

 Open access • Proceedings Article • DOI:10.1109/IROS.2006.281719

SUPERBOT: A Deployable, Multi-Functional, and Modular Self-Reconfigurable Robotic System — [Source link](#)

Behnam Salemi, Mark Moll, Wei-Min Shen

Institutions: University of Southern California

Published on: 01 Oct 2006 - Intelligent Robots and Systems

Topics: Self-reconfiguring modular robot, Mobile robot, Robot and Modular design

Related papers:

- [M-TRAN: self-reconfigurable modular robotic system](#)
- [PolyBot: a modular reconfigurable robot](#)
- [Modular Self-Reconfigurable Robot Systems \[Grand Challenges of Robotics\]](#)
- [Modular ATRON: modules for a self-reconfigurable robot](#)
- [Distributed Self-Reconfiguration of M-TRAN III Modular Robotic System](#)

Share this paper:    

View more about this paper here: <https://typeset.io/papers/superbot-a-deployable-multi-functional-and-modular-self-lunllcp5ug>

SUPERBOT: A Deployable, Multi-Functional, and Modular Self-Reconfigurable Robotic System

Behnam Salemi, Mark Moll, and Wei-Min Shen

Information Sciences Institute
University of Southern California
Marina del Rey, USA
{salemi, moll, shen}@isi.edu

Abstract – *Self-reconfigurable robots are modular robots that can autonomously change their shape and size to meet specific operational demands. Recently, there has been a great interest in using self-reconfigurable robots in applications such as reconnaissance, rescue missions, and space applications. Designing and controlling self-reconfigurable robots is a difficult task. Hence, the research has primarily been focused on developing systems that can function in a controlled environment. This paper presents a novel self-reconfigurable robotic system called SuperBot, which addresses the challenges of building and controlling deployable self-reconfigurable robots. Six prototype modules have been built and preliminary experimental results demonstrate that SuperBot is a flexible and powerful system that can be used in challenging real-world applications.*

Index Terms – *Modular Robots, Self-Reconfigurable Robots, Distributed Robotics, Deployable Robots*

I. INTRODUCTION

Self-reconfigurable robots are modular robots that can autonomously change their shape and size to meet specific operational demands. In domains where the task and environment are known, it is often more efficient to build fixed-shape special purpose robots [1]. However, in applications such as reconnaissance, rescue missions, and space applications where the task and environment are not fully known self-reconfigurable robots can out-perform fixed-shape robots.

Because of their self-reconfigurability, self-reconfigurable robots are more versatile, fault-tolerant, and efficient compared to their non self-reconfigurable counterparts. For example, a self-reconfigurable robot can become a snakelike robot to crawl through a narrow passage and then grow legs and grippers to pick up and carry an object.

Building and controlling self-reconfigurable robots are very difficult. Designing modules that are flexible and allow for efficient performance of locomotion, object manipulation and self-reconfiguration is very challenging. Connectors are critical parts of a self-reconfigurable module [2]. A successful connector design requires solving many challenging problems for enabling modules to reliably dock/undock and create a strong and effective robot.

In the past decade, researchers have addressed the hardware and software challenges of building self-reconfigurable robots. In the area of hardware design, Chain-type [3,4,6,7], and Lattice-based [8,9,10,11] self-

reconfigurable robots have been developed. Lattice-based self-reconfigurable robots consist of a set of modules that can only attach to other modules in discrete locations on a lattice. They require continuous self-reconfiguration for locomotion and manipulation. Chain-type self-reconfigurable robotic systems are made up of linear, looped, or branched chains of homogeneous or heterogeneous modules. This class of self-reconfigurable robots are able to separate locomotion from reconfiguration and for locomotion and manipulation these robots do not require continuous self-reconfiguration.

