



UNIVERSITY POLYTECHNIC OF VALENCIA

MASTER DEGREE ON COMPUTER ENGINEERING

MASTER FINAL WORK

**MONITORING WIRELESS SENSOR NETWORK
NODES WITH A LOW INTRUSION HYBRID
MONITOR**

Author:

Marlon Renné Navia Mendoza

Tutors:

José Carlos Campelo, Alberto Bonastre Pina

Valencia, February 2015

To my family, my support and encouragement.

Contents

Abstract.....	4
1. Introduction.....	5
2. Objectives and Justification.....	7
3. Monitoring Tools.....	8
3.1 Active Monitors.....	8
3.2 Passive Monitors.....	9
3.1 Others Monitors Purposes.....	10
4. The Low Intrusion Active Hybrid Monitor.....	12
4.1. Architecture of the Monitor.....	12
4.2. Monitor Operation.....	15
4.3. Monitor Implementation.....	18
5. Evaluation and Results.....	22
5.1. Intrusion on Time.....	22
5.2. Intrusion on Code.....	24
5.3. Intrusion on Power Consumption.....	25
5.4. Frequency for Event Log Generation.....	26
6. Discussion.....	28
7. Conclusions.....	31
8. Future Work.....	32
References.....	33

Abstract

Several systems have been proposed to monitor sensor networks—specially Wireless Sensor Networks (WSN)—in order to debug and analyze their operation. These systems are based on hardware and/or software technology, and can be active or passive. Each one of them has pros and cons, but none is complete at all. Active monitors generate a lot of intrusion, passive ones do not collect all relevant information about nodes, and most proposals are far away from real systems or are too hardware dependent. This work presents an active hybrid monitor with low intrusion, to be applied on sensor network nodes, which overcomes most of these drawbacks. Monitor is based in a new purposed open architecture for Monitoring Platforms which provides flexibility, universality, and reutilization. Intrusion caused to the sensor node has been evaluated on three aspects: time, additional code, and power consumption. Low intrusion has been achieved in all three issues. We have evaluated interfaces different from traditional serial transmission proposed by most of similar approaches. It has been proved that the use of parallel interface or Serial Peripheral Interface (SPI) allows a higher frequency of event log generation and low intrusion.

Keywords: Monitoring, hybrid monitor, low intrusion, sensor networks.

1. Introduction

Sensor networks have undergone great research and development in recent years. However their massive deployment has not been too great because sensors networks may experience problems or errors in their operation. Many causes can be identified, such as interferences in the transmission medium, security attacks (especially in WSN [1]), adverse environmental conditions, malfunctioning nodes, and others. The node faults, their sources, and detection approaches are diverse, as it is detailed in [2]. Although during development or implementation of this type of network debugging and operation testing is usually made, when sensors are deployed the conditions can be very different and usually unanticipated events arise.

The availability of a suitable sensor network failure diagnostic tool is a key issue in progressing to real-world deployment of WSN. Nowadays there are no standard tools or standard architectures in this area. Most of proposals in monitoring and debugging do not consider enough aspects of sensor networks to be fully useful, or are built for very specific network architectures. There are many challenges in several aspects—architectural, functional, and dynamic—that have to be researched according to [3].

The so called monitoring systems—or simply monitors—are used to evaluate the performance and operation of a sensor network, in controlled conditions or even in a real environment. Monitors can focus on many performance parameters, such as throughput, jitter, response time or reliability, and even to security and intrusion detection in the network, as described in [4].

Monitors usually are built based on one of two possible approaches. Active monitors involve additional hardware and/or software in sensor nodes, interacting with it. In this way, active monitors usually require the modification of the sensor nodes to be monitored. This interferes with its normal operation and measured parameters may vary from unmonitored node. However, the obtained data are more reliable.

On the other hand, passive monitors rely on the observation of the external behavior of the monitored system without any interference with its normal operation. Usually, behavior algorithms are used to evaluate the presence of errors, undesirable operation or unexpected events. No incidence on monitored nodes performance is caused, but only externally observable variables can be measured.

Besides there is another approach for monitors depending if they are based on hardware or software. A *software* monitor is implemented by means of a specific code, application, or plug-in to the operating system of node, which access to the system status and reports relevant information. Usually, a software monitor has deep information about the system functioning, but it may interfere with the operation of the monitored system.

A *hardware* monitor consists on electronic devices connected to the monitored system, which recollect data from interesting system points. Hardware monitors use to be less intrusive than software monitors, but they implies the use of additional components.

Each monitor approach by itself cannot cover all aspects of monitoring tasks, as we will study in the next section. Monitors can also combine both approaches in order to achieve the advantages of both types and obtain a complete vision of the system, trying to keep the interference to the minimum. These are the so called Hybrid monitors [5].

In this work an active hybrid monitor with very low intrusion—based on both hardware and software monitoring methods—is presented. This monitor can record the events which occur in a node of a sensor network and store them in a non-volatile memory for later analysis. Moreover it is capable of being incorporated as a part of a complete monitoring platform that includes other acquisition possibilities, such as passive monitors, as described in [6].

2. Objectives and Justification

The main goal is to build a monitor system that lets to combine the advantages of both types of monitors in a single monitoring platform able to monitor the complete system.

The hybrid monitor is going to work in a standardized way, by using of standard interfaces and available libraries, but with low intrusion on sensor network nodes.

This monitor system allows to register the events and behavior of sensor network nodes, as well as to reconstruct the activity of the network. The generated information can be further analyzed by adequate tools.

The architecture used for hybrid monitor development wants to become a standard for monitoring platforms, and allows to monitor provides flexibility, universality, and reutilization.

3. Monitoring Tools

There are several tools and techniques for monitoring sensor networks, most of them with one approach, and mainly focused to WSN. In [3], an overview and comparison of most of proposed tools are made, not only monitoring tools but also debugging ones. In this section a brief summary of some of the most important and relevant is done.

3.1 Active Monitors

SNMS (Sensor Network Management System) [7] is one of the first and best known monitoring systems. SNMS is a complete management system, focused on working with any type of sensor network. It is built on TinyOS—an open source operative system designed for low-power devices [8]—and allows a review of the state of a node and even save information locally. Nevertheless, it generates substantial intrusion—it increases traffic and power consumption—and is oriented rather to management than monitoring.

Memento [9] and Lightweight Tracing [10] are both examples of active monitors. Both use short encoding with events and information of sensor node. The first one adds its code protocol to a message that is going to be transmitted by node. Memento can detect problems in a node by basing on the information provided by their neighbors in the sensor network; but for detecting new kinds of failures it requires node reprogramming.

In Lightweight Tracing [10] the events are saved by using a very light coding in non-volatile memory for further reconstruction and debugging of node and network behavior. Because both are active monitors, they generate substantial intrusion in nodes operation.

Despite its name, Passive Diagnosis for WSN (PAD) [11] is an active monitor system with little intrusion. It is based in a probabilistic diagnosis approach—based on a Belief (or Bayesian) Network—to infer the root causes of abnormal WSN

operation. This adds a probe in each node that marks the packets with relevant data with very little overhead. . However, PAD has to wait a message transmission to send information and it might not determine when an error has happened. Besides, as far as non-sense nodes, such as router nodes, do not send sensed data, they are not able to send any monitor information to infer possible abnormal operation.

3.2 Passive Monitors

Sympathy [12] works as a passive monitor, and can detect and debug pre-and-post deployment errors. It operates by analyzing the data arriving at the sink of a sensor network, applying metrics, and inferring where in the network a fault or failure can be produced. The implementation of this mechanism depends on the knowledge of the network behavior. It also considers the aggregation of a small overhead on the network to increase its accuracy. However, it was not developed for event-driven applications, losing some important information and reducing its accuracy.

