

Distributed Monitoring Subsystems based on a Bluetooth Implementation

Paulo Bartolomeu, José A. Fonseca, Osvaldo Pacheco
IEETA

Universidade de Aveiro, Aveiro, Portugal
{bartolomeu, jaf, osvaldo}@det.ua.pt

Abstract - The wireless concept goes beyond the reducing of cables and provides, in some standards, mechanisms for Ad-Hoc network construction. Bluetooth as one of these wireless communication standards leads the way in the so-called Personal Area Networks (PAN). Although this wireless protocol was not intended to be used in industrial communications this paper presents a possible implementation on monitoring systems. The monitoring system this paper addresses to is based upon a distributed architecture whose application field is environment monitoring.

Introduction

The monitoring systems present particular constraints in some specific applications, like Coastal Sea Water Quality Monitoring. These constraints are related with the difficult access to the locations to be monitored, large area to be covered, exploitation and maintenance of the system. A common solution for the above problem is to divide the complexity of the overall system in multiple subsystems. These subsystems provide to the upper global system the necessary abstraction to simplify the installation, exploitation and maintenance process. Moreover, the global system is provided with seamlessly access to the data produced within the subsystems. The division of the monitoring system complexity simplifies the addition of extra monitoring spots (see figure1).

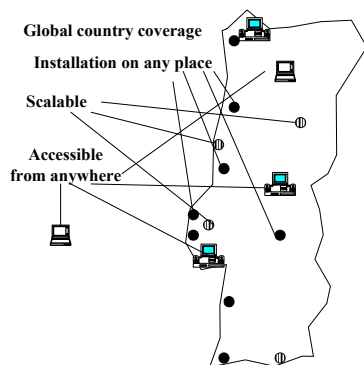


Figure 1 – Global Country Coverage

This paper discusses in detail the architecture of the subsystems developed under the ARMONIO (ARMONIO – “Plug and Play” ARchitecture for a MONItoring System of the Portuguese Ocean) project¹ [1]. This project researches environment monitoring systems with wide geographical

¹ The ARMONIO Project is funded by Fundação das Universidades Portuguesas and the Portuguese Ministry of Defence under the Program “Ocean and its Margins”

area coverage, easy to install and maintain, with seawater quality as one of the target applications. These subsystems are built upon a communication backbone that is implemented with the *Bluetooth* technology.

Overview of the Global ARMONIO Architecture

The overall ARMONIO distributed architecture [2] is based on a publisher subscriber model. This implementation specifies that numerous monitoring subsystems are available and distributed over a large area, in this case the Portuguese Ocean Coast.

Each of these subsystems acquires local information about pH, water temperature, tide level, conductivity, turbidity, air temperature and humidity, wind speed, seismic data, etc. and sends it to the subscribed System Users (SUs). The SUs receive this information, build a measurement database and publish the received information on the web allowing any user with Internet connection to access them. At this moment there are no requisites about the security of the information on the web. An authentication mechanism might be implemented to deny access to non-registered users. As explained, the information produced by the overall monitoring system will be presented in a web page and thus will be available for everyone. Figure 2 shows the developed implementation. As it can be seen there will be more than one SU allowing the required redundancy of the system to possible failures of one of the SUs.

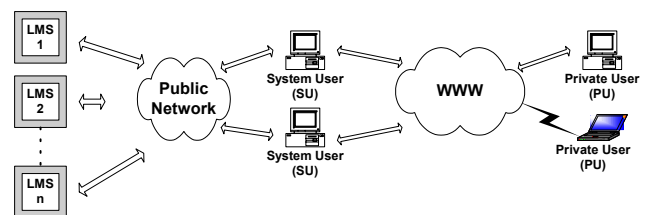


Figure 2 – Overall ARMONIO Architecture

Local Monitoring System (LMS) Architecture

The main objectives of the LMS architecture are the following: ease of installation and maintenance of sensor nodes, installation and maintenance does not require a specialized technician, robustness and tolerance to failures, high level of autonomy and no cables. These objectives lead to a wireless distributed architecture based on the popular producer consumer model.

The producer-consumer model (as the name implies) is based in two types of nodes: producers and consumers. The producer nodes provide the consumer nodes with information about a local measurement. In this sense, a producer-acquired measurement is available for consumption by a consumer node. The producers considered in this LMS architecture have an embedded sensor (or more) and have physical interfaces for connection with Intelligent Sensors (ISs). The ISs are usually assembled with built-in interfaces like RS232, NMEA, RS485, etc.; moreover, the producers have physical interfaces for these standards.

