ARMUS, an ARM Robotic Processing System for Educational Purposes

Jean-Michel Aubin, Marius Bulota, Mathieu Gauthier, Jérome Marchand, Patrick-André Savard, Vincent Simard-Bilodeau, Jean-Luc Ratté-Boulianne, François Michaud

Department of Electrical Engineering and Computer Engineering
Université de Sherbrooke, Sherbrooke, Québec, Canada J1K 2R1

emails: {Jean-Michel.Aubin,Marius.Bulota, Mathieu.Gauthier, Jérôme.Marchand, Patrick-Andre.Savard, Vincent.Simard.bilodeau, Jean-Luc.Ratte-Boulianne, Francois.Michaud}@USherbrooke.ca

***Abstract* – This paper describes the functionalities of ARMUS, an ARM robotic processing system designed by a team of fourth-year undergraduate students in electrical engineering and computer engineering. This project intends to replace the Handy Board, the current microcontroller system used by first-year undergraduate students in our curricula with a more powerful, versatile and up- to- date technology, while preserving ease of use. Our ARMUS processing system offers high processing and memory capabilities at low energy consumption, for a price of around 300$CAD.**

***Index Terms – Robotic processing system, Robotic education, Design project, Project-based curriculum*.**

I. Introduction

Engineering is a discipline concerned with putting scientific knowledge to practical uses. Robotic design always requires the integration of a mechanical structure to a processing element. From smaller microcontroller boards like BasicStamp, Microchip PIC, OOPIC, Lego Mindstorm kit, iCricket [1] to embedded computing done using PC104 or laptop computers, a robot needs onboard processing elements to process sensory data and to control its actuators. The processing system becomes the central element on which to build, and influences greatly what the robot will be used for. In the case of education in robotics and automation, the Handy Board [2] has gained large acceptance over the last 10 years. Designed by Fred Martin at MIT, the Handy Board is built around the Motorola 68HC11 microcontroller. It also uses Interactive C, a cross-platform multi-tasking version of the C programming language. The simplicity of this development environment, the quality of the documentation and the amount of information available on the Internet or shared between users make the Handy Board an excellent tool to use to get introduced to robotics. It has become widely used in robotic competitions such as the Trinity College Fire-Fighting Home Robot Contest [3].

Robotic design projects are very rich learning activities in developing engineering skills. Our curricula recognize this fact and use robotics for introducing electrical engineering (EE) and computer engineering (CE) [4,5] to first-year undergraduate students, and for introducing mechatronics to second-year mechanical engineering students. The Handy Board is used in such activities, allowing students to benefit from its advantages. However, they are also constrained by its technical limitations. For instance, the Handy Board only handles a C-based programming language, while it would be interesting to use the platform to introduce students directly to C++. Communication is limited to serial ports, while an Ethernet link could broaden the type of applications for the robotic device designed. Today's technology can provide more processing power and higher memory capability, improving robotic capabilities, all at an acceptable price. In addition, the 68HC11 microcontroller is no longer produced by Motorola. Finally, more benefit could be reached if students could work right away on devices and tools that could be used in the industry.

To overcome these limitations, we decided to take on the challenge of designing a more "state-of-the-art" robotic processing system that could replace the Handy Board in our pedagogical activities. Important requirements consist of putting up a fully documented open hardware design system that is sturdy, easy to repair and available at a low cost [6]. Compared to the Handy Board, we also want to add a simulation environment for the processor: currently only one robot is available to four students simultaneously, which limits concurrent software development.

To create an optimal learning experience in engineering, sciences, design skills, teamwork and communication, our programs in EE and CE removed the barriers between conventional courses and proposed a new learning paradigm built on a competency-based framework. The approach is built on problem-based and project-based learning (PPBL) [7]. Instead of offering five regular classes (each of 3 credits) during one semester, each semester is organized around a theme and includes two types of activities: problem-based learning units and a design project. Problem-based learning units are conducted on average over a two-week period, each unit being organized around a problem scenario. The design project is conducted one day a week over the entire semester. Each team is made up of 6 to 8 students in EE and in CE, creating conditions in which teamwork situations can be realistically experienced. Compared to the conventional pedagogical approach, PPBL provides more opportunities to contextualize what students have to learn. For first-year undergraduate students, it also gives the opportunity to confirm early on their interest in either EE or CE.