SNIF (Sensor Network Inspection Framework) [13], Pimoto [14], LiveNet [15], SNDS (Sensor Network Distributed Sniffer) [16], NSSN [17], and EPMOS_t (Energy-efficient Passive Monitoring SysTem for WSN) [18] are examples of passive monitors. Their approach consist on deploying a network of sniffers—together with the target sensor network—with an interface to capture all transmissions from nodes. The main difference between them is how the captured data is processed.

SNIF [13] and Pimoto [14] transmit—via a Bluetooth interface—the captured data to other device in order to process it. In the first case the device that receives the packets—tagged with a timestamp—works as a sink and analyzes information. The analysis tool has been developed by the authors.

In Pimoto the device that receives the data—called gateway—is a computer that tag the packets with a timestamp and forwards it, via TCP/IP, to a central server for analysis. Pimoto can also works over more than one sensor network simultaneously. It has a plugin for Wireshark [19]—a traffic analysis tool—in order to analysis data. However, Pimoto may not be practical for monitoring WSN that has too many nodes widely distributed, because it needs more infrastructures.

LiveNet [15] merges the traces obtained by sniffers in order to provide information about dynamics of sensor network, as i.e. node traffic rate analysis, hotspot identification, network topology discovery, and others. However it does not detect problems that do not impact on network traffic, and needs reprogramming for additional information.

SNDS [16] works in a similar way that the two above mentioned, but it uses an Ethernet connection instead of Bluetooth. TCP and UDP protocols are used for data transmission and synchronization respectively. Nevertheless this monitoring system needs additional infrastructure due Ethernet connection.

In NSSN [17] sniffers can detect automatically the work frequency of target WSN. The collected data is sent via wireless—3G, GPRS, Wi-Fi—to a monitor server which parses, pre-processes, and stores data in a database. The information stored in the server can be accessed by clients in order to look it or analyze it. Although the multiple radio channel option is an interesting feature, it cannot avoid the problems of differences in radiofrequency circuits and signals.

EPMOST [18] is a passive monitoring system focused on the reduction of energy consumption of monitoring network. EPMOST selects the nodes that their packets will be captured by each sniffer, so there are not duplicated captured packets. This system provides information using a SNMP (Simple Network Management Protocol) agent.

Most of passive monitors can provides real-time analysis of information about sensor network. Nevertheless all these tools only capture the transmitted frames “on the air” but not obtain information directly from nodes; so some information about sensor network behavior may not be available.

3.1 Others Monitors Purposes

Minerva [20] is not a monitor, but a testbed for WSN. It uses a debugging port and tracing port connected to the sensor node to observe the behavior of the node. Minerva has very interesting features but is not too adequate for monitoring in real environments and is more oriented to debugging.

Finally, Spi-Snooper [21] integrates hardware and software in a hybrid approach. The hardware architecture joins in a piece the sensor node and the monitor in a transparent way, using a SPI interface to communicate them. The software architecture has two operation modes: active and passive. In passive mode the monitor—called co-processor—mainly logs the communication through the SPI bus and check son node data. In active mode it assumes the control of SPI and radio interface. However, it can only be used in sensor nodes that transmit through a SPI port. Only the data transmitted through this SPI interface may be monitored.

Each one of these proposals has pros and cons. Proposed active monitors usually involve much intrusion. Passive monitors only can monitor transmitted data, but they are not able to know what happed inside the node. Addition of monitoring information to a transmitted message — or new messages to the network —produces a decrease of network performance. Monitors based exclusively in software are not able to work if the node fails. Hardware-based proposed monitors are too architecture-specific. A monitor system with wide enough network information coverage, while keeping low intrusion is then necessary. Furthermore, it has to be generic enough to be applied to all hardware architecture. In this work more interesting characteristics of these studied proposals have been considered as a base for design of our monitoring system, while trying to minimize the drawbacks of previous tools.

4. The Low Intrusion Active Hybrid Monitor

In [22] is presented a first approach of the proposal. The system is based on the proposed architecture presented in [23]. The main characteristics of the monitoring platform—according to [6] and [23]—should be:

- It must have the greatest possible independence between the application / network and monitor.
- The intrusion in both software and hardware should be minimal.
- The system must be able to record information of the nodes.

The proposal architecture and operation mode are presented in this section. . It is based in a standard-oriented architecture, in order to provide advantages such as universality, reusability and flexibility. A simple and little intrusive operation mode has been defined for the hybrid monitor

4.1. Architecture of the Monitor

The architecture of this hybrid monitor is based in the proposal presented in [23]. This model is based on the division of the monitoring issues in three layers: Monitoring Layer focuses on relevant information—data capture, analysis and visualization—; Information Layer deals with information coherence—data format, synchronization and triggering—; and finally Interchange Layer supports communication and storage. Figure 1 shows the layers and components of this architecture.

One of the main goals of this model is to aid in the design of new WNS Monitoring Platforms (WNS-MP). This approach offers several new interesting features, such as universality, adaptability, flexibility and simplicity; finally, it is ready to follow the future evolution of sensor networks and their monitoring systems.

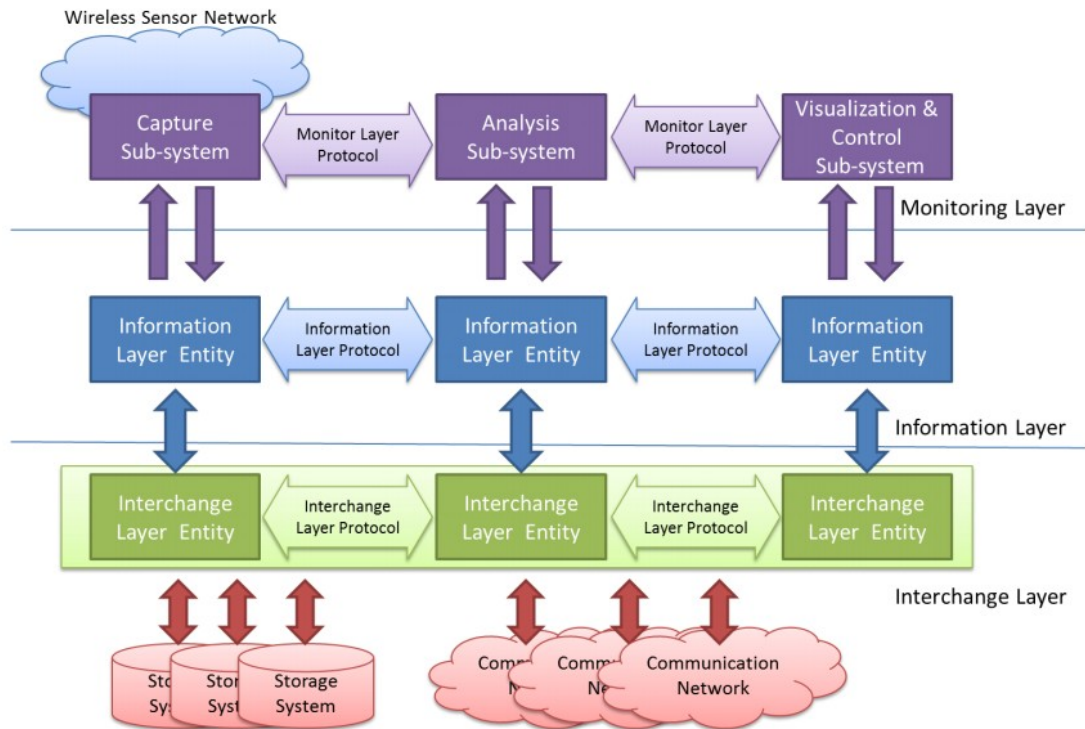


Figure 1. Architecture for WSN monitoring systems proposed in [23].

