Energy Aware HW/SW Integration in an Autonomous Microrobot

A. Sanuy[1], R. Casanova[1], M. Szymanski[2], H. Wörn[2], J. Samitier[1], A. Dieguez[1]
[1]Department of Electronics. Sistemes d'Instrumentació i Control (SIC)
University of Barcelona, C/ Marti i Franques nº 1, Barcelona, Spain

[2]Institute for Process Control and Robotics(IPR)
Kaiser. 12, Universität Karlsruhe(TH), D-76128 Karlsruhe, Germany

Abstract: - This paper addresses the integration of HW and SW for a 3 mm x 3 mm x 3 mm autonomous microrobot called I-SWARM. The robot is intended to be part of a swarm of up to 1000 members for studying large scale swarm behaviour. The robot is the smallest autonomous robot of the world, equipped with a vibrating contact sensor, a four directional infrared communication module, vibrating legs and an ASIC. It presents an overall architecture of the robot and the SW used. The software is comprised of a hardware abstraction layer and an interpreter for a specially designed control language.

Key-Words: - Micorobotic, Swarm, SoC , low power

1 Introduction

Swarm robotics is a new approach to the coordination of multirobot systems which consist of large numbers of relatively simple physical robots. The goal of these systems is to execute tasks, which may be inherently too complex or impossible for a single robot and benefits can be gained from using multiple robots. Those tasks are achieved by building and using several simple and cheap robots than having a very complex and powerful single robot. The requirements for a robot in a swarm depend on the scenario to be executed. The most basic scenarios require usually at least locomotion and basic perception of the environment. With these cooperative mobile robots become an inherent approach in the social sciences (organization theory, economics, cognitive psychology), and life sciences (theoretical biology, animal ethnology). The absence of a central control is the most important fact in these systems and it calls decentralized systems using terms as "emergence" and "self-organization". These terms in a swarm are the ability to distribute itself "optimally" for a given task exhibiting collectively intelligent behaviour with non-intelligent robots. In collective robotics, communication is crucial for coordinating behaviour among robots. Communications among different members of the swarm or a detailed analysis of the environment enable more complex scenarios [1].

These single robots have limited capabilities. Therefore, microrobots need to operate in very large group or swarms to affect the macroworld. Simple agents can be constructed/programmed to achieve collective swarm tasks. These can send orders, share information, change their current task/role if a member of the system fails. They can even reprogram another member of the collective if the robots are provided with this ability [2].

I-SWARM [3] is an autonomous microrobot of 3 x 3 x 3 mm^3 powered with solar cells. It has been designed with the capability to move, communicate, and sense in order to act as a member of a swarm of other I-SWARM robots. Each agent senses its environment and uses this information to take decisions, execute different tasks or learn as shown in Fig.1. Because of the small dimensions of the robot, all the electronics have been integrated on the SoC.

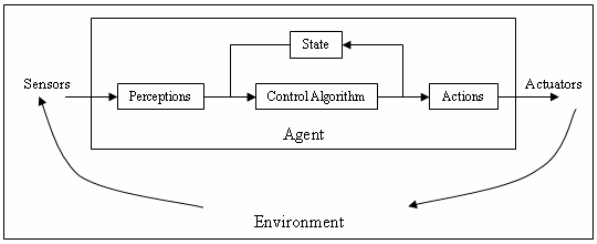

Fig.1. Flow control of a swarm robot

The SW used in this robot is based on finite state machine operating system (FSMO) JaMOS [4] that senses the environment and acts depending on the state and the role of the robot in the swarm.

This paper presents the integration process between HW and SW to achieve that the I-SWARM robot

solves a specific scenario. This means, how the SW uses and operates the HW provided by the robot. A simple processor (the embedded 8051 and integrated memories) is not able to control all the robot components without exceeding the power requirements. The I-SWARM robot is described in section II. The overall architecture of the robot is described in section III. Section IV describes the integration between HW and SW. Section V describes the SW language used. The control cycle of the SW is described in section VI. The combination between the SW and the HW modules is described in section VII, and finally, the section VIII shows the experiments realized.