In the area of software, many centralized and distributed software systems have been presented. Distributed approaches include Digital-Hormone control [16,17], Role-Based control [18], and local search approaches such as [19]. Centralized approaches include gait-table control [1], and M-TRAN control system [5]. Difficulties in controlling Self-reconfigurable robots stem from the fact that a network of self-reconfigurable modules is distributed in nature, modules have limited information, and the overall behaviour of a robot emerges from the coordinated actions of all constituting modules. A control system for such robots needs to be adaptive to dynamic topology of the robot, scalable, fault tolerant and distributed to avoid introducing a single point on failure in the system.

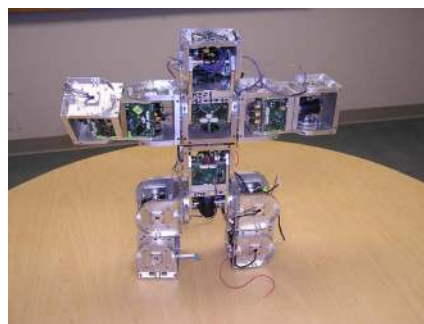


Fig. 1 A network of 6 SuperBot Self-reconfigurable modules

Due to the difficulties of designing and controlling self-reconfigurable robots, in the past, almost all the research efforts have been focused on developing hardware and software systems that can function in the controlled indoor or outdoor environment of a Lab. However, less attention has been on designing fully functional self-reconfigurable robots for real applications. Fortunately, recent advances in design

and control of self-reconfigurable robots have made this dream become a possibility.

Transition from a controlled environment to a real-world situation, however, introduces many new challenges to the field of self-reconfigurable robotics. These challenges include efficient performance of locomotion, manipulation, and self-reconfiguration tasks in the presence of obstacles, power management issues, modules mechanical and electronic endurance and reliability in spite of being in contact with a rough environment, dealing with dust, moisture, and strong light sources, designing reliable and strong connectors, sensing and meaningful interaction with the environment, and efficient human-robot interaction and control.

In this paper, we present a novel deployable and multi-functional self-reconfigurable robotic system called SuperBot. SuperBot is being designed for NASA space exploration programs and addresses the above-mentioned challenges. Figure 1 shows a network of six connected SuperBot modules.

The form factor of SuperBot modules is similar to that of the MTRAN modules [6]. However, SuperBot modules have three degrees of freedom, and such capability has drastically increased the mobility and flexibility of individual and networked modules. SuperBot modules are designed to be strong, flexible, and capable of performing efficient locomotion, self-reconfiguration, and manipulation tasks. Similar to MTRAN, a network of SuperBot modules can perform as both lattice-based and chain-type self-reconfigurable robots. A network of SuperBot modules is capable of sharing power and communicating using high-speed infra-red LEDs. The on-board multi-threaded software controls modules' functionalities and coordinates the behaviours of the network of modules in a distributed fashion.

This paper is organized as follows: In Section two we discuss the considerations in designing SuperBot. In section three we will describe its hardware architecture. In section four we will describe the software architecture. Section five describes some performance evaluation experiments; and section six will conclude and describes the future research directions.

II. DESIGN CONSIDERATIONS

The basic design philosophy of Superbot modules is to develop flexible, powerful and sturdy modules that can efficiently perform tasks in an uncontrolled environment without requiring close attention. In order to accomplish this goal six criteria were considered in the design and construction of SuperBot.

First, as SuperBot is intended to operate in a harsh and rough environment, the design needed to allow for rugged and sealable modules. Modules and connectors needed to cover their internal electronic and mechanical components and protect them from dust, moisture, and physical impact. The building materials needed to be resistant to abrasion and other deleterious effects.

Second, SuperBot is required to perform locomotion, manipulation and self-reconfiguration tasks in the presence of obstacles in an uncontrolled environment. Therefore, it was

essential for the modules to have enough dexterity in order to maneuver around obstacles to perform the task in hand and at the same time conserve energy by minimizing the number of required movements.

