10.2478/v10048-008-0017-8

Application of Wireless Sensor Networks to Automobiles

Jorge Tavares, Fernando J. Velez, João M. Ferro

Instituto de Telecomunicações, DEM - Universidade da Beira Interior, Calçada Fonte do Lameiro, 6201-001 Covilhã, Portugal
e-mail: jorgemstavares@gmail.com, fjv@ubi.pt, joaomferro@gmail.com

Some applications of Wireless Sensor Networks (WSNs) to the automobile are identified, and the use of Crossbow MICAz motes operating at 2.4 GHz is considered together with TinyOS support. These WSNs are conceived in order to measure, process and supply to the user diverse types of information during an automobile journey. Examples are acceleration and fuel consumption, identification of incorrect tire pressure, verification of illumination, and evaluation of the vital signals of the driver. A brief survey on WSNs concepts is presented, as well as the way the wireless sensor network itself was developed. Calibration curves were produced which allowed for obtaining luminous intensity and temperature values in the appropriate units. Aspects of the definition of the architecture and the choice/implementation of the protocols are identified. Security aspects are also addressed.

Keywords: wireless sensor networks, applications, MICAz motes, automobile, architectures, protocols

1. INTRODUCTION

NOWADAYS, the need to collect, interpret and act on real-time data gains increasing interest. However, to collect data using typical wired sensor networks has always been expensive, owing to installation and maintenance costs, and is limited in its range.

Although past wireless measurement solutions have been elusive, the spreading of the use of wireless sensor networks (WSNs) is in fast development. WSN is a term used to describe an emerging class of embedded communication products that provide redundant, fault-tolerant wireless connections between sensors, actuators and controllers. The large amount of research projects in this area allows for the existence of better tiny hardware devices with reduced cost/size, and improvements in software performance. WSNs are typically formed by groups of several sensor nodes, the so-called motes, whose individual constitution is based on actually combining sensor radios and CPUs into an effective robust, secure and flexible network, with low power consumption and advanced communication and computation capabilities, one or more sensors, a communication device (typically a radio), a microcontroller (with memory) and a power supply (battery). Its applications include industry, atmosphere monitoring, and defence, among others. Besides instrumentation concepts, WSNs involve aspects of wireless communications, networks architectures, and protocols.

Due to technological innovations in the area of wireless communications, digital electronics, and personal micro-electromechanical systems, a revolution is occurring in the area of measurement with remote wireless sensors [1]. In particular, WSNs are characterised by a high amount of sensor nodes with multi-hop communication capabilities. These tiny sensors can be spread inside the environment to be monitored or close to it, with positions that are not pre-determined. Indeed, they are set randomly as wireless sensors can be dropped onto places with difficult access from helicopters or airplanes [1]. These motes exchange messages among each other in order to efficiently monitor an environment/process, and operate while balancing the trade-off between low energy consumption and the need to fulfil the assigned tasks.

The application of wireless sensor networks to the automobile constitutes a challenge to be faced in this endeavour; we conceived a wireless sensor system capable to collect, process and supply several types of technical information (to the user) during an automobile journey. The examples are acceleration and fuel consumption, identification of wrong tires pressure value, acknowledgment of illumination failures (turn lights, brake lights, front lights, and register plate lights), and determination of the vital signals of the driver. We chose Crossbow MICAz sensors operating at 2.4GHz (IEEE 802.15.4), and supported by TinyOS. The concepts and the wireless sensor network itself (transmitter/receiver/ interface board) are explained, and aspects of the architecture, and of the implementation of the protocols itself are established. Security aspects are also addressed, and the power consumption issues are discussed.

Section 2 discusses some characteristics of WSNs and their applications to automobile industry, security services, military, environment, and medicine. In Section 3, routing protocols are briefly discussed, security aspects and imperfections are presented, and energy consumption issues are addressed. In Section 4 the use of TinyDB is discussed. Section 5 presents the various components, e.g., flow, tyre pressure, light, acceleration, temperature, heart beat frequency, and blood pressure sensors. Relevant results are presented, e.g., for luminous intensity, temperature and arterial pressure, where the discussion includes the production of the calibration curves. Finally, conclusions are presented in Section 6.

