Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, 2005 (pp47-52)

Low-power aware design: Topics on low battery consumption

ITZIAR MARÍN[†], EDUARDO ARCEREDILLO[†], JAGOBA ARIAS[‡], AITZOL ZULOAGA[‡], IKER LOSADA[‡]
[†]Mechatronics and Precision Department, Fundación Tekniker
Avda. Otaola 20 Apdo 44, 20600 Eibar
[‡]Electronics and Telecommunications Department, University of the Basque Country
Alda. Urquijo s/n, 48013 Bilbao
SPAIN

[†]http://www.tekniker.es, [‡]http://www.ingenierosbilbao.com

Abstract: - In this paper we present the design and implementation of a robust, marketable, general purpose and flexibly configurable device that offers three years of battery lifetime, GSM communications, local and/or remotely configurable behaviour, with wired and/or remote firmware-update facility and "one month before battery depletion" warning. It has been designed based upon WSN's power-aware techniques and empirical obtained results borne laboratory measurements out.

Key-Words: - Low power design, power-aware, wireless sensor network (WSN), battery modelling.

1 Introduction

A great evolution in electronic devices toward low power consumption designing has happened lately as so many specific platforms that have been developed show, such as Telos (based upon TinyOS), EYES, etc [1,2].

The correct election of the microprocessor is essential. Adequate microprocessor election is a very hard task and it has to be carried out very carefully. Upon this selection will depend the rest of the components design and performance.

In this paper we present a very low power consumption device with three years of battery lifetime, GSM communications (point-to-point), local and/or remotely configurable behaviour, with wired and/or remote firmware-update facility and "one month before battery depletion" warning. These all characteristics make it a general purpose and flexibly configurable device that even offers power supply to external sensors. This way, it becomes a completely autonomous and functional system that does not need of other external power supply for the whole system's operation. Its block diagram is presented in figure 1.

Starting from scratch, we have developed a power-aware operating system, for a robust marketable product. A fundamental aim in this development was the low power consumption so that the battery lasts many years. Market demands lasting devices with great calculation capacity and wireless remote control.

Although this device is not a WSN's node, it had to implement similar power-aware design criteria. That's why we found inspiration in WSN's low power-aware philosophy.

The rest of the document is organized as follows. In section 2, power-aware design steps are listed. In section 3 device's power source is presented and section 4 offers comparison between real measurements and theoretical estimations. Section 5 summarizes and presents conclusions and finally, we present some future work in section 6.

2 Power-Aware Design Steps

Low power aware design has to be done in the correct order: from the application to the lower layers. If application is not power-aware designed, whatever done in the rest of the layers will be useless [3]–[5].

2.1 Suitable Application Design

It makes no sense to power-aware design every step in hardware and operating system layers and let application layer be non-power-aware designed. If so, the whole

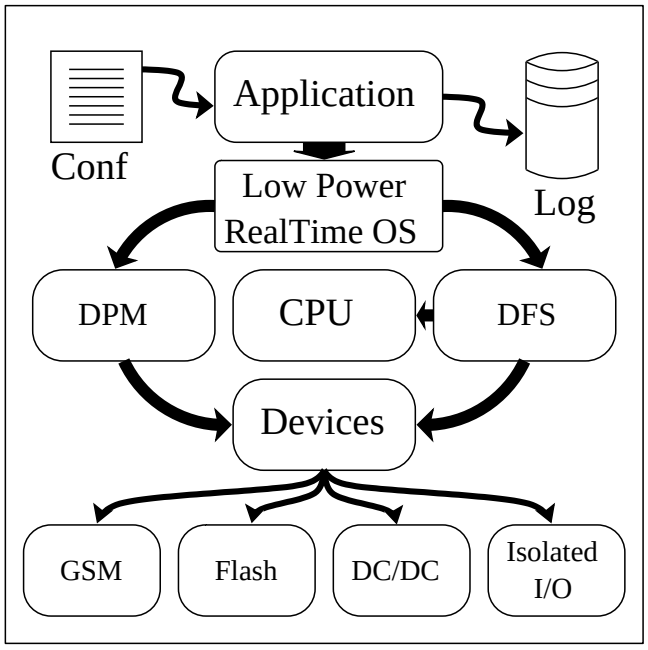

Fig.1. Our device's block diagram.

system will not be low power consuming, no matter how hard you had tried in lower layers.