Third, to be effective in real applications, SuperBot modules should have enough torque to move and lift a reasonable number of neighboring modules and exert force whenever it is needed. This required maximizing the power of actuators while the size and weight of the module are kept minimal.

Fourth, a network of SuperBot modules should be cognizant of their environment through a series of sensors which allow them to avoid obstacles and also navigate in the environment. This also includes the ability of sensing and communicating with other SuperBot modules. Due to the distributed nature of a network of SuperBot modules, sensory information is available in the network of modules in a distributed form. The sensory information might have to be fused for autonomous decision-making or being communicated to a controller host.

Fifth, available power in a network of SuperBot modules should be efficiently used and managed. Some modules may need to move more often and spend more energy while some other modules may not move at all. In addition, the power source of some modules may fail. In such situations, modules should be able to share the available energy in the network. In addition, SuperBot should be able to connect one of its modules to a charging station and charge all connected modules.

Sixth, distributed control software was necessary for effective use of SuperBot. The control software needed to be real-time, fault tolerant and scalable. In addition, it had to accept and execute high-level commands for locomotion, manipulation and self-reconfiguration from a remote host without requiring detailed instructions for individual modules.

In addition to above considerations, the following criteria were also considered in the design of the connectors: First, the connectors needed to be genderless meaning that any connector of a module had to be able to dock to any other connector of another module. Second, two docked connectors could be oriented relative to each other in 90° intervals. Third, connectors needed to enable communication and power sharing among modules. Fourth, either side of two docked connectors had to be able to undock. This is necessary in a situation where one side is dead. In this situation the other side should be able to release itself. Fifth, the connector needed to enforce necessary tension to hold docked faced together. Sixth, connectors needed to sense and guide docking process. Seventh, as it was mentioned earlier, connectors needed to be sand and moisture proof and resistant to the abrasion.

At the time of the preparation of this paper, several designs that fulfill on all the above requirements were being prototyped and tested. The analysis and experimental results of the finalized connector design will be reported in the future publications. For the time being, manual connectors have been used for docking purposes.

III. MECHANICAL DESIGN

The overall body of a SuperBot module is in the form of two linked cubes. The dimensions of each cube are 84x84x84 millimeter and therefore each module is 168 mm long. The current prototypes are made up of a hard aluminium alloy and weigh about 500 grams including the electronics and batteries. Each module consists of three main parts: Two end effectors and a rotating central part. This allows a module to have three degrees of freedom in the form of 180° yaw, 180° pitch, and 270° roll; see figure 2. This design gives the SuperBot module the most flexible movements that we know in the literature, and will allow a single module to bend and twist into many different shapes and provide the needed flexibility for multimode locomotion.

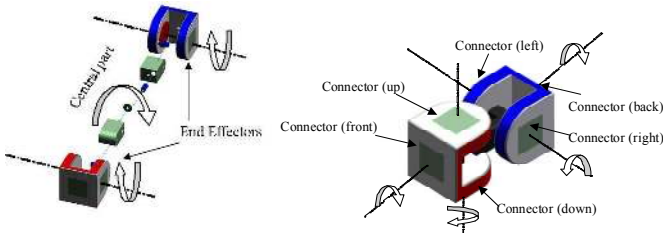


Fig. 2 SuperBot module design and Degrees of Freedom.

Other designs such as MTRAN [6] which do not have the rolling ability of the central part can in many cases produce a similar roll effect of a module through execution of a sequence of bending, docking, and undocking actions. However, that requires consumption of more energy and performing them may not be possible in the presence of other obstacles. Figure 3 shows a prototype of Superbot module where as a result of the 90° rotation of the central part the end effectors are in two different modes. In figure 3a, the end effectors sweeping planes are perpendicular to each other which are equivalent to the CONRO [4] module design and in figure 3b the end effectors sweeping planes coincide which are equivalent to the MTRAN [6] module design.



