Supporting Platform for Heterogeneous Sensor Network Operation based on Unmanned Vehicles Systems and Wireless Sensor Nodes

Edison P. Freitas, Rodrigo S. Allgayer, Marco A. Wehrmeister, Carlos E. Pereira, *Member, IEEE*, and Tony Larsson, *Member, IEEE*

***Abstract*— Advances in vehicle intelligence technology is enabling the development of systems composed of unmanned vehicles, which are able to interact with devices spread on the environment in order to take decisions related to their movements. Sensor networks represent an area that can profit a lot of this new possibilities, as autonomous vehicles can be used to carry sensor devices, which interacting with static sensor nodes can enhance the results provided by the overall system. However, some problems arise in applications' development in such systems due to the network nodes heterogeneity, and also the dynamicity of the environment in which they are deployed, which changes constantly. Thus, new platform solutions are necessary to handle the heterogeneous nodes capabilities in order to facilitate coordination and integration among them. This paper proposes a supporting infrastructure to address these problems composed of an adaptive middleware and a customizable sensor node platform. The goal is to support cooperation in heterogeneous sensor networks, which are composed by static and mobile nodes with different capabilities. The middleware adapts itself in order to manage the very distinct computing resources of the nodes, and also changes in the environment and in the application demands. The customizable sensor node platform allows optimizations in hw/sw modules to meet specific application requirements, allowing the creation of low-end and resource rich nodes that work in an integrated network. In order to illustrate the proposed approach, a system for military surveillance applications is presented as case study.**

I. Introduction

NEW applications based on the interaction between Unmanned Aerial Vehicles (UAVs) and static sensor nodes are emerging. Part of the growing interest in using UAVs in sensor networks comes from the increasing feasibility of these systems due to advances in small and efficient processors, cameras, and wireless networking. The integration of static and mobile sensor nodes enhances capabilities of the overall sensor network. It enables new applications in which fixed sensor nodes achieve an active response capability, while unmanned mobile nodes acquire a spatial vision of the region, allowing larger areas monitoring. Military and civilian applications, such as borderline patrol, search and rescue, area surveillance, communications relaying, and mapping of hostile territory, can take advantage of such kind of sensor networks since these tasks may be repetitive and tedious or dangerous, making them ideal for autonomous unmanned devices [1].

Indeed, applications of sensor network can profit a lot by using different kinds of mobile and sophisticated sensors in addition to static ones which are much simpler and resource constrained. The cooperation between these different kinds of sensors provides advanced functionalities that were not feasible before [2]. The synergy of cooperation between these different nodes opens a vast range of application scenarios. Wireless sensor networks are usually developed to perform specific applications. Sensor nodes have usually a small footprint including only resources necessary to meet specific application requirements. Thus, reconfigurable and customizable architectures are important to make the sensor more flexible. Depending on the application, sensor nodes vary from resource-constrained to high computing power nodes, resulting in a heterogeneous network that can be composed by a variety of sensor nodes.

The main issues in developing such sensor networks are: (i) support for heterogeneous node cooperation; and (ii) sensor nodes customization. The former is related to concerns such as message exchange synchronization, QoS requirements management, task (re-)allocation and network adaptation. The later is related to diversity of node platforms, which may be built upon very distinct hardware components controlled by very different pieces of software.

Considering (i), middleware is a suitable approach to address the mentioned concerns because it can integrate different nodes in order to achieve a common goal in a network by means of promoting a common communication interface and cooperation support. Regarding (ii), customizable architectures can be very useful to build platforms for sensor network nodes. A common base for nodes capability can be provided to all nodes. However, for nodes that need more advanced capabilities, the required resources can be incorporated. Hence, even though all nodes should have the same base capability, some of them could provide additional ones, making the sensor network more powerful by this heterogeneity.

This paper presents a flexible and adaptable infrastructure intended to support heterogeneous sensor network applications. It is based on a proposal of a flexible middleware [3], and on customizable hardware architecture for sensor nodes called FemtoNode [4]. The key idea is to use the customizable platform to deploy different kinds of sensor nodes, from very tiny and resource constrained up to

Edison P. Freitas is with the IDE, Halmstad University – Sweden and the PPGC, UFRGS – Brazil (email: edison.pignaton@hh.se).

