

Energy-aware strategies in real-time systems for autonomous robots^{*}

Giorgio Buttazzo, Mauro Marinoni, and Giacomo Guidi

University of Pavia, Pavia, Italy,
{buttazzo,mauro.marinoni,giacomo.guidi}@unipv.it

Abstract. Most battery operated robots have to work under timing constraints to exhibit a desired performance and must adopt suitable control strategies to minimize energy consumption to prolong their lifetime. Unfortunately, energy saving strategies tend to reduce resource availability and, hence, degrade robot performance. As a consequence, an integrated approach is needed for balancing energy consumption with real-time requirements. In this paper, we present an integrated approach to energy management in real-time robot systems to prolong battery lifetime and still guarantee timing constraints. The method is applied to a six-legged robot controlled by a PC104 microprocessor and equipped with a set of sensors for the interaction with the environment. Specific experiments are reported to evaluate the effectiveness of the proposed approach.

1 Introduction

With the progress of technology, cost and size of robot systems are reducing more than ever, not only for wheeled vehicles, but also for walking machines, which can be used to work in open environments on more irregular terrains. This enables the development of distributed systems consisting of teams of robots, which can cooperate to collect information from the environment and perform a common goal. Typical applications of this type are aimed at monitoring, surveillance, searching, or rescuing. In this type of activities, the use of a coordinated team of small robots has many advantages with respect to a single bigger robot, increasing the probability of success of the mission.

On the other hand, the use of small robot systems introduce a lot of new problems that need to be solved for fully exploiting the potential benefits coming from a collaborative work. Most of the problems are due to the limited resources that can be carried onboard by a small mobile robot. In fact, cost, space, weight, and energy constraints, impose the adoption of small microprocessors with limited memory and computational power. In particular, the computer architecture should be small enough to fit on the robot structure, but powerful enough to execute all the robot computational activities needed for achieving the desired

^{*} This work has been partially supported by the European Union, under contract IST-2001-34820, and by the Italian Ministry of University Research, (MIUR) under contract 2003094275.

level of autonomy. Moreover, since such systems are operated by batteries, they have to limit energy consumption as much as possible to prolong their lifetime.

The tight interaction with the world causes the robot activities to be characterized by timing constraints, that must be met to achieve the expected robot behavior. In order to achieve stability and guarantee a desired performance, timing constraints need to be enforced by the operating system that supports the application. In particular, the operating system should guarantee that all periodic tasks are activated according to their specified periods and executed within their deadlines.

Some of the issues discussed above have been deeply addressed in the literature. For example, the feasibility of a set of periodic tasks with real-time constraints can be easily analyzed if tasks are scheduled with the Rate Monotonic (RM) algorithm (according to which priorities are inversely proportional to task periods), or with the Earliest Deadline First (EDF) algorithm (according to which priorities are inversely proportional to absolute deadlines). Liu and Layland [11] proved that, in the absence of blocking factors, a set of n periodic tasks is schedulable if the total processor utilization is less than or equal to a given bound U_{lub} , which depends on the adopted algorithm. This result has later been extended also in the presence of blocking factors due to the interaction with mutually exclusive resources [13].

In the context of real-time systems, different energy-aware algorithms have been proposed to minimize energy consumption. They basically exploit voltage variable processors to minimize the speed while guaranteeing real-time constraints [14,2,3,12]. What is missing, however, is an integrated framework for energy-aware control, where different strategies can be applied at different levels of the architecture, from the hardware devices to the operating system, up to the application level.

In this paper, we present a system wide approach to energy management applied to all the architecture levels and integrated with the scheduling algorithm to guarantee real-time constraints. The method is applied to an autonomous walking robot controlled by a PC104 microprocessor and equipped with a set of sensors for the interaction with the environment.

