely-controllable power relays, with the additional capability of providing information about their current state (e.g. the artificial lighting relay controller). More complex actuators are the domestic appliances controllable via IR remotes that may be typically found in homes or offices; within this category, we can mention air conditioning units, or automatic motorized electric blinds and curtains. Our hardware infrastructure thus includes the possibility of capturing the interactions between the users and such remotes, with the aim of providing the system with some indication about the users' preferences. An ad-hoc sensor node was designed to this end, and equipped with an IR receiver through a suitable expansion board. We positioned one such node close to each appliance under observation, with the sensor next to its IR receiver. Each setting sent via the remote to the IR receiver is then captured by the sensor node, and sent to the LG along with other sensed data. No modification is thus required to existing devices, which makes the system highly adaptable to different scenarios, and suitable for capturing user-defined

configurations of generic IR-based devices.

In order to develop the sensor node software for decoding the received IR information, a preliminary decoding of the pulse sequence sent by the IR remote needs to be performed. In our case, this has been accomplished by means of the utility library offered by the LIRC (Linux Infrared Remote Control) software package [10], which allows to directly measure the duration of pulses got out from the IR receiver. The validation of these measurements has been achieved by comparing them with those obtained through a digital oscilloscope connected with the same IR received.

SENSOR$_{9K}$ is also meant to reproduce any possible user action, so it provides an IR remote control connected to the LG of each room. This device may be programmed to allow the system to remotely control most of the commonly available domestic appliances, as it just reproduces the same interface as traditional remote controls. The user action sensors, coupled with the USB remote control, contribute to the creation of an unintrusive and flexible testbed, easily deployable in most kinds of environments. The proposed solution also aims to keep the realization costs low since it is does not require to modify the devices to be controlled, or to buy specific versions of them; also it already solves the issue of collecting the relative measurements, that are just regarded as additional sensory inputs, and as such managed by the WSN infrastructure.

5. Middleware

This Section presents the middleware providing the core functionalities for the design of specialized modules on top of the hardware substrate.

From a logical standpoint, the main purpose of the middleware is to decouple the applications from a specific choice for the underlying hardware, so that the developer may focus on the issues concerning AmI aspects specifically. To this end, our testbed provides the modules implementing basic AmI functionalities that may be combined to form complete applications; moreover we also allow the possibility to directly access the lower level functionalities of the sensory and actuation equipment, although we still interpose an abstraction layer hiding the irrelevant details and providing a homogeneous view on the hardware.

In the following we provide some insight into the upper part of the middleware,

by describing some of the specialized modules, currently available in our testbed, for monitoring energy consumption, interacting with the actuators, detecting the presence of users, and profiling their preferences.

5.1. Energy Consumption Management

In order to monitor energy consumption, our testbed has been equipped with specific sensors, as described in Section 4. However, application developers might not be interested in detailed data about individual energy consumption of each device, but they might rather prefer to obtain only higher-level information. To comply with this need, our middleware includes a specialized module for governing energy consumption monitoring, continuous sensing, and for triggering notifications to the application layer only if some predefined thresholds are exceeded. AmI applications will be able to tune the behavior of such software component by acting on its parameters, by means of specific control messages. Besides implementing such basic functionalities, SENSOR$_{9K}$'s middleware also provides support to planning in the context of energy saving by computing predictive models for energy consumption trends for selected devices. We assume that each monitored device is connected to one of the wireless nodes that gather data about energy consumption. The model will be synthetically represented by a table whose entries are pairs of the form $[actuator\_state, consumption]$; the energy consumption associated with each state of the actuator is estimated incrementally via an exponential moving average:

$$c(s) \leftarrow \alpha \cdot \bar{c} + (1 - \alpha) \cdot c(s),$$

where $c(s)$ represents the estimate of the energy consumption of the actuator with respect to its current state s, $\bar{c}$ is the latest reading of energy consumption, and α is a coefficient in the range $[0, 1]$. The cost model may then be queried by AmI applications in order to get information preliminary to planning.

Furthermore, the energy consumption module also includes the definition of the constraints to be taken into account by AmI applications in the planning phase. Although not currently available, it is arguable that in the near future energy providers will be able to supply information about the current contractual offer, scheduled shortages and low-fare hours. We are considering here the possibility of dealing with a limited set of contractual options, so that the module controlling our intelligent energy meter may con-

nect with the energy provider, gather information about possible constraints on monthly average consumptions and, by keeping track of past consumptions, provide an optimal estimate that meets the provided constraints.

Finally, a visionary but realistic scenario could include distributed energy production through the so called "smart grids" [15], whose main goal is to be autonomous by producing energy locally, via the exploitation of renewable energy sources. Such technology allows to feed possible overproduction of energy back into the distributed network in order to satisfy the demand coming from other network areas. In this context, the smart building managed by the AmI system represents a node of the electrical distribution network, and it is crucial to tune its energy consumption with respect to the amount of locally available energy, and to its cost.

5.2. Multi-sensor Data Fusion for Detecting Users at Work

SENSOR$_{9K}$ offers various types of sensors capable of perceiving physical and environmental characteristics useful in order to detect users' presence. However, none of these sensors is sufficient, alone, for performing this kind of elaboration, because it is characterized from an excessive uncertainty or because it does not succeed in monitoring all the premises of interest.

In order to overcome these limitations and to make the most of the available sensory information, SENSOR$_{9K}$'s middleware includes a module that performs the fusion of data coming from multiple sensors. One of the sensors used by this module is a virtual sensor that performs a naïve localization using the information obtained through the few RFId sensors used for users' access control. Several works in literature ([26, 40]) aim to perfect the localization process through the exploitation of a dense RFId grid. However in our work one of the most relevant goals is to maintain low costs and a low intrusiveness degree, so the proposed approach is to exploit several sensory information, each of which suffers from a non negligible uncertainty degree, but is obtainable via inexpensive processes.