Our device has to work in a remote place, unattended and has to last many years so its actions have to be punctual and consume very low power. The only way to achieve this, is with a very low duty cycle as shown in 2.2.

In order to consume as less energy as possible in wireless communications, we have implemented GSM data compression techniques, achieving a faster data transmission. Moreover, we have optimised WSN's transfer performance: instead of transmit every new measured data separately (as WSNs do), we store information in a external flash chip and send it from time to time, improving the whole transfer process.

2.2 Low-Power Real Time Operating System (LPRTOS)

Although there are some operating systems especially designed for power-aware devices, such as, TinyOS, and programming languages, such as nesC, we chose to implement a brand new one because platform portability costs would have been too high and consequently, not affordable in our design.

In order to reduce Operating System's consumption, there some techniques that offer power consumption reduction:

- *Dynamic Power Management* (DPM) algorithms try to reduce system consumed energy making components go to low power consumption state selectively. This can be achieved thanks to a previous analysis of the system duty cycle and each element's load [6]–[9].
- Both *Dynamic Voltage Scaling* (DVS) and *Dynamic Frequency Scaling* (DFS) algorithms reduce power consumption varying CPU or even peripherals functional characteristic: voltage or frequency, respectively [10]–[15]. They make computational unit work faster (with a higher voltage or frequency) when it has load, and slowly when there is no computational activity. This way, power consumption gets reduced to its minimum.

In the development of our device, we have implemented DPM and DFS, but not DVS, because the latter did not offer an evident improvement for our application's objective so it was not worth the implementation.

Moreover, our device has a multi-task non-preemptive operating system over a hardware abstraction layer. Our application has device overlapping tasks and the LPRTOS has to manage these devices' usage. Individualised resources management has been implemented in order to put in its lowest power consumption states those devices that offer that facility and switch off completely those that draw too much current from batteries and it is not affordable to keep them running. For example, GSM module has a great power consumption in idle state, around 100 μA, which makes it impossible to keep it awake continuously.

The operating system has to be designed in order to have a extremely low duty cycle and consequently, there is a very important need for correct choice of the idle state consumption [16,17]. The idle state is the least consuming one in which devices can react acceptably quick to external stimuli. The whole device should be in such a state that would let it consume as less power as possible at the same time as keep it in a quick-react state. That's why it's necessary to carefully determine idle states of every part of the device in order to have a very low idle consumption.

The operating system has to schedule tasks' executions so they last as few seconds as possible. If one task execution needs too much time, that task is divided into different states, and every time the operating system determines it has time to give to that task, that task goes forward within its internal state machine until finishing.

Let A and B be two different tasks. Both A and B need more than one system loop to completely execute

themselves. The Operating System in every loop iteration, takes a look at which one of both has to be executed this time (every task has an associated counter that let the OS know how many times it has been executed). This way, execution is alternate and gives multi-task fashion.

Furthermore, it has to be taken into account that power aware design must be also done at system level, not only at component level. That is, the power saving of a given component must be scaled by its percentage contribution in an entire system. If a component only draws 2 % of the system power, a 50 % reduction in its power consumption would save 1 % of the global system power. That's why it is more efficient to identify power consuming devices more than analyse parts one by one [18].