The consumer nodes on the other hand, are divided in two types: sinks and Debugging Systems (DS). The sink nodes are consumer hardware alike nodes but with interfaces for commercial communications systems. Their function is to aggregate information received from the producers (sensors) and convey it to the outside of the LMS using a predefined format. The Debugging System (DS) is a PC computer (Laptop preferably for mobility) with a communication interface similar to the LMS's communication backbone. It has access to the LMS's produced information by being configured as a regular consumer. This way, it is possible to evaluate *in-situ* the correct operation of the monitoring subsystem.

Bluetooth as the LMS Communications Backbone Solution

The communications solution for implementing a producer consumer network could be very difficult because of the large number of technologies that could be used to perform this operation. If there were no constraints on using wired networks for implementing the LMS communications backbone a possible solution could be the Controller Area Network (CAN) [3] fieldbus. This solution might be a good choice given the high noise immunity of the fieldbus, the perfect adaptation to the producer consumer model and the low cost of its use. However, if the network must be wireless to comply with the required overall system characteristics, the chosen technology must provide Ad-Hoc network construction and its operation mode must be adaptable to provide "Plug and Play" (PnP) connections and producer consumer model operation. *Bluetooth* fits on these requisites.

The *Bluetooth* technology is in fact one of the most widely used wireless technology for low rate communications and its current growth predicts that in a close future it will be extremely cheap, around 5€ per module. This, and other relevant characteristics like low power consumption and robustness [4][5], makes *Bluetooth* the ideal solution for our purposes. Although the application area of this paper does not require a high transmission rate, *Bluetooth* can enable a node with communication rates up to asymmetrical 721kbit/s [6]. This rate gives a wide margin for use in different monitoring application areas and, consequently, to use the developed nodes on them without transmission rate related constraints.

LMS Bluetooth Architecture

The *Bluetooth* LMS architecture is based, as stated before, on the producer consumer model. Each LMS contains at

least one producer and one consumer, corresponding to only one physical parameter to be measured. The *Bluetooth* solution allows a maximum of seven consumers per producer because a piconet can only allow a maximum of seven active slaves plus the master. The producers within the LMS will be configured as masters and the consumers will be configured as slaves. Each producer (master) will create a piconet with the consumers (slaves) in the neighbourhood. There will be several (possibly, accordingly with the number of physical measures to be made) piconets within the LMS. Each producer forms its own piconet and consumers might be configured as slaves in several contiguous piconets. There will be no scatternet formation since the piconets are intended and will be completely independent from each other. A producer is associated with a sensor as stated before and the related information will be transmitted to the sink nodes within the LMS. These sink nodes are global consumers (within the LMS) and are configured as slaves in the piconet of the producers. Debugging systems will be also configured as slaves in the LMS. This occurs because the main objective of those is to provide local LMS monitoring to debug possible errors of the producer nodes. Figure 3 shows these issues.

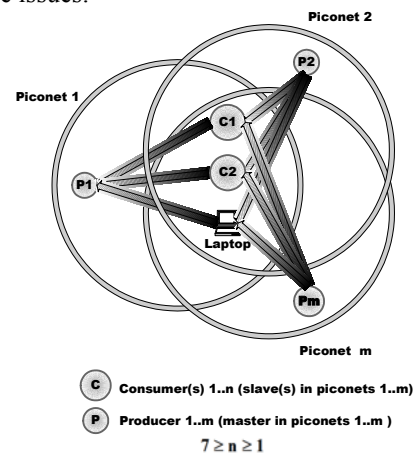


Figure 3 – Bluetooth LMS Architecture

As it can be seen, the producers will generate information related with the local variables that they are measuring and will transmit this information to all consumers in the neighbourhood (LMS). The consumers are common to all piconets in the LMS and thus receive information from all producers (masters). The sink nodes (consumers), after receiving the information from producers and in the adequate instants, will pack it and transmit it to the outside of the LMS. This procedure is possible due to the inner architectures of the sink nodes that have an interface to a commercial communications device that allows the transmission of the information to the outside of the LMS. The information produced inside the LMS is tagged with time and position labels so that the SUs can distinguish it from each other and remotely check for errors of the information producers. The tagging operation is performed in the consumer nodes with an embedded GPS that provides global UTC time information. However since the producing instant might be different from the tagging instant (due to delays in the communication backbone - result of inter-

piconet collisions) the tagged producing instants might not be accurate. Because the number of sensors (and consequently piconets) is expected to be low (a maximum of 7 sensors in each LMS), the inter-piconet interference is predicted to be negligible.

