Sensor-Level Real-Time Support for XBee-Based Wireless Communication

Mihai V. Micea, Valentin Stangaciu, Cristina Stangaciu
Department of Computer and Software Engineering
Politehnica University of Timisoara, Romania
mihai.micea@cs.upt.ro, stangaciu@gmail.com, certejan@gmail.com

Constantin Filote
Computer and Automation Department
Stefan cel Mare University of Suceava, Romania
filote@eed.usv.ro

Abstract— **The ZigBee standard is focused on low-cost, low-power, wireless mesh networking, having a wide applicability mainly in the field of wireless sensor networks. A growing number of such applications require real-time behavior, both at the wireless communication and at the sensor levels. This paper proposes a solution to the problem of providing sensor-level real-time support for wireless platforms using ZigBee-based devices such as the XBee module. The discussion of the experimental results proves the predictable behavior of the XBee sensor platform used as a case study.**

***Keywords-** sensors; real-time; wireless communication; ZigBee; HARETICK kernel*

I. Introduction

Intelligent wired and wireless sensor networks play a key role in a large number of systems and applications currently developed. Many applications require timely response of the sensors and their network, as outdated information could induce control or decision errors with a high potential to harm people or the environment. Real-time sensor networks are used in space, avionics, automotive, security and fire monitoring applications, among many other examples.

A significant amount of research has been recently focused on providing real-time behavior to wireless sensor networks (WSNs) operation. In [1], a scheduling mechanism based on guaranteed time slots (GTSs) is proposed, to provide timing guarantees for message exchange within ZigBee and IEEE 802.15.4 wireless networks [2]. Priority toning strategy is described in [3], as a soft real-time solution to wireless communication over the same type of networks. These works, among many others, focus on solving the real-time problem at medium access control (MAC) level.

The real-time routing aspect has also been investigated. In [4], the authors introduce the real-time power-aware routing protocol (RPAR), which allows the application to specify packet deadlines, leading to dynamic adjustment of the transmission power and the routing decisions. A heuristic solution to finding energy-efficient routes for messages with soft timing specifications is proposed in [5].

Despite the large number of operating systems (OS) proposed for WSNs, providing hard real-time support for the sensor platform still remains an open problem. For example, preemptive multitasking mechanisms are provided by the Nano-RK OS [6]. On the other hand, task preemption could issue predictability problems. Many other WSN operating systems, [7], [8], do not provide native real-time scheduling mechanisms.

A solution to this problem, of providing hard real-time support for wireless sensor platforms, is presented in this paper. The two topical aspects are discussed, based on a case study sensor platform with the XBee wireless module: the operating system level support and the communication application (interface driver).

II. XBee-Based Wireless Sensor Platform

An example sensor platform, able to exchange information over ZigBee wireless networks [2], is the Wireless Intelligent Terminal (WIT) for the CORE-TX platform [9]. For the wireless access, its communication daughterboard uses an XBee module [10] from Digi International, Inc., as shown in Figure 1. This module has been chosen for its features such as low cost, reduced dimensions, good range (up to 100 m indoor and 1500 m outdoor) and ease of use.

An ARM7-based LPC2xxx [11] is used as the main processing unit of the daughterboard. It communicates with the XBee module through the UART interface, with a host PC to exchange debug information, and with the WIT motherboard, through the Serial Peripheral Interface (SPI). As a result, the onboard processor has to execute at least the following main tasks: (1) control of the XBee module and wireless data exchange, (2) communication with the motherboard, which generates and processes the wireless information at the WIT level, (3) exchange and process of debug information with a host PC, as a temporarily development stage, and (4) various local processing operations.

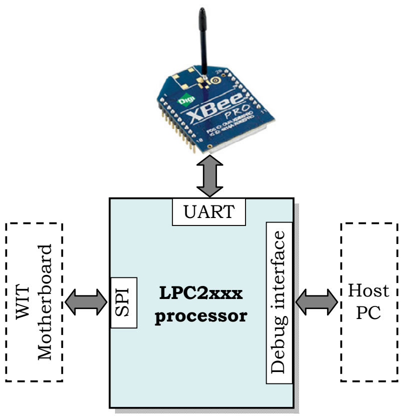


Figure 1. Main blocks of the WIT communication daughterboard.