Rodrigo S. Allgayer, and Carlos E. Pereira are with the PPGEE, UFRGS – Brazil (e-mail: {allgayer, cpereira}@ece.ufrgs.br).

Marco A. Wehrmeister is with the PPGC, UFRGS – Brazil (email: mawehrmeister@inf.ufrgs.br).

Tony Larsson is with the IDE, Halmstad University – Sweden (e-mail: tony.larsson@hh.se).

more sophisticated ones. Both kinds run a common middleware in order to provide the desired interoperability that will allow the cooperation among these different sensors nodes build upon either the FemtoNode architecture or nodes with different hardware platform, such as SunSPOT [5] or another underlying platform.

The text of the paper is presented as follows: Section II presents some related work in the area. In Section III, the concept of network heterogeneity is highlighted. Section IV presents an overview of the proposed middleware, while Section V presents details about the key issues addressed by the middleware. The FemtoNode customizable hardware architecture is described in Section VI. In the following, a case study is presented in Section VII and, finally, Section VIII draws some concluding remarks and directions of the future work.

II. Related Works

AWARE [6] proposes a middleware whose goal is to provide integration of the information gathered by different type of sensors, including low-end sensor nodes in a wireless sensor network and mobile robots equipped with more sophisticated sensors. Our proposal not only addresses heterogeneous sensors and their coordination, but also concerns like QoS and runtime reflection in order to cope with changes in the environment and in the network, which is missing in [6]. Moreover, in our approach, the autonomous decision taken by the mobile nodes characterizes a higher intelligence degree of our mobile robots if compared with the proposal presented in [6].

In [7], an approach, which uses a sensor network to guide a mobile robot with limited sensor capabilities, is proposed. This work presents aspects of cooperation among different sensors, but it does not address the same problem as proposed in our work. In that approach, the mobile node has limited sensor capabilities and it only gathers data from the sensor network to guide the robot's movement. In our work, mobile nodes are equipped with sophisticated sensors that can provide data, which may be merged with data that comes from the other nodes in order to achieve more refined decisions to guide the movement of the mobile nodes.

III. Network Heterogeneity and Application Scenario Overview

In the context of this text, heterogeneity means that nodes in the network may have different sensing capabilities, computation power, and communication abilities. Additionally, they run on different hardware and operating system platforms. Thus such sensor networks are made up of low- and high-end nodes. Moreover, these sensor nodes may have fixed positions or be able to move, being carried by mobile robots on the ground or UAV platforms, which can also vary from very small, such as [8], up to huge aircraft platforms, like GlobalHawk.

Low-end sensors nodes are those with constrained capabilities, for instance piezoelectric resistive tilt sensors, with limited processing support and communication resource capabilities. High-end sensors nodes comprehend powerful devices like radar, high definition visible light cameras or infrared sensors that are supported by moderate to high computing and communication resources.

The mobility characteristic, as mentioned, is also an important characteristic related to the heterogeneity addressed in this work, which requires special attention. Sensor nodes can be statically placed on the ground or can move themselves on the ground or even fly at some altitude over the target area in which the observed phenomenon is occurring. Fig. 1 graphically represents the idea of the heterogeneity dimensions considered in this work, in which each axe represents one of the considered characteristics described above.

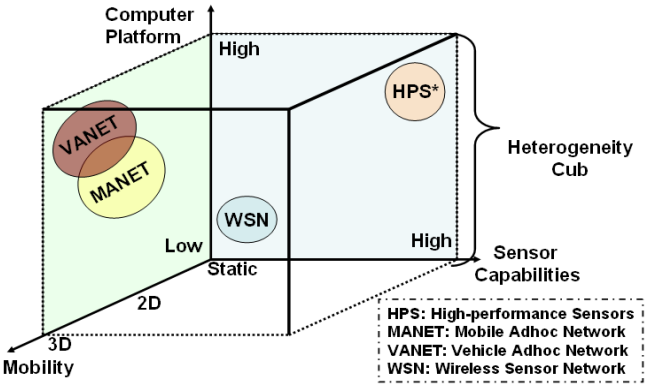

Fig. 1. Heterogeneity Dimensions

The reason for heterogeneity of the sensor nodes is to support applications that deal with very dynamic and changing scenarios, which require different kinds of sensor capabilities. Moreover, those scenarios require also adaptations in the network, in terms of choosing suitable sensors for determined tasks, QoS parameters, among others.