2 System architecture

The robot described in this work is a walking machine with six independent legs, each having three degrees of freedom. The mechanical structure of the robot is illustrated in Figure 1 and a view of the robot with sensors and processing units is shown Figure 2. The robot is actuated by 18 Hitec HS-645MG servomotors including an internal position control loop that allows the user to specify angular set points through a PWM input signal. The internal feedback loop imposes a maximum angular velocity of 250 degrees per second and each motor is able to generate a maximum torque of 9.6 kg-cm.

The robot is equipped with a color CMOS camera mounted on a micro servomotor that allows rotations around its vertical axis. Other sensors include a pair

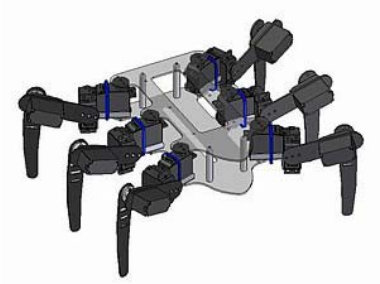


Fig. 1. Mechanical structure of the robot.

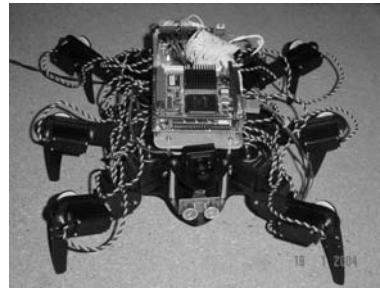


Fig. 2. A view of the robot.

of ultrasound transducers for proximity sensing, two current sensors on each leg for measuring the torque during walking (so detecting possible obstacles in front of the legs) and a battery sensor for estimating the residual level of charge.

A block diagram of the hardware architecture is shown in Figure 3.

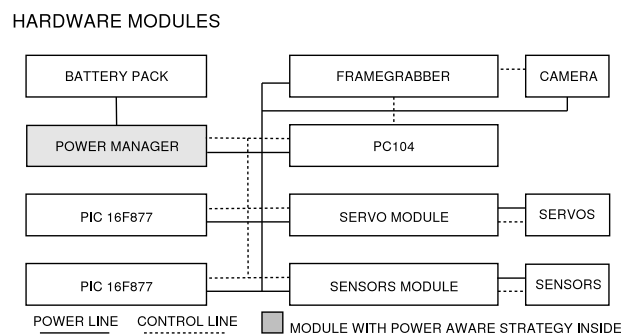


Fig. 3. Block diagram of the hardware architecture.

All software activities carried out by the robot are partitioned in two hierarchical processing units. The high level unit consists of a PC104 Pentium-like computer with a Geode GX1 CPU at 266 MHz mounted on a Eurotech CPU-1432 motherboard. It includes a 128 MBytes RAM and a solid state hard disk of 640 KBytes. A CM7326 PC/104-Plus frame grabber is connected to the motherboard via PCI bus for image acquisition from the camera. The high level unit is responsible for walking control, image acquisition, sensory processing, and power management.

The low level unit consists of a pair of Microchip 16F877 Programmable Interrupt Controllers (PICs), which are dedicated to motor driving, sensory acquisition, and preprocessing. Input set points for the servomotors are received from the walking layer via a standard RS232 serial line, which allows a transfer rate up to 115 Kbaud.

The current consumed by each leg is converted to a voltage signal and is sampled by the PIC through its analog input lines upon a specific command coming from the serial line. The PWM signal generator is interrupt driven and can drive up to 16 motors, hence two PICs are used to drive all the servomotors.

Through the remaining I/O lines of the PIC, it is possible to control external sensors, like infrared or ultrasonic proximity sensors. If a sensor is not used, the PIC interface disables its power line by sending a signal to the Power Manager.

To enforce real-time constraints on critical control activities, the software on the PC104 runs under the SHARK operating system, which is briefly described in the following section.

