Rolling and Climbing by the Multifunctional SuperBot Reconfigurable Robotic System

Wei-Min Shen, Harris C.H. Chiu, Mike Rubenstein, Behnam Salemi

Polymorphic Robotics Lab, Information Sciences Institute
University of Southern California, Marina del Rey, CA 90292
310-448-8710; shen@isi.edu

Abstract. SuperBot is a modular, multifunctional and reconfigurable robotic system built for NASA applications. This paper reports the hardware and software design of the 20 SuperBot modules and experimental results for multifunctional behaviors, which include a set of long-distance running of 1km/hour with batteries, and a climbing on a large sand dune.

Keywords: Modular robots, multifunction, reconfigurable robots.
PACS: PACS numbers 80; choose from this list: http://www.aip.org/pacs/index.html.

INTRODUCTION

Although self-reconfigurable robots are conceptually versatile, fault-tolerant, and efficient for space application, building, controlling, and deploying such robots in the real world is a very challenging problem. The SuperBot project (Salemi, Moll, and Shen, 2006) is a bold attempt at this task. Supported by NASA, the Polymorphic Robotics Lab at the University of Southern California has designed and built 20 deployable modules and conducted many experiments to demonstrate the diversity and multifunction of these modules both indoors and outdoors. These experiments include crawling, slithering, sidewinding, rolling, walking, moving in sand, rolling 1km on carpet, climbing a steep river bank and a sand dune, climbing ropes between buildings, and climbing vertical rope from the ground to the sixth floor. Although there are only 20 modules in total, the reconfiguration of these modules and the ease of programming them have demonstrated so many different behaviors in a short time. The demonstrations have created sufficient evidence that reconfigurable robotic system can indeed provide multifunction that would require many different special-purpose robots to accomplish.

Transition from a controlled environment to a real-world situation, however, introduces many new challenges to the field of self-reconfigurable robotics (Yim, 2007). 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. This paper is organized as follows: Section 2 describes the mechanical design of SuperBot modules. Section 3 and 4 describe SuperBot's hardware and software architecture, respectively. Section 5 describes the 1km/hour rolling experiments with SuperBot modules, and Section 6 describes the climbing experiments on a large sand dune. Section 7 concludes the paper with discussions and future research directions.

We are very grateful that the SuperBot project is sponsored by NASA Cooperative Agreement NNA05CS38A and we thank all members in USC/ISI Polymorphic Robotics Lab for their support and discussions.

MECHANICAL DESIGN

SuperBot is intended to operate in a harsh and rough environment, 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.

To meet these needs, the overall body of a SuperBot module is in the form of two linked cubes with 3DOF as shown in Figure 1. 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 aluminum 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. 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.

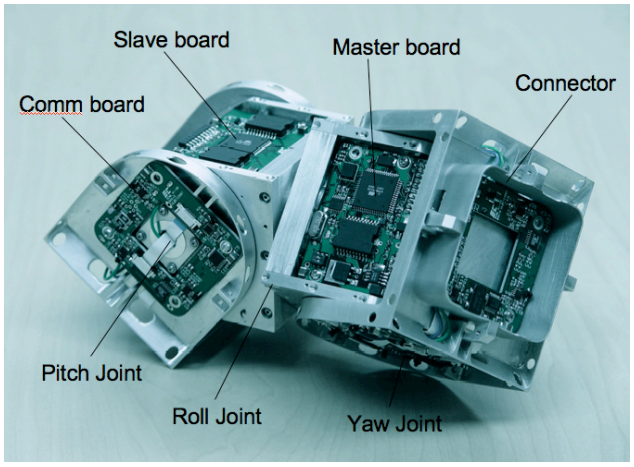

FIGURE 1. SuperBot Module Design and Three Degrees of Freedom.

There are six (currently manual) connectors on each Superbot module; one on each side of the end effectors. Any of the six connectors of a module can connect to any connectors of another module in all 90° interval orientations. It is through connectors that SuperBot modules are reconfigured into different shapes in the experiments. Figure 2 shows the 20 modules configured into different configurations.