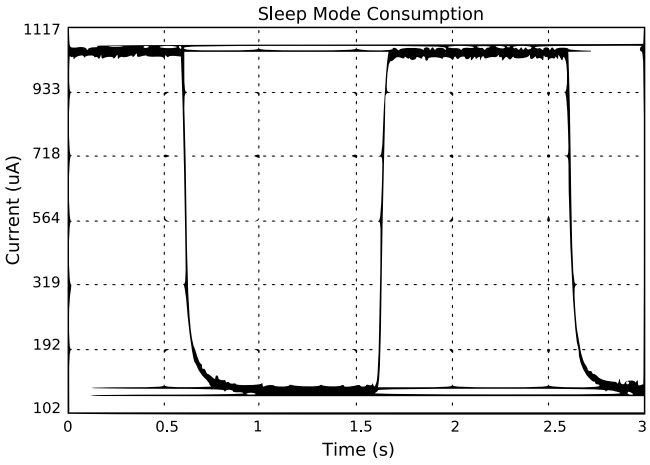

Fig.2. Consumed power in idle and active states.

2.3 Device selection

Selection of ultra-low power consumption specifically designed devices has been a general design rule in the implementation of this device [19]. As a example, flash chips with deep-sleep mode that draw just a few μA and offer a very low reaction time. Moreover, it's considerably more efficient to select power-aware devices than making non-power-aware devices behave as if they were power-aware designed. This selection process must be slow and meticulous in order to achieve satisfactory results [20].

- GSM module: GSM module has been selected taking into account its integration interface, global consumption and its OEM (*Original Equipment Manufacturer*) characteristics.
- Flash chips: Flash chips were selected due to their low power consumption states. Their deep-power-down command puts them into a very low power consuming state and offers a very low wake-up time (see 2.4).
- Micro-processor selection: μP has to be selected following a set of requirements:
    1) Low power consumption function mode: Our application needs to change between active mode and different low-power modes (LPM) depending on power consumption requirements. Those different deep sleep states offer flexibility in power consumption selection. Needless to say, there is a trade-off between those two characteristics. As some calculations show, CPU's idle state consumes more that one third of the total power consumption, so low-power micro-processor is a must in order to have a power aware global performance.
    2) Individualised peripheral control: each ADC pin, each pin of every port, and some other pins have to be individually controllable in order to select as input only the desired pins. Moreover, current low-power μPs offer a maximum leakage current of a few nA.

Although there are some other new technologies, we decided to work with Texas Instruments' MSP430 family, which offers many of the needed capabilities for our design and now it is being used in many famous platforms, such as EYES, Telos, etc [1,2].

2.4 Energetic States Management

Apart from the amount of drawn current, changes between two different energetic states generate a very high delay that should be avoided. There's a trade-off between wake-up times (which are treated as delays) and power consumption. It's very important to decide wisely the instant when the CPU should go to sleep.

The decision is similar to the one you have to make when you stop the car in traffic lights: stop the car engine or wait with the engine on. If you wait with the engine on, you will consume as long as you stay waiting but won't have delay time in running from the traffic lights. On the other hand, if you choose to stop the engine, you will save energy as long as you stay waiting but you will draw a large amount of current in the instant of starting the engine and you will have a great delay in getting ready [21].

Oscillator election can change dramatically the wake-up time, and that's why clock speed should be adapted

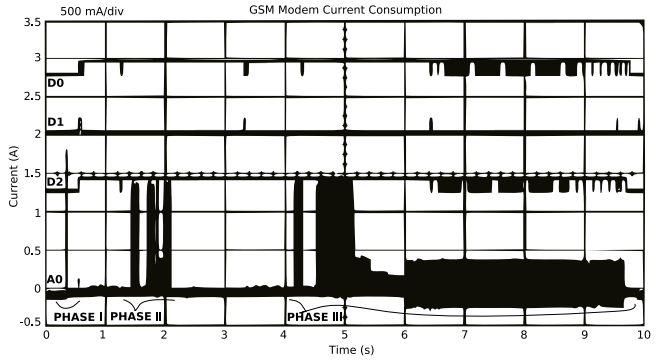

Fig.3. Oscilloscope's trace of GSM consumption.

to every moment's needs. In our device we use three different clocks: internal generator (around 6 MHz) to switch on the module; when precision is needed, then the external 4 MHz crystal is used and this way the system works at high and stabilized speed. When the module is sleeping only the real-time 32 kHz clock is running [22,23].