2.1 The SHARK kernel

SHARK is a real time operating system [9] developed for supporting predictable control applications consisting of tasks with different constraints and timing requirements (e.g., hard, soft, non real-time, periodic, aperiodic, etc.). The most peculiar features of this kernel include:

- **Modularity.** All kernel mechanisms are developed independently of other internal components and several options are available for each mechanism.
- **Dynamic scheduling.** The kernel provides direct support for deadline-based scheduling, which guarantees predictable responsiveness and full processor utilization.
- **Resource reservation.** A *temporal protection* mechanism based on the Constant Bandwidth Server [1] allows the user to reserve a fraction of the processor bandwidth to activities with highly variable computation time. This prevents execution overruns to create unpredictable interference and allows tasks to execute in isolation as they were executing alone on a slower dedicated processor.
- **Bounded blocking.** Priority inversion caused by resource sharing can be avoided through specific concurrency control protocols, including Priority Inheritance, Priority Ceiling [13], or Stack Resource Policy [4]. In addition, a fully asynchronous (non blocking) mechanism is available for exchanging data among periodic tasks running at different rates.
- **Predictable interrupt handling.** A device driver does not fully interfere with application tasks, since can be split into a *fast handler* (executing in the context of the running task) and a *safe handler* (guaranteed by the system).

3 Power-Aware Management

Hardware and software components cooperate to reach the following main goals: low power consumption, onboard sensory processing, and real-time computation capabilities. In order to contain costs, the robot is built with generic mechanical and electrical components, making the low-power objective more difficult to be satisfied. Nevertheless, the adoption of power-aware strategies inside the robot

hardware and software modules significantly increased the system lifetime. To achieve significant energy saving, power management is adopted at different architecture levels, from the hardware components to the operating system, up to and the application.

3.1 Hardware level

The simultaneous activity of 18 servomotors creates sporadic high peak loads of current that must be handled by the power supply circuit. For this reason Lead-Acid batteries are chosen as a preliminary test-set, due also to low cost and fast recharge. A Lithium battery could also be a good alternative.

Without proper precautions, the current instability induced by servomotors can cause unpredictable resets and anomalies inside the microprocessor boards used in the system. To avoid such problems, these systems are typically designed with two different battery packs, one for the servomotors and non-critical electronics, and the other for the microprocessor board. Such a solution, however, is not efficient since does not exploit the full available energy.

The solution we adopted to optimize battery duration uses a single battery pack, with a specific control circuit (the Power Manager) for eliminating peak disturbances caused by servomotors that could reset the processor. The Power Manager is one of the most critical parts of the robotic system. The high current flow and the presence of inductances inside the batteries make the power voltage extremely unstable. If connecting the batteries directly to the servomotors, a simple robot movement would cause a temporary voltage breakdown that would disable all the other boards. In our solution, the voltage breakdown is avoided by a feedback circuit and a set of backup capacitors. When a high peak of current is requested by the actuation system and the power line inductance causes a voltage breakdown, a feedback circuit decreases the current flow and a set of backup capacitor, isolated by a fast Schottky diode, keeps the PC104 and other critical parts alive. The Power Manager can also disable specific subsystems of the robot, like sensors or servomotors, when the power-aware algorithm (running in the PC104) sends a specific control command.

3.2 Operating system level

In a computer system, the power consumption is related to the voltage at which the circuits operate according to an increasing convex function, whose precise form depends on the specific technology [8]. Hence, the amount of energy consumed by the processor can be controlled through the speed and voltage at which the processor operates.

When processor speed is increased to improve performance, we would expect the application tasks to finish earlier. Unfortunately this is not always the case, because several anomalies [10] may occur in the schedule when tasks have time and resource constraints, making the performance discontinuous with the speed.

Conversely, when voltage is decreased to save energy consumption, all computation times increase, so the processor might experience an overload condition that could make the application behavior quite unpredictable.

In [7] it has been shown that, to prevent scheduling anomalies and achieve scalability of performance as a function of the speed, tasks should be fully preemptive and should use non blocking mechanisms to access shared resources. Under SHARK, non blocking communication is provided through the Cyclic Asynchronous Buffers (CABs) [9]. Moreover, to avoid the negative effects of overload caused by a speed reduction, periodic tasks should specify their period with some degree of flexibility, so that they can be resized to handle the overload. In our system, when an overload is detected, rate adaptation is performed using the elastic model [5,6], according to which task utilizations are treated like springs that can adapt to a desired workload through period variations. Viceversa, if the load is less than one, the processor speed can be reduced to get a full processor utilization, thus meeting timing constraints while minimizing energy. The elastic task model is fully supported by the SHARK kernel as a new scheduling module.

