

EvBots – The Design and Construction Of A Mobile Robot Colony for Conducting Evolutionary Robotic Experiments

John Galeotti, Stacey Rhody, Andrew Nelson, Edward Grant, and Gordon Lee*

Center for Robotics and Intelligent Machines
Department of Electrical and Computer
Engineering
North Carolina State University,
Raleigh, NC 27695-7911

*Department of Electrical and Computer
Engineering
San Diego State University
5500 Campanile Drive
San Diego, CA 92182

Abstract

This paper describes a method for building small, inexpensive, autonomous mobile robot systems in order to study into robot colonies. We describe how to build EvBot mobile robot colonies that can be used to navigate through mazes of varying complexity, display intelligent control, produce evolutionary computing algorithms, and use low-bandwidth distributed RF networks to communicate. Experimental results show that EvBots can navigate through a real-world maze using an evolutionary algorithm based on complex fuzzy neural paradigms. These algorithms were first developed in an equivalent simulated maze and transferred to the EvBot operating in the real world. On-going experiments are being conducted using shared learning between individual EvBots in a colony, as well as shared learning in complex robot colony task cooperation.

Keywords: Miniature cooperating mobile robot colonies, fuzzy neural evolutionary computing, control algorithm transfer, on-board desktop PC, low-bandwidth distributed RF networks, sensor-based autonomous self-navigation.

Introduction

The goal of this research is to produce a large colony of small, inexpensive, robust and intelligent machines that are capable of carrying out complex navigation tasks, interacting collectively in remote conditions, and also interacting with humans. Our research into robot colonies, evolutionary computing, smart sensor networks, remote applications, and human-robot interaction can only be achieved by: 1) having the availability of a large number of small, powerful, mobile robots and 2) working in both the simulated and real worlds. The robots, called

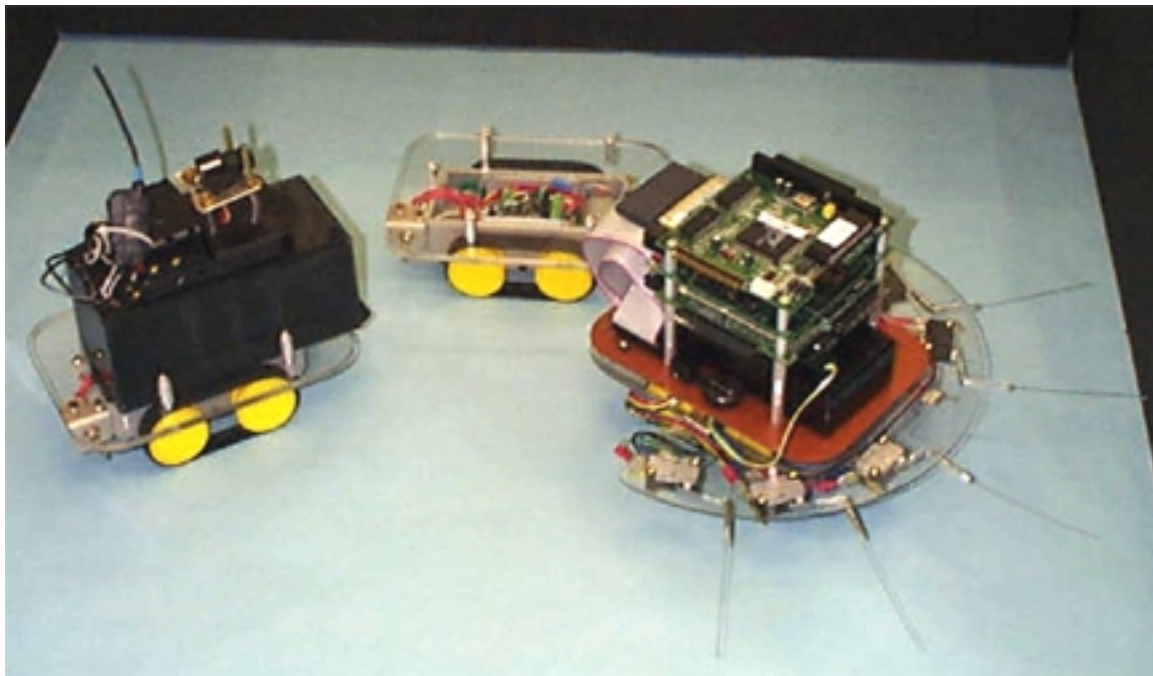


Figure 1 Left: The Prototype EvBot, Center: A Tyco Base Being Retro-Fitted, Right: One Of the EvBot Robot Colony

EvBots, are built from “off-the shelf” components; they are a “plug and play” robot colony that is used for task planning and control. Further, an EvBot can be used individually, e.g., in a security or search and rescue application.

