Paradigms, Frameworks and Prototypes towards the Realization of the Internet of Things

C. Tselikis, A. Alexopoulou, C. Vangelatos, A. Leventis, E. G. Ladis
Hellenic Aerospace Industry
Electronic Systems & Applications Design Dept.
P.O. Box 23, Schimatari 320 09, Greece
tselikis.christos@haicorp.com

Abstract

The problem of integrating the ad hoc networks and especially the Wireless Sensor Networks (WSN) with the Internet has attracted significant research and implementation efforts so far. Admittedly, not all of them have been widely adopted and established as prevalent standards or solutions in this technical area. In this paper we survey the proposed interconnection models and the relevant architectures. We discern three different communication models, we describe some of the existing frameworks towards the development of Internet-based sensor applications and, finally, we present a number of prototypes that implement and use the above concepts. The analysis reveals the pros and cons of each technology and makes hints about the trends and the tracks that will lead to the problem's solution in the near future.

Key Words: sensors, Internet, integration, prototypes.

1. Introduction

In this paper we review different possible techniques which can be used for the integration of networks of small devices (actuators, sensors, PDA and laptops) with the Internet. There has been significant effort devoted to this area because the demand for the users/operators to be able to access seamlessly the resources and the services offered by small devices has become significant.

Quickly we outline the difficulty in this venture by considering the software architecture that describes the world of the very restricted devices, especially sensors with wireless transmission capability. Figure 1 illustrates the basic components of the architecture of such devices. It is agreed in most of the related works that the components form a four-level layered architecture (modules exist at the physical layer, the operating system stack including the device drivers and virtual machines, the middleware layer which exports various services that can be accessed by the user, the sensor applications and/or the management applications of the sensor network), as shown in Figure 1.

A first outcome when inspecting Figure 1 is that the networking functionality is not included as part of the operating system layer. Indeed, in the majority of the WSN that have been developed so far the routing functionality is addressed in the form of networking algorithms and protocols that are integrated as applications, i.e., in many cases the routing components reside outside the operating system boundaries. Moreover, it is evident from Figure 1 that the desired interconnection of the sensor/ad hoc networks with the Internet architecture is difficult due to the inherent restrictions of the devices and due to the strong dissimilarity between the protocols and the applications that are encountered in the sensor networks and the Internet.

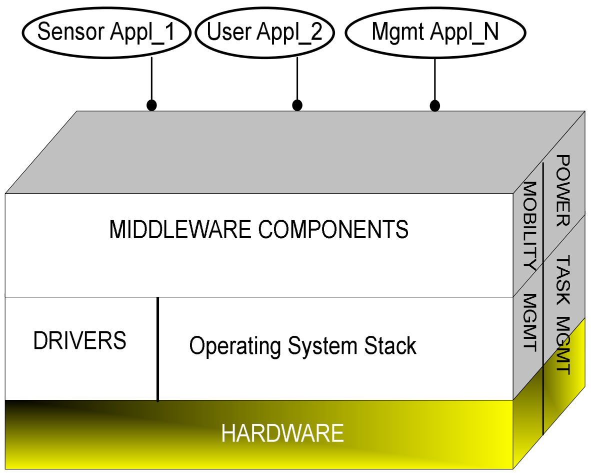

Figure 1. The generic sensor software architecture.

2. Interconnection Architectures

The interconnection between sensors and the Internet can be achieved in various possible levels of the software architecture. In [1] three different overlay networking architectures for micro-sensor

978-0-7695-4160-0/10 $26.00 © 2010 IEEE
DOI 10.1109/DeSE.2010.16

interconnection via the Internet are examined, namely the IP overlay over WSN, the sensor overlay over IP and the higher-level gateway overlays. According to [1], when the Internet overlays the WSN (first architecture), it is necessary for the devices to be addressed in the same way with all the rest hosts in the Internet, a fact that is not the norm in ad hoc networks.

In the second interconnection model the sensor networks overlay the Internet. In this model the sensor readings and the routing-related information can be disseminated seamlessly to remote hosts outside the sensor network.