In order to illustrate the above idea, suppose that a network has the mission to provide a certain kind of information during a given period of time. The set of sensors selected in the beginning of a mission may not be the most suitable one during the performance of the whole mission. Thus the network must be able to choose a better alternative, among the set of all available options, in order to accomplish the mission. For example, an area surveillance system may receive the mission to observe if certain kinds of vehicles that are not allowed to pass through the surveyed area do such violation. Ground sensors are set to alarm in the presence of such unauthorized vehicles. Suddenly, an alarm is triggered by one of these sensors. Then, in order to verify the occurrence, UAVs equipped with visible-light cameras are commanded to fly over the area where the ground sensor has issued the alarm. Due to a sudden change in the weather (e.g. the area becomes foggy or cloudy), visible-light cameras become useless. However the mission must still be accomplished. Thus nodes must coordinate to accomplish the mission by sending UAVs equipped with other kinds of sensors that can provide the required information in bad weather conditions, such as infrared cameras.

IV. Middleware Support Overview

One of the key ideas of the proposed work is to drive the use of sophisticated sensors carried by UAVs using data from low-end nodes. UAVs' intelligent behavior uses such data to decide the trajectory direction during the system runtime. Consequently, the proposed middleware must provide the cooperation among the different sensors, requiring that it fits in both the low-end and rich sensor nodes.

As already mentioned the middleware must be light-weight and provide enough customization in order to address the needs of both kinds of sensors. This goal is achieved by using aspect- and component-oriented techniques likewise [9] and [10], and also a mobile multi-agents approach as discussed in [3]. Fig. 2 depicts the overview of the middleware layers, whose description is provided as follows.

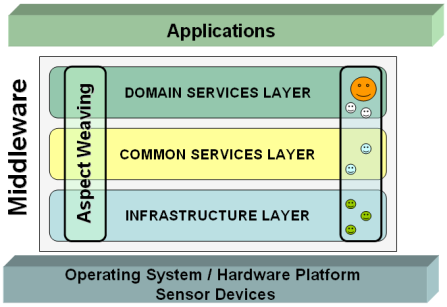

Fig. 2. Overview of the Middleware Layers

The bottom layer is called *Infrastructure Layer*. It is responsible for the interaction with the underlying operating system, and also for the management of node resources, such as available communication and sensing capabilities, remaining energy, etc. This layer is also responsible for resource sharing coordination.

The intermediate layer is called *Common Services Layer*. It provides services that are common to different kinds of applications, such as QoS negotiation, quality of data assurance and data compression. Other concerns are also handled within this layer: deadline expiration alarms; timeouts for data transmissions; number of retries and delivery failure announcements; resource reservation negotiation among applications (based on priorities established by missions and operation conditions); bindings; synchronous/asynchronous concurrent requests.

The top layer is called *Domain-Services Layer*, whose goal is to support domain specific needs, such as data fusion support and specific data semantic support, in order to allow the production of application-related information from raw data processing. Fuzzy classifiers, special kinds of mathematical filters (e.g. Kalman Filter) and functions that can be reused among different applications in the same domain are found in this layer.

"Smile faces" in the Fig. 2 represent autonomous agents that can provide specific services in a certain node at a given moment during system runtime. The *Domain-Services Layer* hosts a special agent (called *planning-agent*), which performs a special task related to the reasoning about the missions that a node is responsible to perform.

Concerns that affect elements in more than one middleware layer, such as security and real-time requirements control, are represented as cross-layer features. These crosscutting concerns are addressed by the aspect-oriented approach presented in [9].

V. Addressing Key Issues of Sensor Networks with the Proposed Middleware

The proposed middleware is based on the publish-subscribe paradigm and inspired in the *Data Distribution Service for Real-time Systems* (DSS) specification, standardized by the OMG [11]. Some nodes publish their capabilities and the offered data, while others subscribe to data, in which they are interested.

