REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

AD-A232 007

2. REPORT DATE: January 1991

3. REPORT TYPE AND DATES COVERED: professional paper

A MODULAR ROBOTIC ARCHITECTURE

5. FUNDING NUMBERS: PR: ZE92 WU: DN300029 PE: 0602936N

6. AUTHOR(S)
R. P. Smurlo, R. T. Laird

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Ocean Systems Center
San Diego, CA 92152-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Chief of Naval Research
Independent Exploratory Development Programs (IED)
OCNR-20T
Arlington, VA 22217

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The development of control architectures for mobile systems is typically a task undertaken with each new application. These architectures address different operational needs and tend to be difficult to adapt to more than the problem at hand. The development of a flexible and extendible control system with evolutionary growth potential for use on mobile robots will help alleviate these problems, and, if made widely available, will promote standardization and compatibility among systems throughout the industry. The Modular Robotic Architecture (MRA) is a generic control system that meets the above needs by providing developers with a standard set of software *and* hardware tools that can be used to design modular robots (MODBOTs) with nearly unlimited growth potential. The MODBOT itself is a generic creature that must be customized by the developer for a particular application. The MRA facilitates customization of the MODBOT by providing sensor, actuator, and processing modules that can be configured in almost any manner as demanded by the application. The Mobile Security Robot (MOSER) is an instance of a MODBOT that is being developed using the MRA.

Published in *Mobile Robots* V *SPIE Proceeding, 1388.*

14. SUBJECT TERMS
robotics
processing architectures
mobile robots

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAME AS REPORT

SPIE MOBILE ROBOTS V
Boston, MA (Nov 90)

A Modular Robotic Architecture

Richard P. Smurlo
Robin T. Laird

Naval Ocean Systems Center
Code 535
San Diego, CA 92152-5000
(619)553-3667/3668

ABSTRACT

The development of control architectures for mobile systems is typically a task undertaken with each new application. These architectures address different operational needs and tend to be difficult to adapt to more than the problem at hand. The development of a flexible and extendible control system with evolutionary growth potential for use on mobile robots will help alleviate these problems, and, if made widely available, will promote standardization and compatibility among systems throughout the industry. The Modular Robotic Architecture (MRA) is a generic control system that meets the above needs by providing developers with a standard set of software and hardware tools that can be used to design modular robots (MODBOTs) with nearly unlimited growth potential. The MODBOT itself is a generic creature that must be customized by the developer for a particular application. The MRA facilitates customization of the MODBOT by providing sensor, actuator, and processing modules that can be configured in almost any manner as demanded by the application. The Mobile Security Robot (MOSER) is an instance of a MODBOT that is being developed using the MRA.

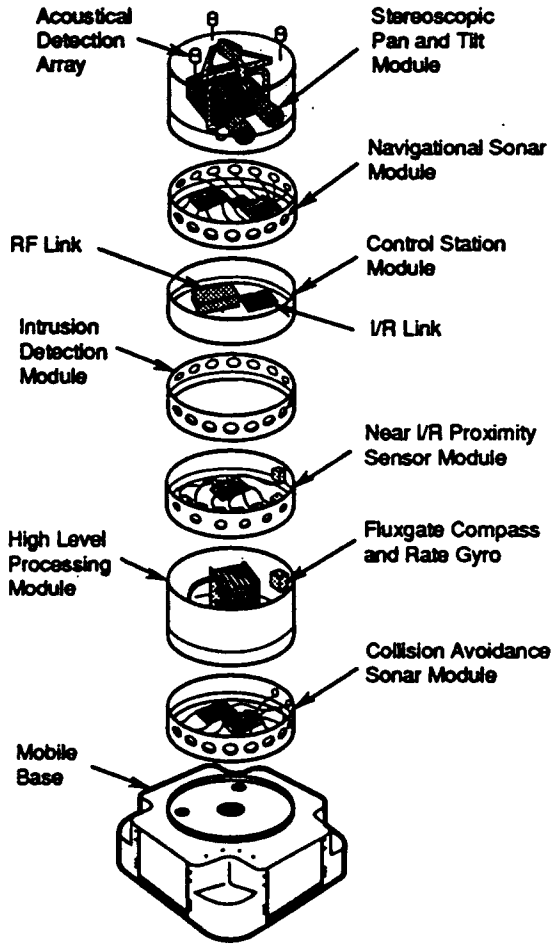

Figure 1. Remote platform module configuration of the Mobile Security Robot (MOSER).

1. INTRODUCTION

Robotics work being done today can be divided into two categories: mobile and stationary systems. The development of a suitable processing and control architecture for mobile systems is a task which historically has been independently undertaken for individual robotics projects, each having a different intended application. Results are often unsuccessful due to insufficient awareness of the issues and alternatives, or have a tendency to rapidly become outdated. Furthermore, the majority of efforts have produced application-specific, non-modular architectures that are difficult to apply to future projects. The development of a flexible, powerful, and widely available "core" high-level control system with expansion capability for use on a wide range of mobile robot applications will greatly alleviate these problems, while fostering an atmosphere of standardization and compatibility among systems throughout the industry.

