

## A Generic Architecture of Modular Embedded System for Miniature Mobile Robots

Yan Meng, Kerry Johnson, Brian Simms, and Matthew Conforth

**Abstract**—Miniature robots have many advantages over their larger counterparts, such as low cost, low power, and easy to build a large scale team for complex tasks. Heterogeneous multi miniature robots could provide powerful situation awareness capability due to different locomotion capabilities and sensor information. However, it would be expensive and time consuming to develop specific embedded system for different type of robots. In this paper, we propose a generic modular embedded system architecture called SMARbot (Stevens Modular Autonomous Robot), which consists of a set of hardware and software modules that can be configured to construct various types of robot systems. These modules include a high performance microprocessor, a reconfigurable hardware component, wireless communication, and diverse sensor and actuator interfaces. The design of all the modules in electrical subsystem, the selection criteria for module components, and the real-time operating system are described. Some preliminary experimental results are also presented.

### I. INTRODUCTION

ONE of the advantages of the miniature mobile robot is its inherent small size. It can easily navigate through smaller, tight spaces, which is impossible for its larger counterpart. The characteristics of low-cost and low-power allows miniature robots to be constructed in large quantity and be used for various complex applications. By combining and coordinating a large number of small, inexpensive robots, various complex tasks can be accomplished with greater efficiency and flexibility, such as include urban search and rescue, de-mining, planet exploration, surveillance, and collective construction. For example, multi-robot coordination have been conducted in our previous work [1][2] for multi-target searching in an unknown area both in a customized Java based environment and using embodied robot simulator Player/Stage [3]. It would be too expensive to have a swarm of bigger robots to conduct experiments, a swarm of cheap and low power miniature robots would be feasible and affordable for this kind of experiments.

Y. Meng is with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, NJ 07030, USA. ( phone: 201-216-5496; fax: 201-216-8246; e-mail: yan.meng@stevens.edu).

K. Johnson is a graduate student in the Department of Electrical and Computer Engineering, Stevens Institute of Technology, NJ 07030, USA. (e-mail: kerry.johnson@stevens.edu).

B. Simms is a undergraduate student in the Department of Electrical and Computer Engineering, Stevens Institute of Technology, NJ 07030, USA. (e-mail: bsimms@stevens.edu).

M. Conforth is a graduate student in the Department of Electrical and Computer Engineering, Stevens Institute of Technology, NJ 07030, USA. (e-mail: matthew.conforth@stevens.edu).

Due to the limited capability of each individual miniature robot, it would be desirable to have different types of robots, such as wheeled robots, legged robots, flying robots, or underwater robots, which can work together to greatly increase the capability of an overall system.

However, it would be very expensive and time consuming to design different embedded systems for each specific robot system. It would be desirable to have a general purpose modular embedded system which can be applied to any type of the miniature robot systems, while minimum efforts would be spent to adapt to each specific type. To this end, a general purpose modular embedded system architecture for miniature mobile robots called SMARbot (Stevens Modular Autonomous Robot) is proposed in this paper, which aims at high performance, low cost, and low power. Our previous work [4] discusses some of our preliminary experiments and potential applications in the field of ubiquitous robotics, while this paper focuses on the modular design of SMARbot.

There are some available miniature robots that have been developed by other research groups. The autonomous mobile robot ALICE, developed by the Swiss Federal Institute of Technology Lausanne [5] is one of the examples. It uses a PIC16 series 8-bit microcontroller, infrared sensors, and a bidirectional radio communications module to perform many tasks, including self localization and map building, and other applications. Kenyon et al. [6] introduces a small, cheap, and portable reconnaissance robot with a unique spherical chassis. This robot provides a modular sensor interface so that its perception capabilities can be changed before runtime. The robot uses a PIC18LF458 8-bit microcontroller for processing and 2.4GHz wireless for communications. Tuci et al. [7] describe a robot system that uses embedded Linux on an XScale processor to run advanced swarming algorithms based around self-assembly, which uses infrared, temperature, light, acceleration, and many other sensors for perception.