Our virtual RFId-localization sensor employes Gaussian filters on the gathered data, according to the pseudocode reported in Figure 5. A badge reading gives timely information about the current presence of the user in the room where the specific reader is placed. The underlying idea is that the *belief* about the presence of the user in a specific location may be represented by a normal probability distribution with mean μ, and

```
— Localization through RFId readings —
Parameters: userID, targetRoom;
Initialization:
1: Graph ← graph(area topology);
2: μ ← [x_ID, y_ID]^T;
3: Σ ← Σ_max;
4: init R;
Main Loop:
5: loop
6:     if new reading for userID in [x, y]^T then
7:         μ ← [x, y]^T;
8:         Σ ← Σ_min;
9:     else Σ ← Σ + R;
10:    end if
11:    propagateBelief(Graph, μ, Σ);          ▷ topology-aware belief propagation
12:    notify targetRoom.belief;              ▷ notify belief update to upper-layer modules
13: end loop
```

Figure 5 Pseudocode for the virtual sensor feeding the Bayesian network, to be run on a TLG.

covariance Σ, as expressed by the following equation:

$$p(x) = det(2\pi\Sigma)^{-\frac{1}{2}} exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right\}. \quad (1)$$

A suitable tool for managing linear Gaussian systems is the Kalman filter [30], which represents the belief function through its moments; we modify the classical approach by specializing the measurement update step, and using a simplified transition state function. The RFId reading produces a precise information about the presence of the user in the same area where the badge is read, so the measurement update step of the Kalman filter may be simply formulated by centering a Gaussian in the reader location, and assigning a minimum variance to it (lines 7–8). Since we cannot predict the direction of the movements of the user, the control vector of the Kalman equation is assumed to be null, thus resulting in the following equations for the measurement update step:

$$\mu_t = \mu_{t-1}, \quad (2)$$

$$\Sigma_t = \Sigma_{t-1} + R; \quad (3)$$

where R represents the covariance of the Gaussian noise signal, with zero mean, assumed to affect the state transition (line 9). If no further readings occur for the same user, the

— **Topology-aware belief propagation** —

Parameters: *Graph*, μ, Σ;
Initialization:

```
updateList ← ∅                                   ▷ list of nodes whose belief is to be updated
for all n in Graph do
    n.updated ← false;
end for
root ← Graph.getNode(μ);                         ▷ the location of the root node is the center of the belief distribution
root.belief ← gaussianValue(μ, Σ, 0);            ▷ evaluate the Gaussian at its center
root.dist ← 0;                                   ▷ physical distance from root node along the graph
root.updated ← true;
neighborList ← neighbors(Graph, root);
for all neighbor in neighborList do
    neighbor.parent ← root;
    updateList.append(neighbor);                 ▷ append the neighbor node to the update list
end for
```

Belief Propagation Loop:

```
while updateList ≠ ∅ do
    n ← updateList.pop();
    parentDist ← distance(n, n.parent);          ▷ Euclidean distance between the current node and its parent
    n.dist ← parentDist + n.parent.dist;         ▷ physical distance of node n from root node along the graph
    n.belief ← gaussianValue(μ, Σ, n.dist);      ▷ evaluate the Gaussian at distance n.dist from center
    n.updated ← true;
    neighborList ← neighbors(Graph, n);
    for all neighbor in neighborList do
        if neighbor.updated is false then
            neighbor.parent ← n;
            updateList.append(neighbor);
        end if
    end for
end while
```

Figure 6 Auxiliary pseudocode of the belief propagation function.

uncertainty on the position estimate, represented by the Gaussian function centered on the location corresponding to the latest reading, increases with time; after some delay, the latest sensory reading does not provide relevant information any longer, so the user might be located anywhere in the building with the same probability.

The software implementing this module is fully aware of the complete topology of the

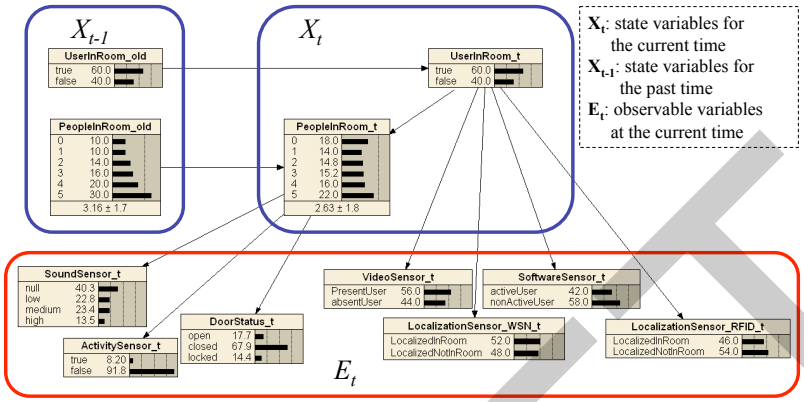


Figure 7 Markov chain for room occupancy evaluation.

building, represented as the graph of the connections between the building areas, in order to propagate the belief about the user's presence. A detailed pseudocode implementing this function is reported in Figure 6; it is worth pointing out that the belief value at a specific location does not depend on the *actual* Euclidean distance from the center of the distribution, but rather on the physical length of the complete path along the topology graph (lines 16–18).

The sensory information achieved through this module is represented by the `LocalizationSensor_RFID` block in Figure 7.

Other sources of information for the multi-sensor data fusion module are:

- `LocalizationSensor_WSN`: the WSN-based subsystem for localization, as previously described;
- `ActivitySensor`: a virtual sensor installed in the LG with the job of detecting the interactions among the customer and the actuators and of merging these information in order to obtain the level of user's activity;
- `SoundSensor`: the sensors able to detect the average level of sound in the room;
- `DoorStatus`: a sensor for detecting the state of the office door (open / closed / locked);
- `SoftwareSensor`: a software sensor for detecting user's activity at his workstation.

The last monitoring device detects the user's logins and logouts at their workstation, thus identifying time intervals when they are not logged in. When the user is performing some activities at the workstation using the peripheral devices, such as mouse and keyboard, the sensor software remains deactivated, and simply acts as a screen-saver; in this case the information on the user's activity is provided by the lack of data. If no activity is detected for a given time interval, it is possible that either the user is no longer sitting at their desk or that they are engaged in some activity that does not requires the use of the workstation; in order to discriminate between the two events, the software sensor activates a visual recognition process through which it tries to recognize the user's face in the images acquired through a webcam, through the approach described in [7].

Since the fusion of these sensory data inevitably involves reasoning with uncertainty, it was decided to consider a model based on probabilistic Bayesian networks as opposed to logical inference engines. Indeed, rule-based expert systems are not suitable for dealing with environmental features characterized by a large uncertainty, as the set of logical rules constituting them is exclusively deterministic; our domain, on the other hand, requires the integration of intrinsically noisy sensory information that, moreover, can only provide partial observations of the system state. Classical Bayesian networks [41], however, may only provide a static model for the environment, which would not be suitable for the proposed scenario; therefore dynamic Bayesian networks were chosen or, more specifically, Markov chains to implement our models which thus allow for probabilistic reasoning on dynamic scenarios, where the estimate of the current system state depends not only on the instantaneous observations, but also on past states. The validity of a Markovian approach for detecting users' presence and behavior by exploiting pervasive sensory information is confirmed by several works presented in AmI literature [8].

Figure 7 shows the Bayesian network designed for detecting users' presence, and clearly shows that sensory information are the only measurable manifestation of the system hidden state. State is here represented by the presence of the considered user in their own office room (associated to the `UserInRoom` variable), and the number of people in in the same room (`PeopleInRoom`). The state is observable through the previously described sensory information described. Variables modeling this sensory information are connected with state variables through sensor probabilistic models, expressed by

conditional probability tables that were learned from an opportune training data set.