One problem with the adopted architecture is that the PC104 is not a new generation power-aware CPU with voltage and frequency scaling. Nevertheless, it is possible to force the CPU in a sleep mode for a specific amount of time, during which the processor enters in standby. Switching the CPU on and off, as a PWM signal, the average processor speed can be continuously varied from the two extreme values. If Q_A is the interval of time the CPU is active at its maximum frequency f_M and Q_S is the interval in which it is in sleep mode, the average frequency is given by $\bar{f} = f_M Q_A / (Q_A + Q_S)$. If $\sigma = Q_A / (Q_A + Q_S)$ denotes the active fraction of the duty cycle in the PWM mode of the CPU, the average frequency of the processor becomes $\bar{f} = f_M \sigma$. Since $\bar{f} < f_M$, all tasks run with an increased computation time given by

$$C'_i = \frac{f_M}{\bar{f}} C_i = \frac{Q_A + Q_S}{Q_A} C_i = \frac{C_i}{\sigma}.$$

Finally, if P_M is the power consumption in the active mode and P_S the one in the sleep mode, then the average power consumption is linearly dependent from σ and is given by

$$\bar{P} = \frac{P_A Q_A + P_S Q_S}{Q_A + Q_S} = P_A \sigma + P_S (1 - \sigma) = P_S + (P_A - P_S) \sigma.$$

3.3 Application level

To achieve a significant reduction in the power consumption, it is essential that appropriate strategies are adopted also at the application level. For some devices, like the camera and the frame grabber, the only strategy that can be adopted is to turn them off while they are not used. For other devices, like the ultrasonic sensor and the servomotors, more careful strategies can be adopted. The ultrasonic sensor, when turned on, can be in three different states: standby, working, and

beam mode. The acquisition period is also important; in fact a short period causes too much energy consumption due to frequent beam generations, whereas a long period can lead to missing some obstacles. Table 1 shows the power consumption of the devices used on the robot.

Device	Power (W)
Frame grabber	1
CCD camera	0.4
Servomotor	0.4 - 20
Ultrasonic sensor	0.15 - 0.75 - 2.5

Table 1. Power consumption of the robot devices.

To minimize energy consumption in the servomotors, it is important not only to move the hexapod with a given speed in a desired direction, but also to select a leg posture and a walking mode that drains less current. This is crucial when comparing walking robots with wheeled vehicles, which have a negligible consumption when they do not move.

The algorithms we propose to coordinate the legs modulate a reference walking step through a number of parameters. An important parameter is the maximum angle that each leg covers in the horizontal plane during its motion. A difference in such angles for the left and right legs causes the robot to turn. Another parameter that can be tuned is the raise that each leg performs in the vertical plane. This value depends on the type of surface on which the robot walks: small values are sufficient for walking on smooth surfaces, whereas higher values are needed in the presence of obstacles or protruding regions.

To guarantee the equilibrium of the robot during walking, the algorithm always keeps at least three legs in touch with the ground, so that the center of mass of the robot always falls in the polygon defined by the touching points. Two different walking modes are considered in this paper, depending on the number of legs that are moved at the same time: one or three. They will be referred to as 1-leg and 3-leg algorithms.

When adopting the 1-leg algorithm, it is necessary to evaluate the exact order in which legs are moved forward. Two very simple rules would be to maintain the support polygon made with the touching legs as large as possible, or to have a little phase difference between adjacent legs.

In the 3-leg algorithm, the specific position set points for the motors involved in the horizontal motion are generated by sampling two cosine functions with opposite phases. Similarly, a pair of sine functions with opposite phases was initially used for the vertical leg motion. However, this solution has been modified since it was causing the robot to have a significant roll while walking. The specific shape of the waveform depends on many actors, including equilibrium requirements, speed, and maximum allowed roll.