2 I-SWARM Robot

I-SWARM robot has been conceived in order to create a swarm of up to 1000 robots as well as to provide a platform to run different swarm scenarios and algorithms. Some of the scenarios are aggregation, pattern forming and object recognition.

The electronics of an I-SWARM robot as well as the actuators are mounted over a flexible printed circuit board (FPCB). This FPCB is folded as a sandwich and is mounted over the locomotion unit. It has been fabricated by the Department of Materials Science (DMS), Uppsala [5]. The three legs are bent out of plane (45º). The VCS is mounted at the middle of one side of the FPCB. The IR communication system is an optoelectronic system composed of four pairs of one photodiode and one LED placed at every side of the robot and it has been fabricated by the Scuola Superiore Sant Anna (SSSA), Italy [6].

Each I-Swarm robot is controlled by a SoC designed specifically for this purpose. The power is supplied by a group of solar cells mounted at the top of the robot. The cells are not connected directly to the electronics but to two tantalum capacitors that store the energy. Two additional smaller solar cells, SC1 and SC2, mounted in opposite sides of the robot top surface are used to program the robots and to provide the global positioning of the robot in the arena **(Fig 2)** [7].

3 Overall Architecture

The collective behaviour of the swarm emerges because each agent takes decisions as a function of the data collected from its environment and his role inside the robotic swarm. In an I-SWARM robot, the decisions and the data processing are performed by an 8051 microprocessor embedded on the SoC provided of 8 kB of program memory that stored the program code and 2 kB for data to store some values and variables. The 8051 work in parallel with specific hardware blocks dedicated to the control of each of the robot functionalities (Fig.3). These are:

- Locomotion Control Unit (LCU) [8]
- Optoelectronic Control Unit (OCU) [9]
- VCS Control Unit (VCSCU) [10]
- Program Control Unit (PCU)

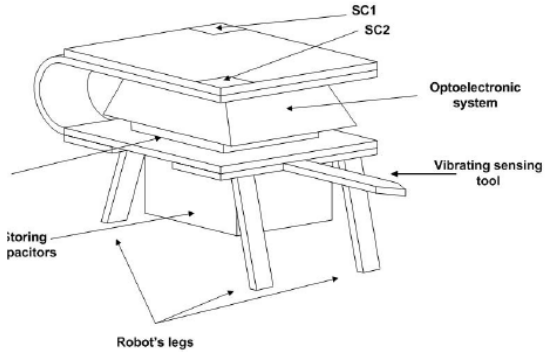

Fig.2. I-SWARM robot concept

The existence of these specific hardware modules reduces the size of the BIOS because basic tasks as forward movements or frame sending can be implemented by hardware. The space not occupied by the BIOS is used to load finite state machine operating system (FSMOS).

The 8051 can adjust its working frequency (Dynamic Frequency Scaling, DFS) as a function of the workload in order to save power. However, software needs the 8051 timers so it can not change the working frequency. To avoid the non use of DFS which is advantageous from the point of view of power management, the SoC is provided with two timers which run with the 12 MHz. With the additional timers the 8051 can modify its frequency and the timers still runs constantly at 12 MHz. These two timers determine the control cycle of the SW execution, as is explained in section 6.

The SoC has a Power Management Unit (PMU) to save power in those cases that no computation has to be performed. The PMU can stop the clock of the processor. The clock can be activated again when one of the controllers generates an interrupt. The way of using the PMU in order to optimize the power consumption of the SoC is running the 8051 at the lowest frequency, 1.46 kHz, during periods of

inactivity or when one or more controllers are performing a task. When any of these finishes, the 8051 has to take a decision and continue executing a SW plan. If it is necessary, the 8051 can speed up this process by incrementing its working frequency. When it finishes, the clock can be set again at the lowest frequency 1.46 kHz or even stopped until a controller requires the 8051 services.