A review of the literature shows that evolutionary robotics is an active research area [5]. Early research was limited to simple sensor/motor control. Currently, more hierarchically ordered architectures are being suggested and there is discussion on whether knowledge-based algorithms are more efficient than evolutionary algorithms [5]. There is also discussion with regard to the scalability issues [2]. Worldwide, many groups have developed miniature mobile robot platforms. Some have been designed and built from discrete components [1][2][4]; others [6] have been adapted from other systems. Whereas the adapted systems leave little choice for microcontroller selection [6], individually designed and built systems can choose a PIC microcomputer [2] or on-board PC [4]. There are advantages and disadvantages associated with either choice. An overview is presented in [7] where wireless local area network technology is used to control the mobile robots. In [6], the authors suggested the use of Khepera, a highly developed mobile robot simulator. Their work dealt with program transfer between the simulated world and the real robots as well as the application of Khepera in the Webots environment [6]. Lastly, NASA continues to conduct research in robot colonies for remote environments [3].

The EvBot Design Specifications

After reviewing the literature on mobile robot designs, a prototype mobile robot was constructed. By retro-fitting a tracked Tyco toy base we have designed and built a simple mobile robot (see left and center of Figure 1). The initial EvBot design used a remote desktop computer to run navigation control software and RF modules to communicate between the computer and the robot. The platform consists of a proto board containing a digital RF receiver connected to a Basic-X controller and a small CMOS camera connected to an analog video RF transmitter. The desktop computer runs Matlab with image capture software. It has a video capture card connected to an analog RF video receiver and a digital RF transmitter connected to a serial port.

The Basic-X controller receives commands from the desktop over its RF link and gives control signals to the two small DC motors in the platform. The camera views the area in front of the EvBot. The image data is sent over the analog RF link to the desktop for processing. Upon receipt of the image data, the desktop computer runs Matlab edge detection algorithms. The algorithms, using differences in neighboring pixel intensity from the captured image, computes whether or not the area in front of the platform is clear for it to travel through. If the area is not clear, a command to turn the EvBot platform is issued and the process is repeated until an unobstructed path is determined and movement initiated. Matlab issues commands by calling a simple batch file that echoes commands to the serial port connected to the digital RF transmitter. It then waits for the robot to respond, reads a new image, and repeats the process. Once the Matlab software was fully developed and the system was tested and proved to function successfully, the design of the next generation of EvBot(s) began.

Testing showed that the use of autonomous visual navigation with this robot suffered from: 1) the choice of the on-board microcomputer system, and 2) control decisions being handled by a remote desktop PC. Although EvBot control is achieved, the time delay between sensing to action is considerable; this is a function of the buffer size of the microcontroller.

A new design specification was needed, one where a desktop PC capability would be on-board the mobile platform and not remote from it. Our eventual choice is a PC/104.

The EvBot Systems Specifications

Although retro-fitting the original prototype met our primary design objectives, the secondary objective was less well defined. The secondary objective was the creation of a robust standard interface for a cooperating EvBot colony. The design specification includes: 1) an onboard microcontroller/CPU, 2) a short-range RF communication module, and 3) a link between the human/computer controller and all available sensor networks. Early in the design process, we determined what items should be on-board the EvBot platform and what items should be remote from it.

The final system design incorporates and manages several modular “plug and play” devices. These “plug and play” devices include: DC motors, remote sensors, on-board sensors, e.g., a CCD Camera, microphone, and range finder. The on-board controller must be able to support all these sensors, either directly or through future expansion. The on-board controller/CPU must handle navigation and routing for resource allocation, e.g., the communication module and onboard sensors. The RF module handles command and control from the human controller or from a remote computer controller. The EvBot systems design is unique; we found no similar designs for comparison.

Design Alternatives

Due to the complex nature of our system design, we had to establish a baseline performance metric. By extending and modifying existing technologies and optimizing our design, we met our performance targets. Producing a design with the potential for future expansion was also part of the design process. We eventually established the following baseline design metrics for our EvBot systems:

- CPU/Controller – handle high level code and operating system
 - C++
 - Linux or a Real Time Operating System (RTOS)
 - Desktop development software such as Matlab
- Interfaces
 - Multiple I/O
 - Serial
 - Parallel
 - USB
- Size – as compact as possible, expected target size is less than 5x7 inches
- Expandable – core components should not have to be re-engineered to meet new applications
- Wireless communication

A ground-up approach to the above system design offered us the best integration and support to achieve our performance targets. Our research showed that if we were to achieve the specified optimum performance, we should use a commercially available integrated CPU board.