The approach based on Bayesian networks allows to connect many state variables through cause effect relationships, thus making simpler the structure of the network. In our case the fact that the customer is present in his office influences indirectly the number of people present in the same room. Therefore sensory signals that are considered directly affected by the feature `PeopleInRoom` are indirectly biased by the feature `UserInRoom`.

In the network model each sensory signal is associated to the state variable from which it is mostly influenced, according to opportune indices of sensors utility [36].

5.3. User Preferences Profiling

SENSOR$_{9K}$ may perceive the actions performed by users via specifically designed sensors. The information obtained may be interpreted as *implicit* user feedbacks in order to learn their requirements; lacking any other kind of explicit feedback, the system resorts to analyzing the actions carried on by users, in order to extract implicit knowledge; for instance, the system is able to detect that a temperature decrease was requested and may use this piece of information to infer that the current environmental conditions are not satisfactory for the user; hence, a plan may be formulated on how to modify the system goals in order to better fit the intelligent environment inhabitants' demands. The availability of these implicit feedbacks, obtainable by means of these special non-intrusive sensors that monitor user's actions, is one of the innovative aspects of the proposed testbed.

The middleware provided with the testbed includes a module that collects the detected feedbacks and produces a model for the user's behavior. User-environment interaction modeling has been extensively studied in the field of Human-Computer Interaction; one of the most relevant issues in formalizing a method for user profiling is the creation of a unique profile for each user valid for all AmI applications possibly running on the testbed [23]. Although such choice may appear as the most elegant, it also raises some issues; firstly, creating a unique user profile for all applications involves processing a wealth of data, originated from many diverse interactions of the users with the actuators; moreover, as the user profile is communicated to the applications, they risk to be overwhelmed by possibly irrelevant data about events they are not interested about.

We therefore opted for the creation of a user profile generation model that were general

but at the same time parameterizable by the applications depending on which aspects of user-environment interactions they need to focus on. The users' profiles are built by regarding them as room occupant, and considering their interaction with the actuators. In particular, the software module may learn the utility value perceived by the user for each $[state, action]$ pair, by observing the environmental conditions and the interactions of the user with the actuators. The utility value is computed via an on-line mechanism based on a phase of action evaluation typical of reinforcement learning, so that the user profile may be built incrementally. Information related to implicit feedbacks is clearly filtered taking into account data about the user's presence obtained by the middleware modules devoted to this task.

Basically, whenever a user interacts with an actuator and changes its state, the correspondence between the current environment state and the current setting for the actuator is bound to a negative utility value; the greater the gap between the current setting and the user-imposed value, the greater the absolute value for the utility. When the user is present in the monitored area and they do not interact with the actuators, the correspondence between the environment state and the actuator setting gets a positive utility value. In order to automate this process, time is divided into slots and each of them is regarded as a discrete event.

According to the reinforcement learning formalization, we refer to the actuator setting at time t as a_t, to the environment state at time t as s_t. The environment state resulting at time $t+1$, as a consequence of the actuator setting, is referred to as s_{t+1}, while the reward obtained according to the users' feedback is r_{t+1}. After the evaluation of users' feedback, the current estimate of the average of the current actuator setting a_t in the current environment state s_t, referred to as $Q(s_t, a_t)$, is updated according to the following equation:

$$Q(s_t, a_t) \leftarrow (1-\beta)Q(s_t, a_t) + \beta[r_{t+1} + \gamma \max_a\{Q(s_{t+1}, a)\}]. \quad (4)$$

The new utility estimate is obtained by merging the previous one with the information about the obtained reward and the future one; namely $\max_a\{Q(s_{t+1}, a)\}$ is the maximum obtainable reward in the new state. The β and γ parameters, both ranging in $[0, 1]$, control the learning mechanism, and represent the learning rate and the discount factor, respectively. The former determines the weight of new information with respect to past

history, and the latter determines the influence of future rewards. By setting the γ parameter to 0, utility is estimated according to a simple exponential moving average, similarly to what is done in the energy consumption module.

The information that may be gathered by AmI applications from the user's preference profile may regard synthetically the optimal action given a specific environmental condition, or broadly the entire utility table learnt so far.

6. A Complete Sample Application: Temperature Control in an Office Room

This Section presents a simple AmI application aimed at illustrating the use of the various hardware and software components of our testbed; the main goal here is to provide a proof-of-concept of the complete sensing-reasoning-acting loop. The testbed has been specialized for the context of temperature control in a work environment, and the physical layer was built by augmenting a classic office room with a set of non intrusive devices. The premises of our department were chosen as a convenient experimental platform, and we aimed to control the temperature conditions not just for improving the users' comfort, but also for optimizing the overall energy consumption.

6.1. The Sensory Infrastructure

The sensor nodes used for this sample application belong to the *mote* family [38]; they are particularly suitable for our purposes, also thanks to the possibility of extending their functionality by adding new types of sensors or actuators. TelosB Motes, in particular, are equipped with 10 kB RAM, 16 kB for configuration EEPROM, and 1024K bytes for data storage into Flash serial memory; the communications among them are based on the IEEE 802.15.4 protocol over a 250 kbps radio channel.

In order to assess the practical usability of our testbed, we compared its requirements in terms of memory occupancy with other widely used applications. The two lowest rows of Table 2 show the separate memory footprints for the SENSOR$_{9K}$ middleware, and for the specific application considered here; for instance, our middleware requires roughly double memory for code, as compared to the basic Blink application for TinyOS. A more reliable comparison can be made with other popular middleware tools for WSNs, such as Maté [34], and Agilla [22], showing that the requirements for code and data are comparable.

Table 2 Memory footprint comparison (data for Maté and Agilla are from [19]).

	App. code (ROM)	**App. data (RAM)**
Blink	2650 bytes	55 bytes
Maté	7.5 kB	600 bytes
Agilla	3.59 kB	41.6 kB
SENSOR$_{9K}$ middleware	4672 bytes	512 byte
Sample application	8.5 kB	3 kB

Nodes have been placed at strategic points in the rooms (see Figure 8), in regions where sensed measurements may present oscillations and unexpected trends; specifically, we deployed nodes in different rooms, close to "sensitive" areas: by the door, by the window, and by the user's desk; additional nodes have been installed on the building facade, close to the office windows, for monitoring outdoor temperature, relative humidity, and light exposure. Besides the sensors provided by SENSOR$_{9K}$, Table 3 shows the main sensors available to this sample application for environmental monitoring.