In this three layer architecture the communication between layers and entities in the same layer is similar to the standardized in the OSI (Open Systems Interconnection) reference model for networks [24]. Each layer defines interfaces to communicate with the next.

One of the advantages of working with a defined architecture is that changes and developments in an entity/layer should not affect the others. Hence, any improvement to the monitor system will be easier to develop and implement. Besides, reutilization of this hybrid monitor in other sensor network is easy by making the changes in the respective layer.

The active hybrid monitor includes both software and hardware parts, as Figure 2 shows. The software part consists of Software Traps added to the sensor network nodes, and constitutes the monitoring layer entity. These software traps—located into the code of the monitored node—send a numeric code when executed, usually related to significant operations in the node, such a state change or event occurring, and optionally additional information (such state variables, message contents) as it will be explained later.

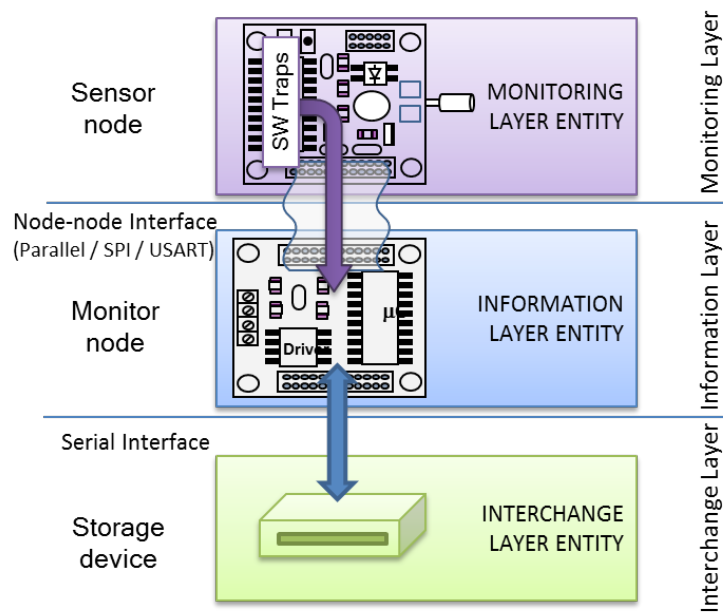


Figure 2. Layered architecture of hybrid monitor.

The hardware part of hybrid monitor covers the others two layers in the reference architecture. The information layer entity of the hybrid monitor is implemented by the monitor node. It is connected to the sensor network node through a standard interface.

In this proposal, interchange layer entity is implemented by means of a storage device attached to the monitor node, where the collected data will be stored. The communication with monitor node is done through a serial interface, as described in Figure 2.

Figure 3 shows monitor components and their location in the architecture, as well as the communication direction. Two routines are included inside the sensor node. First of them—which can be called as a function by the node application—prepares the log data about events to be sent to the monitor. The second one is activated when an ACK (Acknowledgment message) interruption is received. This routine sends the rest of data if necessary. The ACK mechanism was found be necessary for flow and error control in early experiments, especially for higher data sizes. These software routines belong to the Monitoring Layer.

The monitor node implements the information layer entity of the hybrid monitor. It uses its built-in RTC (Real Time Clock) to generate timestamps for the sensor node events. The monitor node sends an ACK every time an event, or part of it, is received and saved. This information, with a defined format that includes timestamp information, is stored for further analysis. Monitor node is responsible for attaching the timestamp to collected data, and store it in an adequate format into the interchange layer. For connection between sensor node and monitor node we have considered standard interfaces available in most of nodes.

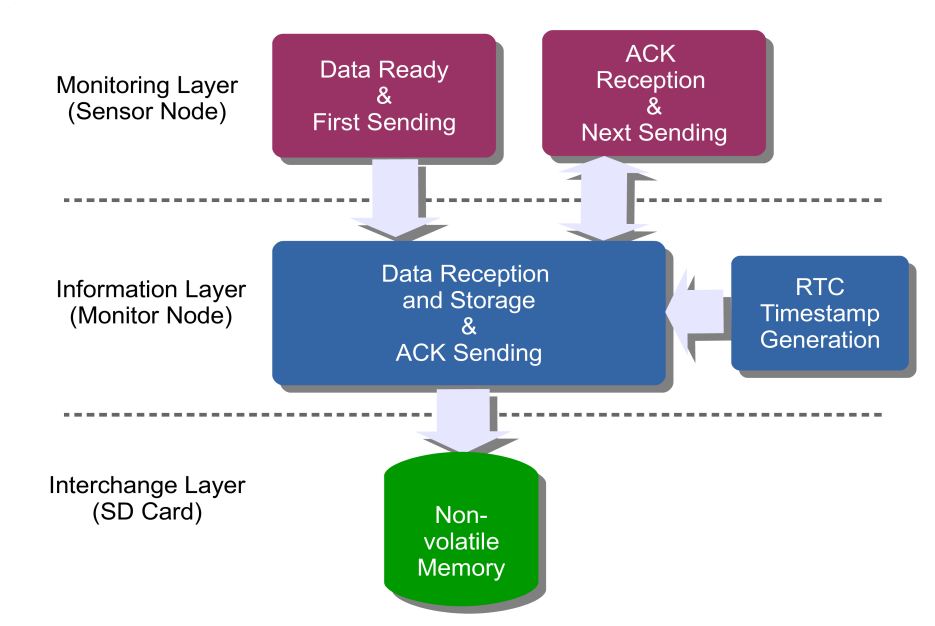


Figure 3. Components of hybrid monitor by layer.

As Figure 3 shows, in the actual implementation the interchange layer consists of a non-volatile memory where the events information will be stored. This approach is similar to the used in Lightweight Tracing [10], but memory is attached to the monitor node instead of to the sensor node. In our implementation a Secure Digital (SD) card is used. Further versions of the interchange layer would include a radio interface to transmit the collected information, either in real time or in scheduled manner.

4.2. Monitor Operation

The main idea of hybrid monitor is that event logs are generated during working time—when sensor node is operating—or when an interruption is activated. Figure 4

shows a scheme of monitor operation. A signal—with information of the event—is sent from sensor node to monitor node when a relevant event—that we want to register—occurs.

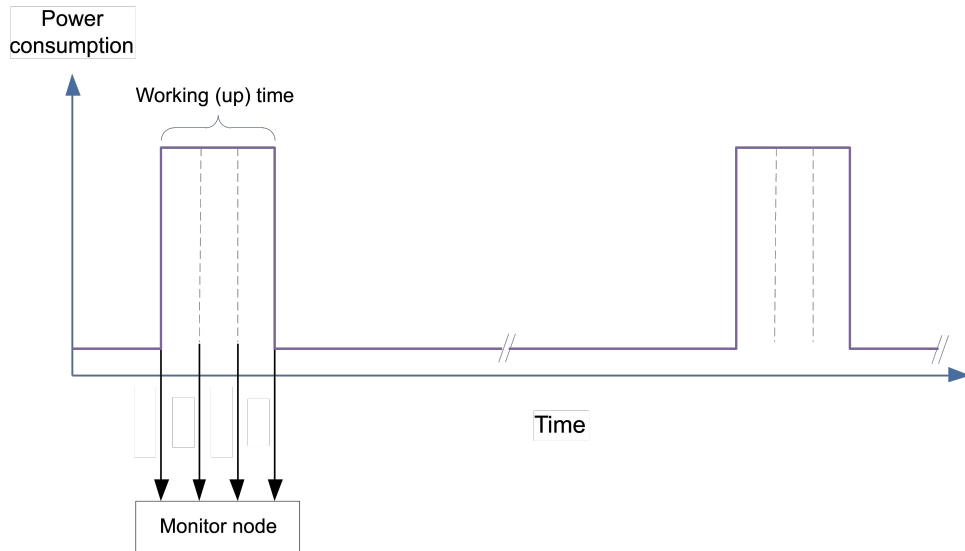


Figure 4. Hybrid monitor operation scheme.