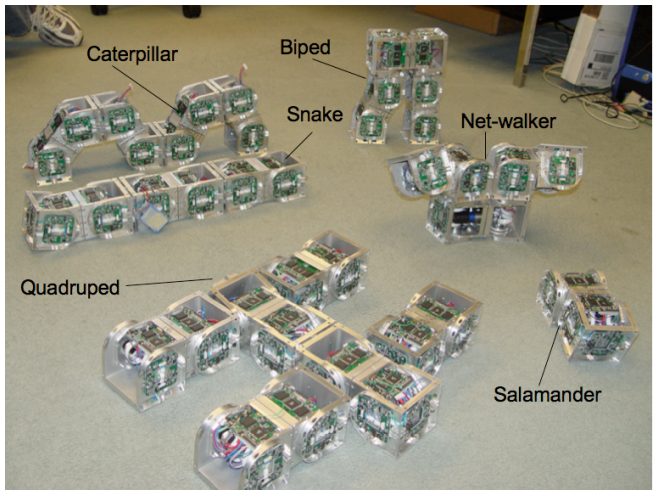

FIGURE 2. SuperBot Modules Configured into Different Shapes (Quadruped, Caterpillar, Snake, Walker, etc).

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. The DC motor outputs between 5.0 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. These result in a maximum of 6.38 Nm torque so that each module can lift three neighboring modules. This has been confirmed by experiments.

HARDWARE ARCHITECTURE

SuperBot has 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 behaviors.

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 3. 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 behavior controller in each module.

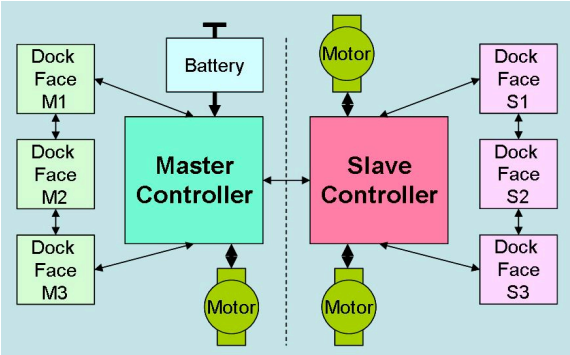

FIGURE 3. SuperBot Hardware Control Architecture.

Each controller is based on a 16 MHZ ATmega128 microcontroller, 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.

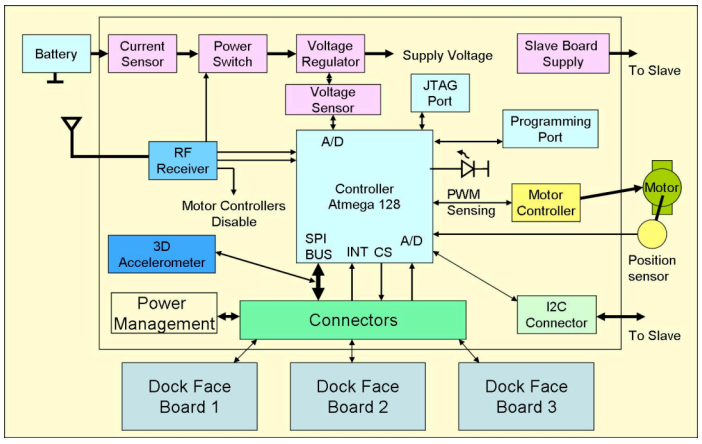

FIGURE 4. SuperBot Master Controller Architecture.

Figure 4 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 Atemga128. A 1.0 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 5 shows the details of the slave controller, which has a similar architecture.

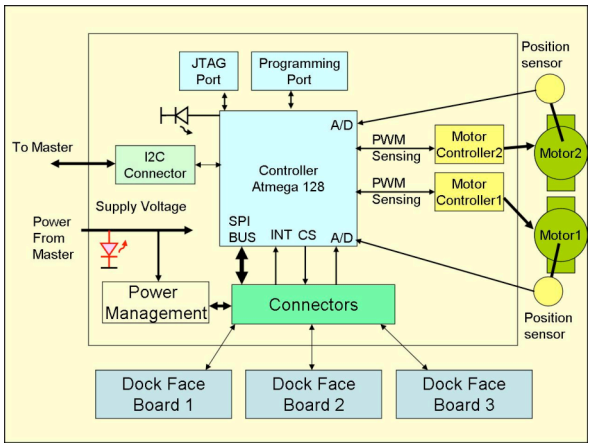

FIGURE 5. SuperBot Slave Controller Architecture.

Figure 6 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.