The first design was based on the Handspring Visor PDA. This device is small, requires low power, and has a respectable processor and up to 8 MB of memory. It also has a Springboard module slot for easy expandability, e.g., sensor interfaces. Unfortunately, Matlab and most other commercial software tools are not supported. Overall, this design alternative would prove to be both limited in application and too time consuming to meet our criteria.

Lastly, we considered building an EvBot system from existing PC-compatible technologies. Technology searches produced several products that have the necessary system components already integrated, such as products developed on the PC/104 standard. This standard “offers full architecture, hardware, and software compatibility with the PC bus, but in ultra-compact (3.6" x 3.8") stackable modules. PC/104 is therefore ideally suited to the unique requirements of embedded control applications - including full standardization” (www.pc104.org/technology). All of the baseline criteria were available and supported using the PC/104 standard.

EvBot Final PC/104 Systems Design Specifications

Based upon our design concept, we chose to use the PC/104 architecture. Our baseline systems design criterion for the CPU board and required interfaces is met by the MZ104 MachZ. This is an embedded PC-on-a-Chip. Specifics directly applicable to our system design include:

- 32 bit CPU core operating at 133 MHz
- SDRAM support
- Integrated I/O to include serial and parallel ports
- Independent USB interface
- Embedded PC BIOS in Fail-safe Boot ROM
- X86 software compatible
- Compact size (3.6" x 3.8" x 0.9")

The communication criteria is handled by:

- Linksys Wireless Ethernet PC-Card, 11 Mbps capability

All programs and software are stored on or utilized by:

- 8 Mb Disc-On-Chip
- 96 Mb ATA Flash Memory PC-Card

- 64 Mb Memory Module (RAM)

The interfacing of the CPU with the hardware adapters is met with:

- PC/104 PCMCIA PC-Card Adapter, handles both the Wireless Ethernet and Memory Cards

The video camera uses a:

- USB RoboCAM, up to 640x480 live motion resolution

The software required to operate and control the system includes:

- Custom Linux Distribution derived from RedHat Linux 7.1
- Trimmed down version of Matlab 5.3 with Image Processing Toolbox.

Other components needed to program and utilize the system include:

- Linksys Wireless Network Access Point, can handle up to 21 devices
- PCM-3521 PC104 Video Card
- MZ104 Cable Set
- Generic computer monitor, keyboard and mouse
- 5 VDC power supply

EvBot Final System Design Description

The same tracked Tyco platform was used for both the prototype and the final design (see Figure 1, right). The prototype control software and interfaces were used. The biggest design change was transferring the remote PC/104 control technology onto the EvBot platform.

The re-designed EvBot system has two main hardware components: the PC/104 stack and the custom PCB (see Figure 2). The PC/104 stack acts as the brain of the system, running Matlab, issuing commands, and interfacing to the camera and wireless network. The PCB receives simple commands from the PC/104 stack and controls the locomotion mechanism accordingly.

The PC/104 Stack consists of two connected PC/104 boards: the MZ104 and the PCMCIA adapter. The MZ104 is a complete computer (minus video display), including an 8 MB bootable Disk-On-Chip flash-chip. A USB port interfaces directly to the USB camera for image acquisition. An RS232 serial ports interfaces to

the Basic-X on the custom PCB to control locomotion. Attached to the MZ104 via the PC/104 bus is the PCMCIA PC-Card adapter that holds two PC-cards. The wireless Ethernet PC-card is used to retrieve data from the self-navigation system for monitoring purposes and for Matlab to acquire a software-license for

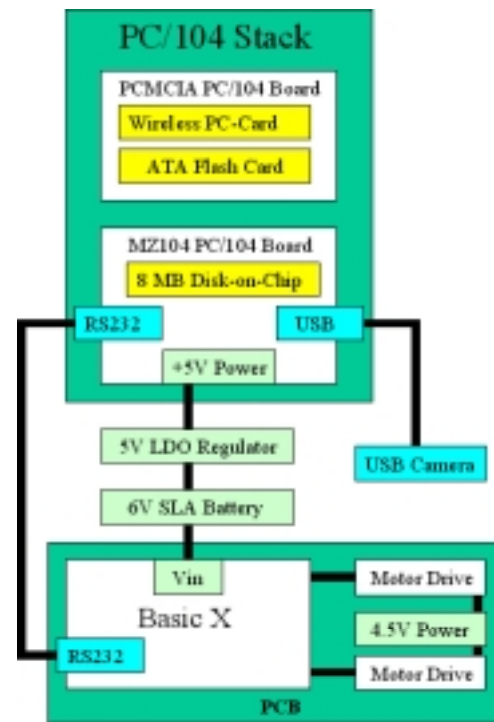


Figure 2 The PC/104 Stack

operation. The 96 MB ATA-flash-card acts as primary program storage for the system, containing a local, trimmed-down installation of Matlab. The EvBot Matlab installation includes the image processing toolbox and an entire custom Linux distribution (except for the kernel, which is booted from the 8 MB Disk-On-Chip). Details of the software system are discussed later.