The operation time of the monitor has to be simple in order to reduce the intrusion in the network nodes. Figure 5 shows the data flow and operation of hybrid monitor.

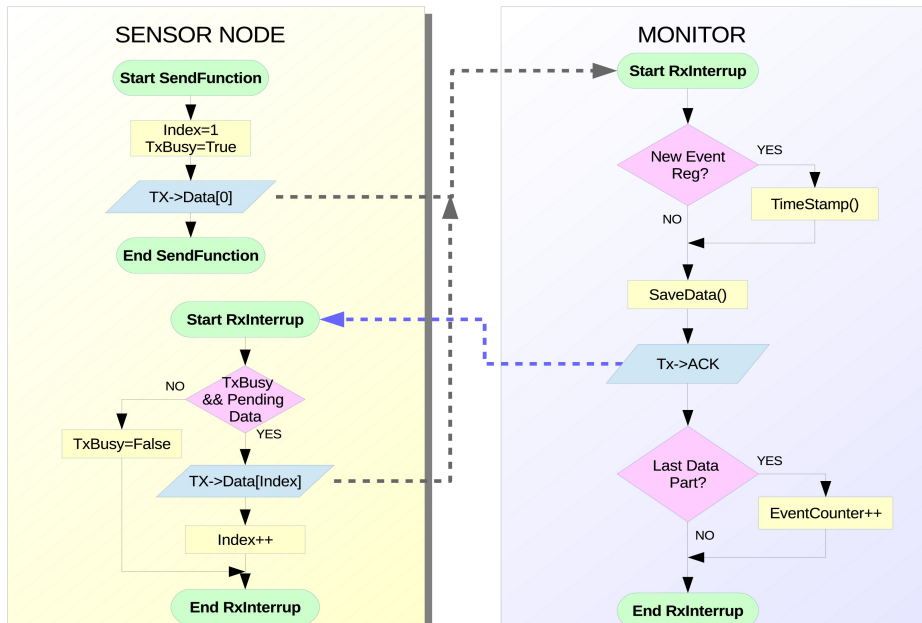


Figure 5. Flowchart of hybrid monitor operation.

Specific functions have been defined to register node events, and they are included in the node. There are two functions added to sensor network node: One can be located where log information required, and the other is activated by an interruption when the ACK is received. When sensor node reaches a software trap, it invokes the Start Send Function in order to make relevant data (contained in TX_Data[] structure) to be sent to the monitor node. This function prepares the data to be transmitted, and then it sends the first part of data to the monitor node, and activates a flag that indicates a transmission process. Then the function returns the control to the main node application.

The data sent to the monitor may include additional information with the event code. The way to obtain the additional data depends of the sensor node.

This possibility allows increasing the accuracy of obtained information. For example, a log of an error event with an additional code that describes the kind of error may be obtained. It is also possible, when dealing with retransmission events, to add to the trap part or the entire message, in order to differentiate between messages.

Finally, it is also possible add a code to indicate the reason of the wakes up of the node. This is a significant improvement in comparison with other proposals that only register events but cannot handle additional information about them.

The monitor node is always waiting for information sent from the sensor node. When data from sensor node is received, monitor node stores it with the appropriated timestamp. This timestamp has to be generated in case of a new event log. Then an ACK message is sent to the sensor node. When the sensor network node receives the ACK the next part of information may be sent if necessary; or transmission flag may be deactivated. This mechanism can be considered semi-blocker and reduces the intrusion for the sensor node.

The events are coded in 4 bits. These codes have already been defined earlier in [22] and are shown in Table 1, although this definition can be modified or increased, in order to enhance the accuracy.

Monitor node stores information with a predefined format. This format includes the timestamp, the log code, and additional information if apply; and it must be easy to analyze with an adequate application.

Table 1. Event codes defined by authors in [22].

Code	Meaning
#define Log_Reset 0x00	//Node Reset/Initialization
#define Log_Sense0 0x01	//Read sensor 0 (first/unique)
#define Log_Sense1 0x02	//Read sensor 1 (second if it's)
#define Log_Sense2 0x03	//Read sensor 2 (third if it's)
#define Log_Wakeup 0x04	//Wake up from sleep/stop
#define Log_RxData 0x05	//Node receives data
#define Log_TxData 0x06	//Node sends data
#define Log_RxACK 0x07	//Node receives ACK
#define Log_RRoute 0x08	//Node reroutes data (if apply)
#define Log_Sleep 0x09	//Node goes to sleep mode
#define Log_Stop 0x0A	//Node goes to stop mode
#define Log_LowBat 0x0B	//Low battery indication
#define Log_SinkRx 0x0C	//Sink receives data
#define Log_SinkTx 0x0D	//Sink sends data
#define Log_SinkEr 0x0E	//Error in sink
#define Log_Error 0x0F	//Error in node

4.3. Monitor Implementation

The monitor node has been developed on a STM32F051R8T6 ARM Cortex-M0 microcontroller (MCU) [25] implemented on a STM32F0 Discovery Board [26]. This microcontroller is a 32-bits core high performance MCU, and it has these main features:

- ARM Cortex-M0 core up to 48MHz.
- Memory: 64KB of flash memory, 8KB of RAM.
- Several GPIO (General Purpose Input-Output) ports.
- Connectivity: USART (Universal Synchronous Asynchronous Receiver-Transmitter), SPI, I2C.
- Control: 3-phase PWM (Pulse-Width Modulation) control timer, PWM timers, basic timer, comparators.

- Real Time Clock onboard or externally supplied.
- Digital-Analog and Analog-Digital converters.

The monitor board has been connected to a SD card through its SPI interface. Figure 6 shows the implementation of the hybrid monitor. The monitor node—at left side—is connected to a sensor node—at right side—that is used for testing. It has attached the SD card used to store data. In case of Figure 6 the SPI interface is also used to connect the monitor node to the sensor node.

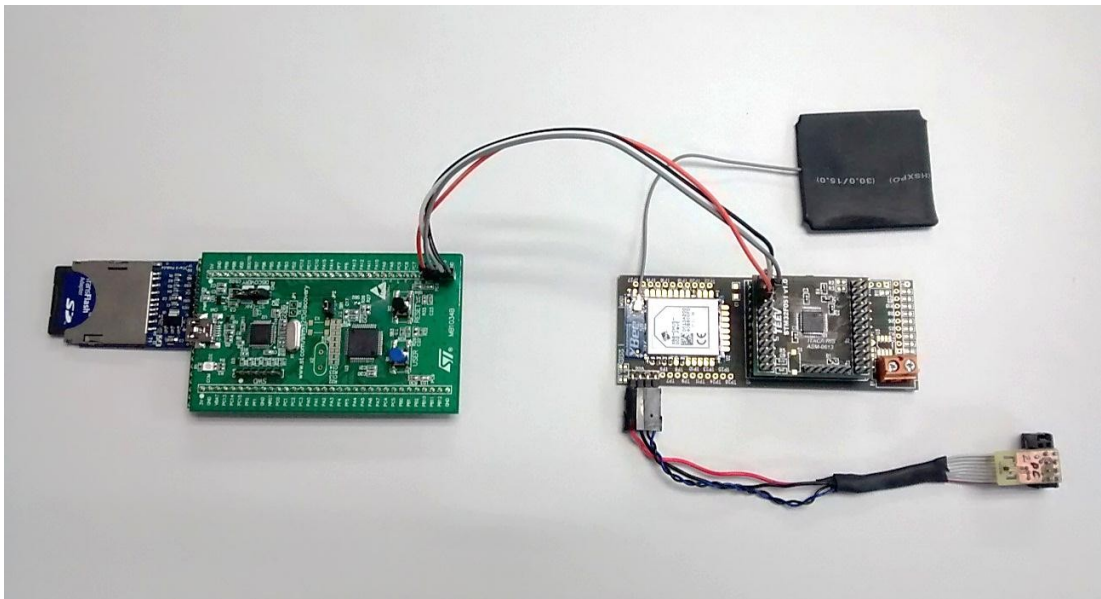


Figure 6. Picture of a sensor node connected to the monitor node.