Although being inspired in the OMG DSS standard, the proposed middleware does not follow the whole specification. As it is intended to fit in both low-end nodes (based on simple and constrained platforms) and more sophisticated ones (carried by the UAVs), it must not only be lightweight but also provide capabilities for customizations to cope with the needs of the different sensors nodes. Consequently, the middleware uses a minimalist approach, keeping it as simple as possible in each node. Required features are included through adding components or weaving aspects to handle real-time and other non-functional cross-cutting requirements in the minimal middleware. Using aspects to tune the middleware is not focused in this paper, for more information the readers are referred to [9] and [12].

The following subsections present how the proposed middleware will address some of the main infrastructure needs in the mentioned heterogeneous sensor networks, enabling the intelligent behavior of the mobile nodes, and providing the required data with real-time guaranties.

A. Flexibility

The middleware provides full communication control, i.e. it does not use underlying communication control mechanisms available in the nodes' network layer. Instead, it provides its own communication control. This means that all parameters related to communication are controlled by the middleware, which uses only basic connectionless communication services offered by the nodes' network layer. The middleware handles parameters such as number of retries, message priority, memory usage for buffering and timing. This communication control provides more flexibility with direct impact in the decrease of message delivery latency.

B. Dynamicity

Using the publish-subscribe paradigm, when a node gets into the network, its services are announced. Then, interested nodes subscribe for those services. This eliminates the need for a dedicated server node that centralizes all available services in the network. Additionally, it reduces latency in acquiring data because there is no intermediary node between the data producer and the consumer.

C. Minimum Message Exchange

Using the publish-subscribe paradigm by itself already reduces the number of exchanged messages due to elimination of intermediate nodes (such as Brokers). However, bandwidth use for control messages still exists. It can be reduced with the use of smart techniques such as QoS contracts and attaching freshness timestamps to data that avoids unnecessary request for data re-send.

D. Multicast Communication

The middleware will use a multicast communication to reach selected destination nodes. This type of communication affect positively the latency and throughput, as data is sent at the same time to several nodes without unnecessary broadcast and delays, which would occur in unicast communication. A negative-acknowledgement (NACK) strategy (controlled by timeout expiration) is adopted in order to reduce acknowledgement messages in the network. However, very sensible data may require a positive acknowledgement to assure its delivery. Hence, positive acknowledgement is also available in middleware services and can be used when required.

E. Network Resources Usage Control

In order to improve the overall system performance, the control of the use of communication media and also transmission buffers is crucial. The middleware performs this task by taking into account two factors: (i) the priority associated to each application; and (ii) the resource sharing policy adopted in the system. There are three available resource sharing policies: **(a) Fair Sharing:** priorities are not considered and thus all applications have the same right to use the resources in a round-robin scheme, which is organized in a incoming FIFO queue; **(b) Soft Priority Sorted**: the priorities are taken in account, but in a relaxed way. If a higher priority application needs to use a resource already used by a lower priority one, it must wait until the resource be released. Due to its higher priority, it will get access to the resource before other applications, which may be waiting for the resource; **(c) Mandatory Priority**: higher priority applications can preempt lower priority applications in order to access the desired resources. Priority inversion issues are handled by a priority inheritance mechanism.

F. QoS Control

The QoS control is done through a contract between the provider and the requester of data. When a node publishes a data service, it informs the offered QoS. Nodes interested in the published data service should accept the offered QoS and subscribe to the service. However, if a node is interested in the data but does not agree with the offered QoS, it has two alternatives: **(a)** if the application that is requiring the data has a priority lower than any other using the same service, the requester node looks for another data provider; **(b)** if its priority is higher than all other applications, the requester node negotiates with the data provider node, in order to obtain the desired QoS. This renegotiation occurs in spite of undesired consequences that may be implied to the other lower priority applications, which need to look for another data provider if the QoS could not be accepted anymore.

G. Use of Cached Values

The use of cache in both data providers and requesters may avoid unnecessary data communication. When the measurement device gathers a new value, the data provider publishes the new value updating its subscribers. If the data size is large requiring many packets to be transmitted, a differential value can be send instead of the whole data value in order to reduce packets transmission. This option is arranged in advance at the time when the nodes are negotiating the QoS contract.