III. SENSOR-LEVEL REAL-TIME XBEE SUPPORT

At sensor level, real-time support must be equally provided by the operating environment and by the communication application (or the interface drivers). This section presents the sensor-level operating support and the design principles of the real-time XBee wireless communication driver implemented on the CORE-TX WIT.

A. Sensor-Level Operating Support

The operating environment plays a key role in providing the required level of timeliness for the sensor platform behavior. Such a real-time operating system is the HARETICK kernel [12], [13], developed in the DSPLabs, Timisoara.

HARETICK (Hard REal-TIme Compact Kernel) is designed to provide execution support and maximum predictability for critical or hard real-time (HRT) applications on sensing, data acquisition, signal processing and embedded control systems. The kernel provides two distinct execution environments:

a) *HRT context*, as a non-preemptive framework for the execution, with the highest priority, of the HRT tasks.

b) *Soft real-time (SRT) context*, for the execution of SRT (or non real-time) tasks, in a classical, preemptive manner.

The HRT context is based on three essential elements: the real-time clock (RTC), which is assigned to the highest interrupt level, the real-time executive (HDIS), and the scheduler task (HSCD). HDIS has the role of framing the execution of any HRT task, thus it is composed by a prefix (HDIS_Pre) and a suffix (HDIS_Suf). HDIS_Pre is called by the RTC interrupt (it *is* actually the RTC interrupt handler) each time a HRT task is scheduled for execution. HDIS_Suf programs the RTC interrupt to activate at the instant the next HRT task is scheduled. The HRT scheduler currently developed and extensively tested implements the so called Fixed-Execution Non-Preemptive (FENP) algorithm [13], providing strong guarantees that all the HRT tasks meet their temporal specifications, even under the worst–case operating conditions.

A particular model has been introduced to specify all the HRT tasks on the HARETICK kernel [12], [13]. The ModX (eXecutable MODule) models the periodic, modular HRT task, with complete and firm temporal specifications, scheduled and executed in the non-preemptive HRT context of the kernel. Among the key temporal parameters of the ModX M_i, are its period, T_{Mi}, and worst case execution time (WCET), C_{Mi}. All these parameters have to be known or derived at the offline application analysis stage, when the feasibility tests are performed, prior to executing the system on the target platform.

B. Real-Time XBee Driver

Due to the hard requirements regarding execution predictability, timing and code analysis, conventional architectures and programming strategies have to be adapted when writing a real time application. One of the most important aspects to be considered when writing a real time task is to provide the means to analyze and calculate its worst case execution time (WCET) [14], [15].

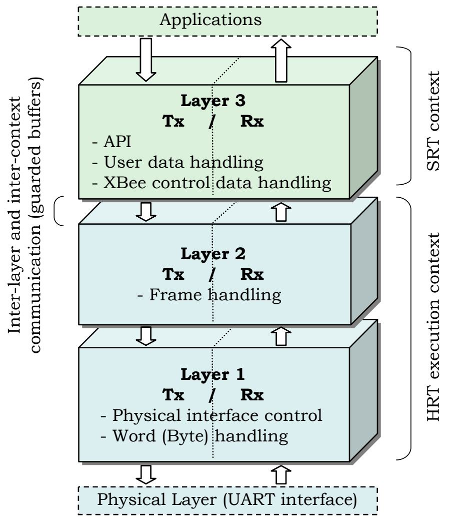


Figure 2. General architecture of the real-time XBee driver.

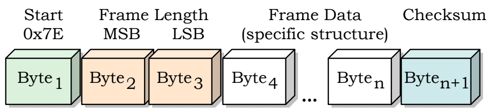


Figure 3. XBee UART frame structure.