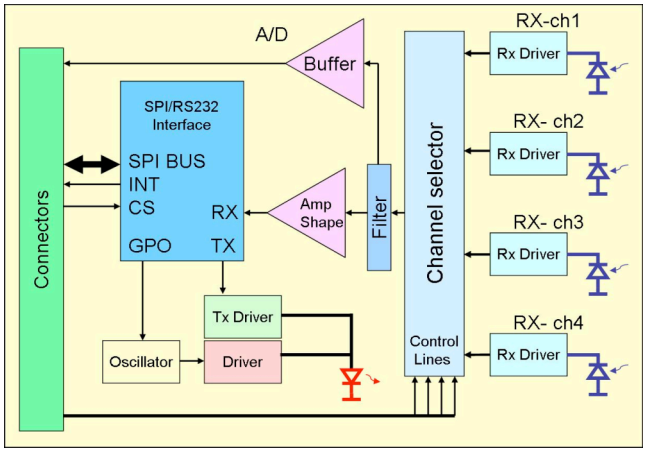

FIGURE 6. SuperBot Communication Interface on a Dock Face.

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. 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.

SOFTWARE ARCHITECTURE

Since SuperBot modules can be dynamically reconfigured into different configurations/functionalities and must support plug-and play with other types of devices, distributed control software is necessary. The control software needed to be real-time, fault tolerant and scalable. In addition, it must accept and execute high-level commands for locomotion, manipulation and self-reconfiguration from a remote host without requiring detailed instructions for individual modules.

Built upon our previous work on hormone inspired distributed control (Shen, 2002), the software on SuperBot modules is distributed and ID-free. It can negotiate with other modules tp select the best actions, adapt to topological changes, synchronize with local clocks, and scale up for different configurations regardless of the shape and size. This software architecture consists of three main parts: low-level drivers, behavior drivers, and remote control.

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 (Barello, 2000). All system-level and user-level code is written in C language as separate tasks. Associated with each task is a message queue (Chiu, 2006). 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 7 shows a simplified diagram of the tasks running on a SuperBot module.

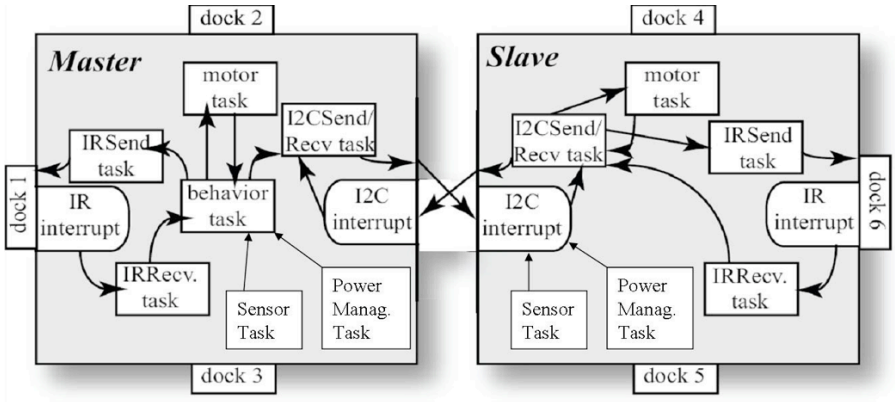

FIGURE 7. Software Tasks on a SuperBot Module.

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.

The behavior-level code runs only on the Master controller. In figure 7 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.

The remote-client 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 remote computer. High-level commands can be sent to SuperBot through the wireless link. SuperBot modules can also use its radio links to accept remote commands.

ROLLING EXPERIMENTS AND RESULTS

One of the required multifunctional behaviors of Superbot is to show the endurance for long-distance travel by running 1km in a single charge of battery. We selected the rolling track for this task for its energy efficiency and speed.

Shown in Figure 8, six SuperBot modules are connected in a closed chain for the rolling track. It rolls forward by changing its shape as shown in Figure 9. The chain was initially a circular shape. To move forward, the robot changes to an oval shape like an American football. With the center of gravity moved forward, the robot rolls. In the meantime when the rolling track has traveled the length of one module, it changes the shape again to keep the center of gravity in the front to keep the rolling track imbalance so that it can continue to roll forward. Experiment has been shown in both simulation (Shen, 2006) and hardware (Sastra, 2006) that the rolling track gait can travel efficiently with a max speed of more than 1meter/second.

(a) Full-circle configuration. **(b)** Squeezed-circle configuration. **(c)** Rolling along a corridor.

FIGURE 8. A 6-Module SuperBot Rolling Track and Changes of Shape.