2. CHARACTERISTICS AND APPLICATIONS

At the University of California, Berkeley, an open code operative system was developed for WSNs with the support of Intel, called TinyOS, which demands little memory (about 8 kb). As an example of the application of TinyOS, it is ~~a~~ worth noting that it is already being used by Crossbow to trace automobile parts in industrial environment [1]. Some examples of possible applications of WSNs follow:

Automobile applications – a modern automobile has about 8km of cables to connect hundreds of sensors [1]. WSNs allow not only to reduce the volume and weight required by the cabling, but also the deployment of sensors with more freedom.

Safety applications - one of the applications sought for domestic use falls in the area of safety. The distribution of temperature and movement sensors along the house allows the detection of fires and intrusions. Besides, it can supervise and control children's and elderly people’s movements within the house.

Industrial applications - WSNs can be designed and implemented by taking the specificities of each type of industry into account, and several applications can be identified in this framework. WSNs are capable to monitor the quality of the air, and the temperature of a building or on an oven. Besides, it controls the produced goods, the complex machinery set, and the conditions of the production system of a certain factory or a group of factories.

Military applications – nowadays, military applications of wireless sensors are quite common, mainly because it is difficult to deploy a communication infrastructure in the theatre of operation, e.g., in a battlefield. The installation of a centralised infrastructure, apart from being time consuming, would become a vulnerable network solution (because the destruction of the central node would totally put an end to the entire network).

Medical applications – WSNs are used to form a so called Body Area Network (BAN), which consists of several sensors placed close to the human body measuring signals such as heart beat rate or breathe rate.

3. ROUTING, SECURITY AND ENERGY CONSUMPTION

There are several protocols in the context of TinyOS. As an example, the TinyOS Beaconing is a protocol used in Mica Motes (wireless sensor nodes) at the University of Berkeley, and operates within networks with restricted hardware [1]. The protocol periodically builds the Minimum Spanning Tree starting from the Base Station. The Base Station propagates the message (beacon call) that is spread through the network with the objective of creating the routing tree. As it is a simple and general protocol, its performance is lower than the one of protocols developed for specific applications.

In terms of security, spoofing is the attempt to change or repeat the direction of information by a malicious node [1]. As a consequence, the information may enter into a loop, and never arrive to the sink, because it will continuously be routed through the same set of nodes, causing energy wasting (to send and to receive data).

TinySec is the TinyOS cryptography layer, and offers authenticity, integrity and confidence [1]. Actually, TinySec just offers the cryptography of symmetrical keys, and the secret key is distributed while programming sensor nodes. The algorithm to produce messages can be any symmetrical one (that can be implemented in TinyOS).

In TinySec there are three operation modes: SendMsgCRC, SendMsg, SendMsgEncryptAndAuth; it can operate both in TOSSIM (a simulator for TinyOS networks), in the MICA, and in the MICA2 platforms but, until the moment the work was performed, it was not yet prepared to work with MICAz. An architecture typically used in WSNs is the Mica Motes of Crossbow one [1].

Among its components, the one with the highest energy consumption is the flash memory. However, in spite of the high consumption of energy during the writing and reading cycles, the use of the memory flash is not essential for link maintenance in the constitution of a WSN. Hence, from the essential hardware components needed, the transmitter is the largest power consumer. Transmitting is costly and receiving can be as costly as transmitting. Even in the idle mode the transceiver wastes energy and so it must be put into sleep mode for as much time as possible. Even when it is in the sleep or idle modes, the transmitter wastes energy. This is done by an appropriate power management scheme.

4. TINYDB

To use TinyDB, it is first important to create an executable file by executing the command make in the directory of the program:

```
cd /opt/tinyos-1.x/tools/java/net/tinyos /tinydb;make
```

This command works in any version of TinyOS. After successfully running it, it is fundamental to load the executable of the application TinyDBApp to each of the MICAz motes. For this purpose, the interface board should be “off”, while MICAz has to be “on”. The executable is created with the following command:

```
cd opt/tinyos-1.x/apps/TinyDBApp; make micaz
```

followed by a connection from the MICAz to the interface board, and the execution of the following command:

```
MIB510=/dev/ttyS0 make micaz install
```

However, it does not work properly in version 1.0 of TinyOS. In this case, the ProgramMote that comes with MoteConfig of Crossbow should be used to load the executable main.exe into the directory TinyDBApp (inside build\micaz). The subsequent commands are the following:

```
    export CLASSPATH=$CLASSPATH:
/opt/tinyos1.x/tools/ java/jars
    cd /opt/tinyos-1.x/tools/java /
    java net.tinyos.tinydb.TinyDBMain
```

Afterwards, one should wait until the graphic interface of TinyDB shows up. Then, one is able to select which of the registers one intends to visualize by pressing on “>> >”, and by choosing “Send Query”. After some seconds needed for network configuration purposes and for the initialisation of the measurement process, the chart will be visualized and it is possible to monitor those values.