Similarly to the sensor overlay model, according to [1], the third interconnection model is based on application-level gateways which translate the different proprietary routing protocols between the WSN and the infrastructure. In this case the sensor network stack (from the routing layer and upwards) is encapsulated into the IP packets. The encapsulated information is communicated end-to-end to remote Internet hosts and/or to remote sensors via TCP/UDP. In effect, we have the formation of *virtual sensors* where the sensors communicate directly and transparently via the Internet, notwithstanding the individual characteristics of the underlying WSN protocols. However, we note that in most cases a proxy node is still needed in order to achieve the "direct" communication between the remote entities.

To our view, the choice of overlaying the Internet protocols over the sensor networks (first model in [1]) is still a strong option. IPv6-based architectures like 6loWPAN protocol [2] and border routers [3] achieve interconnection with additional management features (like auto-configuration and neighbor discovery capabilities) at the TCP/IP level, a solution called the *TCP/IP solution* in the review presented in [4]. Further, we note that in the gateway interconnection model also belong the various ad hoc service discovery protocols which are adapted for restricted devices, such as the UPnP [5], [6], the Jini [7], the SLPv2 [8] and the Bonjour specifications [9], [10].

In [4] the requirement for interconnection between sensor/small devices via the Internet is heavily disputed, mainly due to the security implications of the venture; indeed, the cases where it is sufficient (and more efficient) to achieve interconnection of the small devices only at the local network level are not rare, for example via the industrial intranets where security might be of major concern. Furthermore, the security mechanisms that are necessary when accessing the Internet (primitives or intrusion detection rules) can impose intolerable overhead to the hardware-restricted devices.

We have to stress that the interconnection models should be abstract, open and transparent and this can be guaranteed better if we examine the integration at the middleware level (which does not exclude the observations presented in [1] and in [4]).

Traditionally, the role of the middleware is to glue together heterogeneous systems, i.e., different hardware platforms, different operating systems, different communication protocols and to facilitate the management (development, deployment and update) of distributed applications. In the context of sensor/ad hoc networks some additional middleware requirements are green operation, asynchronous eventing, increased level of efficiency, scalability, service management (e.g. service discovery and update), self-configuration with adaptation to the changing network topology and self-protection to both intended threats and random damages.

3. Communication Models

In the next paragraphs we present the pros and cons of some of the basic communication models (paradigms) in the context of sensor/ad hoc networks.

3.1. Synchronous Client-Server

Traditional client-server interactions are the simplest way for two remote applications to communicate synchronously, either being coded in procedural or in object-oriented form. Client-server communications often rely on infrastructure and run above protocols such as the simple request/reply protocol or on remote method invocations among objects. Therefore, the goal to achieve eventing with asynchronous communications between highly heterogeneous devices can't be matched to the synchronous client-server paradigm.

3.2. Message Oriented Middleware

Messaging systems are prominent applications of the peer to peer distributed computing paradigm and constitute the natural choice for efficient asynchronous interactions among small devices and distributed software modules. Message Oriented Middleware (MOM) fulfills better the requirements that we set previously, namely openness, transparency, loose-coupling, and mainly asynchronous interactions among the components involved in the integration of WSN with the Internet. Prominent example of MOM are the publish/subscribe messaging systems in which multicast semantics are used. In WSN the devices can advertise customized topics (such as temperature, light, motion detection and other) which zero or more peer subscribers receive upon connection.

3.3. Service Oriented Architecture

The Service Oriented Architecture relies on describing the attributes and the operations supported by the small devices (e.g. *getTemperature*, *getHumidity*) with XML. The (web) services invocation can happen either at the gateway brokers that stand at the edge of the ad hoc/sensor network or even directly at the devices which in this case must have installed the WSDL file. It is evident that the regular

SOAP-based Web Services model without modifications (compression, binary adaptation, footprint reduction and message simplification) can't match the restricted environment of sensor nodes. A subset of the regular WS standards has to be used, such as the DPWS specification [11], an open-source implementation of which can be found in [12].

3.4. Data-Centric

The data-centric is an event-based model and according to that the sensors can be virtually viewed as relational tables so that the sensor readings which are cached in-network can be directly accessed via an appropriate database API (e.g. Cougar [27]).

4. Middleware Frameworks

In the next paragraphs we briefly describe the most known open-source middleware frameworks which have been used in the development of prototype solutions for sensor/ad hoc networks.