Based on the measurements of ambient temperature and relative humidity, the application computes the physiological equivalent temperature (PET) [29], and uses this value to make a decision on the proper actuation policy. The equivalent temperature is computed through the index defined in [13], which adds the latent heat of condensation for the water vapor in the air to the actual temperature. The resulting empiric formula

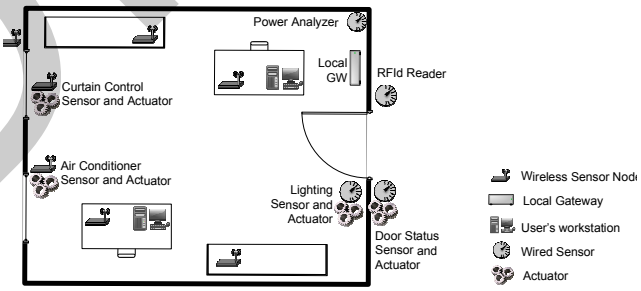


Figure 8 Location of wireless sensor nodes installed in user offices.

Table 3 The main sensors used for environmental monitoring, and their characteristics.

Measure	Sensor	Characteristics
Temperature and relative humidity	Sensirion SHT11	Temperature range: -40 °C to +123.8 °C Temp. accuracy: +/- 0.5 °C @ 25 °C Humidity range: 0 to 100% RH Absolute RH accuracy: +/- 3.5% RH Low power consumption (typically 30 μW)
Barometric pressure and temperature	Intersema MS5534	Pressure range: 300 to 110 mbar Pressure accuracy: +/- 3.5% Temperature range: -10°C to 60°C Temperature accuracy: +/- 2°C Operating range 3.6 to 2.2 volts
Outdoor Light	Hamamatsu S1087	Spectral response range λ: 320 – 730 nm Peak sensitivity wavelength λp 560 nm Photo sensitivity S (A/W) Infrared sensitivity ratio 10%
Ambient Light	Taos TSL2550	Range: 400 to 1000 nm Operating range 3.6 to 2.2 volts

is the following:

$$PET = T + m_h \cdot (r - 2.326 \cdot T)/(c_p + m_h \cdot c_w). \tag{5}$$

For our purposes, we have considered a fixed value for the atmospheric pressure above sea level, and computed all other involved quantities as reported in Table 4, so that the value of the index only depends on the measured ambient temperature T (directly) and relative humidity (through the moisture content parameter m_h). The PET index, for atmospheric pressure in the 800–1100 mbar range, returns meaningful values when the

Table 4 Parameters for computing the PET index.

Quantity	Meaning	Value
m_h	moisture content	$0.620 \cdot \frac{p_v(hum)}{p_{atm} - p_v(hum)}$
p_{atm}	atmospheric pressure	$1013\, mbar$
$p_v(hum)$	vapor pressure	$610.78 \cdot e^{\frac{17.269 \cdot T}{T + 237.30}} \cdot \frac{hum}{100}$
r	latent heat of vaporization	$585\, cal \cdot g^{-1}$
c_p	specific heat of air	$0.24\, cal\, °C\, g^{-1}$
c_w	specific heat of water	$1\, cal\, °C\, g^{-1}$

measured ambient temperature is between 20°C and 45°C; since our test environment might also experience lower and higher temperature, we consider here the following aggregated index PT as representative of a subjective measure of human perceived temperature:

$$PT = \begin{cases} PET & if\ T > 20°C, \\ T & elsewhere. \end{cases} \tag{6}$$

The nodes of the WSN dedicated to environmental control are programmed to directly compute the PT value, by performing all computations and thresholding on board; in fact, the application layer may in all respects assume to deal with a virtual sensor, undistinguishable from other common sensors.

6.2. Exploiting the Middleware Modules

The sample application considered here shows how to profitably make use of the middleware modules described in Section 5. In particular, besides analyzing the perceived temperature, the application will get information about the presence of the users in their office from the middleware module described in Section 5.2, in the form of a probability measure; this module is installed on the LG, which forwards the presence probability, together with the information about the state of the actuators, up to the AmI application.

Furthermore, a profile of the user's preferences is computed thanks to the functionalities provided by the module described in Section 5.3; in this case, the module has been parameterized in order to collect the associations between the environment state and the users' actions, as represented by the following vectors:

$$\begin{aligned} state &= [PT, PT'], \\ action &= [\Delta T, Mode]. \end{aligned} \tag{7}$$

This middleware module returns the user's profile expressed in terms of the utility function as perceived by the user for each $[state, action]$ pair. In our sample application we compute the reciprocal of this utility value, which may be interpreted as a metric representing the distance from the user's preference, and regard this as the first objective function to be minimized.

The sample application also uses the middleware module described in Section 5.1 for predicting the cost of each actuator setting; this cost value is the second objective function to be minimized.

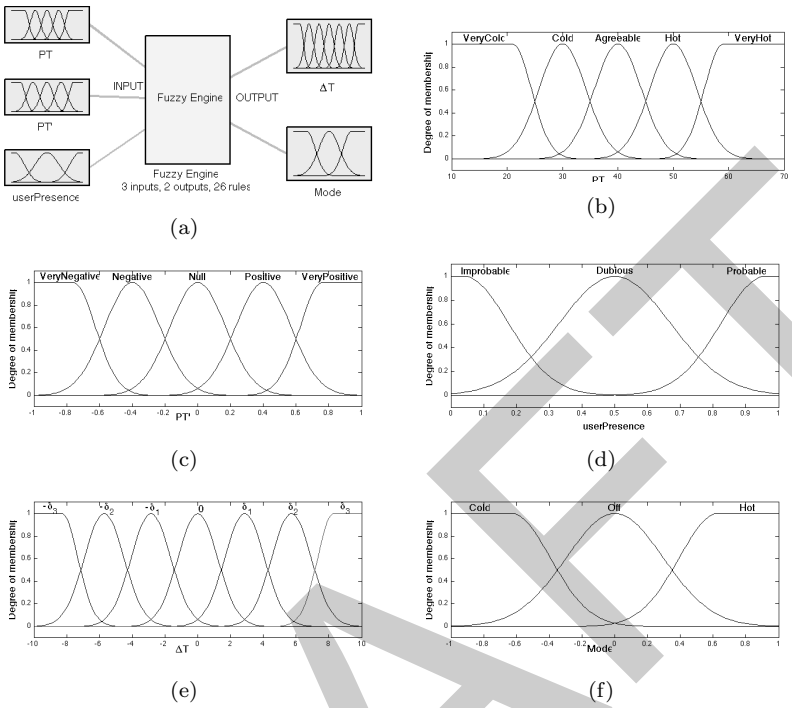


Figure 9 Block diagram of fuzzy controller (a) and membership functions for input and output variables: (b) input Perceived Temperature, (c) input Perceived Temperature Variation Rate, (d) input user's presence, (e) output Temperature Variation , (f) output Operating Mode.