InsBot is a miniature robot based off from ALICE, which is used to cooperate with cockroaches [8][9]. It uses two PIC18LF6720 8-bit processors combined with infrared sensors and a grey level camera for environmental awareness. It also has an 868 MHz radio and chemicals for communication. The iRobot's SwarmBot is equipped with both an ARM processor and a 200k gate FPGA. The infrared sensors, light sensors, and camera to help the robot navigate [10]. Krohling et al. [11] use a mutation algorithm to alter their robot Kephra's behavior by dynamically changing the Xilinx Virtex FPGA configuration during navigation tasks. Good collision avoidance was realized

with an intermediary mutation rate. The Kephera hardware uses 8 IR sensors and an RS-232 port for communications. Also based on this hardware is an implementation of neural networks on the FPGA. In [12], Roggen et al. use this neural algorithm for obstacle avoidance as well. Seshadri et al. [13] describe the development of a real-time operating system kernel for their 68HC12 based robot. The kernel is fully preemptive, and is designed to be efficient and small. The kernel size is nine kilobytes and the processor runs at 2MHz. The robot uses IR sensors, and focuses on self-reconfiguration, though this is not mentioned in detail.

It is obvious from the previous work that these robots generally have some common characteristics. They are compact, low power, and have a large number of sensors. Typically some methods of wireless communication are also implemented. More advanced designs have machine vision capabilities and reconfigurable hardware. The majority of these robots are very specific to their application, and many do not possess advanced computing capabilities. For example, InsBot and ALICE only have 8-bit microcontrollers and their physical construction is highly optimized to provide the smallest possible size. SwarmBot and Kephera provide solutions that are similar to the intent of SMARbot. However, they are supplied as complete systems that are specific to the chassis they are built on. The proposed architecture is intended to be less specialized than the presented previous works, and aims to be generic for various miniature robot types. Exact details concerning sensors and chassis design are purposely left ambiguous. The architecture only intends to provide a low cost, generic microprocessor, software, and reconfigurable hardware framework that will meet or exceed the processing requirements of most small mobile robot applications.

The proposed modular architecture of SMARbot consists of the following design features:

- A modular hardware and software architecture, where individual modules should be easily modifiable and reusable.
- A high performance microcontroller in a common and well supported architecture. Open source compilers and debugging tools are available.
- Reconfigurable hardware, well supported with software tools and documentation.
- Power-efficient wireless communication, where inexpensive and off the shelf module are ideal.
- A large number of interfaces for sensors. Sensors such as infrared proximity sensors, cameras, sonars, and bumper switches need to be connected while leaving room for future expansion.
- Low power. Mobile robots generally run off of small batteries. The system should not consume more than 1000mW.
- Low cost. Using high end microprocessors or FPGAs is out of the question. PCB design should fit the design rules of low cost, fast turn prototypes. Total cost should be under \$600USD each in single quantity.

- Small Size. The complete system should be approximately 3” cubed.
- Low risk design. Proven designs and off the shelf hardware will be used whenever available.
- Ease of fabrication. A reasonably skilled user should be able to assemble the necessary components into a working system.

The rest of this paper is organized as follows. Section II provides a high level description of the proposed modular architecture. Section III describes in detail the modular hardware. Section IV discusses software issues, including development tools, the real time operating system, and reusable and reconfigurable of software modules. Section V covers the current progress of prototype and proof of concept experiments. Section VI concludes and paper and discuss the future research.

## II. GENERIC MODULAR ARCHITECTURE

The objective of this architecture is to provide a modular framework for the purpose of building a generic embedded system for miniature robot system. Generally, there are three layers: the user interface layer, the hardware/software configuration control layer, and the sensor/actuator layer, as shown in Fig. 1.

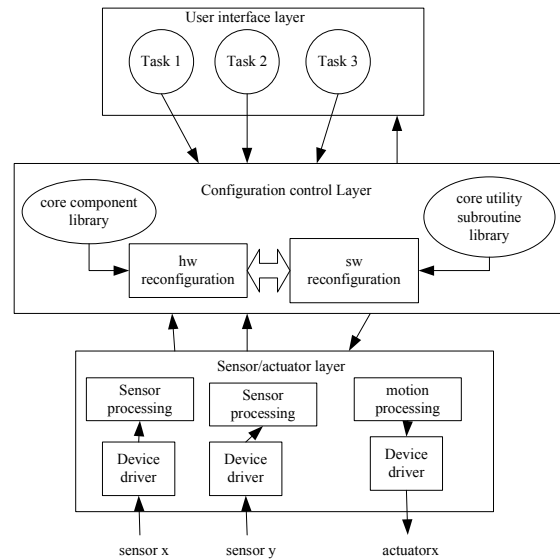


Fig. 1. The system architecture of SMARbot.