The real-time XBee driver has been designed based on a layered approach, following the ISO OSI Model principles and considering the particular communication protocol stack which interfaces the target processor to the XBee module. Furthermore, the driver is also divided between the 2 execution contexts provided by the HARETICK kernel, the HRT and the SRT contexts. The general architecture of the XBee driver is depicted in Figure 2.

Each of the three layers managing the communication protocol has a transmission (Tx) and a reception (Rx) component, with similar functionalities. The first two layers are directly related to the timing requirements of the communication interface (the UART) and, therefore, have been designed to be scheduled and executed as the XBee ModX, within the HRT context of the HARETICK kernel.

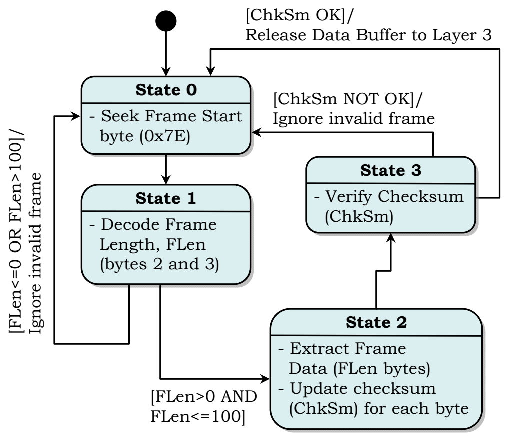

[ChkSm OK]/
Release Data Buffer to Layer 3
State 0
- Seek Frame Start byte (0x7E)
[ChkSm NOT OK]/
Ignore invalid frame
State 3
- Verify Checksum (ChkSm)
State 1
- Decode Frame Length, FLen (bytes 2 and 3)
[FLen<=0 OR FLen>100]/
Ignore invalid frame
State 2
- Extract Frame Data (FLen bytes)
- Update checksum (ChkSm) for each byte
[FLen>0 AND FLen<=100]

Figure 4. Layer 2 frame receive automaton.

Layer 1 handles the physical interface control and the raw data at word level (where a word has a 10 bit structure: START bit + DATA byte + STOP bit).

Layer 2 manages the XBee frames structure (see Figure 3). It implements a distinct but similar finite-state automaton, for the Tx and the Rx components. The basic diagram of the frame receive automaton at the layer 2 is depicted in Figure 4. After successfully extracting the data from a frame, the receive automaton releases the data buffer to layer 3 for further interpretation and processing. Due to the fact that layer 2 operates (along with layer 1) as a ModX, in the HRT context of the kernel, while layer 3 is executed within the SRT context, their data communication is based on guarded (or flagged) buffers [16].

Layer 3 has several important roles, including (a) handling the user/application data, (b) constructing and decoding the XBee status and control data, and (c) application programming interface, providing the required functions and configuration parameters to the upper (application) layer. For example, the XBee modem status data is structured over 2 bytes: <API Id> (0x8A) + <Status byte>. <Status byte> can have the following values: 0 for "Hardware reset"; 1 for "Watchdog timer reset"; 2 meaning "Associated" to a Personal Area Network (PAN); 3 for "Disassociated"; 4 for "Sync lost"; 5 meaning "Coordinator realignment"; 6 for "Coordinator started". Layer 3 is also capable of handling the various AT commands featured by the XBee protocol.

Since layer 3 handles data frame exchanged through buffers, both with the application layer and with layer 2, it is not directly affected by the timing specifications of the physical interface. Therefore, layer 3 task has been designed as a SRT task, and, as a result, its execution does not need to be added to the HRT processor utilization, which should be kept as low as possible for a feasible schedule of this context.

IV. Discussion of the Experimental Results

The WIT communication daughterboard presented in Section II and in Figure 1 has been used to measure and validate the real-time behavior of the sensor-level communication. A set of HRT and SRT tasks have been programmed, analyzed and implemented to provide the basic data exchange capabilities. The tasks directly interacting with the hardware components which require timely behavior have been implemented as ModXs: the HSCD (HRT Scheduler), the XBee ModX (containing the first two layers of the proposed driver), the SPI and the Debug ModXs. The tasks in the SRT execution context include the XBee_Layer3 task, the SPI_Layer4 task and a set of data processing and general debug tasks. They are scheduled with a simple loop algorithm.