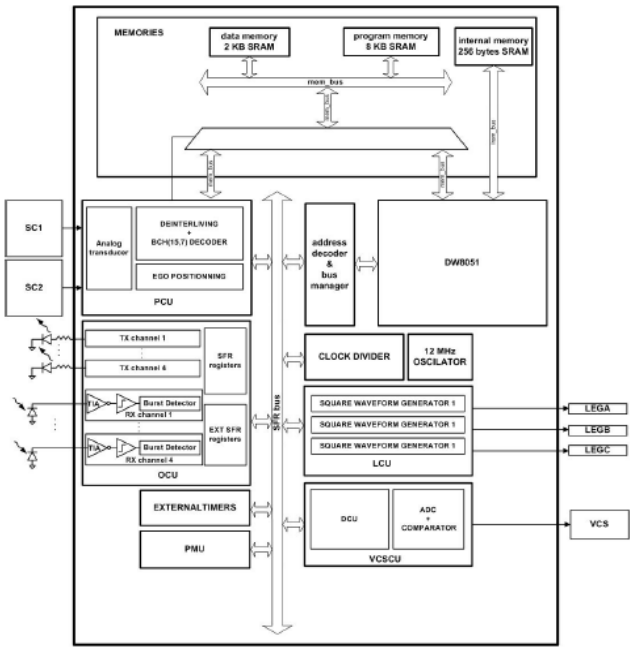

Fig.3. Overall architecture of the SoC

4 Integration between HW and SW

SW consists of two main layers. The hardware abstraction layer (HAL) also called BIOS and an interpreter. The interpreter itself consists of two layers, one that connects the interpreter with the HAL and another one that does the interpretation.

The HAL configures the HW and puts it in the correct state. After configuring the hardware the interpreter starts interpreting its program given in a special language as a byte code. This language is the extended Motion Language Two (MDL2ϵ), and it is explained at the next section. The decision to use an interpreted language was guided by the fact, that programming the robot via the beamer needs 45 minutes for 8kB. The interpreter enables us to load programmes of 256 – 512 Bytes without touching the HAL and the interpreter. This reduces the programming time during different experiments by a factor of up to 32.

5 Extended motion description language two

5.1 Overview

The extended Motion Description Language Two (MDL2ϵ) is a high-level language for implementing behaviour-based control programs for relative simple robots. MDL2ϵ is based on MDLe by Manikonda et al. [11] but incorporates several improvements leading from a control language driven by control theory to a fully functional control language for autonomous robots. MDL2ϵ therefore aims on:

1. hiding low-level robot implementation details from the user,
2. reducing code-size,
3. providing the user with a simple and easy to use control language,
4. simple concept of integrating MDL2ϵ in robot simulation environments and different robotic platforms.

5.2 Concept

MDL2ϵ is conceptual similar to a regular expression. MDL2ϵ-programs are called plans and are usually implemented in an xml-file. MDL2ϵ provides the user with an alphabet, which is a set of so called atoms (ATOM), and a set of non-terminals or MDL2ϵ-operators that control the control flow. Atoms let the robot interact with its environment or change its internal state. Typical atoms for interacting with the environment are AMOVE, which makes the robot move forward, or ASEND, that sends a message to other robots. An atom that changes the internal state is for instance ASETROLE, which changes the role of the robot for instance switching from a worker to a soldier role showing different behaviours.

The non-terminals are multiplicity (MULT), union (UNION), random union (RUNION), behaviour (BEHAVIOUR) and plan (PLAN). The multiplicity is a looping operator that loops for a given time or infinite. The union acts like a if-than-else-statement which selects the first valid subnode stated in the xml expression <UNION multiplicity=”5”> <subnode/> . . .<subnode/> </UNION>. How a subnode can be valid or invalid will be described later.

The random union randomly selects one of its subnodes. Each subnode holds as attribute its own probability value, which is used to calculate a simple normalised probability distribution for the selection

process.

The behaviour operator is a simple bracket around a set of arbitrary MDL2ϵ operators and atoms. However, behaviours and atoms are in some way special. Their attributes include an interrupt and a duration. The duration is a simple timer that states how many MDL2ϵ-cycles a behaviour or atom has to be executed also infinite execution is possible. The interrupts connect the behaviours and atoms to the robot's internal and external state. If the interrupt of an atom or behaviour is true the atom and behaviour will be called valid or active otherwise invalid or inactive. If an atom or behaviour becomes invalid a new valid atom will be selected from the plan. Multiplicity, union and random union are considered as always valid. To distinguish between those interrupts and the hardware interrupts generated by the ASIC. We will refer to those either as MDL2ϵ-interrupts or hardware interrupts. Finally is a plan a special behaviour that has no interrupt just a duration. The duration of a plan is usually set to infinite a plan is the entry point for a MDL2ϵ-program and surrounds all other nodes.