4.1. OSGi

Figure 2 illustrates the four-level software architecture of the java-based OSGi service platform [13] (the JNI API allows to access native code). Basic components of the OSGi technology are the bundles, (i.e., OSGi applications in the form of deployable web service packages which can be imported/exported), the OSGi framework (i.e., the run time system where the services are deployed and maintained via the OSGi lifecycle model), the service registry (inside the run time system) and the vertical security layer (used for bundle protection). A number of standard services has been defined which map several external protocols (like HTTP, Jini and UPnP) to OSGi services.

The purpose of the OSGi platform is to offer a standardized and integrated environment to facilitate the development, discovery, tracking, communication and cooperation of component-based dynamic Web applications by following the SOA and the Service Oriented Device Architecture (sensors as utilities in enterprises [28]).

The dynamicity here refers to the capability to choose/control the services at run time. There are numerous implementations of the specifications from both the commercial world and the open source community; especially regarding the run time system, there are currently six known open source implementations ([21]-[25]), namely the Eclipse Equinox, the Knopflerfish, the Oscar, the Apache Felix, the Concierge and the IBM Service Management Framework.

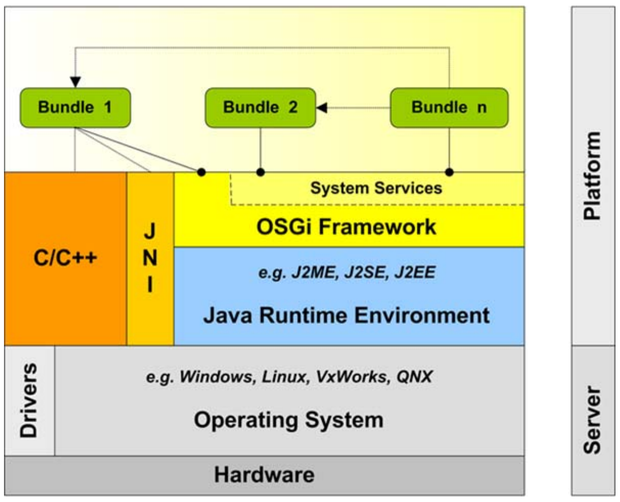

Figure 2. The OSGi software architecture.

4.2. Bonjour

Bonjour [10] is a set of specifications for service/device discovery. Bonjour allows for zero-configuration networking of devices and services in IP networks (i.e., without the need of DHCP). However, in order to perform service discovery across the Internet one can capitalize on DNS which is included in one of the Bonjour protocol specifications.

4.3. OGC

The Open Geospatial Consortium (OGC) [19] has provided specifications for introducing SOA in scenarios like environmental monitoring and critical infrastructure protection. Prominent OGC specifications are the Sensor Web Enablement (SWE) suite of standards, the language for modeling the sensors (sensorML), the location services and information models (openLS core service) and Sensor Observation Servers (SOS) which are appropriate to host sensor observation data and execute fusion procedures with geo-references [20]. Notably, the most popular OGC specification with respect to the number of implementations is the OpenGIS Web Map Service.

5. Prototypes

The paragraphs that follow describe in more detail certain prototype implementations which belong to the aforementioned paradigms and frameworks. The target applications include home energy management, pervasive healthcare, homeland security, environmental monitoring, surveillance, data streaming and critical infrastructure protection. Obviously, the existing implementations are driven by application-specific requirements and also by the specific characteristics of the targeted platforms (e.g. tinyOS, CLDC or embedded Linux). However, some projects (e.g. VITRO) seek application neutrality via virtualization.

5.1. MIRES

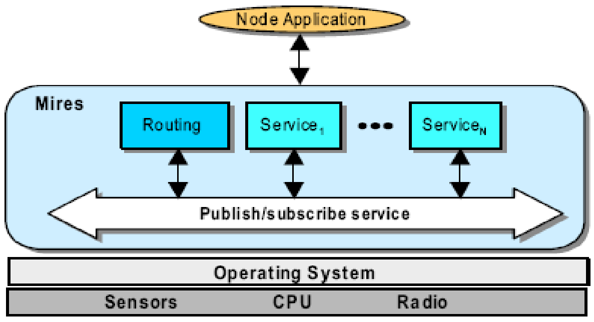

Figure 3. The architecture of the MIRES middleware.