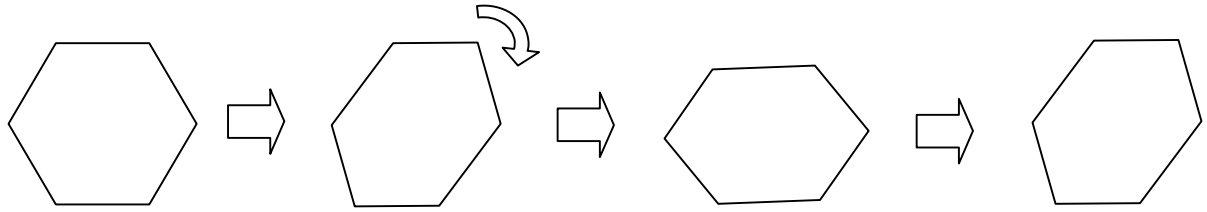

FIGURE 9. The Shape Changing Sequence for Rolling Track.

The control of shape changing is implemented in a distributed fashion with sensor feedback. All modules are running the same algorithm shown in Figure 10. Each module first gets sensor values from its accelerometers. The values are divided into 6 different ranges representing 6 different orientations of module with the rolling track in a position shown in the 1st step in Figure 9. Sensors values obtained are then checked to see what orientation the robot is and determine the state it is in. The motors of a module act accordingly to the state the module is in. The gait is synchronized by the proper arrangement of accelerometer ranges and thus the modules do not need to communicate to their neighbours for synchronized actions.

```
Loop {  1. Get accelerometers reading;

        2. Check sensor value if it falls in one of the six ranges and determine the state;

        3. Run the motors to the angle according to the state detected. Motor angle assigned are for the rolling
track changing to a football shape;

}
```

FIGURE 10. Distributed Control Algorithm for Rolling Track.

Six SuperBot modules are connected in a closed chain and each module has a fully charged battery running the algorithm described. The robot is commanded to roll and go around the building of Information Sciences Institute in laps. Each lap is 107m, so it takes the robot 9.34 laps to achieve the 1-kilometer run. The turning at each corner is done manually.

Two runs have been carried out in the office environment. In both runs, the average time elapsed for each lap is about 4-7 minutes. In the first run, the robot run 9 laps plus 7 meters (about 970 m) in 45 minutes. A power shutdown is observed in a module at the first lap. After resetting the module with the same battery, the experiment continues. No other problems have been observed for the rest of the run. Table 1 shows the average battery voltage in individual modules after the rolling track completely exhausted its batteries in one of its modules. Batteries in the other five modules are still in good condition at the end of the run.

In the second run, the rolling track has traveled 1142.5 meters in 54 minutes before some batteries are exhausted. In the first 6 laps, the rolling track experienced 5-6 times power shutdown but the experiment continues after resetting the module with the same battery. The exact cause of power shutdown is not completely determined but we suspect that they are due to some high current serge caused by the mechanical motion conflicts between modules in the closed kinematic chain.

From the result, the rolling track is capable of running a 1km within an hour in a single charge of batteries. The performance can be further improved with higher stability and robustness of the gaits with better software parameters. Video clips of the experiments can be found at the following website: http://www.isi.edu/robots/superbot/movies/.

TABLE 1. Average Remaining Battery Voltage (V) after the Running Experiments. A Charged Voltage is 8.2V.

Module Number	1st Run (970 m)	2nd Rub (1142 m)
Module 1	3.63	3.48
Module 2	7.41	5.19
Module 3	7.45	3.63
Module 4	7.43	7.23
Module 5	7.43	6.70
Module 6	7.44	7.63

CLIMBING A LARGE SAND DUNE

Another multifunctional task that SuperBot is designed to perform is to climb steep slopes. With a legged configuration of 4 modules, SuperBot is capable of climbing slopes up to 45 degree on hard surfaces. Such experiments have been conducted on carpet boards in office environment and on riverbanks with concrete surface. However, climbing sand dune is very challenging for legged robots because the sand on a steep slope so loose that it is difficult to firmly anchor the legs without sliding down. The following experiment is designed to test SuperBot's capability to perform such a task.