In our work, TinyDB was used for reading data from external sensors attached to the sensorial board, e.g., from the flow meter, the tire pressure sensor, and the blood pressure sensor. In these cases we used the Digi-Key H2163-ND connector. By consulting the datasheet of the sensorial board, it is possible to verify how internal sensors are connected, as well as to know how to read some information. In this work, the option was to connect the external sensors to terminals 42 and 51 (Ground), corresponding to the light sensor in the sensorial board.

5. HARDWARE COMPONENTS AND RESULTS

For the transmission and processing of the signals we have selected the MICAz ZigBee (MPR2400) from Crossbow. This module uses an ATMega 128L to collect the data from the network, and to program MICAz motes. The MIB510 interface board was connected to a computer by using the serial port, and the MICAz MTS310 sensor node was then used to connect the sensors that are not on the board via the Digi-Key H2163-ND connector, Fig.1.

Flow sensor - We chose a flow sensor with the technical, economical, and weight specifications closest to the requirements. The choice was the RS 508-270, which allows to measure a wide range of flows, from 0.05 to 10 [l/min].

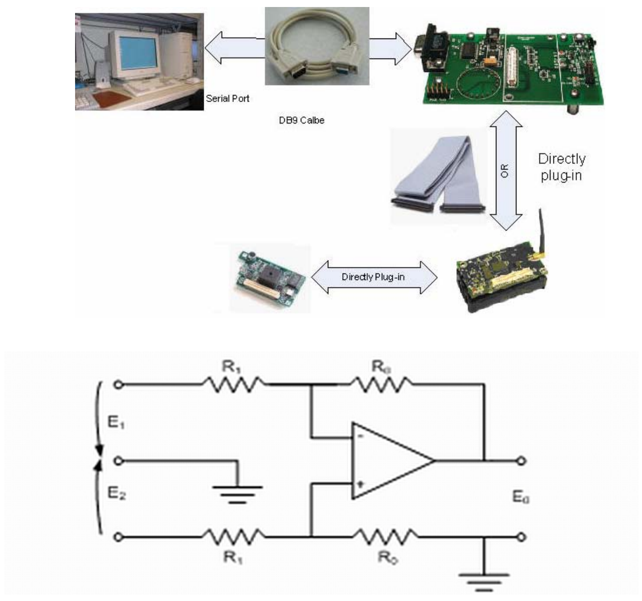

Fig.1 MIB510 Interface board and related components, and differential amplifier for the pressure sensor

Its sensibility is not high enough to measure small values of fuel flow, as is in those involved in the low fuel consumption cars from Shell ECO-MARATHON, the scenario we used as an initial motivation to our work, i.e., ~0.0054 l/min. However, they are useful in the context of competition cars with higher fuel consumption flows. As a possible scenario, two or three cars of the same team, and a central node in the team box were considered; each car can serve as a relay for the other nodes. As the output of the flow sensor is proportional to frequency, we used the frequency to voltage converter LM2917N whose output varies 1 V for each variation of 67Hz in frequency. We had to select the nodeId related to the sensor of light because of TinyDB functioning, which uses the terminal from the sensor of light for data acquisition when MICAz is connected to the MTS310 sensorial board.

Tire pressure sensor - The pressure sensor included in tire pressure reader Sensor Monza 2 in 1, is the one that was used. As the sensor does not have any identification, it was not possible to find its datasheet. However, the sensor has four terminals, which indicates that its internal circuit should be a Wheatstone bridge. To have access to the output of this sensor it is thus necessary to measure the voltage in each of its four terminals, in two different cases: sensor reading the ambient pressure, and sensor under pressure (close to the limit is the ideal). We had to remove the pressure sensor from the Monza 2 in 1 kit in order to connect it to the MICAz. To monitor the pressure value we used TinyDB, and the sensor node executes the TinyDBApp program available in the directory /opt/tinyos-1.x/apps/TinyDBApp. Fig.1 presents the scheme of the amplifier used between the MICAz and the sensor. E0 is the output voltage amplifier circuit, and is connected to the input of MICAz; E1 is a reference voltage of 0.2V, which is used to calibrate the sensor; E2 is the voltage at the output of the pressure sensor; R1=1500Ω, and R0=10Ω.