This project also benefits from this approach by involving seventh and eighth-semester undergraduate students. These last two semesters address specialized EE and CE skills such as artificial intelligence, robotics, automation, software engineering, etc. One difference with

the previous six semesters is that the design project is now conducted over two semesters, and it is worth 6 credits for each student per semester. Design topics are opened and can be determined by students, professors, researchers, companies, etc., as long as the topics involve EE and CE skills. ARMUS' design is done in such a context with a team of 4 EE students and 3 CE students. Having worked with the Handy Board four years ago, these students are well qualified to work on improving the robotic setup to better address the need of first semester students, while at the same time perfecting their EE, CE, design, cost analysis and project management skills. Therefore, this project provides double benefits in education and in the training of students in robotics and automation.

The paper is organized as follows. Section II presents the design specifications. Section III presents ARMUS' hardware and software characteristics. Section IV compares ARMUS with the Handy-ARM project [6] underway. Section V concludes the paper by presenting the current status of the design and its intended impacts.

II. Design Considerations

The Handy Board [2] operates from a 9.6 Volt electric source and is based on the Motorola 68HC11 microprocessor (8-bit). It includes a 32K battery-backed static RAM, built-in recharging circuit, outputs for four 1 Amp DC motors, a 16x2 LCD screen, a piezo beeper, two user-programmable buttons, one knob, IR transmitter/receiver, SPI circuit (1 Mbaud serial peripheral interface), and inputs for a variety of analog (7) and digital (9) sensors. It is also possible to use an expansion board with the Handy Board, providing 10 additional analog sensor inputs, 4 inputs for active LEGO sensors, 9 digital outputs, 6 servo-motors and one ultrasonic range sensor. Our first year undergraduate EE and CE students also use a sound generating device that allows the robot to play messages recorded on a ISD ChipCorder (a single chip device for voice recording and playback) [5]. This device is interfaced directly with the Handy Board. The programming environment for the Handy Board is Interactive C, a custom C-language compiler. It was developed for educational robotics applications with three considerations in mind: interactivity (using an interpreted computer language environment to easily interact while other programs are running on the microcontroller); stability (reporting runtime error for common programming problems rather than crashing the system); and multitasking [2].

Since our objective is to replace such systems, this sets out the minimal specifications for our project. Just like the 68HC11, the processor must be easily available and well supported. The system should also consider its use in a modular, distributed embedded processing approach [1,9]. Instead of designing a system that offers a limited set of drivers and conditioning circuits for interfacing with sensors and actuators, more versatility can be achieved if the system is made to interface with other peripheral modules using communication ports (e.g., SPI, CAN, USB).

Systems based on 68HC11 processors have shown to build impressive robots in the past. However, capacities required for advanced tasks (e.g., image analysis, high bandwidth transmissions, complex audio encoding and decoding, real-time closed loop control system) are profoundly altered with the use of the 68HC11. On the other hand, single board PCs in the PC104 form or any adaptation of a small PC features strong support for today's top level language (C++, Java) and many open-source applications. The problem is usually cost and power consumption. Those processors are usually taken directly from a PC, and their architecture was never intended for very low-power embedded applications. In addition, the external peripherals on those systems are based on PC models, requiring the purchase of additional stack-up cards to provide analog to digital channels or multiple high speed I/Os. An interesting alternative could be embedded Linux boards which are small, less expensive (e.g., KwikByte, LOM9), low consumption, usually featuring an ARM7 or ARM9. The use of open source software is an important factor in reducing cost and facilitating portability and acceptance, compared to proprietary real-time operating systems (OS) (e.g., VxWorks). However, they are not robotic-oriented, meaning that they do not offer servo drivers, analog-to-digital converters, multiple I/Os, etc.