A plan is an arbitrary combination of the latter control flow operators and atoms. Valid operators and atoms will be executed sequentially as long as no control flow operator changes the flow of execution.

6 Mdl2ϵ Control Cycle

The whole control process is embedded into the MDL2ϵ control cycle. This cycle usually consist of the following three steps:

1. Check if the current behaviours or atom is valid:
   yes: continue executing the current atom
   no: select the next valid atom from the plan and execute it.
2. Gather sensor information and update MDL2ϵ-interrupts.
3. Decrement the timers of the behaviours and the atom.

In the I-SWARM-robot sensors do not have to be polled for information. All actuators and sensors inform the CPU by a hardware interrupt that they entered a new state. For instance a message arrived/was send or a motion has been finished. This allows a very energy efficient integration of the MDL2ϵ control paradigm exploiting all the features of the Power Management Unit (PMU). The integration will be described in detail in the following section.

7 Combining Mdl2ϵ and the Power Management Unit

The PMU enables the software engineer to power down all sub modules of the ASIC independently. This makes energy saving very convenient from the MDL2ϵ point of view. Acting modules like the Motion Control Unit can be turned on by an atom when needed and also turned of either in an interrupt service routine (when motion has finished) or by another atom. Switching of the Optical Control Unit (OCU) or the vibrating needle makes no sense from the robot's point of view as it is the only sensor that makes the robot aware of other robots or obstacles. However, if energy is scarce those units can also be turned off automatically by appropriate atoms. For instance sending and receiving a message may not possible while walking, than the AMOVE would also turn off the OCU.

Significantly most of the energy is consumed when the DW8051 μController accesses to the internal memories. Here the PMU provides us with a very important feature. The DW8051 can be set to a sleep mode, by simply disconnecting it from the clock signal, for a given time. Instead of using an external timer for the MDL2ϵ-duration of an atom or behaviour the DW8051 can be set in sleep mode. This significantly reduces the frequency of step one of the execution cycle.

The DW8051 one can be awaken by all peripheral hardware interrupts during the sleep mode and handle them in the interrupt service routines (ISR). Some hardware interrupts affect the MDL2ϵ-interrupts. Table 1 describes the pseudo-code of the implemented loop the DW8051 is continuously send to sleep mode, when it awakes it is checked if it has been awaken by the sleep timer or any other interrupt. If it was the sleep timer all software timers (durations) are decremented and the controller is send back to sleep mode. If any other interrupt occurred or a software timer overflow occurred it is checked if the current behaviours and the atom are still valid. If not a new atom will be selected from the plan.

8 Experiments and Evaluation

Experiments have been performed to evaluate the concept. The MDL2ϵ-cyle duration has been set to 20ms. We selected 20ms as it corresponds with the integration step time of our robot simulator. This improves the transfer of plans written in the simulator

to the robot. However, the cycle time can be adapted to reduce the power consumption.

As test scenario one of the final I-SWARM scenarios has been selected. In this scenario the robots have to collectively optimise a function given by the environment. The algorithm is inspired by honeybees and has bee developed by our partners from Graz T. Schickel et al. The implementation of the algorithm in MDL2ϵ is given in Table 2.

Table 1. MDL2ϵ cycle for the I-SWARM-robot.

```
01 while (true) {
02        select-valid-atom;
03        do {
04           clear-software-interrupts;
05           do {
06                    PMU-send-DW8051-to-sleep;
07                    if (waken-up-by-PMU-sleep-timer)
08                      decrement-software-timers;
09           } while ( no-new-message
10                      AND no-collision
11                      AND motion-not-finished
12                      AND no-software-timer-overflow);
13        } while( no-MDL2e-interruption );
14 }
```

The algorithm it self is very simple. The robots perform a random walk until they meet another robot. Then they measure the value of the environmental pattern and stop as a linear function of the measured value. This leads to an aggregation in the spot with the highest value.