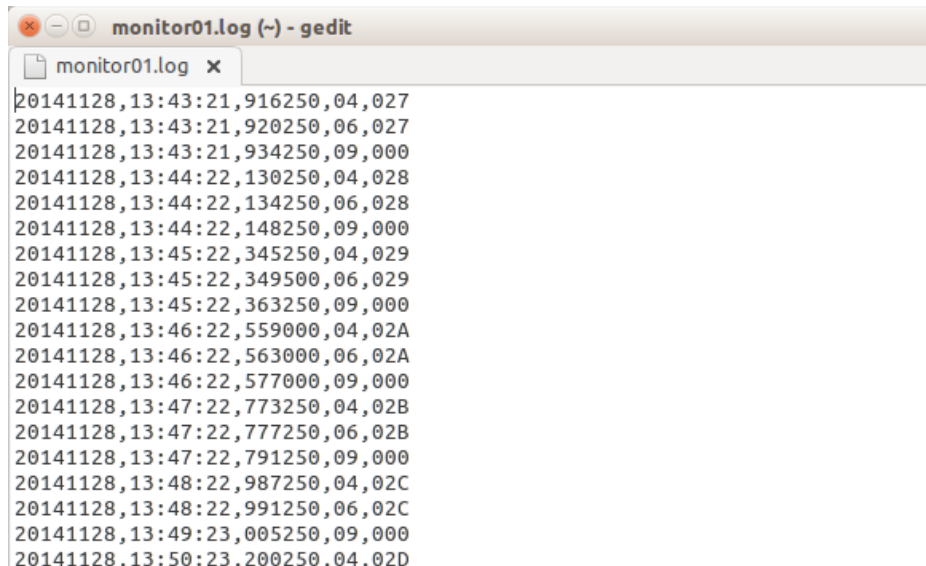
In order to evaluate the most appropriate communications interface between sensor node and monitor node, three different interfaces implementation—usually found in most of microcontrollers—have been used: SPI, USART, and finally parallel transmission, using 16 GPIO ports. Parallel and SPI data width is 16 bits, and USART data width is 8 bits. Besides, three transmission speeds were evaluated for SPI interface. To compare the performance of each interface, four sizes of transmitted data (common sizes of 16, 32, 64, and 128 bits per trap) have been assumed. . The bigger the data being transmitted, more detailed the information provided in each trap, but higher the interference.

A library should be added to the code implementation on sensor node with the instruction sets for initialization and data transmission. As far as the ARM

architecture may be the most used in sensor node implementation, and in order to provide a standardized level CMSIS (Cortex Microcontroller Software Interface Standard) has been used to create this library. CMSIS is a vendor-independent hardware abstraction layer and can be used on many microcontrollers [27]. This library can be adapted easily for others architectures with the correspondent instructions. As far as the messages sent through the interface between sensor node and monitor node (SPI, USART or Parallel) are the same that previously cited ones, monitor node needs no modification.

For evaluation of the monitor node a sensor node based in an ARM Cortex-M0 at 48MHz was used. Every time this sensor node wakes up from sleep mode, it takes a measurement from a temperature sensor, transmits the measurement via wireless, and turns back to sleep mode for 60 seconds again.

Traps have been inserted in the program of the sensor node to register three kinds of events: wake up, transmission, and sleep. These codes are sent to the monitor node, which reflects them into ASCII text separated by commas, and stores these data and its timestamp in a SD card. Figure 7 shows part of the information generated by the hybrid monitor and recovered from this SD card.



```
monitor01.log (~) - gedit
monitor01.log x
20141128,13:43:21,916250,04,027
20141128,13:43:21,920250,06,027
20141128,13:43:21,934250,09,000
20141128,13:44:22,130250,04,028
20141128,13:44:22,134250,06,028
20141128,13:44:22,148250,09,000
20141128,13:45:22,345250,04,029
20141128,13:45:22,349500,06,029
20141128,13:45:22,363250,09,000
20141128,13:46:22,559000,04,02A
20141128,13:46:22,563000,06,02A
20141128,13:46:22,577000,09,000
20141128,13:47:22,773250,04,02B
20141128,13:47:22,777250,06,02B
20141128,13:47:22,791250,09,000
20141128,13:48:22,987250,04,02C
20141128,13:48:22,991250,06,02C
20141128,13:49:23,005250,09,000
20141128,13:50:23,200250,04,02D
```

Figure 7. Information recovered from the SD card attached to the monitor node.

In case of Figure 7 information includes date, time, microseconds, event code, and additional data. Event codes 04, 06 and 09 (Table 1) have been used for wake up,

transmission and sleep events, respectively. Additional values for wake up and transmission events—also sent as additional information—indicate the iteration number. All this data can be delivered to the visualization and control subsystem (Figure 1) by using the information layer services.

5. Evaluation and Results

In order to evaluate the characteristics of the proposed monitor, intrusion analysis is required. Three are the main intrusion aspects that must be considered. Time intrusion deals with the increment of execution time on sensor node caused by the monitor. Taking into account that sensor nodes use to be limited in RAM memory, Code intrusion evaluates how many bytes have to be added to source program of this sensor node. Finally, Power intrusion evaluates the amount of energy that this monitoring technique requires from the sensor node, and thus reducing its battery life. Several experiments have been performed, considering the previously cited interfaces and data sizes, to measure the intrusion of the active hybrid monitor. Obtained results are shown in this section.

5.1. Intrusion on Time

Time intrusion was determined by measuring the amount of time needed to fulfill one thousand wake-up iterations in the sensor node without traps (reference time), and then measuring when traps were added. Three interface implementations and four data sizes were combined to provide the table 2. This table shows the difference between the obtained time and the reference time, and thus the time intrusion.

Table 2. Intrusion on time for each log event with different interfaces and data sizes (microseconds).

Data Size	Parallel (16bits)	SPI 18mbps	SPI 4.5mbps	SPI 2.25mbps	USART 115.2kHz
16bits	3.80	3.10	3.10	3.20	6.20
32bits	7.30	6.10	6.20	6.20	13.60
64bits	16.50	13.00	13.10	13.10	28.80
128bits	31.90	28.10	28.10	28.20	60.60

As shown, the intrusion time is similar for different interfaces with the same data size, due to the implementation of the monitor code in the sensor node: As far as data transmission through the interface is performed by specific hardware (SPI controller, USART, others), trap routines only write the outgoing data into the interface buffers

and the sensor node program may continue. These hardware controllers take care of transmission in parallel to this program execution.

Parallel interface has no dedicated hardware controller. That is the reason that the time is little bigger than the others. An additional line is used to generate the transmission interruption in the monitor node. As expected, intrusion time increases for larger data sizes.

However, values in Table 2 cannot be used as a reference to know the maximum frequency for the event log generation, because it depends on the peripheral using time (communication hardware) and the monitor node hardware. Besides, the times of Table 2 are a sum of non-continue values. Time limitations on SPI controller, USART and parallel ports must also be considered. In case of USART the intrusion time is about double because it only sends 8 bits per transmission and it has to manage twice of interruptions.