Fig. 3 Two modes of Superbot; similar to (a) CONRO and (b) M-TRAN

There are six connectors on each Superbot module; one on each side of the end effectors; see figure 2. Any of the six connectors of a module can connect to any connectors of another module in all 90° interval orientations.

The drive train of each degree of freedom of a module consists of a MicroMo® DC electric motor, a planetary gearbox, and an external gearbox; see figure 4. The DC motor outputs between 5 to 21.18 milli-Newton-meter torque. The gear ratio of the planetary gearbox is 1:86 and its efficiency is

70%. The gear ratio of the external gearbox is 1:5. Based on the following calculation this results a maximum of 6.38 Nm torque.

$$21.18 \times 86 \times 70\% \times 5 = 6375 \text{ mN-m} = 6.38 \text{ N-m}$$

Given the size and weight of each module, this amount of torque is enough for reliably lifting three neighboring modules.

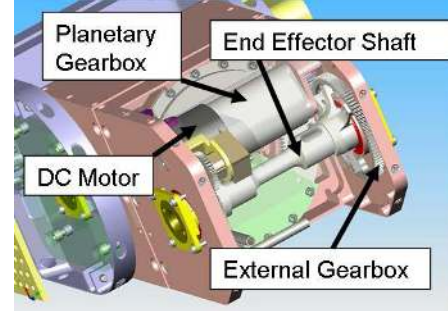


Fig. 4 SuperBot Motor, Gearboxes and end effector drive shaft

IV. HARDWARE ARCHITECTURE

SuperBot possesses a modular hardware architecture. Each module's on-board hardware is responsible for controlling the actuators, connectors and sensors, power management, communicating with neighboring modules, autonomous decision-making, and distributed control of high-level behaviours.

Each half module (cube) has a controller. The controller of the half module containing the battery and one motor is called the 'master controller' and the controller of the other half is called the 'slave controller'; see figure 5. Both controllers are connected through power lines and a bi-directional 400 Kb/S I2C bus. I2C is a two-wire bus and is selected to provide enough bandwidth between half modules and at the same time keep the number of wires among the cubes low. Each controller is responsible for managing the motors, sensors, communication, power and docking of its corresponding cube. In addition, the master controller is responsible for running the high-level behaviour controller in each module.

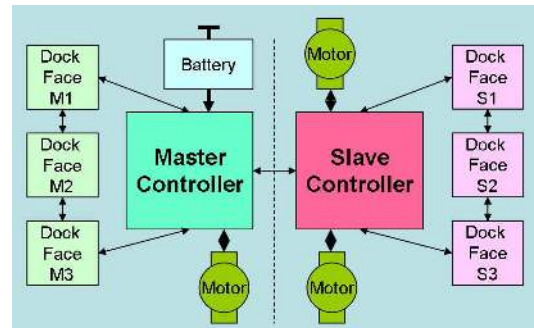


Fig. 5 SuperBot hardware control Architecture

Each controller is based on a 16 MHz ATmega128 microcontroller [20], which is an 8-bit low power AVR processor with 128 Kbytes of flash program memory, 4

Kbytes of EEPROM and 4 Kbytes of internal SRAM. The ATmega128 also includes an 8-channel 10-bit ADC, three timers, and several bus interfaces including two USARTs, SPI, and I2C.

Figure 6 shows the details of the master controller. A wireless receiver is considered for remote on/off, motor disable, to stop modules while the control program is running, and receiving serial commands. The ATmega128 can measure the voltage and output current of the battery. PWM pulses are interfaced to the motor through an H-bridge for controlling the motor speed. The angular position of the end effector is sensed by a potentiometer that is coupled to its shaft and is connected to an A/D line of the ATmega128. A one Mb/s SPI communication bus is used for communicating with dock faces. This provides enough bandwidth to communicate with three dock faces that communicate with their neighboring modules through 230K Baud RS-232 lines. Details about the communication circuits are given below. A 3D accelerometer/inclinometer is also interfaced through the SPI bus. A JTAG port is used for debugging purposes. Figure 7 shows the details of the slave controller, which has a similar architecture.