H. Data Segregation

There are two kinds of data exchanged among nodes in the network: control data and application data. Control data is small and cannot experience latency or unexpected delays to achieve its destination. Thus, control data is segregated from application data by receiving higher priority to be forwarded. Moreover, data from different kinds of applications are handled according to their kind and semantics, e.g. the communication of a video stream has a different handling if compared with the communication of character strings. It can be handled with higher or lower priority, depending on the semantic of the specific data to a given application.

I. Synchronous and Asynchronous Calls

The middleware is intended to support both synchronous and asynchronous calls. Synchronous calls are time bounded in order to avoid unpredictable waiting periods. The waiting time and number of retries are configurable and negotiated during the QoS negotiation. Asynchronous calls are also provided in order to support asynchronous event handling.

VI. FemtoNode – Wireless Sensor Architecture

The architecture of a sensor node aims to efficiently support specific application needs. It requires a dedicated processing module, including a wireless communication interface, which meets both energy and performance requirements, as well as respect footprint constrains. The fact that application requirements or operation/environmental conditions may change during system operation imposes a major challenge [13]. In this context the use of reconfigurable hardware [14] appears as an interesting alternative. Therefore, a customizable sensor node called FemtoNode is proposed. It contains a customizable ASIC and a wireless communication interface, which are configured according to application requirements.

The processor used is the RT-FemtoJava processor [15], a stack-based microcontroller that natively executes Java byte-codes. It implements an execution engine for Java in hardware, through a stack machine that is compatible with the specification of Java Virtual Machine. The customized code application is generated by the Sashimi design

environment [16]. The code includes a VHDL description of the processor core and ROM (programs) and RAM (variables) memories. The Sashimi environment has been extended to incorporate an API that supports concurrent tasks, implementing the RTSJ standard [17].

As the RT-FemtoJava is customizable, its code can be optimized according to the application requirements, reducing the occupied hardware area, and also the power consumption and dissipation. The customizable hardware architecture of the FemtoNode allows the use of sensor-node as either a low- or high- end node. If the application requires higher performance resources to handle more complex data, such as image processing, additional available resources in the FemtoNode architecture can be included. However, if the application is aimed to process simple data, such as those from presence sensors, a reduced set of resources in the architecture is used. This feature benefits the sensor node, because energy consumption is a great concern in wireless sensor networks, due to the nodes' limited energy resource. Besides, reducing the unused resources during the sensor node architecture synthesis allows the implementation in reconfigurable architectures with fewer available logical units, which provides great application portability between these architectures.

In the current implementation, the FemtoNode includes a wireless transceiver of Texas Instruments CC2420 which utilizes the IEEE802.15.4 standard communication protocol targeted to wireless sensor network applications with a low data rate. A module adapter described in VHDL performs the interface with the wireless transceiver. The module uses data and address buses to communicate with the processor, performing the exchange of data and allowing the transceiver parameters configuration.

As the data transfer rate from the wireless transceiver is low, compared with the processor frequency, the wireless communication module implements a buffer to store data, preventing delays to provide the necessary data to the processor. The module uses an interruption system to inform the processor when a reception was made.

To facilitate the use of the wireless communication module by the application developers, a communication API has been developed. The API-Wireless abstracts details of communication means between the sensor nodes, offering a simplified form for the configuration of the data transfer module.

The API-Wireless is used by the middleware's communication services in the *Common Services Layer*, and also by resource management services in the *Infrastructure Layer*, in order to implement end-to-end communication with the desired QoS and reliability control.

VII. CASE STUDY

In order to illustrate the use of the proposed platform infrastructure, i.e. the customizable FemtoNode and the adaptable middleware, an area surveillance application is studied. In this application, low-end sensors nodes are scattered on the ground along a borderline. In case of an unauthorized vehicle crosses the borderline limit, the sensors issue an alarm which will trigger the use of Unmanned Aerial Vehicles (UAVs) equipped with a more sophisticated sensors, such as radars or visible light cameras, in order to perform the recognition of the vehicle. Fig. 3 presents this scenario.