Regarding software programming, many implementations of interpreted C exist today, providing a console-based approach and nearly 100% C/C++ capability, such as CInt. The main drawback to using an interpreted C implementation is that it may be demanding to implement it in an embedded system that aims to be as simple as possible. For example, when it comes to shared library handling, large projects suffer from a significant overhead that may slow down development (e.g., standard libraries need to be converted to the interpreted C environment). On the other hand, a standard cross-compilation based approach may seem too complex for a system aimed at first-year undergraduate students. Nevertheless, it may be very powerful and easy to use if it is well integrated in a simple development environment.

Thus, overall, the challenge in this project is to come up with a system that can be accessible for untrained users to work on robotic aspects (and not have to learn about processors, hardware, sophisticated programming), while at the same time remain open to more complex applications and advances computing capabilities.

III. ARMUS

A. Hardware

Fig. 1 presents the block diagram of ARMUS' general structure. It is based on the AT91RM9200 processor. This choice is based on the following reasons:

- Atmel's high quality documentation and support to small capacity users.
- Having been on the market for some years now, the AT91RM9200 processor has shown to be reliable for our intended use, and easily available.
- It has very efficient power consumption, which is important in mobile robotic applications.
- It is Linux compatible, ensuring low development cost.

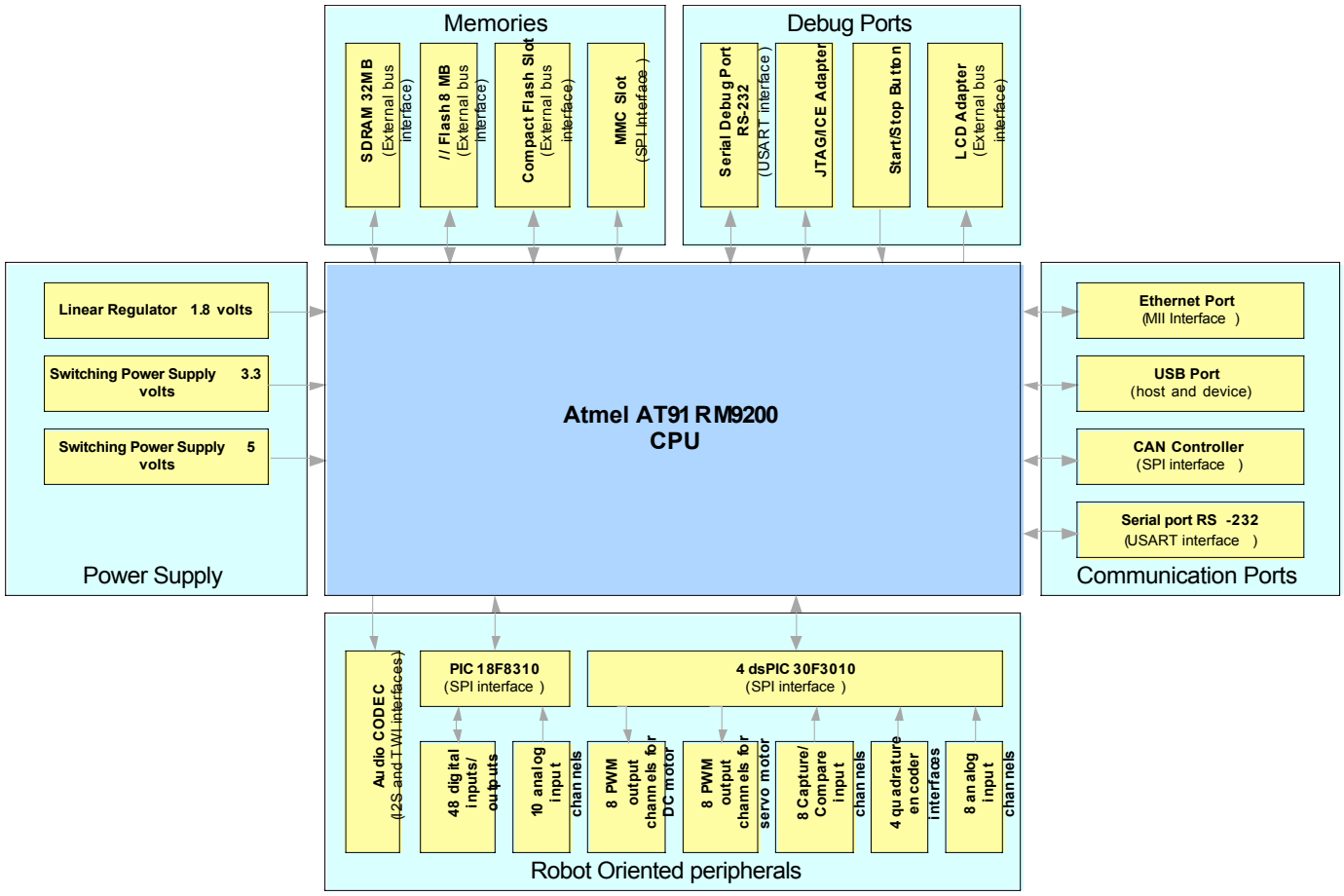

Figure 1: Bloc diagram of ARMUS' hardware design.

- It features a 200 MIPS ARM920T processor with 16K-byte instruction and 16K-byte data cache memories, 16K bytes of SRAM, 128K bytes of ROM, External Bus Interface featuring SDRAM, Burst Flash and Static Memory Controllers, USB Device and Host Interfaces, Ethernet 10/100 Base-T MAC, Power Management Controller, Real Time Clock, System Timer, Synchronous Serial Controller, 6-channel Timer-Counter, 4-channel USART, Two-Wire Interface, Serial Peripheral Interface, Multimedia Card Interface and Parallel I/O Controller.