The whole algorithm has been analysed regarding the needed CPU time. The DW8051 ran with a fixed clock of 6 MHz. The input to the ASIC had been controlled via an RS232 connection by a specially designed application. In this way it was possible to have a look at the single parts of the plan.

Fig.4 exemplary shows what has been measured during the test. Interesting for the HW and SW integration using the PMU is the duty time of the DW8051 during the interpretation of the MDL2ϵ byte code. The duty during the interpretation which includes operant fetching, interrupt analysis and including the execution of a newly found atom lay between 10 and 40ms. The MDL2ϵ-cyle for sending the DW8051 back to sleep and reducing the software timers took about 280us. In this picture we can see the RUNION code part of the plan presented in Table 2. On the left side of this image, we can see the AROT_L atom. This means the only one of the two frontal legs are moving (D13 and D14 on Fig.4.). This atom is executed during 5 MDL2ϵ cycles and then in section B, the interpreter selects the new atom at an initial clock frequency (D7). AMOVE atom starts during 10 MDL2ϵ cycles (Fig.4. C) The next process executed is the RUNION (Fig 4. D) and means that an atom is randomly selected with its programmed probability (lines 06-08 of Table 2). AROT_R atom is executed (Fig.4. E) during 5 MDL2ϵ cycles, and finally an AMOVE (line 04 on

Table 2.: MDL2ϵ implementation of the bee inspired aggregation algorithm by T. Schmickel et al.

```
00 <MDLeScript>
01  <PLAN name="BeeAlg" duration="infinite">
02   <BEHAVIOUR name="collisionAvoidance" Interrupt="NOT(GEQ(VNROBOTS,1))">
03    <MULT multiplicity="infinite">
04              <ATOM name="AMOVE"  interrupt="NOT(ITOUCH)" duration="10"/>
05              <RUNION>
06                <ATOM name="AMOVE"  interrupt="NOT(ITOUCH)" duration="10" probability="18"/>
07                <ATOM name="AROT_L" interrupt="ITRUE"      duration="5"  probability="1"/>
08                <ATOM name="AROT_R" interrupt="ITRUE"      duration="5"  probability="1"/>
09              </RUNION>
10              <BEHAVIOUR name="avoid" interrupt="ITOUCH">
11                <ATOM name="AROT_L"  interrupt="AND(ITOUCH0,ITOUCH1)" duration="5"/>
12                <ATOM name="AROT_R"  interrupt="AND(ITOUCH0,ITOUCH3)" duration="5"/>
13                <ATOM name="ASTOP"   interrupt="ITRUE" duration="1"/>
14              </BEHAVIOUR>
15    </MULT>
16   </BEHAVIOUR>
17   <BEHAVIOUR name="measurement" interrupt="OR(ISEMAPHORE,GEQ(VNROBOTS,1))">
18    <ATOM name="ASETSEM"  duration="1" /> <!-- sets a semaphore -->
19    <ATOM name="ASTOP"    interrupt="NOT(IMEASURE)" duration="40"/>
20    <UNION>
21              <ATOM name="ASEND" arg0="180" interrupt="GEQ(VXPOSL,180)" duration="180"/>
22              <ATOM name="ASEND" arg0="120" interrupt="GEQ(VXPOSL,120)" duration="120"/>
23              <ATOM name="ASEND" arg0="60"  interrupt="GEQ(VXPOSL,60)"  duration="60"/>
24              <ATOM name="ASEND" arg0="30"  interrupt="GEQ(VXPOSL,30)"  duration="30"/>
25    </UNION>
26    <RUNION>
27              <ATOM name="AMOVE"  interrupt="NOT(ITOUCH)" duration="10"  probability="1"/>
28              <ATOM name="AROT_L" duration="5" probability="1"/>
29              <ATOM name="AROT_R" duration="5" probability="1"/>
30    </RUNION>
31    <ATOM name="ARELSEM" duration="1" /> <!-- releases the semaphore -->
32   </BEHAVIOUR>
33  </PLAN>
34 </MDLeScript>
```

Table 2) process is executed to select the next atom (Fig.4. F).