It is worth mentioning that if the middleware modules for user profiling and cost prediction provided only static, off-line models, the resulting application policy would be characterized by overfitting, and the system would only correctly react to ambient conditions analogous to those experimented during the training phase. In our case this is avoided by allowing each module to incrementally update its model, based on on-line information.

6.3. The Sample Application

The considered sample application, running on the central AmI server, deals with actuator management and enforces advanced AI algorithms for optimizing environmental control, also taking into account constraints about energy consumption. In particular, the management is performed through a simple fuzzy controller operating on the sensory inputs.

Fuzzy Logic allows to model uncertainty of sensory data and inaccuracy of human-based definitions; in our case, for instance, it allows to manage vague concepts such as *Cold* while speaking about perceived temperature. An AmI application whose core is a fuzzy controller is able to mimic human reasoning, and to simply model a non-linear mapping from inputs to outputs.

Considered input variables are the PT index, its variation rate PT' and the *userPresence* probability value, opportunely fuzzified, while output variables are the variation of the controlled temperature (ΔT) with respect to the current assigned value, and the operation mode of the air conditioner ($Mode$), which can be set to `Cold`, `Off` and `Hot`. A block diagram of this fuzzy controller is shown in Figure 9(a).

The fuzzy knowledge base comprises five gaussian membership functions for each input variable, seven gaussian membership functions for the output variable ΔT and three gaussian membership functions for the output variable $Mode$, as shown in Figure 9.

The fuzzy engine has been designed so that with a high probability that the user is present in his office the air conditioner is tuned by acting on its functioning mode and temperature, in order to found the best setting that fits with the actual perceived temperature and its variation rate. Rules designed for this case are in the following form:

if ($userPresence$ **is** $Probable$) $\wedge$ (PT **is** val_{PT}) $\wedge$ (PT' **is** $val_{PT'}$) **then**
(ΔT **is** $val_{\Delta T}$) $\wedge$ ($Mode$ **is** val_M);

where val_{PT}, $val_{PT'}$, $val_{\Delta T}$ and val_M are linguistic values defined over the ranges of PT, PT', ΔT, and $Mode$ respectively. A sample of a complete mapping from $[PT, PT']$ to $[Mode, \Delta T]$, when $userPresence$ is `Probable`, is summarized in Tables 5 and 6.

If $userPresence$ is `Improbable` a unique rule is activated, that turns off the air condition in order to meet energy saving goal:

Table 5 Fuzzy rules for temperature variation.

$PT \setminus PT'$	**VNeg**	**Neg**	**Null**	**Pos**	**VPos**
VeryCold	δ_3	δ_3	δ_2	δ_1	0
Cold	δ_3	δ_2	δ_1	0	0
Agreeable	δ_2	δ_1	0	$-\delta_1$	$-\delta_2$
Hot	0	0	$-\delta_1$	$-\delta_2$	$-\delta_3$
VeryHot	0	$-\delta_1$	$-\delta_2$	$-\delta_3$	$-\delta_3$

Table 6 Fuzzy rules for the operating mode.

$PT \setminus PT'$	**VNeg**	**Neg**	**Null**	**Pos**	**VPos**
VeryCold	`Hot`	`Hot`	`Hot`	`Hot`	`Off`
Cold	`Hot`	`Hot`	`Hot`	`Off`	`Off`
Agreeable	`Hot`	`Hot`	`Off`	`Cold`	`Cold`
Hot	`Off`	`Off`	`Cold`	`Cold`	`Cold`
VeryHot	`Off`	`Cold`	`Cold`	`Cold`	`Cold`

if ($userPresence$ **is** $Improbable$) **then** ($Mode$ **is** Off).

When dealing with the conflicting goals of adjusting the perceived temperature according to the users' preferences while minimizing energy consumption, the traditional approach of merging them into a single objective function presents several limitations, mainly because it would require an accurate knowledge of the different objective functions, either in terms of relative priority or relevance. On the contrary, we chose to keep two independent objective functions, one expressing the dissimilarity from the users' preferences and the other expressing a degree of energy consumption, as previously mentioned.

The associations between $[state, action]$ pairs (as defined by Eq. 7), and the corresponding distance and cost values, are collected by the AmI application as shown in Table 7, where n is the number of discrete environment states and m is the number of possible actions.

For each state s_i, optimizing only with respect to the user's preference would involve selecting the action a^* that minimizes the $d_{s_i}(a^*)$ value, whereas optimizing only with respect to the energy consumption would involve minimizing the $c_{s_i}(a^*)$ value.

Since we intend to consider both objective functions at the same time, we adopt a Pareto-dominance criterion for evaluating the actions; this implies the selection of

Table 7 Evaluation of $[state, action]$ pairs via two objective functions.

State	**Action**	**Dissimilarity**	**Cost**
s_1	a_1	$d_{s_1}(a_1)$	$c_{s_1}(a_1)$
s_1	a_2	$d_{s_1}(a_2)$	$c_{s_1}(a_2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
s_n	a_m	$d_{s_n}(a_m)$	$c_{s_n}(a_m)$

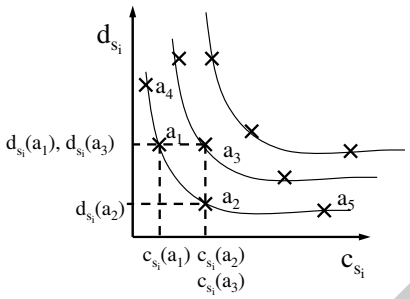


Figure 10 Graphical example of the Pareto-dominance analysis.

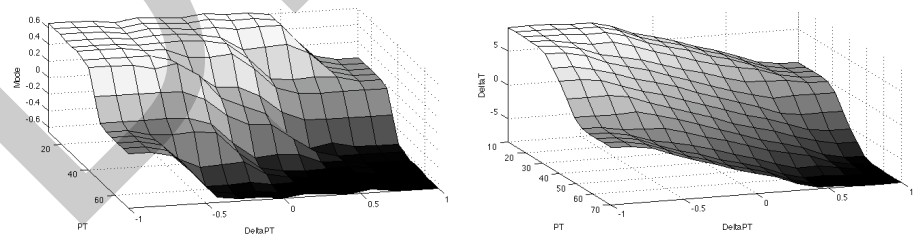
multiple optimal actions.

For a given state s_i, an action a_j Pareto-dominates another action a_k if $d_{s_i}(a_j) \leq d_{s_i}(a_k) \wedge c_{s_i}(a_j) \leq c_{s_i}(a_k)$. An action a^* is Pareto-optimal if no other solution has better values for each objective function, that is if the following equation holds:

$$d_{s_i}(a^*) \leq d_{s_i}(a_j) \wedge c_{s_i}(a^*) \leq c_{s_i}(a_j), \quad \forall j = 1 \ldots m. \tag{8}$$

Figure 10 represents an example of the Pareto-dominance analysis for a given ambient state s_i: actions a_1 and a_2 belong to the same non-dominated front because $d_{s_i}(a_2) \leq d_{s_i}(a_1)$ and $c_{s_i}(a_1) \leq c_{s_i}(a_2)$, while both actions a_1 and a_2 dominate action a_3; the set of optimal actions is $\{a_1, a_2, a_4, a_5\}$.