LMS Bluetooth Implementation

The LMS implementation is based on producer and consumer nodes as referred through the text. The producer and consumer nodes can have a different operation but are alike with respect to the software and hardware inner architecture. In terms of software the distinction of the nodes is made configuring the Inquiry Scan and Page Scan of the *Bluetooth* modules that integrate the LMS node. These parameters allow the software designer to force the desired operation of the module, master or slave. An example can be when these parameters are disabled, that is, the *Bluetooth* module will not reply to those packets and therefore will never be configured as slave in a piconet. Although the LMS *Bluetooth* communications are based on the producer consumer model, they are not strictly implemented as in the model [7] in this solution because the model states that every node in the network receives the data produced by other distinct node and then consumes it accordingly to a proper consumer criteria. In this case the producers (masters) will never receive data from other producers, so this producer consumer model is only an approximated version of the usual model.

The developed software is divided in two software modules. The first is the *Bluetooth* Stack Module (BSM) and the second is the Hardware Driver Module (HDM). The BSM abstracts from the software designer the inner complexity of the *Bluetooth* communications Protocol and provides a user-friendly interface for *Bluetooth* communication. The HDM does the same but for the sensor hardware and the external devices that can be plugged in the nodes (like GSM and GPS modules).

The hardware of the producers and the consumers is identical and is based on a Core Hardware Node (CHN). The CHN is a small stackable board that contains four different modules: Microcontroller Unit (MCU), Communications Unit (Bluetooth Module), Power Supply Unit and finally an Interface Unit. The power supply unit is designed to provide status on the batteries remaining power and control of the CHN and Bluetooth module supply. The interface unit provides connection between the CHN and external devices such as Intelligent Sensors, GSM modules or GPS modules.

The DS is a special case because it is a consumer but it doesn't use the same platform to process data. The DS is basically a PC (laptop preferably) with a *Bluetooth* interface that allows slave operation. This PC runs an application that processes the received LMS data to provide a user-friendly debug interface to the data produced locally.

Software

The main objective of the software developed for the nodes is to provide PnP operation in the LMS, that is, when the application requires an additional physical measure, for example, it is only necessary the addition of an extra

producer to the LMS and, after this operation, the information of the newly added producer will be available to the sink node. Although this seems easy to understand, it is hard to implement because it involves several consumer nodes in the neighborhood. This can only be achieved if the consumer nodes within the LMS are ready to reply to inquiries from the producers. Another example is when the maintenance engineer detects remotely a fail on the LMS and comes to the local with a DS to fix the problem. He (or she) should only be within the *Bluetooth* range of the LMS with his (her) DS to start receiving the locally produced data. This means that all producers have to recognize a new consumer node arrived in the neighborhood and start the transmission of their information to it. This method allows the debug of the LMS's operation and sub consequently the identification of malfunctions in the different producer's operation.

These constraints lead to the solution of building an application for the nodes that, with minimum effort, produces adequate operation for the different possible configurations that a node can operate with. Following, the operation mode is defined by a simple pre programming that "informs" the node of its future functioning. This pre programming is not definitive and the node can be reused in some other distinct application.

Bluetooth Stack Module (BSM)

The *Bluetooth* Stack Module implements the group-oriented communication. This group-oriented communications is built upon the Logic Link Control and Adaptation Protocol (L2CAP) layer [8] that provides the primitives for this operation mode. The main advantage of working with the developed software stack is that there is no need for inner knowledge of the *Bluetooth* protocol and thus the consequent abstraction of its components to the application developer. Figure 4 shows the interface between the application and the BSM. As it can be seen, the BSM implements the L2CAP and Host Controller Interface (HCI) layers. These layers allow the group operation as well as provide the necessary mechanisms to implement it. The BSM is then the interface for application that handles with the operation of the *Bluetooth* module.

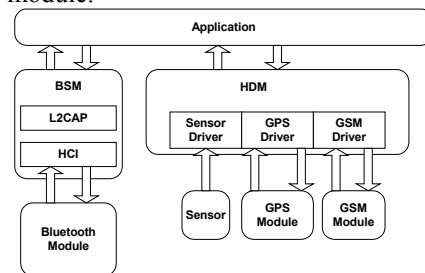


Figure 4 – BSM and HDM modules

The group operation that the BSM implement is based upon the primitives identified in the L2CAP Layer. These primitives are: Group Create, Group Add Member, Group Remove Member, Get Group Membership and Group Close. The group construction is based on the responses to the producer inquiry packet.