The sand dune we chose for SuperBot to climb is located at the Sand Dune Park in Manhattan Beach, California (33°53'55.75"N 118°24'45.11"W). The dune is made up of dry, fine sand, with an average slope of 20 to 25 degrees (measured from horizontal). The surface was bumpy with many footprints. A straight path was marked on the dune, directly up the dune, 90 meters long. To overcome the anchoring problem on loose sand, we select a climbing rolling track configuration shown in Figure 11. This rolling track is again made of 6 SuperBot modules, connected in a loop configuration. Each module had its roll and pitch axis aligned and parallel to each other. The rolling track was collapsed to keep its Centre of Mass as low as possible. There are 2 shapes for this rolling track, a "U" shape and a Square shape. Using communication between modules, the modules will synchronize their actions and move between these shapes in synchrony. For every cycle of these two shapes, the rolling track will move a half a module forward (1 cube). Each module will have a "state", which represents where they are in the rolling track, and what configuration the track is in (U or Square shape). Using its state, a module will know what angle its motors should be at, based on a pre-determined table. One module is chosen arbitrarily and loaded with the "leader" program, and the other 5 modules are loaded with the "follower" program. The leader program will start at state zero, and increment its state every second by 1, modulus 24. It will then send current, (state+4)mod24, to the module connected to its front dock. The module on the front dock of the leader will update its state to be the value it received, then continue passing the message forward, by sending (state+4)mod24 to the module connected on its front dock. The incrementing message is passed in this fashion down the entire rolling track. This messaging scheme will synchronize all 6 modules, and each module will be at a state four ahead (mod24) of its front neighbour. After 24 state changes, the modules will have returned to their original position in the rolling track, and the track will have moved 1 meter forward. One full rotation of the rolling track is accomplished in about 24 seconds.

(a) The bending-up shape. **(b)** The normal bending shape.

FIGURE 11. The Two Shapes of the Climbing Rolling Track.

The sand-climbing rolling track was covered in a plastic bag to seal out the sand, and then covered in black nylon to protect and hold the plastic. Six aluminium cleats were secured to the outside of the rolling track, at evenly spaced intervals (see Figure 12). The cleats served two purposes, first to widen the footprint of the rolling track to prevent it from tipping over, second to dig into the sand and adding traction. Without this traction, the rolling track would just roll in the loose sand without reliably making forward movement.

(a) The entire sand dune course. **(b)** SuperBot during climbing. **(c)** SuperBot with cleats and sand protection.

FIGURE 12. The Overview of the Sand Dune and the Climbing Rolling Track with Sand Protection.

The robot climbed the entire sand dune without any stopping up the full length of the course in approximately 42 minutes. Figure 12 shows the overview of the sand dune and some snapshots of the climbing process. The only human intervention was to occasionally steer the robot, because the robot was not capable of turning in this configuration autonomously. The video clip of this experiment can be found at the website: http://www.isi.robots/superbot/movies.html.

CONCLUSION

This paper described the SuperBot robotic system designed and constructed for deployable, multifunctional self-reconfigurable robotic missions. We discussed the detailed architecture of the SuperBot modules and experimental results for running 1km/hour and climbing a large sand dune. Future research directions include fine-tuning the software parameters for these and other multifunctional behaviors, designing and performing more functions, and completing the self-reconfigurable connectors. In addition, we will also complete power-sharing mechanism among modules so that SuperBot can be more robust in terms of power endurance.

ACKNOWLEDGMENTS

We are very grateful that the SuperBot project is sponsored by NASA Cooperative Agreement NNA05CS38A.

REFERENCES

Barello, L., "AvrX Real Time Kernel," Available at http://www.barello.net/avrx/, 2000, (access date 2005).

Chiu, H.C., and Shen, W.-M., "Concurrent and Real-Time Task Management for Self-Reconfigurable Robots," In Proceedings of *the Third Intl. Conf. on Autonomous Robots and Agents*, Edited by S.C. Mukhopadhyay and G. Sen Gupta, Massey University, Palmerston North, New Zealand, 2006.

Salemi, B., Moll, M. and Shen, W.-M. "SUPERBOT: A Deployable, Multi-Functional, and Modular Self-Reconfigurable Robotic System," In Proc. *2006 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Edited by Yunhui Liu, IEEE Robotics and Automation Society Publisher, Beijing, China, October 2006.

Sastra, J., Chitta, S. and Yim, M., "Dynamic Rolling for a Modular Loop Robot," in Proceedings of *International Symposium on Experimental Robotics*, Edited by O. Khatib, V. Kumar, D. Rus, Springer-Verlag Berlin Heidelberg, Rio de Janeiro, Brazil, 2006.

Shen, W.-M., Salemi, B., and Will, P., "Hormone-Inspired Adaptive Communication and Distributed Control for CONRO Self-Reconfigurable Robots," *IEEE Transactions on Robotics and Automation*, 18(5), 2002.

Shen, W.-M., Krivokon, M., Chiu, C.H., Everist, J., Rubenstein, M., and Venkatesh, J., "Multimode Locomotion for Reconfigurable Robots," *Autonomous Robots*, 20(2), 2006, pp 165–177.

Yim, M., Shen, W.-M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., and Chirikjian, G.S., "Modular Self-Reconfigurable Robot Systems -- Challenges and Opportunities for the Future," *IEEE Robotics and Automation Magazine*, March 2007, pp 43–53.