The action to perform in a given ambient state is selected inside the set of Pareto-optimal actions; namely, it is the action that represents the median point on the curve



(a) (b)

Figure 11 Nonlinear surfaces for the proposed AmI fuzzy application concerning (a) the Functioning Mode and (b) the Temperature Variation.

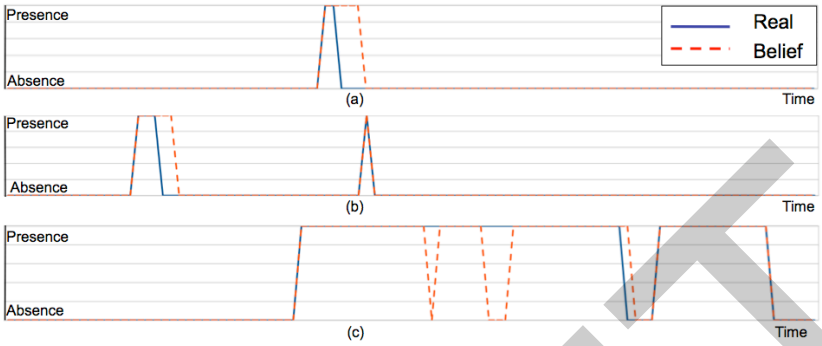


Figure 12 Results of the Bayesian multi-sensor data fusion process for user's presence detection (Test A).

of the optimal front.

Finally, Figure 11 shows the nonlinear surfaces produced by the Pareto-optimal fuzzy controllers that models the mapping among inputs and outputs.

7. Experimental Evaluation

We assessed the performance of our AmI sample application as a whole, and of the individual modules of the testbed; we report here a representative subset of our tests conducted over a three days time period. We considered one target user, who was unaware of the ongoing experiment, and so did not modify his usual behavior. In order to validate the subsystem for detecting the user's presence, its outcome was compared with information obtained by manually analyzing the output of a video-surveillance system.

The plots reported in Figure 12 show how the system performs in detecting user's presence. Figures 12(a), 12(b), and 12(c) consider three different time intervals, and each of them shows two superimposed plots: the continuous line represents the actual trend of the user's presence signal; the dashed line represents the belief of the system about the user's presence, as obtained by the Bayesian multi-sensor data fusion process, previously described in Section 5.2 (see Figure 7).

In order to get a deeper understanding of the behavior of this module in scenarios including a limited number of sensory devices, we carried on further experiments highlighting the impact of specific sensory information; namely, we iteratively excluded some of the sensory inputs from the probabilistic inference process, and run the tests on the

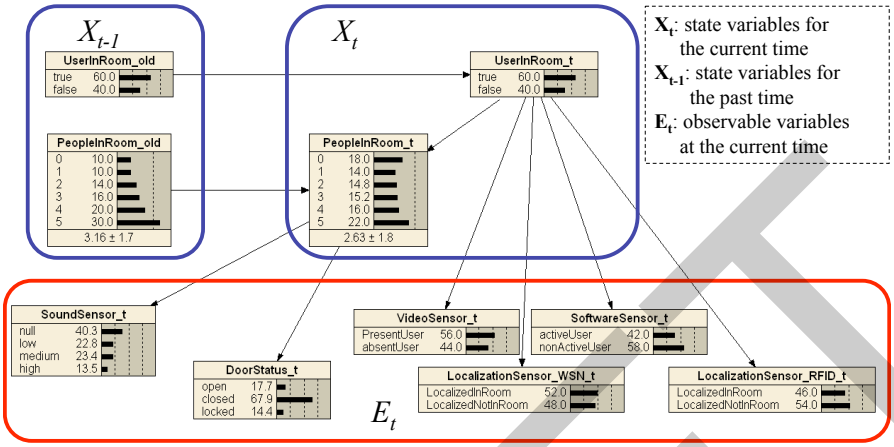


Figure 13 Structure of the Bayesian network without the ActivitySensor (Test B).

same data in order to get comparable results. The new tests indicate that the user detection module is robust with respect to the exclusion of a limited number of sensor types. It is however clear that there exists a critical threshold for the number of excluded sensor types beyond which the results deteriorate intolerably. Figure 13 shows the Bayesian network obtained after excluding the `ActivitySensor`; the corresponding test is labeled "Test B", as opposed to the original test including all the available sensors, and labeled as "Test A". As shown by the results in Figure 14, the presence detection system succeeds in compensating the lack of information by exploiting what is still available. The greatest difference appears at the beginning of time interval (b), when the software module detects the user only with some delay. Figure 15 reports the results from "Test C", where the `DoorStatus` sensor is missing in addition to the `ActivitySensor`; the figure shows that additional errors are present as compared to "Test B", both in time interval (b) and (c).

In order to obtain a statistical evaluation of the system performance we discretized the time intervals and computed false positives and false negatives; the specificity and the sensitivity of the system are computed according to the following definitions:

$$specificity = \frac{\#true\ negatives}{\#true\ negatives + \#false\ positives};$$
$$sensitivity = \frac{\#true\ positives}{\#true\ positives + \#false\ negatives}. \quad (9)$$

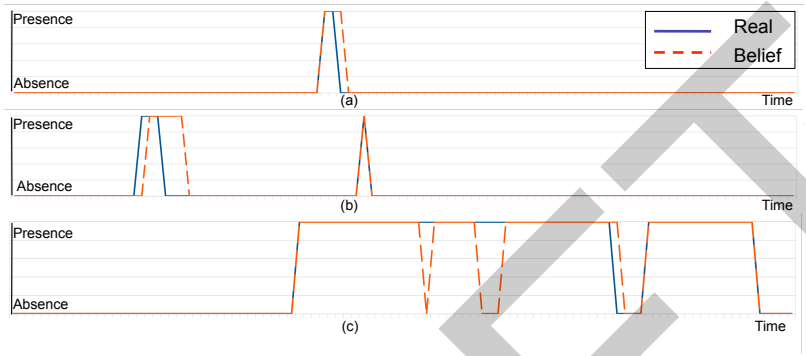


Figure 14 Results of the Bayesian multi-sensor data fusion process for user's presence detection without the ActivitySensor (Test B).