Multiple tasks, which are high-level descriptions of functions to be performed by the robot (for example “move to point x”), are defined in the user interface layer. When the post-conditions of one task and the pre-conditions of the next are satisfied, a reconfiguration can be executed by the microprocessor’s configuration control layer. The sensor/actuator layer consists of multiple sensors and actuators with corresponding device drivers and signal processing modules.

Our model of a control module provides a generic structure that is applicable to both periodic and aperiodic

real-time tasks. Multiple control modules execute in parallel and operate independently or cooperatively. Each control module has input ports, output ports, and resource connections. Input and output ports are used for communication between tasks in the same subsystem, while resource connections are used for external communication to other subsystems or the user interface. In order to provide automatic integration of the control modules, it is necessary that they are implemented as a set of basic components. All of the data flow, communication, synchronization, and scheduling is handled automatically by a real-time operating system.

### III. MODULAR HARDWARE DESIGN

The block diagram of a complete hardware system is shown in Fig. 2. Since this is a reconfigurable system, the users can select any number and kind of sensors and motors they like for their specific systems.

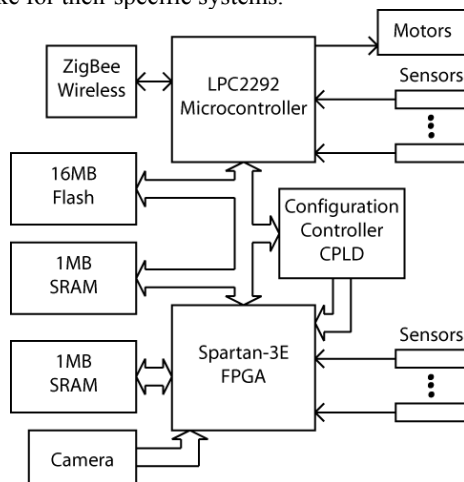


Fig. 2. Block diagram of the hardware modules

#### A. Microprocessor System Module

The microprocessor selected is the ARM7TDMI based NXP Semiconductor LPC2292 [14]. This 32-bit RISC ARM architecture was chosen because of its high performance, low power consumption, and pervasiveness throughout the microcontroller market. The LPC2292 in particular was selected because an external memory bus is required. The module also includes an additional 1MB of static RAM, and 16MB of flash. The large flash ROM is necessary for the storage of reconfigurable software modules and hardware configuration information.

#### B. Reconfigurable Hardware: FPGA Module

The reconfigurable hardware component is implemented as an FPGA connected directly to the microcontroller's external memory bus. This allows any FPGA resources to be directly mapped to system memory, providing the fastest and simplest access. Since low cost is an important design goal, and this system is more focused on computation than I/O, the low cost, high logic density Xilinx Spartan-3E [15] series of components was chosen.

Hardware development is done with Xilinx's free ISE WebPack design software.

Typically, an FPGA is configured by downloading a bitstream, which describes the internal logic functions and connections, from an external ROM. Since software controlled reconfiguration during runtime is a stated design goal, an alternate solution must be used. As in Khepera [12], configuration of the FPGA is accomplished by writing to the configuration port directly from the memory bus. Xilinx provides a reference design [16] utilizing a complex programmable logic device (CPLD) as glue logic to interface between the FPGA's configuration port and the microprocessor's memory bus. Since this method is supplied as a reference design and has been successfully used in a similar system, it easily satisfies the "low risk" design criteria.

The FPGA module also includes 1MB of SRAM. It is clear from the examination of several FPGA development boards that a flexible, general purpose FPGA system should include some amount of SRAM. The 1MB figure was chosen because it uses the same components as the microprocessor module. The external SRAM has the additional advantage of saving context during reconfiguration. Since the Xilinx Spartan-3E series of FPGAs do not possess the dynamic reconfiguration capabilities of high end devices, they must be reset and a new bitstream has to be loaded whenever reconfiguration is desired. If all of the important variables are stored in the external SRAM, they should be preserved through reconfiguration.