Light Sensor - To verify the state of the automobile lights, a light sensor has been used close to each lamp. In our case, we opted for the light sensor already included into the MTS310 sensorial board, Fig.2. To collect data from this sensor, the mote is programmed with OscilloscopeRF application, while being placed onto the zone to be monitored.

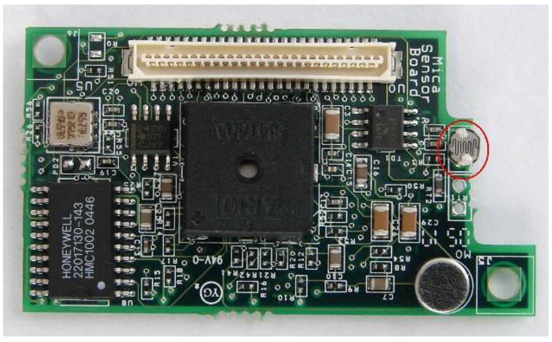

Fig.2 Placement of the light sensor on the MTS310

Another node running the TosBase application is placed on the interface board connected to the computer. The programming board is switched off while the nodes are switched on. To generate the executable files one has to type:

```
cd /opt/tinyos-1.x/apps/OscilloscopeRF; make micaz
```

To send the generated files to the motes, the MoteConfig executable file can be used, or alternatively the following command can be used:

```
MIB510=/dev/ttyS0 make micaz install
```

To compile the Java directory one has to type:

```
cd /opt/tinyos-1.x/tools/java;make
```

For every new session we need to select the frequency of the communication through the serial port connection (which is 57600Hz for MICAz and MICA2, and 19200Hz for MICA and MICA2DOT) by typing the following command line (example for 57600Hz)

```
export MOTECOM=serial@COM1:57600
```

The received data can be easily visualised by typing:

```
java net.tinyos.oscope.oscilloscope
```

Then, we can watch live the variation of the light brightness on the photovoltaic sensor of the mote, Fig.3. This allows to know what the state of lamp is. The experimental results presented in Table 1 allowed for obtaining the calibration curve presented in Fig.4. The raw value 545 corresponds to a luminous intensity of 0 lux. The curve is piecewise and presents two linear regions with different slopes. One linear region is from 0 to 50 lux whereas for the other the luminous intensity varies from approximately 180 lux to 450 lux.

Acceleration sensor - The acceleration sensor is incorporated into the MTS310 board of sensors. The signal acquisition is performed by using TinyDB, as the TinyDBApp program is very slow. In the future, this aspect should be improved.

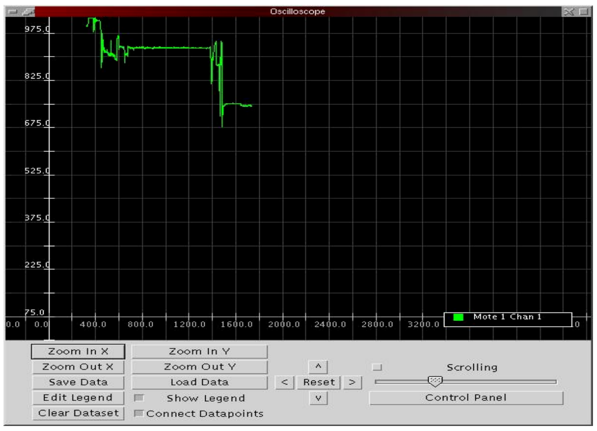

Fig.3 Graphical representation of the luminous intensity extracted from the mote sensor by using OscilloscopeRF

Temperature sensor - The temperature sensor should be installed in contact with the driver's body in order to monitor his/her temperature. As the sensorial board already has a built-in temperature sensor, it is considered that this sensor, whose data is read by TinyDB, serves our initial objectives.