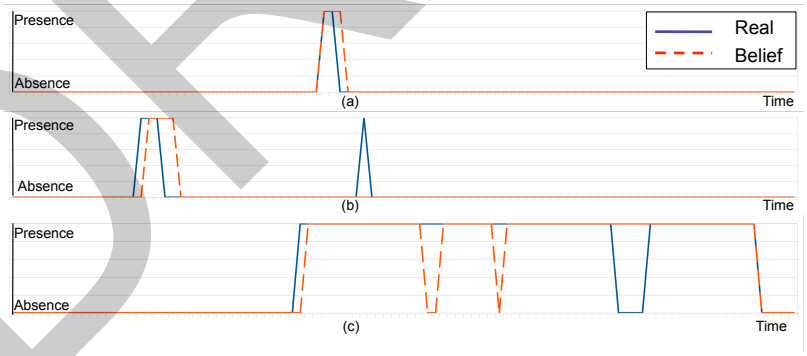


Figure 15 Results of the Bayesian multi-sensor data fusion process for user's presence detection without the ActivitySensor and the DoorStatus sensor (Test C).

Table 8 Specificity and sensitivity of the module for detecting the user's presence, in different sensor devices available.

Test	Excluded Sensor Types	Specificity	Sensitivity
Test A	none	97.50%	93.33%
Test B	`ActivitySensor`	97.92%	91.67%
Test C	`ActivitySensor`, `DoorStatus`	97.08%	90.00%

If the gathered sensory information is not sufficient to infer the user presence, the software module opts for a default strategy that indicates the user's absence. Such choice implies that sensitivity is the most relevant factor for the assessment of the system performance. In order to get a deeper insight on that, we can imagine a scenario where the software module is constantly unable to detect the user's presence; in such scenario the module keeps indicating user absence, hence no false positives may arise and specificity amounts to a constant 100%; however the anomalous situation shows clearly by looking at the sensitivity parameter, which falls to 0. Both the specificity degree and the sensitivity degree are reported in Table 8 for all conducted tests, and the worsening performance of the system are shown by the lowering sensitivity degree throughout the tests. It is however worth noting that the system still produces good results, even when lacking groups of sensory devices.

In order to assess the effectiveness the system's decisions as regards the resulting environmental conditions, we measured the mismatch between the current ambient state, and the desired ambient state inferred by observing the interactions of the user with the actuators. We used the same time discretization as for the localization system, and labeled each time interval as "satisfactory" if the user was in fact present and no interaction was detected, whereas the opposite label is used if the user modified the actuators settings at least once in that interval.

In order to assess the performance of the AmI application, we need to take into account 6,6% of the cases where the user was present in the office, but the localization system failed in detecting them. This in fact prevented the triggering of the rules for environmental control. This testbed-dependent error must be added to the application-dependent error, so that overall the AmI application sets the environmental state coher-

ently with the user's preferences in 84,5% of the cases, which is a remarkable behavior nonetheless.

8. Conclusion and Future Developments

This work presented a testbed for experimenting with AmI-based applications, which is characterized by the use of a low-cost and non intrusive technology, such as WSNs. Besides providing the hardware substrate, our testbed is also equipped with a middleware that allows for a full exploitation of the offered capabilities; the physical component is in fact decoupled from the higher-level module devoted to drive the operations of the entire system. A sample application involving the estimate of the users' presence in a given area, and a fuzzy controller for tuning the heating and air conditioning, was considered as a proof-of-concept of the role of the middleware in processing raw sensory data in order to extract high-level information.

SENSOR$_{9K}$'s main contribution relies in addressing energy efficiency in the context of pervasive systems, by providing the designer with comprehensive support for enabling energy-aware AmI systems, in order to obtain a globally sustainable development.

Future developments will focus on providing further capabilities, such as additional complex middleware modules for supporting advanced reasoning and planning, and more refined management services for the sensory infrastructure.

For instance, the multi-sensor data fusion module will be extended to use the high-level information extracted from raw environmental data, in order to perform dynamic selection of sensor devices, thus implementing self-optimization based on QoS and cost metrics. Additionally, SENSOR$_{9K}$ will implement self-monitoring mechanisms in order to timely detect potential malfunctioning in the low-level devices, so as to allow our uncertainty-based reasoning to exclude such faulty data.

In order to enhance the overall fault tolerance, such self-monitoring functionality will be coupled with basic primitives for self-reconfiguration and self-optimization, allowing for optimal tuning of sensing parameters.

Finally, our research will focus on investigating further techniques for sustainable energy management, specifically considering the optimization of the energy consumption of sensors and actuators. This topic is particularly challenging when some of the sensory

devices are allowed to access renewable energy sources, which suggests the adoption of energy harvesting policies in order to extend the overall lifetime of the pervasive sensor networks. Although this issue is very application-specific, and usually deeply affects all of the node architectural levels, we will focus on how to provide useful high-level tools, such as a middleware module for accessing information about residual node energy (e.g. measured voltage supply, discharge function), or specialized modules for the prediction of physical phenomena influencing the replenishment of the energy sources. Such information may be exploited for planning accurate strategies of load balancing that optimize the overall network lifetime.

References

[1] K. Aberer, M. Hauswirth, and A. Salehi. A middleware for fast and flexible sensor network deployment. In *Proceedings of the 32nd International Conference on Very large data bases*, pages 1199–1202. VLDB Endowment, 2006.

[2] H.S. Ahn and K.H. Ko. Simple pedestrian localization algorithms based on distributed wireless sensor networks. *IEEE Transactions on Industrial Electronics*, 56(10):4296–4302, 2009.

[3] M.J. Akhlaghinia, A. Lotfi, C. Langensiepen, and N. Sherkat. Occupant Behaviour Prediction in Ambient Intelligence Computing Environment. *Journal of Uncertain Systems*, 2(2):85–100, 2008.

[4] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communication Magazine*, 40(8):102–114, 2002.

[5] H. Alex, M. Kumar, and B. Shirazi. MidFusion: An adaptive middleware for information fusion in sensor network applications. *Information Fusion*, 9(3):332–343, 2008.

[6] F. Amigoni, N. Gatti, C. Pinciroli, and M. Roveri. What planner for Ambient Intelligence applications? *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35(1):7–21, 2004.

[7] E. Ardizzone, M. La Cascia, and M. Morana. Face Processing on Low-Power Devices. In *Proceedings of the 4th International Conference on Embedded and Multimedia Computing*, pages 1–6, Jeju, Korea, 2009.

[8] L. Atallah and G.Z. Yang. The use of pervasive sensing for behaviour profiling - a survey. *Pervasive and Mobile Computing*, 5:447–464, 2009.

[9] R. Barr, J. C. Bicket, D. S. Dantas, B. Du, T. W. Kim, B. Zhou, and E. G. Sirer. On the need for system-level support for ad hoc and sensor networks. *ACM SIGOPS Operating Systems Review*, 36(2):1–5, 2002.