A simple example of the FPGA module's importance is the implementation of a quadrature decoder/counter. This particular functionality is necessary to utilize encoders on the wheel motors of a robot for motion control and odometry measurement. If an off-the-shelf integrated circuit is used, it will consume a great deal of space and I/O resources. In the FPGA case, the counter registers are directly mapped to the microprocessor's memory space, and no additional board area and I/Os are required. Since this particular component would most likely be in every configuration used, it can be loaded in the initial configuration, and differenced out of subsequent bitstreams.

#### C. Sensors and Wireless Communication Module

Sensors and wireless communication are critical to the operation of a robot system. Most details relating to sensors are left to the end users since they are highly application specific. The microcontroller system provides a large number of inputs and outputs, such as synchronous/asynchronous serial, timer/pulse width modulation, analog to digital converters, and CANbus. There are a large number (40) of extra digital I/Os from the FPGA which are also available to the end users. If the microcontroller is found lacking in any particular type of I/O, it is easy to attach additional interface circuitry to the microcontroller module or FPGA module. For example, if a large number of analog inputs are necessary, additional analog to digital converters could be attached to the

microcontroller's synchronous serial interface. A more complicated sensor, such as a camera, which requires a high speed parallel interface and buffering, could be attached to the FPGA module with an appropriate interface designed in the user's hardware description language of choice.

For wireless communication, an off-the-shelf MaxStream XBee ZigBee module [17] was chosen due to its low power consumption. The ZigBee module draws approximately 165mW with a 15m range, as opposed to over 1000mW for 802.11. It also has the capability of mesh networking, which allows data to reach a remote node by multiple hops. The MaxStream module has the particular advantages of small size, low cost, and transparent operation. Transparent operation is considered as a major advantage because it greatly simplifies the communications software module.

#### D. Power Module

The power module contains power supplies for the rest of the robot components. Again, most of the details of this module are left to the end users, as they are dependent on the application of the system. Since this module also has connections to the microprocessor GPIO, motor drivers and battery supervisors will also be included in most systems. An example of a system which will not include circuitry in the power module, other than power supplies, is a legged robot. In this case, the motor drivers are too large to fit into the power module and would dissipate a significant amount of heat. A separate motor driver module would be provided that also includes battery supervisory circuitry suitable for a high current draw.

#### E. Overall Hardware System Construction

Since the main goal of this design is modularity, the system is constructed as a set of stackable circuit boards. A typical design will consist of four boards: the power board, the microprocessor board, the FPGA board, and the sensor/interconnect board, as shown in Fig. 3. The power board is typically at the bottom of the stack, and contains, at a minimum, 3.3V and 5V power supplies. Generally, battery supervisory circuitry and motor drivers would also be included on this board. The microprocessor board is stacked on top of the power supply board, and contains the LPC2292 microcontroller, flash ROM, SRAM, and the core voltage power supply. Board-to-board connectors allow access to the microcontroller's GPIO and external memory bus. The FPGA board contains a Spartan-3E FPGA, configuration controller, SRAM, and associated power supplies. In addition to connections to the microcontroller's external memory bus, and a pass through for GPIO, a board-to-board connector for FPGA I/Os is included on the FPGA board.

The sensor/interconnect board provides connections and interface circuitry for any attached sensors, as well as wireless communication by connecting the ZigBee module to the microcontroller's serial port. It should be noted that if a tradeoff in performance for lower cost is desired, the FPGA board can be omitted, and the sensor board is connected directly to the microprocessor board. An optional debugging/programming board, which may not be part of

the final system, is also provided to temporarily allow access to the microcontroller and FPGA JTAG ports. Since the power and sensor/interconnect boards are very specific to the application of the system, their design is left to the end users.

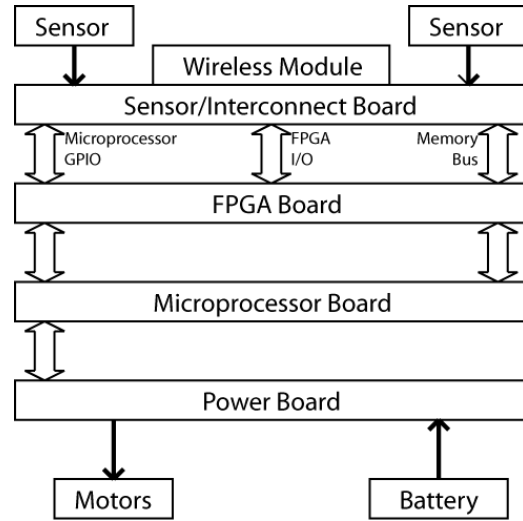


Fig. 3. Physical connection of the modules.