Apart from the wake-up time, clock speed is also important in CMOS power consumption as the latter is proportional to the crystal frequency as equation 1 shows. That's why it's not always possible to work at maximum speed because power resources could get depleted.

$$Power = CfV^2 \tag{1}$$

3 Batteries

A very important part of our long-living device is its batteries. Our device has non-rechargeable batteries as a design requirement, but there are more sophisticated solutions that could offer a quasi-permanent performance, such as the one presented in [24]. Batteries are not simple devices: their operation depends on many factors, including battery dimensions, type of electrode material, temperature, discharge profile, etc.

3.1 Battery family selection

There are many different types of batteries nowadays and not all of them have the desired characteristics.

1) *Size*: Our design has size restrictions because it has fixed dimensions, and this gives few choices in battery selection process.
2) *Capacity*: We have to last many years and any battery can't give such a quantity of constant current.
3) *Voltage maintenance*: Our devices work in between 6 and 7.2 V; lower voltage values would not be enough.
4) *Maximum peak current*: The selected batteries must withstand the maximum peak current drawn from the GSM module (around 2 A), during the network connection process.

For example, if we use a pair of AAA 400 mA-h alkaline batteries to supply our μP, apart from the 2 μA of current drawn from the CPU, these batteries need a regulator in order to behave correctly, which means a total current of 22 μA. Considering that these kind of batteries can only offer 40% of their total capacity, the lifetime of the system would just be a couple of years. However, a 220 mA-h lithium cell can offer the 90% of its total capacity and does not need a regulator, so the system would have around 10 years of operation. As a result, a lithium battery with less total capacity can offer a longer lifetime than a more powerful alkaline battery.

3.2 Battery behaviour

Apart from correct battery selection, battery characterization is needed in order to know how the nonidealities can change its performance. For example, some kind of batteries suffer the *Relaxation effect*, caused by some nonidealities, that can partially mitigate the consequence that a high discharge rate can produce. If the discharge current is reduced or cut off, battery capacity can get recovered from the partial depletion suffered because of the discharge [25]–[27]. Hence, correct battery type selection is fundamental for a correct behaviour of the device [28,29].

3.3 Battery modelling

Moreover, it is very important to know as close as possible, the way the battery is being discharged and how much capacity is still remaining. What we have done in the design is to implement an application-level battery consumption prediction model in order to know the remaining battery capacity [30,31]. We have modelled the most consuming tasks, the power consumption in sleep-state (LPM3 with only external LF clock running—see figure 2) and measured the total current consumption. We have performed an analysis in depth of every part that is involved in this device's functionality, and in

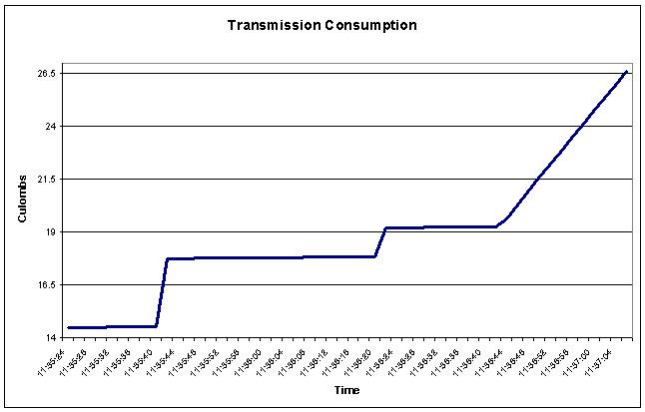

Fig.4. Estimated GSM consumption.

our application circumstances, only CPU's and GSM module's consumptions appear to be relevant.

We know how much current it has been drawn during sleeping and how much current it is drawn whenever CPU executes every task, so we subtract from the total current capacity of the batteries, the measured consumption.