TABLE I presents the experimental setup, along with some of the most important parameters taken into consideration or calculated at the offline system/application analysis step, prior to launching it on the target platform. The worst execution scenario has been considered the case when the maximum possible data is transferred through the XBee module, thus involving the use of all the corresponding buffers at their maximum capacities.

TABLE I. System and XBee ModX Configuration Parameters

Parameter	Value
CPU Clock	58.9824 MHz
HARETICK Real-Time Clock (RTC)	14.7456 MHz
Total ModXs on target system (HSCD scheduler included)	4
Prefix executive (HDIS_Pre) WCET	112 RTC cycles (7.59 µs)
Suffix executive (HDIS_Suf) WCET	68 RTC cycles (4.61 µs)
Scheduler ModX (HSCD) WCET	4052 RTC cycles (274.79 µs)
XBee ModX WCET	4568 RTC cycles (309.79 µs)
XBee UART transfer rate	57600 bps
XBee ModX minimum frequency	360 Hz

TABLE II. Measurement Results for the Worst Case and Typical Execution Times

Parameter	Approx. Value
XBee ModX WCET	305 µs
Xbee ModX typical execution time	75 µs
Scheduler ModX (HSCD) WCET	271 µs
HSCD typical execution time	271 µs
Debug ModX WCET	41 µs
SPI ModX WCET	21 µs
HDIS executive (Prefix + Suffix) WCET	11.76 µs

To avoid losing information during the wireless transactions, the minimum execution frequency of the XBee ModX has to be respected. This frequency has been calculated taking into consideration the operating characteristics of the UART interface between the XBee module and the target processor:

$$F_{\min \text{XBee}} = TR_{\text{UART}} / (SizeW_{\text{FIFO}} \cdot WLen_{\text{UART}}), \quad (1)$$

where TR_{UART} is the UART transfer rate (57600 bps), $SizeW_{\text{FIFO}}$ is the size in words of the hardware UART FIFO buffer (16 words), and $WLen_{\text{UART}}$ is the number of bits per UART frame (10 bits/word). From the minimum frequency of the XBee ModX, the maximum value of its period as a task can be derived:

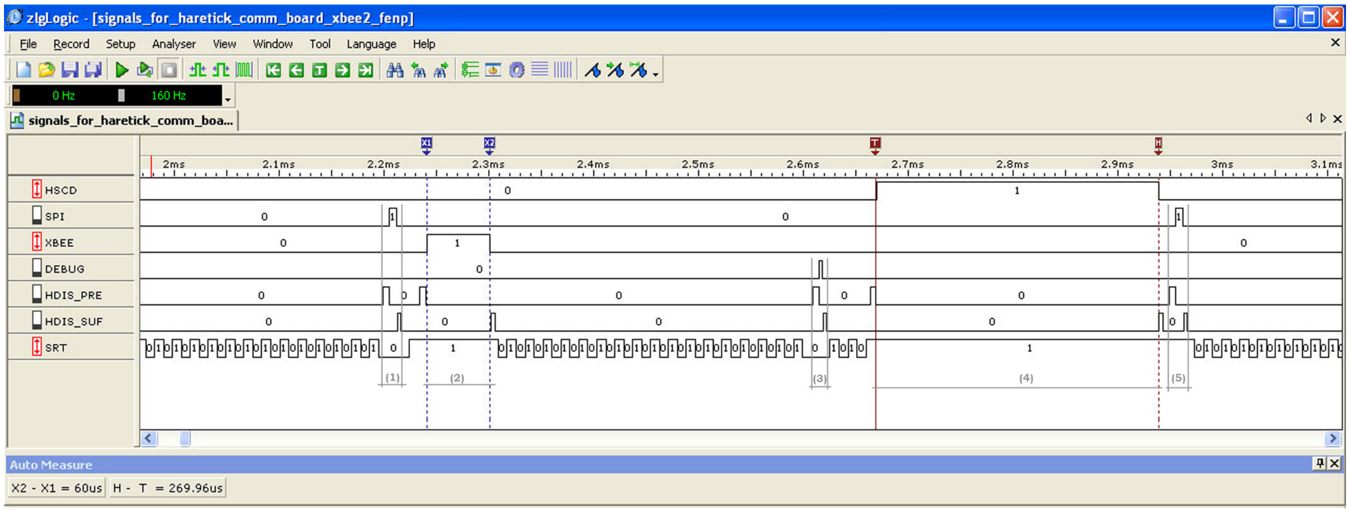
$$T_{\max \text{XBee}} = 1 / F_{\max \text{XBee}} = 2777.77 \text{ µs}, \quad (2)$$