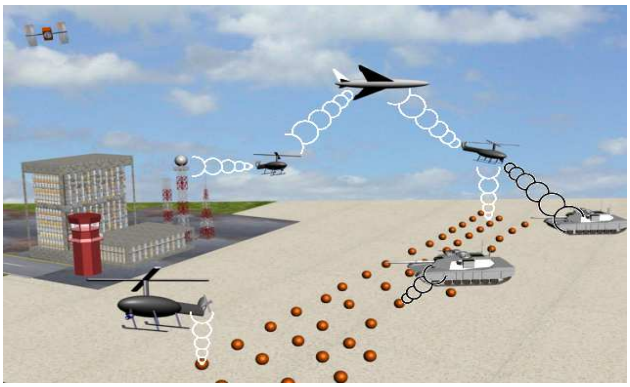

Fig. 3. Area Surveillance Application Scenario

Sensors nodes with different architectures compose the described surveillance system. Each of them includes all necessary resources to meet the requirements of their usage. Thus, based on the application specifications, different customizations of the FemtoNode architecture are performed. The UAVs' architecture is a FemtoNode with a great set of available resources capable to process image data, for instance. On the other hand, the architecture used in the ground nodes is very simple and constrained in terms of available resources, and hence they are only capable of processing simple data likewise those produced by piezoelectric sensors, which inform only if an object over a certain weight threshold passed on it.

The communication of the low-end nodes, among them and with high-end richer nodes, requires a special concern in order to provide the intended cooperation that may exist among all nodes in the network. However, as they have very different capabilities, some mediation (in terms of QoS, data aggregation and data delivery assurance) must be provided. Low-end sensors nodes cannot spend all their energy (re)trying to transmit a message because of "not-received" acknowledgment. Thus a policy to control communication and energy usage is required. In order to fulfill these outlined application needs, the proposed middleware minimizes the message exchanged among nodes, without compromising the information meaning that is transmitted by using the mechanisms outlined in Section V.

In the *Domain Services Layer*, the middleware provides data aggregation, making the data fusion of many presence sensors in an area, in order to provide richer information, such as the direction of a crossing vehicle. The delivery and temporal validity of such information has an expiration time, since after a given time threshold, it is probable that the vehicle may have changed its trajectory, making the previous collected information useless. Therefore, the *Common Services Layer* associates QoS with the messages delivery as well as guarantee assurance mechanism. The *Infrastructure Layer* uses these QoS parameters to manage resources usage.

For instance, it manages the energy consumption and the communication device of the low-end nodes. The use of the public-subscribe approach (outlined in Section V) assists in providing the data distribution among the nodes, which allows nodes to perform the date aggregation required to provide semantic information from the raw data.

In the UAVs, the middleware makes the complementary task, providing data fusion of images and low-end nodes alarm data that matches, in the *Domain Services Layer*. QoS verification of incoming messages is performed by the *Common Services Layer*, which checks if that data can be used by the application, or a request for fresh data must be sent. At this point, a verification of the established QoS among data providers and consumers is tough performed. In case of errors produced by lack of desired accuracy or precision, renegotiations in QoS contracts may occur, following the criteria described in Section V.

Depending on the results from data processing, as described above, the unmanned vehicles carrying sensors may autonomously decide different placements over the surveillance area. Additionally, other factor influence in this decision, e.g. specific needs presented in the current situation and also sensible data which must be sent to the base station or another vehicle. In this case, data segregation and network usage control play crucial role to the efficiency of the system. Important data must be prioritized. Thus, network resources are allocated to data transmission according to the established priorities. Possibly, the specific needs faced by the system in a given situation may require adjustments in the resource usage policy. For instance, a change to a "Mandatory Priority" may take place in order to assure that data about a detected enemy arrives within a given deadline at the base station via relied communication through UAVs.

VIII. Conclusion

This paper has presented the proposal of an infrastructure to provide interoperability support for heterogeneous sensor networks composed by static and mobile sensor nodes. This infrastructure is composed of customizable sensor nodes, and an adaptive middleware to manage different resources available in each customized sensor node. The FemtoNode platform provides a support to develop both low-end and rich nodes. The adaptive middleware provides support for the network heterogeneity, enable adaptations to fulfill requirements that may change during the system run-time, and also promotes the necessary coordination among nodes.