[10] C. Bartelmus, P. d'Angelo, H. Langos, T. Wheely, K. Scheibler, J. Paris, P. T. Jochym, and M. Pikula. LIRC - Linux Infrared Remote Control. Available online at `http://www.lirc.org/`, 2002.

[11] L. Benini, E. Farella, and C. Guiducci. Wireless Sensor Networks: Enabling Technology for Ambient Intelligence. *Microelectronics Journal*, 37(12):1639–1649, 2006.

[12] M. Botts and A. Robin. OpenGIS sensor model language (SensorML) implementation specification. *OpenGIS Best Practices Paper OGC*, pages 5–86, 2006.

[13] W. Bründl and P. Höppe. Advantages and disadvantages of the urban heat island - an evaluation according to the hygro-thermic effects. *Meteorology and Atmospheric Physics*, 35(1):55–66, 1984.

[14] I. Chatzigiannakis, G. Mylonas, and S. Nikoletseas. 50 ways to build your application: A Survey of Middleware and Systems for Wireless Sensor Networks. In *Proceedings of the 12th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2007)*, pages 466–473. IEEE, 2008.

[15] S. Y. Chen, S. F. Song, Li L. X., and Shen J. Survey on Smart Grid Technology. *Power System Technology*, 33(8):1–7, 2009.

[16] D. J. Cook, J. C. Augusto, and V. R. Jakkula. Ambient Intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298, 2009.

[17] D. J. Cook and S. K. Das. MavHome: Work in progress. *IEEE Pervasive Computing*, 2004.

[18] D. J. Cook and S. K. Das. How smart are our environments? An updated look at the state of the art. *Pervasive and Mobile Computing*, 3(2):53–73, 2007.

[19] N. Costa, A. Pereira, and C. Serodio. Virtual Machines Applied to WSN's: The state-of-the-art and classification. In *Systems and Networks Communications, 2007. ICSNC 2007. Second International Conference on*, page 50. IEEE, 2007.

[20] A. De Paola, S. Gaglio, G. Lo Re, and M. Ortolani. Human-ambient interaction through Wireless Sensor Networks. In *Proceedings of the 2nd Conference on Human System Interactions (HSI)*, Catania, Italy, 2009.

[21] K. Ducatel, M. Bogdanowicz, F. Scapolo, and J.-C. Burgelman. *Scenarios for Ambient Intelligence in 2010, Tech. Rep.* Information Soc. Technol., Advisory Group (ISTAG), Inst. Prospective Technol. Studies (IPTS), Seville, 2001.

[22] C. L. Fok, G. C. Roman, and C. Lu. Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 4(3):1–26, 2009.

[23] G. González, C. Angulo, B. López, and J. L. de la Rosa. Smart user models: Modelling the humans in ambient recommender systems. In *Proceedings of the workshop on decentralized, agent based and social approaches to user modelling (DASUM)*, pages 11–20, Edinburgh, Scotland, 2005.

[24] S. Hadim and N. Mohamed. Middleware: Middleware challenges and approaches for wireless sensor networks. *IEEE Distributed Systems Online*, 7(3):1–1, 2006.

[25] H. Hagras, I. Packharn, Y. Vanderstockt, N. McNulty, A. Vadher, and F. Doctor. An intelligent agent based approach for energy management in commercial buildings. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2008)*, pages 156–162. IEEE, 2008.

[26] S. Han, H. S. Lim, and J. M. Lee. An efficient localization scheme for a differential-driving mobile robot based on rfid system. *IEEE Transactions on Industrial Electronics*, 54(6):3362–3369, 2007.

[27] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen. The Gator Tech Smart House: A programmable pervasive space. *Computer*, 38(3):50–60, 2005.

[28] A. Holmes, H. Duman, and A. Pounds-Cornish. The iDorm: Gateway to heterogeneous networking environments. In *International ITEA Workshop on Virtual Home Environments, Paderborn, Germany (February 2002)*, pages 1–8, 2002.

[29] P. Höppe. The physiological equivalent temperature – a universal index for the biometeorological assessment of the thermal environment. *International Journal of Biometeorology*, 43(2):71–75, 1999.

[30] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[31] J. A. Kientz, S. N. Patel, B. Jones, E. Price, E. D. Mynatt, and G. D. Abowd. The Georgia Tech Aware Home. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 3675–3680. ACM, 2008.

[32] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Computer Networks*, 43(4):499–518, 2003.

[33] R. G. Lee, C. C. Lai, S. S. Chiang, H. S. Liu, C. C. Chen, and G. Y. Hsieh. Design and implementation of a mobile-care system over wireless sensor network for home healthcare applications. In *Proceedings of IEEE Conference on Engineering in Medicine and Biology Society (EMBS '06)*, pages 6004–6007, New York City, USA, 2006.

[34] P. Levis and D. Culler. Maté: A tiny virtual machine for sensor networks. *ACM SIGARCH Computer Architecture News*, 30(5):85–95, 2002.

[35] T. Liu and M. Martonosi. Impala: a middleware system for managing autonomic, parallel sensor systems. In *Proceedings of the ninth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 107–118. ACM, 2003.

[36] C. H. Lu and L. C. Fu. Robust Location-Aware Activity Recognition Using Wireless Sensor Network in an Attentive Home. *IEEE Transactions on Automation Science and Engineering*, 6(4):598–609, 2009.

[37] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems (TODS)*, 30(1):122–173, 2005.

[38] Memsic. TelosB datasheet. Available online at `http://www.memsic.com`, 2010.

[39] M.M. Molla and S.I. Ahamed. A survey of middleware for sensor network and challenges. In *Proceedings of the 2006 International Conference on Parallel Processing (CPP 2006) Workshops*, pages 6–228. IEEE, 2006.

[40] S. Park and S. Hashimoto. Autonomous Mobile Robot Navigation Using Passive RFID in Indoor

Environment. *IEEE Transactions on Industrial Electronics*, 56(7):2366–2373, 2009.
[41] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.
[42] G. Roussos and V. Kostakos. RFId in pervasive computing: State-of-the-art and outlook. *Pervasive and Mobile Computing*, 5(1):110–131, 2009.
[43] A. Savvides, H. Park, and M.B. Srivastava. The n-hop multilateration primitive for node localization problems. *Mobile Networks and Applications*, 8(4):443–451, 2003.
[44] R. Vastamaki, I. Sinkkonen, and C. Leinonen. A behavioural model of temperature controller usage and energy saving. *Personal and Ubiquitous Computing*, 9(4):250–259, 2005.
[45] J. Wang, R. K. Ghosh, and S. K. Das. A survey on sensor localization. *Journal of Control Theory and Applications*, 8(1):2–11, 2010.
[46] M. Weiser. The computer for the 21st century. *Scientific American*, 272(3):78–89, 1995.