As shown in Figure 3, the MIRES middleware [14] stands between tinyOS and the applications layer. MIRES is an implementation of the publish/subscribe paradigm (following the network establishment, the topics are advertised to the base station via the routing component; in sequence, the data messages are published throughout the network to the subscribed interested users) by utilizing the primitives supported by the tinyOS. Apart from the messaging system, the MIRES middleware includes services and a multi-hop routing component. Three types of notification events are supported and an in-network data aggregation service is implemented with respect to the published topics that reach the nodes of the sensor network.

The MIRES stack is installed on the sensor nodes and on the base station while the end-user application interacts with it through configuration messages which are applied as input from within a user interface.

5.2. MORE

The MORE prototype [15], [16] is a prototype designed for embedded devices (CLDC platform). The MORE services are externally utilized by following the SOA model (publish, discover and invoke service) by relying on a proxy server.

What is interesting in MORE is its internal software architecture. MORE is a combination of the benefits of the OSGi framework and the DPWS set of WS specifications [11] (WS-Eventing –MORE implements publish/subscribe messaging functionality–, WS-Addressing, WS-Discovery and WS-Transfer). Figure 4 illustrates the internal software architecture of the MORE middleware which effectively is an integrated (Web) services management environment most suitable for remote service deployment.

As shown in Figure 4, the MORE runtime is implemented as a DPWS-compliant stack which is installed on small devices as a regular OSGi bundle that has been hooked to the OSGi runtime.

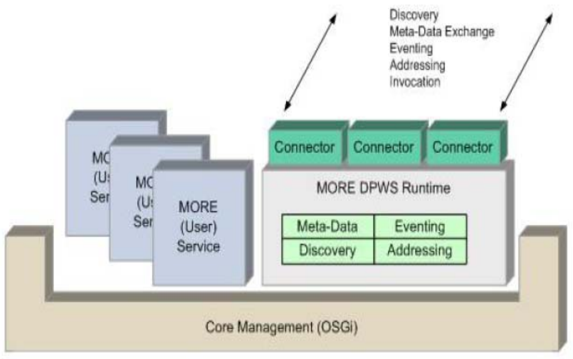

Figure 4. The MORE run time system and the services.

In effect, the DPWS stack is responsible for event-handling and addressing mechanisms, while one MORE service can be discovered by an external user by looking-up the DPWS registry at the enterprise proxy level. Notably, the DPWS service registry is automatically updated with the service bundles that have been previously registered with the OSGi runtime's registry.

Another advantage of the MORE middleware is that its services are included in a toolkit of building blocks. That enables the selective deployment of MORE services on small devices, however depending on the specific requirements per use case. The prototype of the distributed middleware was tested with WS message interactions among a Gumstix server [17] with the open source DPWS4J stack [12], a gas sensor equipped with a GSM/GPRS wireless module (on which the DPWS stack was installed) and a web services client that made the remote invocations. In the results it is recognized that the DPWS standard is difficult to be ported on highly restricted embedded devices, unless further adaptation is performed.

5.3. MULLE

MULLE [9] is a lightweight implementation of the standards-based Bonjour service discovery protocol [10] which has been developed for constrained peer devices. The prototype is with 20kBytes of RAM and runs the TCP/IP and the Bluetooth stack in a lightweight form. According to [9], the Bonjour is chosen as a standardized service discovery protocol because it is based on DNS; in effect, the prototype can be tested for local (in a Bluetooth piconet) and remote sensor service discovery.

Contrary to others protocols, such as UPnP and Jini, it is recognized [9] that in Bonjour (and in SLP) the format of the advertised services and return values is unspecified and therefore an IP-based application protocol is necessary in order to access the sensor data.

5.4. SOA-WSN

The SOA-WSN prototype [18] implements the SOA model for sensors running the tinyOS.

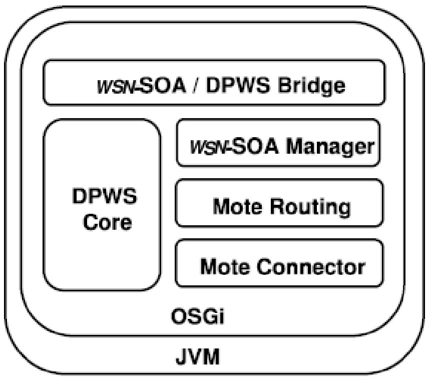

Figure 5. The software architecture of the gateway.