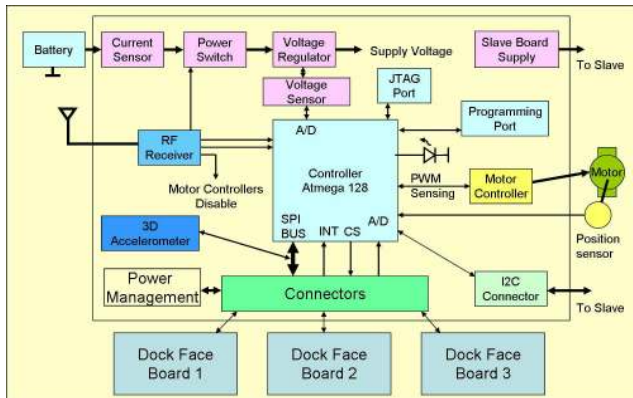


Fig. 6 Master Controller Architecture

Figure 8 shows the details of the communication interface on a dock face. A communication interface has four infra-red receiver LEDs and a transmitter LED. Any combinations of the receiver channels can be selected which results the sum of the received signals on each receiver LED to be delivered to a buffer stage. The output of the buffer is connected to an A/D channel of the corresponding controller. As a result the controller can measure the intensity of the input signal. This analogue value is proportional to the distance and angle of a nearby docking face and is used for guiding the docking process of two modules. This analogue value ranges from 0 to 4.5 volt for a transmitter LED at 40cm distance to coincided docking faces, respectively. The four channels on each module engaged in a docking process results eight channels of information, which allows for guiding the docking process in 3D space. In addition, receivers of a module can read the analogue value produced by the reflection of the module's own transmitter LED. This can be used to measure the distance of a docking face from a reflective object.

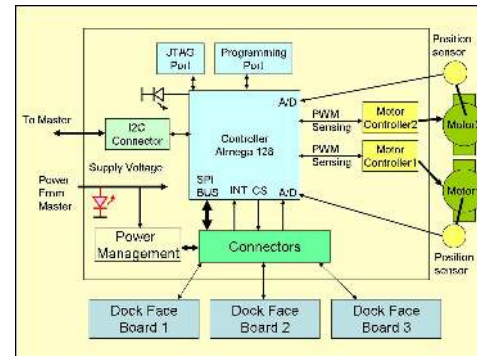


Fig. 7 Slave Controller Architecture

The amplifier stage is used to amplify and shape a digital signal received from another module during communication. Modules can communicate as far as up to one meter. The communication speed is 230K Baud and an IrDA timing mode is used. When a byte of data is received from a neighboring module, the SPI/RS232 interface, via a MAX3100 chip, generates an interrupt and the corresponding controller reads the received byte through the SPI bus. This interrupt driven architecture allows the controllers to use their time to perform other tasks and attend to the communication module only when there is a byte of information to be retrieved.

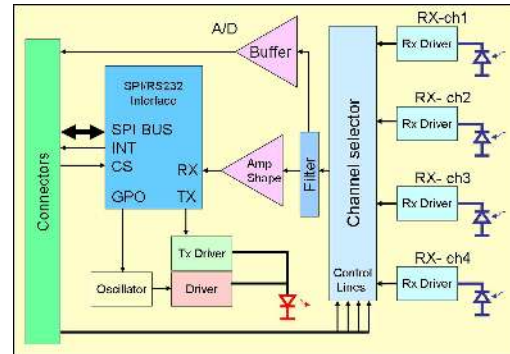


Fig. 8 Communication interface on a dock face.