ARMUS has five peripheral blocks aiming to maximize the processor capabilities in the context of robotic applications:

- **Memories:** There are two types of ROM memory: a 8 MB parallel Flash which is write-protected and contains the Linux Operating System; a Flash removable card, ranging from 2MB to 8 MB capacity (for instance we use Atmel's 8 MB Dataflash AT45DCB008) for the user-defined code. This is interfaced using a Serial Peripheral Interface (SPI) protocol and is entirely supported with our processor. For RAM memory, two 16 Mbytes SDRAM from Micron technology (MT48LC16M16A2P-7E) for a total of 32 Mbytes, not battery backed. Note that upgrading to 64 Mbytes is possible. The Compact Flash drive provides additional storage for miscellaneous data (e.g. MP3 files). It is a rather inexpensive option, fully supported by Linux.
- **Debug Ports:** Based on the design considerations, our system is equipped with a RS-232 serial port for debugging and downloading purposes. This port is also used for outputting the Linux console. Also included are a JTAG port (only used by our design team for low-level debugging), a LCD adaptor port and a simple start/stop button.
- **Communication Ports:** The design options were prioritized by general industry standards and for providing extensibility to the system. First, an Ethernet 10/100 port is mandatory with current technology. The MAC layer is fully supported by the CPU. The physical layer chosen is the Intel LXT971A via a MII interface. Second, because of its robustness and increased use in the design of modular robotic systems [9,10], a CAN port is included on the board. It uses the MPC2515 and MCP2551 controllers made by Microchip that communicate via SPI bus to CPU. Finally, USB host and device connexions are a valuable option for several peripheral connexions or simply for communication with a desktop computer.