The WSN-SOA architecture, regarding the highly constrained nodes, adds one management layer above the tinyOS which implements service discovery, service information announcement and event-based messaging according to the Web Services specifications. HELLO messages are sent periodically to announce the available services and routing is performed by implementing in the stack a distance-vector multi-hop routing protocol that makes use of periodic beacons.

In SOA-WSN the DPWS stack is installed on powerful gateway nodes. The gateways have two fundamental functionalities, namely to translate as bridges between WSN-SOA and DPWS and to implement WS-Discovery, WS-Topics and WS-Eventing in order to achieve data dissemination via appropriate event handlers. Therefore, in SOA-WSN the tiny nodes and the bridges can communicate as peers according to the publish/subscribe model. As shown in Figure 5, all the presented modules are deployed as bundles into the OSGi framework.

The DPWS stack is also installed on the tier of even more powerful clients who search for and invoke the WSN services. Initial benchmarking tests prove good performance during the network initialization phase (including registration of all services), approximately 4 seconds for up to 10 motes, and good performance for service invocation (2 seconds).

6. VITRO

VITRO [26] is a research project accepted for funding by the EU 7th Framework Programme. The authors of this paper constitute the Hellenic Aerospace Industry research team that participates in the project. The VITRO proposal focuses on providing support for collaborative and multi-purpose Virtual Sensor Networks. In this context, the identified research areas include:
a) Optimization of energy efficiency by the use of a distributed network-channel coding and network coding division multiplexing heavily tailored to the characteristics of the embedded system. b) Virtualization with emphasis put on the design and development of a distributed multi-layer and service-oriented middleware aiming to bring the benefits of SOA to low capacity nodes. c) The design of an innovative multi-channel, flexible MAC solution in support of seamless connectivity and performance constraints. d) Trusted routing which in conjunction with the underlying IEEE 802.15.4e MAC specifications will provide an interoperable and trust-aware management scheme, e) Formal validation methods that will be applied to the proposed protocol stack and f) Real-time reconfiguration of the VITRO nodes. The latter will be performed in user friendly and fully automated ways by the means of a specially developed toolset which will take advantage of the developed protocol stack.

7. SMART

SMART [29] is a project of ARTEMIS joint undertaking which currently is at the development phase and in which the authors participate as a research team. SMART targets to design and implement a highly reconfigurable Wireless Visual Sensor Node (WVSN) defined as a miniaturized, light-weight, secure, low-cost and battery powered sensing device. The SMART node utilizes a wireless interface and a mobile ad-hoc network and is able to capture images and video streams utilizing embedded micro-cameras.

The objective is to pre-process the content in reconfigurable hardware resources and to efficiently transmit it over an ad-hoc network, while providing high-levels of security.

To this end, SMART proposes a new architecture for various sensor nodes, which would be based on reconfigurable hardware devices (FPGA). The reconfigurable resources will execute the tasks (e.g. data and H.264/AVC video coding, encryption and authentication) more efficiently by the means of dedicated hardware modules while the rest of the processing tasks will be executed by an ultra-low-power CPU. Numerous and diverse application domains will take advantage of the corresponding features and efficiency of the SMART basic infrastructure.

8. Conclusion

We investigated different possibilities regarding the integration of WSN into the Internet with a focus on the middleware for the reason that a higher level of abstraction is provided at this layer.

Numerous current trends became obvious from this short overview. Asynchronous messaging technologies are necessary to handle the events coming from devices and their services; the SOA model guarantees flexibility and interoperability while constantly increases its momentum. Admittedly, the WS stacks based on the eloquent XML can't be afforded in highly constrained devices. Standards-based services and architectures is the least means to advance the integration efforts, given the huge disparity of the existing platforms and the different requirements per use case in the realm of small devices. The communication model of the sensor/ad hoc services (inter-service communication within the run times and remote service invocations) already plays a vital role and will be a determinant factor for enhancing the usage scenarios in the near future.

Most important is that the components-based, open-source frameworks and tools which already have been introduced are now being extended with additional features, such as generic service creation, dynamic ad hoc service composition, service cooperation, service tracking for enhanced service availability, security and real-time streaming with orientation to quality of service guarantees.

9. Acknowledgment

The work presented in this paper is supported by the EU-funded VITRO project, contract number *INFSO-ICT-257245*.