The coordination and cooperation among distinct nodes in the network allow intelligent behavior of the unmanned vehicles, resulting in autonomous decisions about their movement, and also in how they can complement the work performed by other nodes. This intelligent behavior is based on data exchanged by nodes that are aggregated in order to support the autonomous decisions.

The simulations of the ideas presented in this paper are being done. Moreover, before the final implementation, there is some additional work to be done in order to include task (re)allocation features, presented in [12], to support additional aspects of the adaptation capability.

IX. Acknowledgments

E. P. Freitas thanks the Brazilian Army for the given grant to follow the PhD program in Embedded Real-time Systems in Halmstad University in cooperation with UFRGS in Brazil.

References

[1] D.A. Schoenwald. "AUVs: In Space, Air, Water, and on the Ground". IEEE Control systems Magazine, Vol. 20, No. 6, December 2000, pp. 15-18.

[2] D. Culler, D. Estrin, and M. Srivastava. "Overview of sensor networks". IEEE Computer, vol. 37, no. 8, pp. 41–49, 2004.

[3] E. P. Freitas, M. A. Wehrmeister, C. E. Pereira, and T. Larsson. "Reflective middleware for heterogeneous sensor networks". In Proceedings of 7th Workshop on Adaptive and Reflective Middleware (ARM'08), ACM. pp. 49-50, 2008.

[4] R. S. Allgayer, A. Cavalcante, C. E. Pereira. "Wireless Sensor Networks on Heterogeneous Platforms using RTSJ". 10th Real-Time Linux Workshop. Colotlan, Mexico, pp. 37-41, 2008.

[5] Sun Microsystems. 2008. SunSPOT. www.sunspotworld.com

[6] A.T. Erman, L. Hoesel, P. Havinga. "Enabling Mobility in Heterogeneous Wireless Sensor Networks Cooperating with UAVs for Mission-Critical Management". IEEE Wireless Communications. Vol. 15, Issue 6, pp. 38-46, 2008.

[7] M. A. Batalin, M. Hatting, and G. S. Sukhatme. "Mobile Robot Navigation using a Sensor Network". IEEE Intl. Conf. on Robotics and Automation, 2004.

[8] MSB Co. web site. Submeter-scale aircraft, http://spyplanes.com

[9] E. P. Freitas, M. A. Wehrmeister, C. E. Pereira, F. R. Wagner, E. T. Silva Jr., F. C. Carvalho. "DERAF: A High-Level Aspects Framework for Distributed Embedded Real-Time Systems Design". LNCS 4765/2007, pp. 55-74, Springer, 2007.

[10] E. P. Freitas, M. A. Wehrmeister, C. E. Pereira, and T. Larsson. "Using Aspects and Component Concepts to Improve Reuse of Software for Embedded Systems Product Lines", In Proc. of 13th Early Aspects Workshop at SPLC'08, pp.105-112, 2008.

[11] Object Management Group (OMG). Distribution Service for Real-time Systems (DSS) Specification. v. 1.2. January 2007.

[12] E. P. Freitas, A. P. D. Binotto, C. E. Pereira, A. Stork, and T. Larsson. "Dynamic reconfiguration of tasks applied to an UAV system using aspect orientation". IEEE Int. Symp. Parallel and Distributed Processing with Applications, pp.292–300, 2008.

[13] H. Hinkelmann, P. Zipf, M. Glesner. "A Domain-Specific Dynamically Reconfigurable Hardware Platform for Wireless Sensor Networks". Int. Conf. on Field-Programmable Technology, pp. 313-316, 2007.

[14] P. Garcia et al. "An overview of reconfigurable hardware in embedded systems". EURASIP J. Embedded Systems, New York, NY, USA, n.1, p.13-13, 2006.

[15] S.A. Ito, L. Carro, R. P. Jacobi. "Making Java Work for Microcontroller Applications". IEEE Design and Test of Computers. Los Alamitos, v.18, n.5, pp. 100–110, 2001.

[16] Sashimi Manual. 2006. www.inf.ufrgs.br/~lse/sashimi/

[17] M. A. Wehrmeister, C. E. Pereira, L. B. Becker. "Optimizing the generation of object-oriented real-time embedded applications based on the real-time specification for Java". DATE 2006. Belgium, pp. 806-811, 2006.