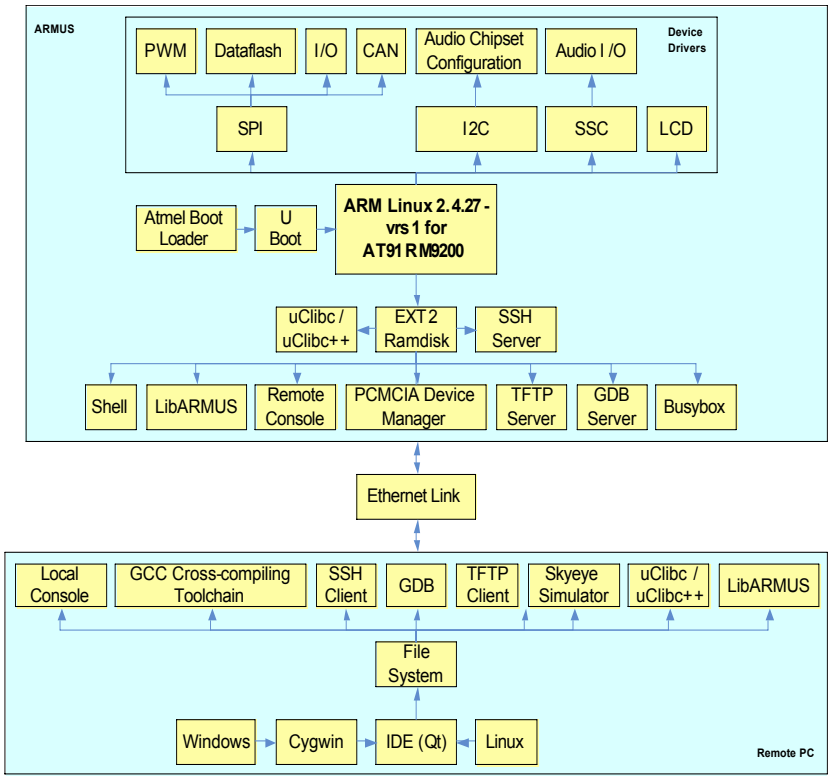

Figure 2: Block diagram of ARMUS' software programming environment

- **Robotic-Oriented Peripherals:** This block groups the necessary input/output (I/O) ports for robotic projects. ARMUS is equipped with a stereo audio CODEC interfaced with microphone input from Texas Instruments, (TLV320AIC23B, which is a commonly used device in the industry), 48 digital I/Os, 8 analog inputs, 4 PWM ports for DC motors (high frequency), 4 PWM for servo-motors (low frequency), 4 digital capture ports and 4 quadrature input ports for encoders. A PIC18F8310 is used to interface the digital and analog I/Os. Four dsPIC30F3010 are used for the PWM and other motor-related control devices. These will allow students to also get introduced to PIC microcontrollers. Note that an H-bridge is not implemented directly on the board, making it possible to use motors with the appropriate power requirements for the intended application.
- **Power Supply:** A 5 Volts switching power supply serves the PIC and the USB port. A 3.3 Volts switching power supply feeds the I/O port and the CPU. A 1.8 Volt linear regulator is used for the CPU's core demand. Note that because of the potentially high power demand for our application, the board is equipped with a switching power supply providing over 90% efficiency, thus higher battery lifetime.

B. Software

Fig. 2 illustrates ARMUS' software architecture. It represents software on the ARMUS board (installed on the robotic platform), and software running on an external computer (for code development of the intended robotic application using the board). Communication between the two is done using Ethernet, allowing rapid exchange (e.g., software download). It consists of three major elements:

- **ARMUS Operating System:** Combining our initial specifications with our hardware design, we chose ARM Linux as the OS. It is free, reliable, requires little space (between 100 kB to 1 MB ROM memory), offers a complete, sophisticated, multitasking environment, and is of course supported by ARM architecture. Fig 2. shows the services that should be implemented within the OS, providing the required runtime tools for user space applications: SSH server assures a remote link to the platform; a GDB server provides developers with a remote debugging interface; a TFTP server provides easy file transfer capability.
- **Device Drivers:** Although ARM Linux has been thoroughly ported to the AT91RM9200 microcontroller, its initial support targets the AT91RM9200EK evaluation kit. Thus, several driver designs are put in place to fulfill ARMUS' functionalities, such as sound (I1O), CAN bus, and LCD.
- **Development environment:** Fig. 3 illustrates the development environment created for ARMUS. The main challenge in designing our own development environment is to fully adapt it to first year undergraduate's knowledge and learning objectives. A QT-based interface (IDE (Qt)), providing a graphical frontend to manipulate the underlying tools, will give