8. References

[1] “Unifying Micro Sensor Networks with the Internet via Overlay Networking”, Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN’04).

[2] G. Montenegro, N. Kushalnagar, J. Hui and D. Culler. RFC 4944: Transmission of IPv6 Packets over IEEE 802.15.4 Networks, Request for Comments, September 2007.

[3] J. W. Hui, D. E. Culler, “IP is Dead, Long Live IP for Wireless Sensor Networks”, SenSys’ 08, November 5–7, 2008, Raleigh, North Carolina, USA.

[4] R Roman, J Lopez, C Alcaraz, “Do Wireless Sensor Networks Need to be Completely Integrated into the Internet?”, Technical Report, Future Internet of People, Things and Services (IoPTS) eco-Systems, Brussels, December 2009.

[5] UPnP Forum, UPnPTM Device Architecture v1.0.1 Draft, December 2003. http://www.upnp.org.

[6] Gsottberger, Y., Shi, X., Stromberg, G., Sturm, T.F., and Weber, W., “Embedding low-cost wireless sensors into universal plug and play environments”. In Wireless Sensor Networks. First European Workshop, EWSN 2004. Proceedings. (Lecture Notes in Computer Science, Vol.2920), 2004, pp. 291 – 306.

[7] Sun Microsystems, JiniTM Architecture Spec.

http://www.sun.com/software/jini/specs/jini2_0.pdf.

[8] E. Guttman, C. Perkins, J. Veizades, and M. Day, Service Location, Protocol, Version 2, RFC 2608, June 1999, IETF.

[9] A. Ostmark, P. Lindgren, A. van Halteren, L. Meppelink, “Service and Device Discovery of Nodes in a Wireless Sensor Network”, IEEE CCNC 2006 Proceedings.

[10]Bonjour Developer Web Site, http://developer.apple.com/darwin/projects/bonjour

[11]Devices Profile for Web Services Specification, □□□□□□□ http://schemas.xmlsoap.org/ws/2006/02/devprof.□

[12]Open Source DPWS implementation, https://forge.soa4d.org.

[13]OSGi: OSGi Alliance, “OSGi Service Platform”, http://www2.osgi.org/Specifications/HomePage.

[14]E. Souto, G. Guimarães , G. Vasconcelos, M. Vieira, N. Rosa and C. Ferraz, “A Message-Oriented Middleware for Sensor Networks”, Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing, Toronto, Canada, 2004, pp. 127-134.

[15]MORE: “Network-centric Middleware for Group Communication and Resource Sharing across Heterogeneous Embedded Systems”, WWW page http://www.ist-more.org.

[16]S. Michaelis, A. Wolff, J. Schmutzler: MORE – Architecture and Services, Public Deliverable 2.1, EU Project MORE, Dortmund, Germany 2007, http://www.ist-more.org, accessed 18.02.2008.

[17]Gumstix, Inc. 3130 Alpine Rd, Suite 288-606, Portola Valley, California 94028, http://www.gumstix.com.

[18]J. Leguay, M. Lopez-Ramos, K. Jean-Marie, V. Conan: “Service oriented architecture for heterogeneous and dynamic sensor networks”. Proceedings of the Second International Conference on Distributed Event-Based Systems, DEBS 2008, Rome, Italy, July 1-4, 2008.

[19]Open Geospatial Consortium standards, http://www.opengeospatial.org/standards.

[20]S. Kunz, T. Uslander, K. Watson, “A Testbed for Sensor Service Networks and the Fusion SOS: towards plug & measure in sensor networks for environmental monitoring with OGC standards”, 18th World IMACS / MODSIM Congress, Cairns, Australia 13-17 July 2009. http://mssanz.org.au/modsim09.

[21]http://www.eclipse.org/equinox/

[22]http://www.knopflerfish.org/

[23]http://felix.apache.org/site/index.html

[24]http://forge.ow2.org/projects/oscar/

[25]http://concierge.sourceforge.net/

[26]http://www.VITRO-ICT.eu

[27]Cougar Project, www.cs.cornell.edu/database/cougar

[28]C. Mauro, J. M.Leimeister, H. Krcmar, “Service Oriented Device Integration – An Analysis of SOA Design Patterns”, 2010 43rd International Conference on System Sciences, Hawaii.

[29]http://www.artemis-smart.eu.