The duty time of the DW8051 induced a shift of the MDL2ϵ pseudo clock, cf. Fig.4. This shift can be reduced by measuring the duty time, using the additional external timers, and respectively reducing the software timers and the sleep time. This strategy has been successfully applied. However, there is a trade-off, as the external timer interruption for counting the duty time runs at a minimal frequency of 1.46kHz and handling those interrupts also induces a significant increase of the workload. This gets even worse when the main clock of the µC is decreased as it is interrupted more often in respect to the conducted work. The advantages or disadvantages of this approach have to be measured with the real robot as it is a matter of the consumed energy which strategy is the best.

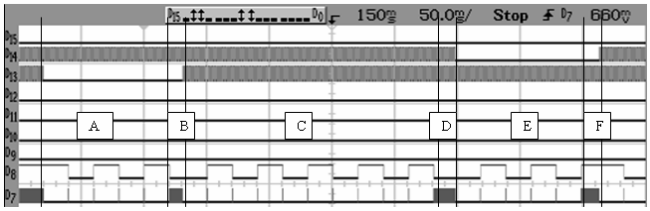

Fig.4. Collision avoidance behaviour with random motion (Table 2. line 2 – 16).

9 Conclusion

We presented the integration of software in a hardware platform designed for an autonomous microrobot forming part of a swarm. The specifically architecture designed for the hardware allows to simplify the software integration and to manage power efficiently. It has been shown how this can be achieved with simple examples.

References:

[1] Y. Uny Cao, Alex S. Fukunaga, Andrew B. Kahing, "Cooperative Mobile Robotics: Antecedents and Directions", Int. Conf. on Intelligent Robots and Systems 95, Pittsburgh, August 5-9, 1995, pages 226-234 vol.1.

[2] Menciassi A, Seyfried J, Crailsheim K, Corradi P, Dario P, Valdastri P, Schmickl T, "Micromanipulation, communication and swarm intelligence issues in a swarm microrobotic platform", Robotics and Autonomous Systems, 789—804, vol 54. 2006

[3] http://www.i-swarm.org/

[4] M. Szymanski, and H. Wörn "JaMOS – A MDL2e based Operating System for Swarm Micro Robotics", IEEE Swarm Intelligence Symposium, May 2007, Honolulu, USA, pages 324–331.

[5] N. Snis, E. Edqvist, U. Simu, S. Johansson, "Multilayered P(VDFTrFE) Actuators for Swarming Robots", Actuators 2006, 10th International Conference on New Actuators, 14 – 16 June 2006, Bremen, Germany, pages 390–393.

[6] P. Corradi, O. Scholz, T. Knoll, A. Menciassi, P. Dario, "Micro-Optical system for Communication and Perception in Swarm Microrobotics", unpublished, to be submitted to Institute of Physics Journal of Micromechanics and Microengineering.

[7] Boletis, A. Brunete, W. Driesen, and J.-M. Breguet. "Solar Cell Powering with Integrated Global Positioning System for mm3 Size Robots", IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2006), Bejing, October 9-15, 2006, pages 5528-5533, 2006.

[8] R. Casanova, A. Sanuy, A. Dieguez, J. Samitier, "Addressing the locomotion problem on a mm^3-sized robot equipped withPVDM legs", to be publihed to XXII Conference on Design of circuits and Integrated Systems (DCIS), Sevilla, Spain, 2007.

[9] O. Alonso, A. Dieguez, R. Casanova, A. Sanuy, O. Scholz, P. Corradi, J. Samitier, "An Optical Interface for Inter-Robot Communication in a Swarm of Microrobots", Fisrt International Conference on Robot Communication and Coordination (ROBOCOM), Athens, Greece, 2007.

[10] A. Arbat, J. Canals, R. Casanova, A. Dieguez, J. Brufau, M Puig and J. Samitier, "Design and control of a Micro-cantilever tool for micro-robot contact sensign", European Conference on Circuits Theory and Design, Aug. 2007, pp 100-103.

[11] Manikonda, Krishnaprasad, and Hendler. Languages, Behaviors, Hybrid Architec-tures, and Motion Control. Mathematical Control Theory, 1998. C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.