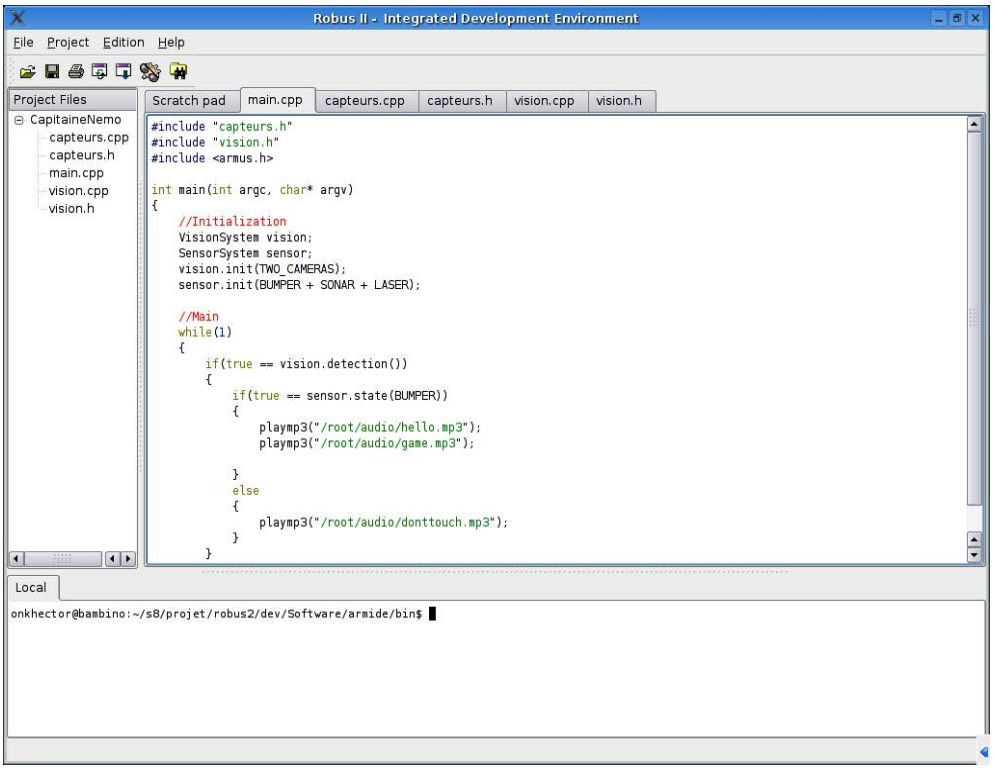

Figure 3: ARMUS' Integrated Development Environment.

sufficient functionalities and ergonomy. It is also a cross-platform toolkit, which means the interface can be accessed both from Linux and Windows. Different aspects are included in our design to provide an easy setup for students. For instance, the makefile used to cross-compile robotic applications will be taken in charge automatically by ARMUS' development environment. This means that GCC will be used transparently through the graphical interface. This ensures that users unfamiliar with GCC and its command-line interface will still be able to work with the development environment. The environment will also interact with various clients, that will communicate with the target platform to enable services (debugging, remote management, file transfer, etc.).