4 Field Tests

Field tests have reaffirmed our theoretical estimations and have shown that our device works quite similar to what it was designed to. Only minor changes have been done to tune the algorithm.

Every task consumption has been measured with the necessary resolution degree. Low-power aware designed devices cannot waste their precious time and energy in evaluating an excessively accurate measure of their consumption, so it has been necessary to balance a trade-off. As shown in figure 3 and 4, power consumption has been very accurately calculated, using non-invasive methods (measures were taken with a Hall effect probe).

Our application has to warn when it has only one month of battery power left. This calculation has to be done regarding the current programmed configuration, which determines the behaviour of the device. Current consumption is not the same for every configuration, as each configuration can have different type and quantity of tasks. That's why remaining current capacity has to be calculated on-line based upon estimated consumptions of every task, as explained in section 3. That's why it is necessary to have a discharge foresight. Even so, we have used the most pessimistic measures, to be sure to warn at least one month before.

5 Conclusions

As explained in this paper, low-power aware design has to be done from the application to the lower layers in order to achieve acceptable results. Moreover, in the whole design process, energetic management is crucial for achieving consumption objectives, apart from the correct selection of the power supply, i.e. batteries. WSN's research community is working hard in low-power aware design apart from energy scavenging methods, which seem to be the basis of future wireless autonomous devices.

6 Future work

As explained above, our application has to warn one month before battery depletion. If battery depletion could be more exactly foreseen, more accurate measures would be used and it would be possible to make the most of battery capacity.

It is very difficult to find the relationship between GSM module's exact power consumption and actual coverage, so measures have to be done with average coverage and the least operative coverage. A more accurate knowledge of the actual coverage would offer a better estimation of battery lifetime and a more exact estimation of GSM module's consumption.

There are more improvements in power-aware designing, for example, measuring the exact amount of current that CPU, memory chips or communication drivers draw from the battery each time they perform their functionality.

Finally, its important to remember that it is not only important to manage existing resources correctly in order to avoid wasting them and reserve some of them for critical situations, but also regenerating or *harvesting* consumed energy as much as possible, to make the network live longer [32].

References:

[1] [www.tinyos.net]. Tinyos operating system home page. 2005.

[2] [www.xbow.com]. Crossbow technology inc. home page. 2005.

[3] T. Šimunic, L. Benini, and G. de Micheli. Energy-efficient design of battery-powered embedded systems. *Proceedings of the International Symposium on Low Power Electronics and Design, ISLPED*, pages 212–217, 1999.

[4] G. Lim. Trends in low power digital system-on-chip designs. *Proceedings of the International Symposium on Quality Electronic Design*, page 73, 2002.

[5] L. Niu and G. Quan. Reducing both dynamic and leakage energy consumption for hard real-time systems. *Proceedings of the International conference on Compilers, Architecture, and Synthesis for Embedded Systems, CASES*, pages 140–148, 2004.

[6] L. Benini, A. Bogliolo, and G. de Micheli. A survey of design techniques for system-level dynamic power management. *The Morgan Kaufmann Systems On Silicon Series Readings in hardware/software co-design Section: Analysis and estimation*, pages 231–248, 2001.

[7] H. Muller and C. Randell. An event-driven sensor architecture for low power wearables. *Workshop on Software Engineering for Wearable and Pervasive Computing*, June 2000.

[8] T. Šimunic, L. Benini, P. Glynn, and G. de Micheli. Event-driven power management. *Proceedings of International Symposium on System Synthesis*, 1999.

[9] P. Pop, P. Eles, and Z. Peng. *Analysis and Synthesis of Distributed Real-Time Embedded Systems*. Kluwer Academic Publishers, 2004.

[10] W. Kim, D. Shin, H.-S. Yun, J. Kim, and S. Lyul Min. Performance comparison of dynamic voltage scaling algorithms for hard real-time systems. *Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, page 219, 2002.