The processing time in the monitor node was measured too. Each event log has to be processed in one or more data reception interruption, according to data size transmitted from sensor node. The processing time is considered from the arrival of the first data to the receiving of the last piece of information. The timestamp generation takes 15.8 microseconds, and is done when the first piece of data arrives. Table 3 shows the obtained values.

Table 3. Processing time in monitor node for each event log (microseconds).

Data Size	Parallel (16bits)	SPI 18mbps	SPI 4.5mbps	SPI 2.25mbps	USART 115.2kHz
16bits	17.80	17.40	17.40	17.40	88.40
32bits	22.90	23.40	25.40	27.80	262.00
64bits	32.90	34.40	40.10	48.20	612.00
128bits	52.80	56.20	69.80	88.00	1300.00

As shown in Table 3, when dealing with 16 bits events the obtained times are similar for every interface, as far as process time depends mainly on timestamp generation. For larger data size the processing time is a bit less when using parallel interface instead of SPI, especially when the SPI speed is low.

In the case of the USART interface, the processing time is very larger, because its low transmission speed and the fact that it only can send 8 bits at a time, whereas other considered options can send 16 bits at a time. As expected, the processing time increases for larger data sizes.

Should be noted, values of Tables 2 and 3 may change in some cases. A monitor node based on a faster microcontroller will reduce the timestamp generation time, hence will reduce the processing time and increase the performance. Besides, the time intrusion and processing time could change depending of sensor network node characteristics, such as core frequency, availability of buffers, architecture, interface to monitor, and others.

USART interface has not been considered as an option to be used in our monitor because of its low speed and high processing time—compared with the others interfaces—, which results in low performance. Thus it is not considered in further results.

5.2. Intrusion on Code

As far as memory resources are limited for sensor nodes, the evaluation of the intrusion on program code is relevant. Code used in the sensor node was generated by Keil MDK (Microcontroller Development Kit) version 5, a comprehensive software development environment for Cortex-M processor based microcontrollers that includes IDE (Integrated Development Environment), Compiler, and Debugger [28].

Size difference in bytes has been considered between the pure application code and the trap-modified code, in binary compiled program. Main differences between both codes consist on the addition of the port initialization subroutine and several data transmission instructions.

Table 4 shows the intrusion in bytes on program code in sensor node. As expected, intrusion is not related to the communication speed, thus this parameter has not been considered. Since transmission code is reused for all the traps, a new event may be monitored by just adding to the code of the sensor node a trap call of 8 bytes.

Table 4. Intrusion on Code: Initialization and one event log (Bytes).

Interface	16bits	32bits	64bits	128bits
Parallel	228	252	268	276
SPI	428	452	468	476

Code intrusion ($S_{Intrusion}$) may be predicted by means of the Equation (1), where S_{init} is the initialization value appearing in Table 4, and n is the number of trap calls included in the application code. As expected, for a small number of traps monitored, the code intrusion is mostly determined by the initialization code.

$$S_{Intrusion} = S_{init} + 8 \times (n - 1) \quad (1)$$

5.3. Intrusion on Power Consumption

Taking into account that sensor nodes usually are powered with batteries or harvesting techniques, power consumption is a key aspect in sensor network nodes. Due the working time is very low—some milliseconds as maximum—power consumption was determined by modifying the application program of the sensor node, avoiding sleep mode and keeping used peripherals enabled. About 7000 samples of sensor node electrical power were taken—18 samples per second—and averaged them. Measurements were taken with an Agilent 34405A Multimeter. This multimeter has a 5.5 digit resolution, with an accuracy of $\pm(0.05\%$ of reading + 0.015% of range) for our measuring conditions [29].

The obtained results showed that the instantaneous power consumption was the same for all the evaluated communication speeds. Data size was also found to be irrelevant in power consumption.

The sensor node electrical current required in clear operation, without monitoring, was found to be 53.4 mA. The total electrical current required when monitoring was performed through each interface and the differences with normal operation are shown in Table 5.

Table 5. Electrical Power by interface (milliamps).

Interface	Total Power	Difference with no-monitor operation
Parallel	53.61	0.21
SPI	53.95	0.55

Since power consumption seems not to be related with trap implementation, the power intrusion may be considered directly related with the wake-up time. In this sense, it is possible to evaluate the impact of each technique by multiplying the values of power (Table 5) and time intrusion (Table 2). Table 6 shows the obtained values. The power consumption has been found to be proportional to the intrusion time, and it can be considered low in relation of normal operation of the sensor node (below 2%). It can be seen in Table 6 that the power consumption intrusion increases with the data size, and is little higher for parallel interface, as expected for the number of lines involved in trap transmission.

Table 6. Intrusion on Power Consumption at pico-Amps-hours (pAh).

Data Size	Parallel (16bits)	SPI 18mbps	SPI 4.5mbps	SPI 2.25mbps
16bits	56.58	46.46	46.46	47.96
32bits	108.70	91.42	92.91	92.91
64bits	245.70	194.82	196.32	196.32
128bits	475.01	421.11	421.11	422.61

5.4. Frequency for Event Log Generation

Values of Tables 2 and 3 are not enough to calculate the maximum frequency of event log generation. Intrusion time consists of non-continuous values, and processing time does not include times for preparing data, first and last transmission times, and last ACK reception in sensor node.

Figure 8 shows a scheme of communication between sensor network node and monitor node. The period for event log generation—the total time to record an event—starts when software trap is activated and ends when the last ACK is received and transmission flag is deactivated. Hence, for the same monitor node, this period will depend of the interface used to communicate with sensor network node, and of the sensor node characteristics.

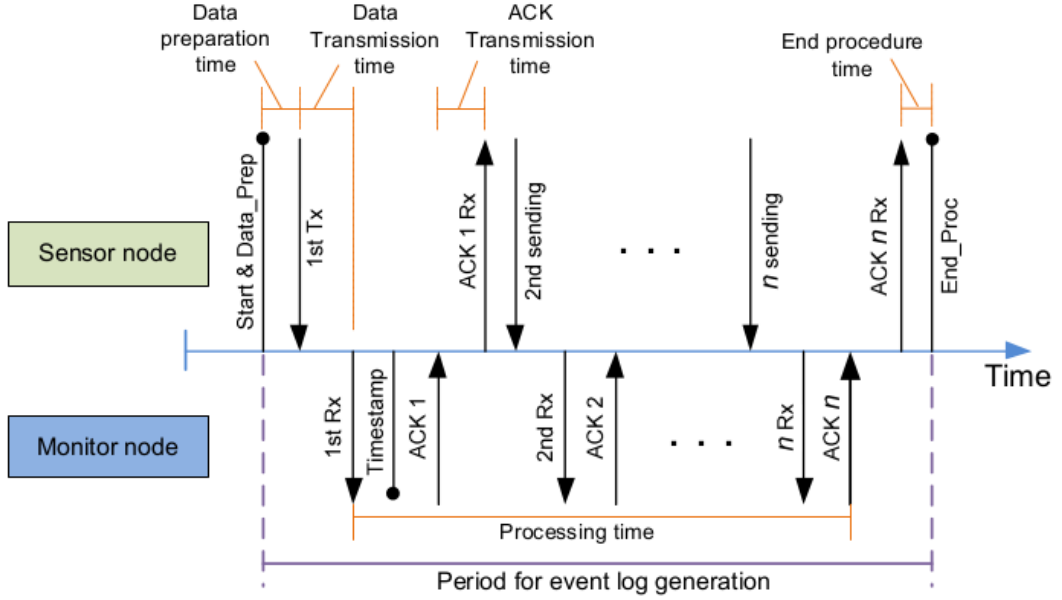


Figure 8. Time scheme for event log generation.