- **Driver interface library:** To ensure easiness of use by undergraduate students, a software library design is created to wrap driver (i.e. kernel-side) functions into a simple API. Each driver module is wrapped using a straightforward interface that, when used with the ARMUS development environment, will help students program robotic applications more easily.
- **Simulator:** Designing a simulator for a processing system is a challenging task that takes important time and effort. To reduce development time, we decided to use SkyEye, an Open Source Software Project (http://www.skyeye.org). SkyEye's goal is to provide an integrated simulation environment in Linux and Windows. SkyEye environment simulates typical embedded computer systems (e.g., Atmel AT91), running some OS such as ARM Linux, uClinux, uc/OS-II (ucos-ii) and analyze or debug them at source level. Although still under development, Skyeye provides a GBD (GNU Debugger)-like interface that supports the simulation of many widely used processors and their peripherals. This project may bring, in the near future, a free and open simulation opportunity that may help schools and universities reduce the number of platforms needed for basic development.

IV. ARMUS VS HANDY-ARM

In [6], Martin et al. presents the design specifications of the next generation of the Handy Board. It is intended to be more powerful than an 8-bit processor, but not as much to run a full operating system such as Linux. Their motivation is to do a lot with little, creating a platform simple and yet extremely capable. The board is based on Atmel AT91R4008 32-bit 66 Mhz ARM processor, coming with 256 Kbytes of internal static RAM. It will be equipped with 1 to 4 Mbytes of SRAM, 4 to 16 Mbytes of Flash ROM, a Crystal Semi CS8900 10 BT Ethernet port, serial communications ports, a LCD screen and motor drivers. It is intended to run with a small operating system (e.g., eCos OS), capable of running code compiled in C, C++ on a desktop computer, or interpreted language such as Lisp or Smalltalk.

ARMUS differs from the Handy-ARM by targeting Linux as the OS and more widespread programming languages (Lisp or Smalltalk are not taught to first-year undergraduate students). ARMUS provides more memory capability, communication ports, I/O ports, and considers the LCD and motor drivers as external modules for increased versatility. It does however take the same approach

of the Handy-ARM for I/O, using separate low-cost microcontrollers (Microchop PICs). This provides features that the Atmel processor does not offer (i.e., analog-to-digital conversion), and electrically isolate robot circuits from the Atmel processor. This prevents voltage spikes and other disturbances that could cause hardware failures in the main processor circuit.

IV. Design Status and Future Work

As of January 2006, the first ARMUS processing system prototype was built, rigorously tested and minor modifications were made to its design. Fig. 4 shows a picture of the designed board (dimension 16 cm x 13 cm). Several Linux device drivers were developed and tested to provide solid peripheral functionality. A beta version of the development environment has been created and, we believe, showcases the ARMUS functionalities very well while preserving easiness of use and ergonomy. The simulator's implementation has been delayed, since Skyeye is in active development. Team members are keeping a close eye on the project's evolution and consider using a stable version as its first candidate for implementation. The total budget for the project is 3000 $CAD. Our effort in this project is evaluated at 4000 hours. Based on our cost analysis, assuming that a 100 system would be produced in a first production phase, the estimated cost of the system would be around 300 $CAD, with a 6% contingency and profit margin. This is a little bit higher than the Handy Board (which is around 150$CAD, without labor for the assembly of the parts), but is still a reasonable price for the functionalities provided by ARMUS. As a possible extension to the system, a graphical user interface for the simulator and other functionalities of the programming environment would be useful. Such work could be carried out by another senior design team in 2006, ensuring the progress of the project and its use in Fall 2006 with first-year undergraduate EE and CE students.