In order to transmit a byte of data, a controller just needs to write the byte into the SPI/RS232 interface buffer and the rest of the process is taken care of by the interface. The output infra-red light of the transmitter LED can also be modulated through a command from a General Purpose Output (GPO) pin. This will generate a continuous modulated infra-red light to be received by the receiving LEDs for guiding the docking process. The modulated signal in combination with the filter module is used for removing DC level noise such as sun light in an outdoor environment.

Figure 9 shows the power sharing schematic. In each of the six docking faces there is switch/diode combination, which allows the current to always flow in. However, the out current is only possible if the switches are closed, which are controlled by the controller of the corresponding half module. The default position of the switch connected to the battery is on the charger side and the rest of the switches are normally open.

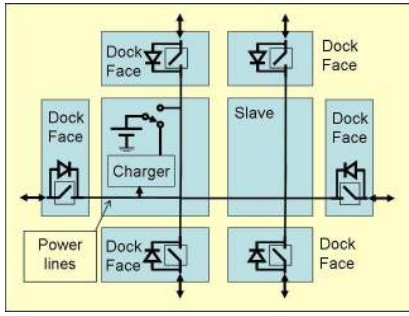


Fig. 9 Power Sharing circuit schematic.

In an initial situation where all the batteries of a connected network of modules are fully discharged (all modules are dead) as soon as a charging source is connected to one of the connectors of a module, its battery starts charging through the input current and the battery switch. Once the module battery is charged, the controllers will become active and the output switches can be connected one by one to allow other modules to start charging without overloading the charging source. This design allows modules to share the battery power at will and in situations where the inside batteries fail, modules can get power from other modules. In the current prototype a 1600mAh, 7.4 volt lithium-polymer battery is used.

V. SOFTWARE ARCHITECTURE

The control of the SuperBot system is a challenging task, for modules must be able to dynamically reconfigure into different configurations/functionalities and support plug-and-play with other types of devices. Our approach to this challenge will build upon our previous work on (i) hormone inspired distributed control, (ii) table based control for fast prototyping, and (iii) phase automata for coordinating module activities. This approach allows the SuperBot system to be:

- **Distributed:** to support decentralized control and avoid single point failures (i.e., a single module failure would not paralyze the entire system). A module must select its actions based not on its absolute address or unique identifier, but based on its topological location in the current configuration.
- **Collaborative:** to allow modules to negotiate the best actions for a global task. For example, if a snake's head module wants to move forward while the tail wants to move backward, then they must negotiate to select the best action for the entire system.
- **Dynamic:** to be able to adapt to the topological changes in the module network and support all possible configurations.
- **Asynchronous:** to synchronize modules actions without a global clock.
- **Scalable:** to work for any configuration regardless of the shape and size.

The SuperBot software consists of three main components:

A. Low-level Software

The low-level software on the modules hides the details of low-level control of the hardware from the behavior software programmer and is built on top of AvrX, a small real-time

kernel for embedded processors [21]. All system-level and user-level code is written in C language as separate tasks. Associated with each task is a message queue. Tasks can communicate with each other by placing messages into each other's queue. Tasks can be set up to run periodically or to be run "on demand." Figure 10 shows a simplified diagram of the tasks running on a SuperBot module.

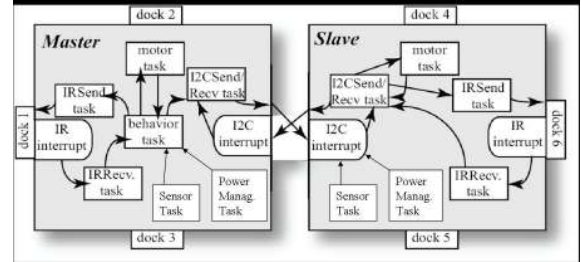


Fig. 10 Tasks running on a SuperBot module

The AvrX kernel runs on each of the Atmega128 controllers, together with a number of tasks. The Master and Slave use I2C serial communication to send messages to each other. The communication with other modules via the docks is handled by the IR tasks. For simplicity, the IR related tasks for only one dock on the Master and Slave are shown. Although the large number of tasks seems to add significant complexity, it actually minimizes the time that the CPU is blocked waiting on a task or resource.