4 Experimental results

This section presents some experimental results performed on the robot to evaluate the power consumption related to different postures and walking modes.

In a first experiment, we evaluated how the power consumption is affected by the robot posture. To do that, we measured the current drained by the motors as the angles, α and β , of the joints corresponding to the two horizontal axes of the legs were varied in a given range. The angles of the joints rotating around the vertical axes were set to keep the legs parallel to each other. The results are reported in Figure 4. A dash in the table means that the leg was not touching the floor, so consuming a negligible power. As intuitive, the minimum power consumption was reached when the legs were vertical (that is, $\alpha = 0$ and $\beta = -90$), however such a posture resulted to be quite critical for the stability of the robot. As shown in the table, the minimum current consumption in a stable configuration was obtained with $\alpha = 45$ and $\beta = -45$.

$\alpha \backslash \beta$	-45	-30	-15	0	15	30
0	0.62	0.86	-	-	-	-
15	0.43	0.72	0.73	-	-	-
30	0.13	0.55	0.78	-	-	-
45	0.09	0.26	0.55	0.77	-	-
60	0.18	0.14	0.30	0.58	-	-
75	0.19	0.13	0.15	0.35	0.25	-
90	0.28	0.20	0.20	0.19	0.55	0.20

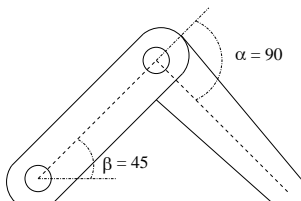


Fig. 4. Current values for different postures.

A second experiment has been carried out to test the power consumption for different walking modes, namely the 1-leg and 3-leg modes. The 3-leg mode was tested for three different speeds, obtained by changing the period T_s of a basic leg step. To compare the two modes, we monitored a set of parameters while the robot was walking along a straight line path 1 meter long. In particular, we measured the time T_d to complete the path, the energy E consumed by the system in T_d , and the average current I_{avg} drained by all the motors. The results of this experiment are reported in Table 2.

Note that, when legs are moving at the same speed ($T_s = 0.9$), the 3-leg mode is faster than the 1-leg mode and consumes less current. This counterintuitive result can be explained by considering the non-linearity in the current/torque function, which makes the robot to consume more current when its weight is distributed on five legs rather than on three legs. Hence, the 3-leg mode resulted to be more efficient both in terms of energy and performance. We also observed that the energy consumed by the system to accomplish the task decreases as the leg speed gets higher. This happens because, in a fast walking, the reduction

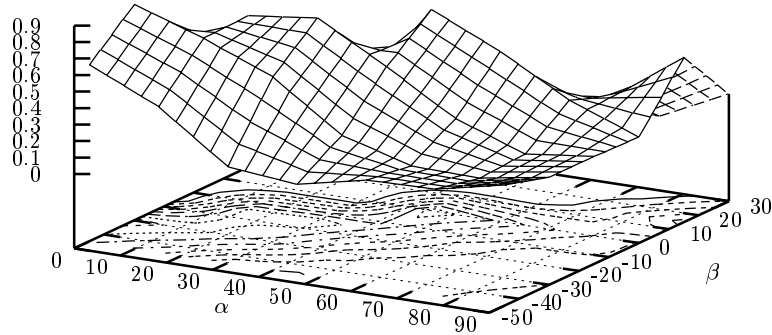


Fig. 5. Current drained in function of α and β .

	1-leg	3-leg	3-leg	3-leg
T_s (s)	0.9	1.8	0.9	0.23
T_d (s)	34.3	21.8	9.87	3.42
E (J)	512	273	140	73.6
I_{avg} (A)	1.86	1.56	1.77	2.69

Table 2. Comparing different walking modes.

of time T_d for completing the task is more significant than the increase of the average current drained by the motors, making fast walking more effective.