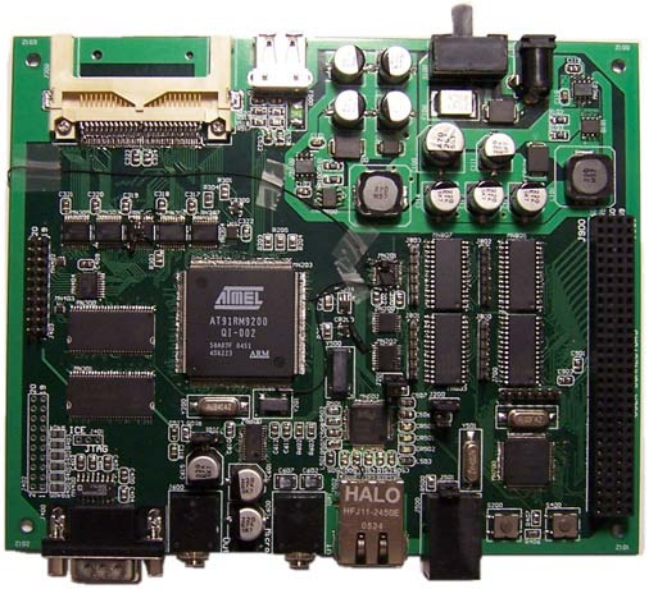

Figure 4: ARMUS' hardware.

As senior undergraduate students in EE and CE, we have found this project to be an incredible design experience. Conducting this project has allowed us to elaborate our capacity to acquire important management skills, overcome new technical difficulties in our engineering training, and make good use of our design skills. Our involvement goes farther than just satisfying the academic criteria of our specialized design courses, for which we are doing the work. We are also very encouraged by the fact that our work will contribute to the training of future engineers with state-of-the-art tools; four years ago we benefited in a similar way when we received training using the Handy Board. Our interest in robotics was initiated then, and we hope that by using ARMUS, others will become passionate about it as well.

Acknowledgment

F. Michaud holds the Canada Research Chair (CRC) in Mobile Robotics and Autonomous Intelligent Systems. This research is supported by Atmel Canada, Halo Electronics and the Department of Electrical Engineering and Computer Engineering of the Université de Sherbrooke.

References

[1] F. Martin, F., K. Par, K. Abu-Zahra, V. Dulskiy, A. Chanler, A. (2005). "iCricket: A programmable brick for kids' pervasive computing applications." *Proceedings 2nd International Workshop on Ubiquitous Computing*, Miami Beach, FL, May 2005.

[2] F. Martin, *The Art of Robotics: An Introduction to Engineering*, Addison-Wesley, 1998.

[3] D.J. Ahlgren, J.E. Mendelssohn, "The Trinity College Fire-Fighting Home Robot Contest: A medium for interdisciplinary engineering design", in *Proc. American Society for Engineering Education*, June 1998.

[4] F. Michaud, G. Lachiver, M. Lucas, A. Clavet, "ROBUS – A mobile robotic platform for Electrical and Computer Engineering Education", *IEEE Robotics and Automation Magazine*, Special Issue on *"Robotics in Education: An Integrated Systems Approach to Teaching"*, 20(3):20-24, 2003.

[5] F. Michaud, G. Lachiver, M. Lucas, A. Clavet, "Designing robot toys to help autistic children - An open design project for Electrical and Computer Engineering education", *Proceedings American Society for Engineering Education Conference*, St-Louis Missouri, 2000.

[6] F. Martin, G. Pantazopoulos, "Designing the next-generation Handy Board," Proceedings of the Spring 2004 AAAI Symposium, American Association for Artificial Intelligence, Stanford, CA.

[7] G. Lachiver, D. Dalle, N. Boutin, A. Clavet, F. Michaud, J.-M. Dirand, "Competency- and project-based programs in Electrical and Computer Engineering at the Université de Sherbrooke", *IEEE Canadian Review*, 41:21-24, 2002.

[8] S. J. Ylönen, A. J. Halme, "WorkPartner – Centaur like service robot," *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 727-732, 2002.

[9] F. Michaud, D. Létourneau, J.F. Paré, M.A. Legault, R. Cadrin, M. Arsenault, Y. Bergeron, M.C. Tremblay, F. Gagnon, M. Millette, P. Lepage, Y. Morin, S. Caron, S., "Multi-modal locomotion robotic platform using leg-track-wheel articulations", *Autonomous Robots, Special Issue on Unconventional Robotic Mobility*, 18(2):137-156, 2005.