The handling of incoming data through IR and I2C is interrupt-driven. I2C communication is very fast and relatively reliable. The sending and receiving of data is therefore wrapped into single task: in this case the task switching cost is expected to be higher than the cost of not being able to send and receive simultaneously. The motor task implements a PID controller, which is being executed every one millisecond.

The IR communication is much slower and tends to be noisier. For IR communication we have implemented the stop-and-wait ARQ (Automatic Repeat reQuest) protocol. Once the IR interrupt handler receives a complete packet, it passes the packet on to a Receive Task. This task checks for transmission errors. If no errors are found the Receive Task will place the message into the appropriate message queue and ask the Send Task to send an ACK signal (acknowledgment) to the original sender. If there is an error, the Receive Task will ask the Send Task to send a NACK signal. A task on a neighboring module cannot directly send a message to a low-level task on a module, but only to a behavior task. So a message received on any of the docks is routed to a behavior task. If the destination task specified in the header of the message does not run on a receiving module, then the message is simply ignored.

The sensor task is executed by the behavior task and once activated, it reports the status of the on-board sensors to the behavior task directly or through I2C channel. The power management task is responsible for checking each connector current, the status of the battery, charging the battery, and set/resetting the power switches in each docking face.

B. Behavior-Level Software

The high-level behavior code runs only on the Master controller. In figure 10 only one behavior task is shown, but in principal several behavior tasks can run simultaneously. Examples of behavior tasks are power management, locomotion, manipulation, and self-reconfiguration. For control and coordination of multi module robots we have used a distributed approach, called "Digital Hormone Control" [13,14,15,16,17].

B. Remote-Client Software

This software module is the interface between the high-level controller (usually a human) and SuperBot. The Remote-client software is developed in Java and runs on a hand-held PC. High-level commands are sent to SuperBot through the wireless link; see figure 11.



Fig. 11 The SuperBot remote commander.

VI. EXPERIMENTS

We have conducted several experiments to evaluate the performance of SuperBot modules. These experiments include single and dual module gaits, and sensors.

Single module gaits demonstrate the outstanding ability of a single SuperBot module to move around, flip and change its direction. This ability is very critical in connecting detached SuperBot modules to make a connected network.

The dual module experiments have shown that SuperBot modules can synchronize their activities using communication and perform caterpillar-like, creep, drunken or S moving gaits and also being able to change direction in each case. The speed of creep gait has been 12.5cm/s. A dual module Caterpillar-like SuperBot has been able to move of small rocks and go through a pipe.

We have used the 3D accelerometer/inclinometer sensor to balance a cup of Dr Pepper. Also, we have used this sensory information to develop single module gaits that are dependent on the orientation of the module on the floor. As more modules are assembled we will use them to perform multi-module gaits, manipulation and self-reconfiguration tasks. For videos of the above-mentioned and other experiments please visit: <http://www.isi.edu/robots/superbot/movies/>.

VII. CONCLUSION AND FUTURE WORK

SuperBot, A deployable, multifunctional self-reconfigurable robotic system was presented. It was discussed how SuperBot can be used in real applications which require flexible, efficient, sturdy, strong, and durable Robots.

In the future, we plan to assemble twenty SuperBot modules and evaluate the performance of larger networks of modules. We plan to add intelligent docking connectors and evaluate self-reconfiguration tasks. We also plan to build module using titanium and develop space qualified electronics.

ACKNOWLEDGMENT

This research is supported in part by NASA's Cooperative Agreement NNA05CS38A, and in part by US Army Research Office under the grants W911NF-04-1-0317 and W911NF-05-1-0134. The authors are also grateful to other members of the Polymorphic Robotics Laboratory for their useful comments on the earlier drafts of this paper.