#### IV. SOFTWARE DESIGN

In a sophisticated embedded system, it is often necessary to utilize a real time operating system (RTOS) to schedule and coordinate the different processes that the system must execute. The project budget excluded most commercial RTOS options, but many open source solutions [18] are available. RTLinux is extremely powerful; consequently it is rather demanding on the hardware. Also, such an extensive feature-set creates more complexity [19] than this project requires. Given the low-level and invasive nature of some of our processes (e.g. resetting the FPGA during operation), it is advantageous to use an RTOS that allows more direct access to the hardware than is typical of RTLinux. After a survey of available open source RTOS implementations for the ARM7 architecture, the team decided that FreeRTOS is the best compromise, providing the needed capabilities and devoid of features we will not use. One of the major advantages of the ARM architecture is the ability to use open source tools for everything from system software to development tools. As per [20][21], we have used the GNU tool chain compiler along with the Eclipse IDE [22], and OpenOCD [23] for JTAG on-chip debugging.

FreeRTOS is a portable, light weight real-time kernel. It is essentially the same software as the IEC 61508 certified commercial product, SafeRTOS [24]. The goal of the FreeRTOS project is to be simple, portable, and concise [25]. It is written completely in C, except for a few assembler functions. It supports both preemptive and cooperative real time task scheduling. It optionally supports co-routines, which are basically light-weight tasks. Both

message queues and semaphores/mutexes are provided to facilitate interprocess communication and synchronization. The small and easy to work with source code has been ported to a wide variety of platforms, including ARM7. Extensive documentation and demo source code is also provided [26]. Furthermore, ports exist for Windows and the Keil software simulator, which was quite helpful when the team had only one demo board. Thus far, FreeRTOS has been more than sufficient for the demands of this project.

The key to developing reusable and reconfigurable software is to first create a hardware abstraction layer (HAL) and then write all of the operational code using the HAL interface. The HAL hides features which are specific to our hardware and instead presents high level logical interfaces. For example, the PWMs are abstracted as “motor” objects that have settings such as “throttle” and “direction.” When writing, e.g. the navigation code, we do not reference any of the hardware interfaces directly; we use the HAL. Thus, that code does not know about concepts such as PWM duty cycle. In this way, any software developed for the SMARbot can easily be ported to other hardware, since only the code for the HAL itself needs to be modified. The operational code is also divided into modules. Each module provides a type of general functionality not tied to any particular task or hardware, such as “communication,” “movement,” “sensing,” etc. Tasks are built as compositions of simpler tasks, e.g. a box pushing task could be a composition of a bump sensor monitor task and a localization task. The localization task could, itself, be the composition of an IR sensor monitor task and a mapping task. The idea is to build up a library of simple tasks which can be composed in different ways to achieve a diverse range of high level tasks. The entire system has many layers, and each layer is coded using the abstractions created by the layers beneath it. This makes the software reusable and reconfigurable.

## V. SYSTEM PROTOTYPE

### A. Current Prototype

The current prototype is shown in Fig. 4, which addresses all aspects of the architecture we proposed in Fig. 1, 2 and 3.



Fig. 4. The current prototype of a SMARbot

The LPC2292 was chosen because it is generally well supported and its widespread usage. The FPGA board uses an FPGA in a 256 pin 1mm pitch BGA package. This is somewhat marginal in satisfying the “ease of fabrication” and “low cost” design requirements due to the fact that the package must be reflow soldered using specialized equipment. The power board contains only a 3.3 and 5 volt supply. A battery fuel gauge circuit and motor driver chip was added, providing a complete power solution for a small robot.

The latest revision of the chassis provides bumpers, mounting for the boards, and brackets for the IR and sonar sensors. Construction is largely of laser cut ABS plastic, with a minimum of machined components. A protective housing covers most of the components to prevent snagging. The new chassis also provides plenty of room for a larger two-cell lithium battery pack.