Once the period ($T_{EventGen}$) is determined, the maximum frequency for log event generation (f_{Log}) can be calculated by applying the Equation (2). However, this frequency is theoretical, because there may be others incidents that affect it.

$$f_{Log} = \frac{1}{T_{EventGen}} \quad (2)$$

6. Discussion

The intrusion generated by the hybrid monitor at the three evaluated aspects—time, code, and power consumption—may be considered low. This section studies the influence of these intrusion effects when applied to real applications.

The study about percentage of time intrusion during working time on a sensor node Figure 9 shows an example graphic with different estimated working times for a sensor network node similar to the used one in tests. At the example of Figure 9 we consider the log of 4 events—e.g. waking up, sensing, transmission, go to sleep—and a data size of 128 bits. The percentage of intrusion is very low—less than 1.5% in the worst case—even at the smallest case (10 milliseconds).

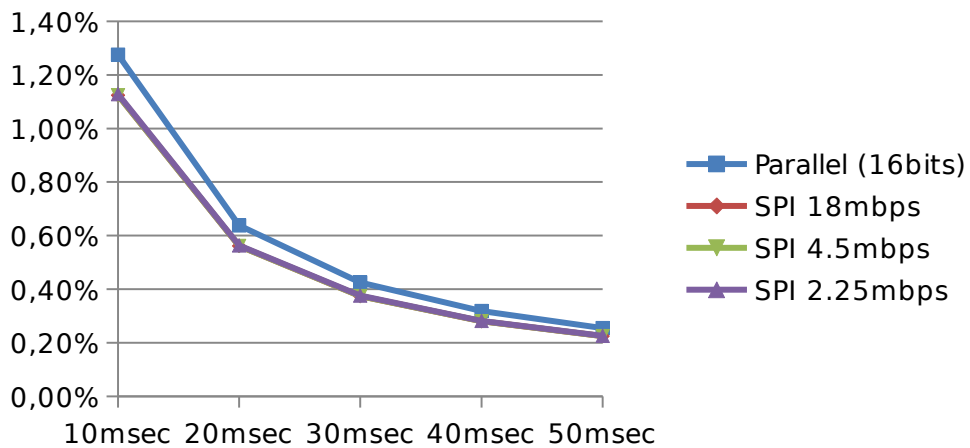


Figure 9. Examples of percentage of time intrusion (128bits / 4 event logs).

The estimated values used as reference in Figure 9 are similar to real implementations found in literature. For example in [30]—a wireless sensor node to monitor track bicycle performance—the sensor node active time is 30msec. In others cases as [31]—an application for heating and cooling loads—and as [32]—a wireless node enabled by wireless power with some sensors—the active time reaches 100msec or more. However, times of Table 2—intrusion time—cannot be extrapolated to these cases because the microcontroller characteristics of these nodes are different to our sensor node. Anyway, the percentage of time intrusion— despite being three times bigger than the values shown in Figure 9—should be considered low.

Figure 10 shows an example graphic with different application code sizes for having a better idea about percentage of code intrusion on sensor node. Again, the log of 4 events and a data size of 128 bits were considered. The percentage of intrusion is low—only in SPI and 8 KB of program code it is little up of 6%—and is smaller with parallel interface.

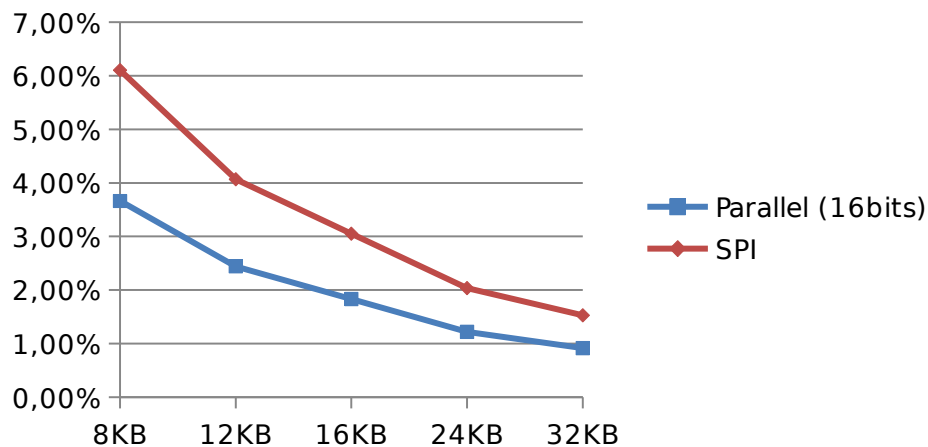


Figure 10. Examples of percentage of code intrusion (128bits / 4 event logs).

The maximum event generation frequency in the SPI bus, at high speeds, is a little faster than the parallel transmission. This is because in the second case it is necessary to generate interrupts to the monitor, while for SPI is not necessary. However, code intrusion when we use SPI is nearly twice that using parallel transmission, but it not pass from 500 Bytes, with a low percentage intrusion as Figure 10 shows. In all cases, for each event that we want to monitor, 8 bytes are added to the application code.

Moreover, processing time in monitor, when SPI is used instead of parallel interface, is little higher. Besides, this time increases for SPI lower transmission speeds, as it is shown in Table 3. Hence, used interface—as well as hardware characteristics of sensor network node and monitor node—affects the frequency for events log generation.

Since the power consumption in the sensor node is directly proportional to time intrusion, and that the difference in additional electrical current by using the peripheral to pass information is very low, this aspect will not be too transcendental when choose an interface to connect the monitor node.

A subject pending to define and standardize is the criteria and aspects for establish levels and categories—e.g. very low, low, medium, high—of monitor intrusion. Not only for sensor network monitors but for any monitor system. In this topic is needed be done a future work.

7. Conclusions

In this work an active hybrid monitor has been presented. It is able to collect directly from a sensor node detailed information about its operation with a very low intrusion, and thus without perturbing the node performance. It is based on an Open Architecture that provides many advantages, such as flexibility, reusability, and standardization.

As a part of the Open Architecture, many interfaces may be used to communicate sensor node. Performance of these interfaces influences the overall intrusion caused to sensor node operation, and thus reducing the accuracy of the obtained results. Several interfaces have been evaluated and characterized, being parallel or SPI interfaces the ones with the best performance. However, as far as eighteen pins are necessary for parallel communication, which are not always available for monitoring purposes in sensor network nodes, it can be concluded that SPI, when available, uses to be the better option in most of cases.

Due intrusion in sensor network node is considered very low, the main aspects to choose an way to active monitoring are intrusion on time and code, but not necessarily intrusion on power consumption that is proportional to time.

To achieve this goal it is necessary the standardization of the data format. Diverse monitoring data sources, even from different manufacturers, may then be joined to observe the whole network behavior.

Finally, it can be concluded that hybrid monitoring allows a deep knowledge of the internal operation of sensor nodes in a sensor network with a bounded—and usually very small—intrusion to its operation. Thus, it can be considered a very interesting option when designing, implementing, deploying and debugging sensor networks.

8. Future Work

Adding the monitor to a passive monitoring system—e.g. a sniffer network like mentioned proposed above—can result in a complete monitoring system that can observe the whole network behavior. Hence, a future work that integrates the active monitor to a passive monitoring network (sniffers) should be developed.

For a faster analysis of log data captured by the monitor, a radio interface can be added to the monitor node to transfer the data from memory card to a host. This host can be the same device that works as sniffer.

The format of stored data—in the information layer—can be standardized, so it can comply with the main goal of purposed architecture in which the hybrid monitor is based. A practical and easy way to do it is to use an existing standard format or adapt it to this purpose.

A model to determinate intrusion is necessary. The considered intrusion aspects of this model must be at least the three covered in this work. The model should define levels and thresholds of intrusion, as well as formulas to calculate these ones.

A Program for collected data analysis must be developed. This program can be a plugin for be used with any known network traffic analyzer—e.g. Wireshark—or a new application developed for this goal.