REFERENCES

- [1] Yim, M., Locomotion with a unit-modular reconfigurable robot (Ph.D. Thesis), in Department of Mechanical Engineering, 1994, Stanford University.
- [2] Nilsson, M., Essential Properties of Connectors for Self-reconfiguring Modular Robots, . 2001, Autonomous Functional Lab: Kista, Sweden.
- [3] T. Fukuda, Y. Kawauchi, and F. Hara, "A Study on a Dynamically Reconfigurable Robotic System "CEBOT"," Japan Soc. Mech. Eng. (JSME) Int'l Journal , Series C, Vol. 37, No.1, pp.162-171 1994
- [4] Castano, A., W.-M. Shen, P. Will, CONRO: Towards Miniature Self-Sufficient Metamorphic Robots. Autonomous Robots, 2000.
- [5] E. Yoshida, et al.: "A Self-Reconfigurable Modular Robot: Reconfiguration Planning and Experiments", International Journal of Robotics Research, Vol. 21, No. 10, pp.903-916, 2003
- [6] Murata, S. et al M-TRAN: self-reconfigurable modular robotic system, Mechatronics, IEEE/ASME Transactions on Volume: 7, Issue: 4 On page(s): 431 - 441, 2002
- [7] Yim, D. G. Duff, and K. D. Roufas, "PolyBot: a modular reconfigurable robot," in International Conference on Robotics and Automation, (San Francisco, California, USA), IEEE, Apr. 2000. In press.
- [8] Lee, W.H., Sanderson A. C. Dynamics and Distributed Control for Tetrobot Robots. in ICRA. 1999. Detroit.
- [9] Murata, S., H. Kurokawa, S. Kokaji. Self-Assembling Machine. in Proc. IEEE Robotics and Automation. 1994.
- [10] Murata, S., H. Kurokawa, E. Toshida, K. Tomita, and S. Kokaji. A 3-D self-reconfigurable structure. in ICRA. 1998.
- [11] Rus D., Marsette Vona: Crystalline Robots: Self-Reconfiguration with Compressible Unit Modules. Auton. Robots 10(1): 107-124 (2001)
- [12] Kotay, K., D. Rus, M. Vona, and C. McGray. The self-reconfiguring robotic molecule. in ICRA. 1998.
- [13] Salemi, B., WM. Shen and P. Will. Hormone Controlled Metamorphic Robots. in ICRA. 2001.
- [14] Salemi B., Peter Will, and Wei-Min Shen. "Distributed Task Negotiation in Modular Robots". IEEE/Robotics Society of Japan, Special Issue on "Modular Robots", 2003.
- [15] Salemi B., Wei-Min Shen. "Distributed Behavior Collaboration for Self-Reconfigurable Robots". International Conference on Robotics and Automation. April - May 2004, New Orleans, LA, USA.
- [16] Salemi B. Peter Will, and Wei-Min Shen. "Autonomous Discovery and Functional Response to Topology Change in Self-Reconfigurable Robots". International Conference on Intelligent Robots and Systems. September - October 2004, Sendai, Japan.
- [17] Shen, W.-M., Salemi, B., and Will, P. 2002. Hormone-Inspired Adaptive Communication and distributed control for CONRO self-reconfigurable robots. *IEEE Transactions on Robotics and Automation*, 18(5).
- [18] [Stoy 2002a] Stoy, K., Shen, WM., Will, P., Using Role-Based Control to Produce Locomotion in Chain-Type Self-Reconfigurable Robots. IEEE/ASME Transactions on Mechatronics, 2002. 7(4): p. 410.
- [19] Bojinov, H., A.Casal,T.Hogg. Multiagent Control of Self-reconfigurable Robots. in ICMAS. 2000. Boston,MA,USA.
- [20] http://www.atmel.com/dyn/products/product_card.asp?part_id=2018
- [21] L. Barello, "AvrX real time kernel real time kernel." [Online]. Available: <http://www.barello.net/avrX/>