[11] J. Pouwelse, K. Langendoen, and H. Sips. Dynamic voltage scaling on a low-power microprocessor. *Proceedings of the ACM International Conference on Mobile Computing and Networking, MobiCom*, pages 251–259, 2001.

[12] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. *Proceedings of the annual conference on Design Automation*, pages 275–280, 2004.

[13] I. Brynjolfson and Z. Zilic. Dynamic clock management for low power applications in fpgas. *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 139–142, 2000.

[14] S. P. Mohanty, N. Ranganathan, and V. Krishna. Datapath scheduling using dynamic frequency clocking. *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pages 65–70, April 2002.

[15] S. P. Mohanty, N. Ranganathan, and S. K. Chappidi. Peak power minimization through datapath scheduling. *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, pages 121–126, February 2003.

[16] L. Chakrapani, P. Korkmaz, V. J. Mooney III, K. V. Palem, K. Puttaswamy, and W. F. Wong. The emerging power crisis in embedded processors: What can a (poor) compiler do? *International conference on Compilers, Architecture, and Synthesis for embedded systems, CASES*, November 2001.

[17] F. Gruian and K. Kuchcinski. Uncertainty-based scheduling: energy-efficient ordering for tasks with variable execution time. *Proceedings of the International Symposium on Low Power Electronics and Design, ISLPED*, pages 465–468, 2003.

[18] J. Liu, P. H. Chou, N. Bagherzadeh, and F. Kurdahi. Power-aware scheduling under timing constraints and slack analysis for mission-critical embedded systems. *Proceedings of the 38th ACM/IEEE Design Automation Conference, DAC*, pages 840–845, 2001.

[19] K. Roy and S. C. Prasad. *Low Power CMOS VLSI Circuits*. John Wiley and Sons, 2000.

[20] Geoffrey C-F Yeap. Leakage current in low standby power and high performance devices: trends and challenges. *Proceedings of the 2002 international symposium on Physical design*, pages 22–27, 2002.

[21] H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley and Sons, 2005.

[22] H. Wu, B. Ravindran, E. D. Jensen, and P. Li. Cpu scheduling for statistically-assured real-time performance and improved energy efficiency. *Proceedings of the IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, pages 110–115, September 2004.

[23] C. Lynch and F. O' Reilly. Processor choice for wireless sensor networks. *Workshop on Real-World Wireless Sensor Networks, REALWSN*, June 2005.

[24] X. Jiang, J. Polastre, and D. Culler. Perpetual environmentally powered sensor networks. *Proceedings of ISPN/SPOTS*, April 2005.

[25] C. F. Chiasserini and R. R. Rao. Pulsed battery discharge in communication devices. *Proceedings of the ACM International Conference on Mobile Computing and Networking, Mobicom*, 1999.

[26] C. F. Chiasserini and Ramesh R. Rao. A traffic control scheme to optimize the battery pulsed discharge. *IEEE Military Communications Conference, MILCOM*, 1999.

[27] S. Park, A. Savvides, and M. B. Srivastava. Battery capacity measurement and analysis using lithium coin cell battery. *Proceedings of the International Symposium on Low Power Electronics and Design, ISLPED*, 2001.

[28] I. Buchmann. *Batteries in a portable world*. Cadex Electronics Inc., 2nd edition, May 2001.

[29] C. F. Chiasserini and R. R. Rao. A model for battery pulsed discharge with recovery effect. *Wireless Communications and Networking Conference, WCNC*, 1999.

[30] C. Krintz, Y. Wen, and R. Wolski. Application-level prediction of battery dissipation. *Proceedings of the International Symposium on Low Power Electronics and Design, ISLPED*, pages 224–229, 2004.

[31] C. F. Chiasserini and R. R. Rao. Energy efficient battery management. *IEEE Journal on selected areas in communications*, 19(7), July 2001.

[32] S. Roundy, P. Kenneth Wright, and J. M. Rabaey. *Energy Scavenging for Wireless Sensor Networks with Special Focus on Vibrations*. Kluwer Academic Publishers, 2004.