References

1. Patel, M.; Aggarwal, A. Security Attacks in Wireless Sensor Networks: A Survey. In Proceedings of the IEEE International Conference on Intelligent Systems and Signal Processing (ISSP), Anand, India, 1–2 March 2013; pp. 329–333.
2. Mahapatro, A.; Khilar, P.M. Fault Diagnosis in Wireless Sensor Networks: A Survey. *Communications Surveys & Tutorials*, **2013**, Vol. 15, No. 4, pp 2000–2026.
3. Rodrigues, A.; Camilo, T.; Silva, J.S.; Boavida, F. Diagnostic Tools for Wireless Sensor Networks: A Comparative Survey. *Journal of Network and Systems Management*, **2012**, Vol. 21 Iss. 3, pp. 408–452.
4. Rachedi, A. Monitoring Mechanism for Wireless Sensor Networks: Challenges and Solutions. In *Wireless Sensor Networks*, El Emary, I., Ramakrishnan, S., Eds; CRC Press: Boca Raton, FL, USA, 2013; pp. 595–621.
5. Schoofs, A.; O'Hare, G.M.P.; Ruzzelli, A.G. Debugging Low-Power and Lossy Wireless Networks: A Survey. *IEEE Communications Surveys & Tutorials*, **2012**, Vol. 14, No. 2, pp. 311–321.
6. Piqueras, I.; Campelo, J. C.; Ors, R.; Serrano, J. J. Hybrid monitoring of wireless sensor networks. In Proceedings of the IEEE International Conference on Wireless Information Technology and Systems (ICWITS), Maui, HI, USA, 11–16 November 2012; pp. 1–4.
7. Tolle, G.; Culler, D. Design of an Application-Cooperative Management System for Wireless Sensor Networks. In Proceedings of 2nd European Workshop on Wireless Sensor Networks (EWSN), Istanbul, Turkey, 31 Jan. –2 February 2005; pp. 121–132.
8. FAQ – TinyOS Wiki. Available online: <http://tinyos.stanford.edu/tinyos-wiki/index.php/FAQ> (accessed on 15 December 2014).
9. Rost, S.; Balakrishnan, H. Memento: A Health Monitoring System for Wireless Sensor Networks. In Proceedings of 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON), Reston, VA, USA, 25–28 September 2006; Vol. 2, pp. 575–584.
10. Sundaram, V.; Eugster, P.; Zhang, X. Lightweight tracing for wireless sensor networks debugging. In Proceedings of the 4th International Workshop on Middleware Tools, Services and Run-Time Support for Sensor Networks (MidSens 09), Urbana Champaign, IL, USA, 30 Nov. – 4 December 2009; pp. 13–18.

11. Liu, Y.; Liu, K.; Li, M. Passive Diagnostics for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, **2010**, Vol. 18, No. 4; pp. 1132–1144.
12. Ramanathan, N.; Chang, K.; Kapur, R.; Girod, L.; Kohler, E.; Estrin, D. Sympathy for the Sensor Network Debugger. In Proceedings of 3rd ACM Conference on Embedded Networked Sensor Systems (ACM SenSys 2005), San Diego, CA, USA, 2–4 November 2005; pp. 255–267.
13. Ringwald, M.; Römer, K.; Vialetti, A. SNIF: Sensor Network Inspection Framework. Department of Computer Science, ETH Zurich, Technical Report No. 535; 2006.
14. Awad, A.; Nebel, R.; German, R.; Dressler, F. On the Need for Passive Monitoring in Sensor Networks. In Proceedings of 2008 11th EUROMICRO - Conference on Digital System Design Architectures, Methods and Tools (DSD); Parma, Italy, 3–5 September 2008; pp. 693–699.
15. Chen, B.; Peterson, G.; Mainland, G.; Welsh, M. LiveNet: Using Passive Monitoring to Reconstruct Sensor Network Dynamics. In Proceedings of IEEE Conference on Distributed Computing in Sensor Systems (DCOSS), Santorini Island, Greece, 11–14 June 2008; pp. 79–98.
16. Kuang, X.; Shen, J. SNDS: A Distributed Monitoring and Protocol Analysis System for Wireless Sensor Network. In Proceedings of 2010 Second International Conference on Networks Security, Wireless Communication and Trusted Computing (NSWCTC), Wuhan, China, 24–25 April 2010; pp. 422–425.
17. Zhonghua, Z.; Huangfu, W.; Linmin, S. NSSF: A Network Monitoring and Packet Sniffing Tool for Wireless Sensor Networks. In Proceedings of 8th International Wireless Communications and Mobile Computing Conference (IWCMC), Limassol, Cyprus, 27–31 August 2012; pp. 537–542.
18. Garcia, F.; Andrade, R.; Oliveira, C.; de Souza J.N. EPMOST: An Energy-Efficient Passive Monitoring System for Wireless Sensor Networks. *Sensors*, **2014**, 14, pp. 10804–10828.
19. Wireshark. Available online: <https://www.wireshark.org/> (accessed on 20 January 2015)
20. Sommer, P.; Kusy, B. Minerva: Distributed Tracing and Debugging in Wireless Sensor Networks. In Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys), Rome, Italy, 11–14 November 2013.
21. Hossain, M.S.; Lee, W.S.; Raghunathan, V. Spi-Nooper: A Hardware-Software Approach for Transparent Network Monitoring in Wireless Sensor Networks. In Proceedings of the 8th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and Systems Synthesis (CODES+ISSS), Tampere, Finland, 7–12 October 2012; pp. 53–62.

22. Navia, M.; Campelo, J.C.; Bonastre, A.; Serrano, J.J. Low Intrusion Active Hybrid Monitor for Nodes of Sensor Networks. In Proceedings of Workshop on Innovation on Information and Communication Technologies (ITACA-WIICT), Valencia, Spain, 4 July 2014; pp. 111–120.
23. Capella, J.V.; Campelo, J.C.; Bonastre, A.; Ors, R. Proposal of a Reference Model for WSN's Monitoring Platforms. *Journal of Network and Computer Applications*. Received (under revision.)
24. ISO/IEC. Information Technology – Open Systems Interconnection – Basic Reference Model. International Standard ISO/IEC 7498-1, second edition, 1994.
25. STM32F051R8 ARM Cortex-M0 MCU. Available online: <http://www.st.com/web/catalog/mmc/> (accessed on 2 December 2014).
26. STM32F0DISCOVERY Discovery Kit for STM32F051. Available online: <http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF253215> (accessed on 20 January 2015)
27. CMSIS - Cortex Microcontroller Software Interface Standard. Available online: <http://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php> (accessed on 8 December 2014).
28. Keil MDK-ARM Version 5. Available online: <http://www2.keil.com/mdk5/> (accessed on 10 January 2015).
29. 34405A Digital Multimeter, 5½ digit | Keysight (Agilent). Available online: <http://www.keysight.com/en/pd-686884-pn-34405A/> (accessed on 10 January 2015).
30. Gharghan, S.; Nordin, R.; Ismail, M. Energy-Efficient ZigBee-Based Wireless Sensor Network for Track Bicycle Performance Monitoring. *Sensors*, **2014**, *14*, pp. 15573–15592.
31. Molina-Garcia, A.; Fuentes, J.A.; Gómez-Lázaro, E.; Bonastre, A.; Campelo, J.C.; Serrano, J.J. Development and Assessment of a Wireless Sensor and Actuator Network for Heating and Cooling Loads. *IEEE Transactions on Smart Grid*, **2012**, Vol. 3 No. 3, pp. 1992-1202.
32. Lee, D.S.; Liu, Y.H.; Ling, C.R. A Wireless Sensor Enabled by Wireless Power. *Sensors*, **2012**, *12*, pp. 16116–16143.