Table 1 Experimental results for the calibration of the light sensor

Node value [raw units]	Luminous intensity [lux]
545	1.14
605	15.3
730	45.8
730	45.8
805	85.6
820	100.0
845	125.0
890	182.0
910	293.0
925	384.0
935	445.0
982	633.0

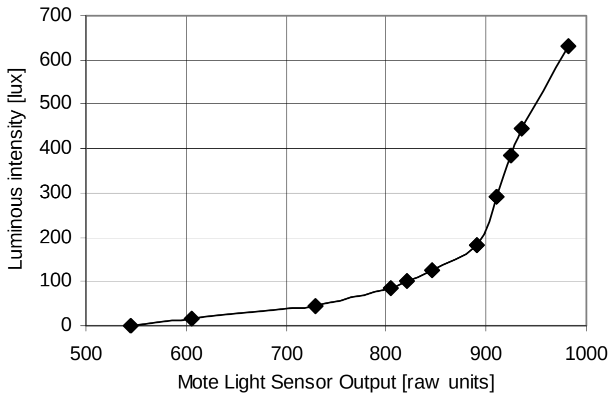

Fig.4 Calibration curve for the mote light sensor

An example of the variation of the temperature with time is presented in Fig.5. The corresponding calibration curve obtained from the measurements presented in Table 2 is sketched in Fig.6. We moved one of the temperature sensors within the laboratory and also in the corridor whilst keeping the other one static. For calibration purposes, we collected the temperature variation near and far from a heating system in a multimeter while comparing the instantaneous values with the curve obtained with TinyDBApp. By using the calibration curve we concluded that the temperature measured in mote 2 is approximately 31ºC.

Heart frequency and arterial pressure - The equipment selected to measure the heart frequency and the diastolic and systolic blood pressure is a prototype for testing and monitoring biomedical signals, developed in the Department of Computer Science of University of Beira Interior by Prof. Pedro Araújo and his student Pedro Ussman, Fig.7.

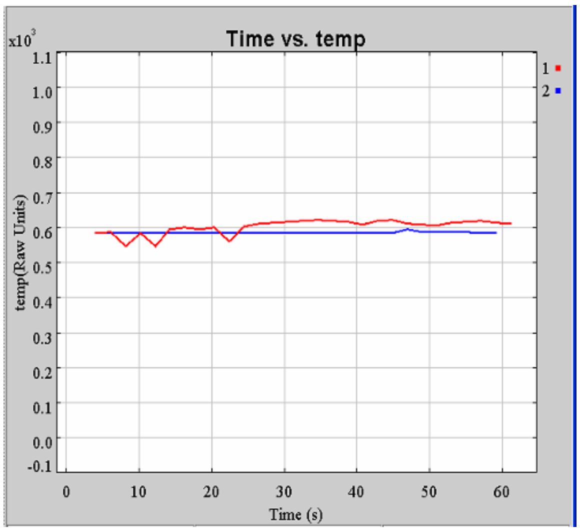

Fig.5 Measurement of the temperature with the TinyDBApp application installed in two MICAz sensor motes

Table 2 - Experiences for the calibration of the temperature sensor

Node value [raw units]	Temperature [ºC]
412	20
467	22
577	31
665	34

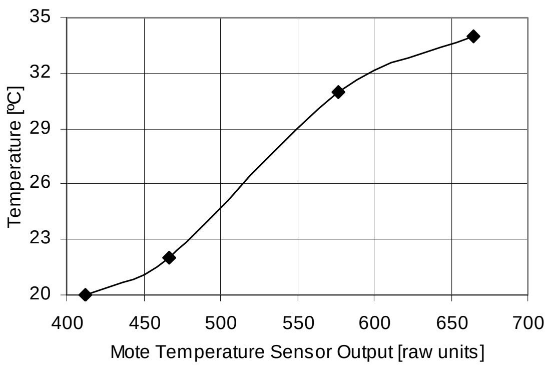

Fig.6 Calibration curve for the mote temperature sensor

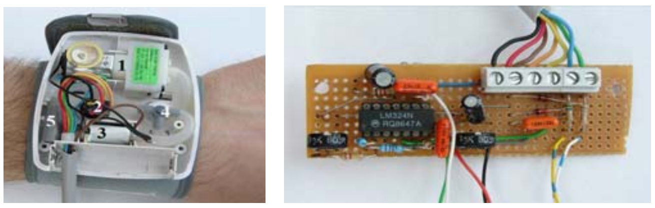

Fig.7 Arterial pressure monitoring prototype placed on the human pulse, and circuit for signal amplification

This prototype uses an inner tube placed under pressure. During its emptying process the pressure is measured through a sensor. This signal is amplified, Fig.7, and sent to the computer that runs the software to interpret the signal while extracting the blood pressure and the heart beating frequency values.