For the latest version of the chassis, a sensor/interconnect board was constructed that provides interfaces for three IR range sensors, two sonars, four bumper switches, two quadrature encoders, and a camera. The sensor interfaces are trivial, consisting of a MOSFET to control power to the IR sensors, wiring for the sonars, pullup resistors for the bumper switches, 5v to 3.3v level shifters to interface the quadrature encoders to the FPGA, and wiring to connect the camera to the FPGA. The ZigBee module is mounted on the top of the board and attached to one UART, while the other is reserved for future expansion or debugging, and attached to a connector. Since there was a large amount of extra space, connectors for the FPGA, CPLD, and microcontroller JTAG interfaces were added, thus avoiding the necessity of a separate debug board. It should be noted that there are a large number of extra microcontroller and FPGA IOs, which provide ample interfaces for future expansion.

### B. Preliminary Experiments

The current prototype was put through some preliminary tests in order to evaluate its performance. First, collision avoidance was implemented, using IR sensors to avoid an object placed in the robot’s path. Fig. 5 shows the robot approaching the object, backing up and turning to avoid it.



Fig. 5. Autonomous obstacle avoidance. The robot approaches an obstacle, stops, reverses, and turns away

In order to test the wireless communications, a Zigbee development board was connected to a computer. HyperTerminal was then used to send commands to the robot to change its speed and direction. Also, data from the infrared sensors was sent back to the computer for

inspection. This test setup proved very useful in testing the initial prototype.

For the software, target detection algorithms were prototyped using OpenCV. The preliminary software does basic color blob tacking by first converting RGB to HSV color space. The image is then thresholded based on H to form a binary representation, from which the centroid can be determined. In Fig. 6, the left side shows the original image, while the right side shows the thresholded image. The main goal of prototyping this algorithm is to ensure it functions as expected before implementation in hardware. VHDL code for this functionality was also simulated.

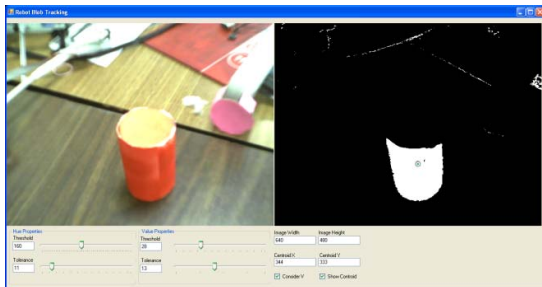


Fig. 6. The blob tracking development program

## VI. CONCLUSION AND FUTURE WORK

A generic modular embedded system architecture for miniature mobile robots was presented in this paper, which provides a large amount of functionality in a small, low-power, low-cost, and highly modular platform. With a completely modular architecture that allows individual components to easily be added, replaced, or modified and a large number of sensor and actuator interfaces, many different miniature mobile robot systems can be constructed.

Since the hardware design is finalized and fabricated, we will mainly focus on developing the software on the miniature robot systems. Algorithms for sensor data acquisition and motor control will be developed first. This will be followed by some higher level tasks, such as sensor perception, robot localization, and navigation techniques. To make the platform user friendly, most basic functions will be provided as libraries to the end users, who will only concentrate on high-level task-specific development.

This generalized modular embedded system platform will also be applied to our next miniature robot platform: a bio-inspired quadruped robot. Each leg has three DOF, so the system requires at least twelve PID control loops. The FPGA can be used for multi-axis PID control as in [27].

## REFERENCES

- [1] Y. Meng, O. Kazeem, and J. Muller, "A Hybrid ACO/PSO Control Algorithm for Distributed Swarm Robots", *2007 IEEE Swarm Intelligence Symposium*, April 1-5, 2007, Honolulu, Hawaii, USA.
- [2] Y. Meng and J. Gan, LIVS: Local Interaction via Virtual Stigmergy Coordination in Distributed Search and Collective Cleanup, *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2007)*. Oct. 29 - Nov. 2, 2007, San Diego, CA, USA.
- [3] B. Gerkey, R. T. Vaughan and A. Howard. "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems". In