and the minimum processor utilization of the XBee ModX can be calculated:

$$U_{\min \text{XBee}} = \frac{C_{\text{XBee}}}{T_{\max \text{XBee}}} = \frac{\text{WCET}_{\text{XBee}}}{T_{\max \text{XBee}}} = 11.15\,\%. \quad (3)$$

TABLE II shows the worst case and the typical execution times measured during the experiments, for the HRT context of the system. As seen in the table, the typical execution of the HSCD ModX is equivalent to the worst case, as the scheduler applies each time the same algorithm to a constant set of tasks (ModXs) and temporal parameters.

It is worth mentioning that, although the HDIS executive is distinctly shown from the execution times of the ModXs in TABLE I and TABLE II, the actual (total) WCET of any ModX includes also the HDIS execution time. This total WCET is considered when the ModX is scheduled (or at the offline analysis step).



zlgLogic - [signals_for_haretick_comm_board_xbee2_fenp]
File Record Setup Analyser View Window Tool Language Help
0 Hz
160 Hz
signals_for_haretick_comm_boa...
X1
X2
T
H
2ms
2.1ms
2.2ms
2.3ms
2.4ms
2.5ms
2.6ms
2.7ms
2.8ms
2.9ms
3ms
3.1ms
HSCD
SPI
XBEE
DEBUG
HDIS_PRE
HDIS_SUF
SRT
(1)
(2)
(3)
(4)
(5)
Auto Measure

X2 - X1 = 60us | H - T = 269.96us

Figure 5. Logic analyzer capture of the system operation.

Figure 5 shows one of the system operation timeframes captured with a logic analyzer. It depicts the execution of the HRT context, with its 4 ModXs (HSCD, SPI, XBee and Debug) and the two components of the executive (HDIS_Pre and HDIS_Suf), which frame the executions of each ModX. The SRT context operation has also been captured.

Several interesting intervals have been marked in the figure. The execution of the SPI ModX, prefixed by the HDIS_Pre component of the executive, and ended by the HDIS_Suf, is highlighted with (1) and (5). Interval (2) marks the execution of the XBee ModX (elapsing approx. 60 µs, here). It also is framed by HDIS_Pre and HDIS_Suf. The Debug ModX is outlined at (3), while (4) corresponds to the execution of the HRT scheduler (HSCD), measured here at around 270 µs.

The SRT context captured in Figure 5 shows a cyclic behavior, due to the implemented scheduling policy. The diagram also outlines the intervals in which the SRT tasks are interrupted by the higher priority HRT context: (1) – (5).

ACKNOWLEDGMENT

This work is supported in parts by the Romanian Ministry of Education and Research, through the grant PNCDI II PDP 22-137/2008-2011.

REFERENCES

[1] J. Francomme, G. Mercier, and T. Val, "A simple method for guaranteed deadline of periodic messages in 802.15.4 cluster cells for control automation applications", Proc. IEEE Conf. Emerging Technologies and Factory Automation (ETFA 2006), IEEE Press, Sept. 2006, pp. 270-277, doi: 10.1109/ETFA.2006.355357.

[2] IEEE Standard, "Specific requirements part 15.4: Wireless medium access (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)", IEEE-SA, 2006, Online: http://standards.ieee.org/about/get/802/802.15.html.