The monitor of arterial pressure is formed by a motor that insufflates air into the inner tube, by a Metrodyn MPS-1001 pressure sensor, by an inductor, and by exhaust valves for the inner tube (fast and slow ones). The connections from the circuit amplifier to the MICAz are the following:

- White wire - analogue signal of the absolute pressure, - connected to the terminal 42 of the MICAz;
- Green wire - analogue output signal of the relative pressure, - connected to the terminal 42 of another MICAz;
- Yellow/blue wire - digital output signal to turn the motor that inflates the arm brace on/off, - connected directly to a relay that is commanded by the terminal 9 of MICAz;
- White/blue wire - digital output signal that closes/opens the valve, - connects to the relay that is commanded by the terminal 10 of MICAz.

These processes were controlled by running the *SimpleLedCmd* application, followed by the execution of the following MICAz command:

```
java net.tinyos.tools.BcastInject <parameter>
```

where the parameter can have the following values: *led_on* - closes the valve; *led_off* - opens the valve; *led_A_on* - activates the pump; *led_A_off* - turns the pump off.

6. CONCLUSIONS

This work addressed the conception of a WSN capable of measuring, processing and supplying diverse types of information to the user during an automobile journey. The examples are acceleration and fuel consumption, identification of incorrect tire pressure, failures of illumination, and evaluation of the vital signals of the driver. Beside a survey on the concepts, the wireless sensor network itself (transmitter/ receiver/control board) was configured, and aspects of the architecture and protocols were addressed. By using the calibration curves for the light and temperature sensors, precise experimental values were extracted. Security aspects were also identified, and the difficulties and solutions were discussed. Competition cars in a controlled environment constitute a suitable scenario for experimental work. Besides, the evolutions in this field promise a lot in the automobile industry, e.g., for cooperation among cars for road safety purposes.

ACKNOWLEDGMENT

This work was partially funded by MobileMAN (an internal project from Instituto de Telecomunicações/LA), by IST-

UNITE, by CROSSNET (a POSC project with FEDER funding), by the PhD FCT (Fundação para a Ciência e Tecnologia) grant SFRH/BD/36742/2007, and by "Projecto de Re-equipamento Científico".

REFERENCES

[1] GTA/UFRJ Grupo de Teleinformática e Automação (2004, June). *Redes de Sensores Sem Fio, Características*. Retrieved July, 2006 from http://www.gta.ufrj.br/~rezende/cursos/eel879/trabalhos/rssf1/caracteristicas.htm

[2] Teixeira, I. (2005) *Roteamento com Balanceamento de Consumo de Energia para Redes de Sensores Sem Fio*. Rio de Janeiro, Brasil: Universidade Federal do Rio de Janeiro. (http://www.gta.ufrj.br/ftp/gta/TechReports/Ingrid05/tese.pdf)

[3] Akingbehin, K., Patel, N., Richardson, P., Yoon, D., Chen, J., Abdu, H. (2003) *Proposal for a hybrid wireless harness for automotive applications*. Michigan: Institute for advanced Vehicle Studies, University of Michigan-Dearborn.

[4] Luz, G.D. (2004) *Roteamento em Redes de Sensores*. São Paulo, Brasil: Instituto de Matemática e Estatística, Universidade de São Paulo. (http://grenoble.ime.usp.br/movel/roteamentosensores.pdf)

[5] Araújo, R.C. (2004) *Um Estudo do Impacto do Uso de Criptografia em redes de Sensores Sem Fio (RSSFs).* Recife, Brasil: Centro de Informática, Universidade Federal de Pernambuco. (http://hantuanne.com.br/faculdade/7semestre/redes/criptografia.doc).

[6] Correia, L.H.A., Macedo, D.F., Santos, A.L., Nogueira, J.M.S., Loureiro, A.A.F. (2005) *Uma Taxonomia para Protocolos de controle de Acesso ao Meio em Redes de Sensores Sem Fio*. Belo Horizonte: Departamento de Ciência da Computação, Universidade Federal de Minas Gerais. (http://www.dcc.ufla.br/~lcorreia/bibtex/rt0505.pdf).