The custom PCB is specific to the EvBot treaded base. It is generic for any locomotion control mechanism accepting RS232 input. In the EvBot application it is used for locomotion and self-navigation. This is briefly documented here to explain system interfacing and to clarify the overall system functionality. The PCB contains the Basic-X connected to the two motor drivers (dual high-current CMOS-inverters connected to each pole of the motors). The important concept

is that the Basic-X can actuate the motors so as to move the platform forward and backward and to turn the platform in-place when so instructed from the MZ104 via the RS232 link. Details of the communication link are further described later with the software description.

custom 2.4.2 Linux kernel after loading an image of an initial ramdisk for PCMCIA access into memory. The kernel is pointed to the ramdisk image and is then given control of the MZ104's CPU. After following normal bootup, the kernel creates a temporary ramdisk, loads it with the initial ramdisk image, and transfers control to the

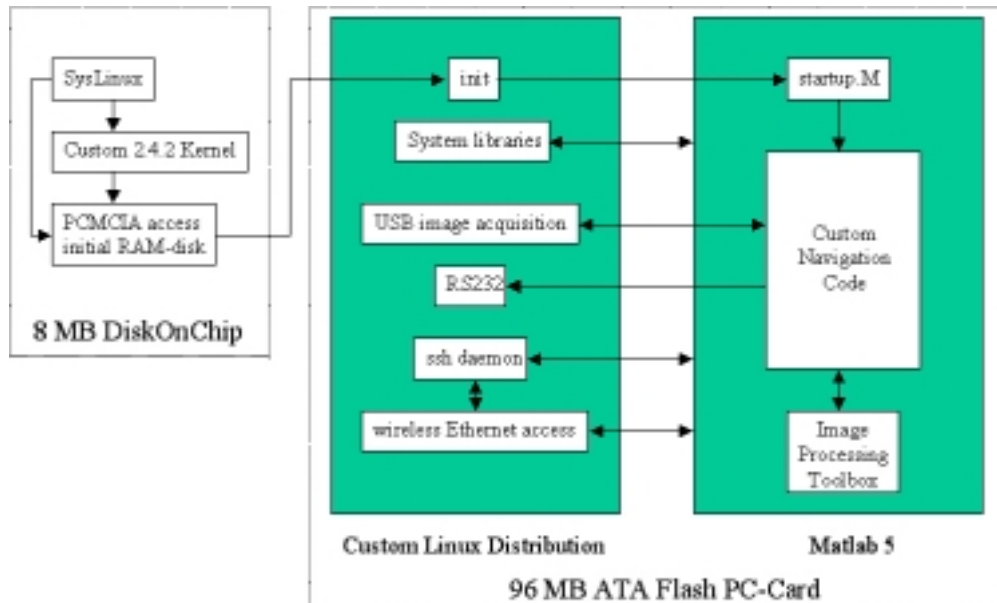


Figure 3 The EvBot Software Architecture

Several hardware components and accessories are necessary for the EvBot system to function. These include (1) the treaded locomotion base with its PC/104-base Plexiglas top and 3 AA battery compartment providing the 4.5 V power supply for the motor drivers (see Figure 1, center) and (2) the 6 V sealed-lead-acid battery mounted on a PC/104-stackable perf-board connected to a low-drop-out 1.5 Amp, 5 Volt voltage regulator that powers the PC/104 stack via a connector on the MZ104.

The software architecture is physically divided into two components by a storage medium; it is logically divisible into several interconnected modules within three primary categories: bootup code, custom Linux distribution, and Matlab 5 (please reference the block diagram in Figure 3).