5 Conclusions

In this paper we presented an integrated approach for designing robot systems with real-time and energy-aware requirements. We showed that to achieve a predictable timing behavior and a significant saving in the energy consumption, a combined effort is required at different architecture levels. At the hardware level, the processor must provide different operational modes to balance speed versus power consumption. At the operating system level, a specific power management layer should set the appropriate operational mode to minimize energy consumption while guaranteeing the timing constraints. Finally, at the application level, the control strategies should be tunable to trade performance with energy con-

sumption, so that the robot can switch to a different behavior to prolong its lifetime when the batteries are low, still performing useful tasks.

We showed how the techniques discussed above can be implemented in a small walking robot using commercial low-cost hardware components. As a future work, we plan to perform a more extensive experimentation on the robot, in order to derive a complete set of strategies to allow the power management unit to select the most appropriate operational mode based on the task to be performed and on the residual energy available in the batteries.

References

1. Abeni, L. and Buttazzo, G.C.: "Integrating Multimedia Applications in Hard Real-Time Systems". *Proc. of the IEEE Real-Time Systems Symposium*, (1998).
2. Aydin, H., Melhem, R., Mossé, D. and Mejia Alvarez, P.: "Determining Optimal Processor Speeds for Periodic Real-Time Tasks with Different Power Characteristics". *Proceedings of the Euromicro Conference on Real-Time Systems*, (2001).
3. Aydin, H., Melhem, R., Mossé, D. and Mejia Alvarez, P.: "Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems", *Proceedings of the IEEE Real-Time Systems Symposium*, (2001).
4. Baker, T.P.: "Stack-Based Scheduling of Real-Time Processes". *The Journal of Real-Time Systems* 3(1), 76–100, (1991).
5. Buttazzo, G.C., Lipari, G. and Abeni, L.: "Elastic Task Model for Adaptive Rate Control". *Proc. of the IEEE Real-Time Systems Symposium*, 286–295, (1998).
6. Buttazzo, G.C., Lipari, G., Caccamo, M., and Abeni, L.: "Elastic Scheduling for Flexible Workload Management". *IEEE Transactions on Computers*, Vol. 51, No. 3, 289–302, (2002).
7. Buttazzo, G.C.: "Scalable applications for energy-aware processors". *Proc. of the 2nd Int. Conf. on Embedded Software (EMSOFT 2002)*, Grenoble, France, Vol. 2491 of Lecture Notes in Computer Science, Springer-Verlag, 153–165, (2002).
8. Chan, E., Govil, K. and Wasserman, H.: "Comparing Algorithms for Dynamic Speed-setting of a Low-Power CPU". *Proceedings of the First ACM International Conference on Mobile Computing and Networking (MOBICOM 95)*, November (1995).
9. Gai, P., Abeni, L., Giorgi, M. and Buttazzo, G.C.: "A new kernel approach for modular real-time system development". *Proc. 13th IEEE Euromicro Conf. on Real-Time System*, (2001).
10. R. L. Graham: "Bounds on the Performance of Scheduling Algorithms". Chapter 5 in *Computer and Job Scheduling Theory*, John Wiley and Sons, 165–227, (1976).
11. Liu, C.L. and Layland, J.W.: "Scheduling Algorithms for Multiprogramming in a Hard real-Time Environment". *Journal of the ACM* 20(1), 40–61, (1973).
12. Melhem, R., AbouGhazaleh, N., Aydin, H. and Mosse, D.: "Power Management Points in Power-Aware Real-Time Systems". In *Power Aware Computing*, ed. by R. Graybill and R. Melhem, Plenum/Kluwer Publishers, (2002).
13. Sha, L., Rajkumar, L.R. and Lehoczky, J.P.: "Priority Inheritance Protocols: An Approach to Real-Time Synchronization". *IEEE Transactions on Computers*, 39(9), (1990).
14. Yao, F., Demers, A. and Shenker, S.: "A Scheduling Model for Reduced CPU Energy". *IEEE Annual Foundations of Computer Science*, 374–382, (1995).