Hardware Driver Module (HDM)

The main function of the hardware driver module is to provide abstraction to the application about the interface complexity of the sensor or the externally connected devices. Figure 4 also shows the possible hardware to be driven by this module. The module provides simple primitives of configuration and operation of the referred hardware. In the previous section we stated that, in some particular cases, the consumer would process tasks. These tasks are related with the local acquisition of time/position information and future processing of them to label the built frames in the sink. These frames will after be sent to the registered SUs.

System Operation

When a producer “wakes-up” it starts by programming the default parameters related to its operation and then tries to find consumer nodes within its *Bluetooth* operation range.

After the reply of the consumer nodes within a specified timeout, the producer builds an information distribution group with the data received in those replies. When the group is formed, the data locally produced will be sent to the group and the consumers will have access to it. After finishing the transmissions, the producer node will destroy the previously formed group (giving information to the consumers that no information will be present by the producer node) and shuts down for a pre-determined amount of time that is its production period. This operation allows significant power savings due to the substantial difference between the awake interval and the measurement period for this application (Environmental Monitoring).

On the other side, the consumer nodes will always be ON. When a consumer node is plugged into the LMS, it starts (like the producer) by programming its default operation parameters. After acknowledging its function (consumer) it will periodically check for connection attempts from the producers. When a connection request is received in, the consumer responds with the necessary information for the producer to connect to it and then listens periodically for data remotely sent from that producer. A dynamic database saves the received data from the different active producers with the necessary information to distinguish them. This procedure is based on the idea that each producer has a distinct measure to provide to the sink.

When a producer sleeps, it sends a Connection Destroy packet that informs the consumer that it will not send any further data until the next local measure occurs. This packet may contain information about the amount of time that the producer will be in sleep state. This information may be used to optimise the operation mode of the consumer that knows that it will not receive any further data from that producer within that specified period of time.

The regular operation of the consumer node is assumed to be the post-processing and subsequent handling of the received data. When some event occurs (receive data buffer is full or optimum amount of data to be routed outside the LMS is received are examples of possible events) the consumer shall pack the already received data accordingly to a pre-defined format and subsequently transmit that packet to the registered SUs. These communications are achieved

through SMS messages that contain the locally produced data under a specified format that guaranties error correction (messages with CRC). The fault tolerance mechanism used in the LMSs is the consumer redundancy, that is, there are multiple consumers within the LMS that send the same information to the global monitoring system. The elimination of duplicates is made at the SU level with information contained in the received SMSs.

Conclusions and work in progress

This paper presents an adaptation of the *Bluetooth* technology for use in distributed monitoring systems with high level of complexity, difficult installation and maintenance, and a large monitoring area. The use of this technology envisages the reduction of installation and maintenance problems that might arise from the target application area (sea water quality monitoring along the Portuguese coast). Although this is a specific application area, the architecture seems promising for use in monitoring systems with different characteristics. A relevant characteristic is that it is predictable that the architecture will work continuously for a period of about six months without the need for maintenance. This autonomy results from the choice of low power components and devices to build the LMS as well of the overall system architecture.

Currently, a prototype of the LMS is being developed in order to have a fully functional demonstrator ready in July. Future work includes evaluation of the architecture performance and the analysis of the use of Plug and Play smart sensors that follow the IEEE P1451.4 standard.

References

- [1] <http://www.ieeta.pt/armonio>
- [2] Fonseca J.A., Bartolomeu P.J., Pacheco O.R., Duarte P., Macedo A., ARMONIO – “Plug and Play” ARchitecture for a MONItoring System of the Portuguese Ocean, *in Proceedings of RTLIA'03 – “2nd Intl. Workshop on Real Time LANs in the Internet Age”*, Porto, July 2003.
- [3] Bosch, “*CAN Specification Version 2.0 - Technical Report*”, Bosch GmbH, Stuttgart, Germany, 1991.
- [4] Jaap C. Haartsen, “The Bluetooth radio system”, *Personal Communications*, IEEE, Volume: 7 Issue: 1, Page(s): 28–36, Feb. 2000
- [5] Sairam K.V.S.S.S., Gunasekaran N., Reddy S. Rama, “Bluetooth in Wireless communication”, *IEEE Communications Magazine*, Page(s): 90–96, June 2002.
- [6] James Kardach, “*Bluetooth Architecture Overview*”, Intel Corporation, 1998
- [7] P. Verissimo, L. Rodrigues, *Distributed systems for systems architects*, Kluwer Academic Publishers, 2001.
- [8] Bluetooth SIG, *Specification of the Bluetooth System v1.1*, Volume 1, 2001.