- Proceedings of the 11th International Conference on Advanced Robotics (ICAR 2003)*, pages 317-323, Coimbra, Portugal, June 2003.
- [4] Y. Meng, K. Johnson, B. Simms, M. Conforth, C. Creamer, and C. Hawley, "SMARbot: A Miniature Mobile Robot Paradigm for Ubiquitous Computing," *International Conference on Intelligent Pervasive Computing*, 2007.
- [5] G. Caprari, K. O. Arras, and R. Siegwart, "The Autonomous Miniature Robot ALICE: from Prototypes to Applications," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000.
- [6] S. Kenyon, D. Creary, D. Thi, and J. Maynard, "A small, cheap, and portable reconnaissance robot," *Sensors and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IV*, Vol. 5778, 2005.
- [7] E. Tuci, R. Gross, V. Trianni, F. Mondada, M. Bonani, and M. Dorigo, "Cooperation Through Self-Assembly in Multi-Robot Systems," *ACM Transactions on Autonomous and Adaptive Systems*, Vol. 1, No. 2, pp. 115-150, Dec, 2006.
- [8] A. Colot, G. Caprari, and R. Siegwart, "InsBot: Design of an Autonomous Mini Mobile Robot Able to Interact with Cockroaches," *IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 2418-2423, 2004.
- [9] G. Caprari, A. Colot, R. Siegwart, J. Halloy, and J.-L. Deneubourg, "Building Mixed Societies of Animals and Robots," *IEEE Robot. Automat. Mag.*, Vol. 12, No. 2, pp. 58-65, 2005.
- [10] J. McLurkin, and J. Smith, "Distributed Algorithms for Dispersion in Indoor Environments Using a Swarm of Autonomous Mobile Robots," *7th International Symposium on Distributed Autonomous Robotic Systems*, 2004.
- [11] R. Krohling, Y. Zhou, and A. Tyrrell, "Evolving FPGA-based robot controllers using an evolutionary algorithm," *First International Conference on Artificial Immune Systems*, pp. 41-46, Sept, 2002.
- [12] D. Roggen, S. Hofmann, Y. Thoma, and D. Floreano, "Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot," *NASA/Dod Conference on Evolvable Hardware*, 2003.
- [13] A. Seshadri, M. Ferretti, A. Castano, and P. Will, "A Distributed Embedded System for Modular Self-Reconfigurable Robots," *ISI Technical Report ISI-TR-551*, Dec, 2001.
- [14] NXP Semiconductors, "LPC2292/LPC2294 Datasheet," Feb, 2007.
- [15] Xilinx Incorporated, "Spartan-3E FPGA Family: Complete Data Sheet," May, 2007.
- [16] M. Ng and M. Peattie, "Using a Microprocessor to Configure Xilinx FPGAs via Slave Serial or SelectMAP Mode," *XAPP502 (v1.4)*, Nov, 2002.
- [17] Maxstream Incorporated, "XBee/XBee-PRO OEM RF Modules Product Manual v1.xAx - 802.15.4 Protocol," Oct, 2006.
- [18] T. Straumann, "Open Source Real Time Operating Systems Overview," *8th International Conference on Accelerator & Large Experimental Physics Control Systems*, 2001.
- [19] K. Andersson, and R. Andersson, "A comparison between FreeRTOS and RTLinux in embedded real-time systems," <http://www.streambag.se/files/rtproj.pdf>
- [20] J. Lynch, "ARM Cross Development with Eclipse Version 3," Dec, 2005.
- [21] R. Barry, "Tutorial: Using a completely open source tool chain for ARM software development," <http://www.elektronikpraxis.vogel.de/filesserver/vogelonline/files/419.pdf>
- [22] The Eclipse Software Foundation, <http://www.eclipse.org/>.
- [23] <http://openocd.berlios.de/web/>
- [24] Wittenstein high integrity systems, "SafeRTOS - an IEC 61508 derivative of FreeRTOS for safety related systems," <http://www.highintegritysystems.com/safertosfaq.html>
- [25] R. Barry, "FreeRTOS," <http://www.freertos.org>
- [26] R. Goyette, "An Analysis and Description of the Inner Workings of the FreeRTOS Kernel," *SYSC 5701*, 2007.
- [27] W. Zhao, B. H. Kim, A. Larson, and R. Voyles, "FPGA Implementation of Closed-Loop Control System for Small-Scale Robot," *12th International Conference on Advanced Robotics*, pp. 70-77, 2005.