The bootup code, at a minimum, contains all the software on the 8 MB DiskOnChip boot chip. SysLinux is automatically loaded by the MZ104's BIOS boot-loader according to normal PC bootup procedures. SysLinux then loads a

linuxrc script on that image. Linuxrc then loads the driver modules necessary to access the PCMCIA adapter and its cards, notably the ATA-flash-card containing the bulk of the software system. Once the ATA-flash-card is accessible, it is re-mounted as the root device and control is transferred to its init executable, which executes a standard set of System V scripts (specifically, the RedHat 7.1 Beta scripts with unnecessary and uninstalled features removed) and eventually transfers control to Matlab which in-turn runs its startup.m file to load the custom navigation code that ultimately drives the robot. However, for organization purposes init is logically grouped as part of the custom Linux distribution, which is discussed next.

The contents of the custom Linux distribution were patterned after Linux Care's Bootable Recovery Disk and Tom's Root/Boot Disk; but all the files are RedHat 7.1 compatible (with all binaries stripped to reduce space). Basically, only common system utilities, programs, and

libraries that make the system boot and run Matlab were added. Key components of the distribution include:

- Trimmed-down and customized init scripts
- Only necessary libraries
- USB image acquisition software which uses the script to take pictures to capture an image to /tmp/dump.jpeg
- Wireless Ethernet support (currently, DNS is neither needed nor supported)
- Secure Shell daemon (used as desired to monitor the platform remotely; no special features)

The trimmed-down version of Matlab 5.3 had the image processing toolbox. The startup.m file was written to load and begin executing the visual-navigation Matlab code as soon as Matlab started. When Matlab starts, navigation starts. The only other piece of custom code for Matlab is the com_init shell-script that initializes the RS232 serial port. It is called by Matlab before entering the main loop to control the locomotion platform. Like the custom Linux distribution, the trimmed-down version of Matlab 5.3—complete with navigation code—is also included on the attached CD-ROM. The format for the RS232 commands to the locomotion platform is as follows:

- F: go forward for 1.5 seconds
- B: go backward for 1.5 seconds
- L: turn left for 0.4 seconds
- R: turn right for 0.4 seconds

Conclusion

EvBot robot platforms were designed to be a generic base for mobile robotic systems research. The EvBot design is proving to be more versatile than the distributed robot application it was intended for. When outfitted with sensors and evolutionary algorithms, it can carry out maze navigation tasks. Evolutionary algorithms were developed and tested separately in simulated world maze navigation scenarios that reflect real-world maze navigation tasks. The successful algorithms were then transferred to the real EvBots and tested to determine their performance. This approach is a form of transferable and shared learning, suitable for robot colony research. The production of a cheap and robust EvBot that provides lots of data

has opened up many avenues of research for the CRIM research team.

Acknowledgements

The authors would like to thank the following: NC State student Kent Marshall, Tyco Inc., the College of Engineering at NC State, and Dr. Elana Etheridge, DARPA MTO (Grant No. 97-41) for her encouragement and support.

References

- [1] Agah, A., and Pierik, C. T., “Design and Fabrication of a Team of Robots in Hardware”, In: 22nd IEEE International Conference on Industrial Electronics, Control, and Instrumentation, August 5-10, Taipei, Taiwan, pp 1577-1582, 2000.
- [2] Caprari, G., Arras, K. O., Siegwart, R., “The Autonomous Miniature Robot Alice: from Prototypes to Applications”, In: Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 31-November 5, Takamatsu, Japan, pp 793-798, 2000.
- [3] Engwirda, A., Engwirda, T., Vlacic, L., and Sattar, A., “Cooperating Autonomous Robot Colonies”, In: Proceedings of the IEEE International Conference on Industrial Technology 2000, January 19-22, Goa, India, pp 454-459, 2000.
- [4] Kagami, S., Kabasawa, M., Okada, K., Matsuki, T., Matsumoto, Y., Konno, A., Inaba, M., and Inoue, H., “Design and Development of a legged Robot research Platform JROB-1”, In: Proceeding of the IEEE International Conference on Robotics and Automation”, Leuven, Belgium, May 16-20, pp 146-151, 1998.
- [5] Meyer, J-A., “Evolutionary Approaches to Neural Control in Mobile Robots”, In: Proceedings of the 1998 International Conference on Systems, Man, and Cybernetics, October 11-14, San Diego, CA, pp 2418-2423, 1998.
- [6] Wang, L. F., Tan, K. C., Prahald, V., “Developing Khepera Robot Applications in a Webots Environment”, In: 2000 International Symposium on Human Micromechatronics and Human Science, October 22-25, Nagoya, Japan, pp 71-76, 2000.
- [7] Winfield, A.F.T., and Holland, O.E., “The Application of Wireless Local Area Network Technology to the Control of Mobile Robots”, In: Microprocessors and Microsystems 23 (2000), Pub: Elsevier, pp 597-607, 2000.