[3] T.H. Kim and S. Choi, "Priority-based delay mitigation for event-monitoring IEEE 802.15.4 LR-WPANs", IEEE Communication Letters, vol. 10, no. 3, IEEE, Mar. 2006, pp. 213-215.

[4] O. Chipara, Z. He, G. Xing, et al., "Real-time Power-Aware Routing in Sensor Networks", Proc. 14th Intl. Wshop. Quality of Service (IWQoS 2006), New Haven, CT, Jun. 2006, pp. 83-92.

[5] P.K. Pothuri, V. Sarangan, and J.P. Thomas, "Delay-constrained, energy-efficient routing in wireless sensor networks through topology control", Proc. IEEE Intl. Conf. Networking, Sensing and Control (ICNSC 2006), Ft. Lauderdale, FL, 2006, pp. 35-41.

[6] A. Eswaran, A. Rowe, and R. Rajkumar, "Nano-RK: An energy-aware resource-centric rtos for sensor networks", Proc. 26th IEEE Intl. Real-Time Systems Symp. (RTSS 2005), Miami, FL, Dec. 2005, pp. 256-265, doi: 10.1109/RTSS.2005.30.

[7] Q. Cao, T.F. Adbelzaher, J.A. Stankovic, and T. He, "The LiteOS operating system: towards Unix-like abstractions for wireless sensor networks", Proc. 7th Intl. Conf. Information Processing in Sensor Networks (IPSN 2008), 2008, doi: 10.1109/IPSN.2008.54.

[8] S. Bhatti, J. Carlson, H. Dai, et al., "MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms", ACM/Kluwer Mobile Networks and Applications Journal (MONET), vol. 10, no. 4, Aug. 2005, pp. 563-579.

[9] R. D. Cioarga, M. V. Micea, B. Ciubotaru, D. Chiciudean, and D. Stanescu, "CORE-TX: Collective Robotic Environment - the Timisoara Experiment", Proc. 3-rd IEEE Intl. Symp. Applied Computational Intellig. Informatics (SACI 2006), Timisoara, Romania, May 2006, pp. 495-506.

[10] Digi International, "XBee^{TM}/XBee-PRO^{TM} OEM RF modules: Product manual v1.xAx - 802.15.4 protocol", Digi International, Inc., May 2007, Online: www.digi.com.

[11] NXP Semiconductors, "UM10114: LPC21xx and LPC22xx user manual", Rev. 03, NXP Semiconductors N. V., Apr. 2008, Online: http://www.nxp.com/documents/user_manual/UM10114.pdf.

[12] M. V. Micea, and V. I. Cretu, "Highly predictable execution support for critical applications with HARETICK kernel", Intl. J. Electron. Commun. (AEU), vol. 59, no. 5, Elsevier GmbH, Jena, Germany, May 2005, pp. 278-287, doi: 10.1016/j.aeue.2005.05.011.

[13] M. V. Micea, V. I. Cretu, and V. Groza, "Maximum predictability in signal interactions with HARETICK kernel", IEEE Trans. Instrum. Meas., vol. 55, no. 4, IEEE, Aug. 2006, pp. 1317-1330.

[14] P. Puschner, "Algorithms for dependable hard real-time systems", Proc. 8th Intl. Wshop. Object-Oriented Real-Time Dependable Systems (WORDS 2003), IEEE Press, Jan. 2003, pp. 26-31.

[15] R. Wilhelm, J. Engblom, A. Ermedahl, et al., "The worst-case execution time problem – overview of methods and survey of tools", ACM Trans. Embed. Comput. Syst., vol. 7, no. 3, 2008, pp. 36-88.

[16] M. V. Micea, C. Certejan, V. Stangaciu, R. D. Cioarga, V. I. Cretu, and E. M. Petriu, "Inter-task communication and synchronization in the hard real-time compact kernel HARETICK", Proc. IEEE Intl. Wshop. Robotic Sensors Environments (ROSE 2008), Ottawa, Canada, Oct. 2008, pp. 19-24, doi: 10.1109/ROSE. 2008.4669174.