A standardized, modular control system will also reduce the costs associated with development of customized architectures by providing the framework around which to build those architectures; only the configuration of the pieces would have to be done each time, not the re-design of the entire system. In addition, a standardized architecture will promote software/hardware re-usability in that identical modules and components will be used in several places, reducing both development time and cost.

The *Modular Robotic Architecture (MRA)* is a generic architecture that provides developers with a standard set of hardware and software tools that can be used to design modular robots with a high degree of flexibility and extensibility. The MRA is designed to support the development of modular robots and modular control systems (in the general case); it emphasizes standard hardware (electrical/mechanical) and software interfaces to allow development of different capabilities by various working groups, the products of which can then be easily integrated to create or extend a modular robotic system.

The first MODBOT to be developed under the MRA will be the *Mobile Security Robot (MOSER).* MOSER (figure 1) addresses the need for physical security within the confines of a structure such as an office building or a warehouse. MOSER will be an autonomous, modular, mobile robot responsible for detecting intruders and responding to the assessed threat. MOSER is an attempt at duplicating several of the sensor and processing components found on ROBART II (Everett and Gilbreath, 1989) with several improvements in the area of distributed processing such as system networking, modularity (module design), and dynamic system configuration.

2. THE MODULAR ROBOTIC ARCHITECTURE

The Modular Robotic Architecture (MRA) describes both the hardware and software components that are used to create a *Modular Robot (MODBOT).*

2.1 Overview

Conceptually, the MODBOT is similar to the IBM PC with its expansion slots; adding a module to a MODBOT is like adding a peripheral card to a PC. One simply plugs a card into an available slot, installs the supplied software drivers, and immediately incorporates the new capabilities of the card into his system. Adding modules and capabilities to a MODBOT will be as simple, making the integration of new, smarter, better, faster modules very easy. It is the ability of the MODBOT to accept modules of increasing complexity that provides the MRA with its evolutionary growth po-

tential, and was a primary rationale for the development of the architecture.

Simply stated, a MODBOT is a collection of independent modules of varying intelligence and sophistication connected together by a generalized, distributed network. The MRA does not require a particular physical module configuration nor does it require that all modules be located physically together. The generic MODBOT is illustrated in figure 2a.

Typically, however, and specifically for systems involving direct human supervision, a MODBOT is divided into two physically separate computing systems: the control station (CS) and the remote platform (RP). The control station is a single module that is remote from the rest of the MODBOT. The remote platform consists of several modules and is connected to the control station by a telemetry link that acts as a network bridge. Two possible implementations to this approach are given in figures 2b and 2c. Only in systems that are strictly autonomous would the control station be located with the remote platform (figure 2b).

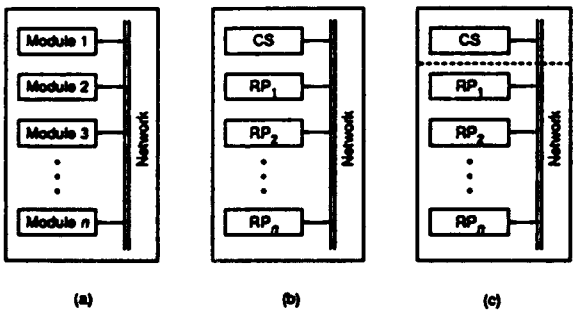

Figure 2. Modular Robot (MODBOT) Block Diagram.

2.2 Hardware

As illustrated in figure 2, the MODBOT is made up of several modules which are connected together by a generalized network. This network distributes power and communications to all modules and is referred to as the *MODBOT Bus*, or *MODBUS*. The standard components of each robot module are the *Power Distribution Node (PDN)*, the *Intelligent Communications Node (ICN)*, and the *Module Processing Unit (MPU)*. A *Platform Power Conditioning Unit (PPCU)* resides in the robot base and is responsible for the conditioning and converting of the base power source to supply the individual modules via the MODBUS. The three module components and their interconnections are shown in figure 3 below.

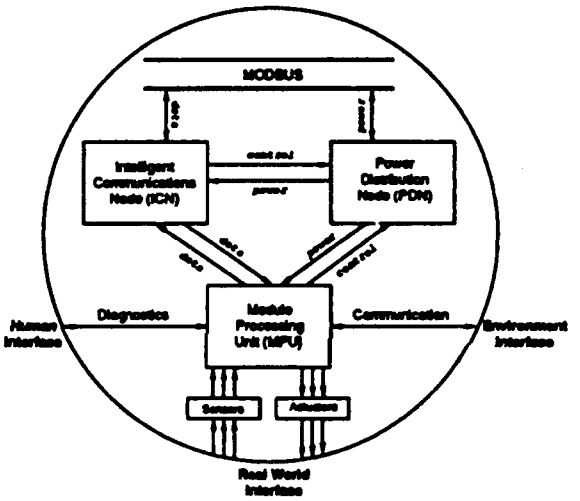

Figure 3. Generic robot module. The communications link to the external environment (Environment Interface) and the sensors and actuators (Real World Interface) are optional.

2.2.1 Power Distribution Node

The Power Distribution Node (PDN) provides local power conditioning and short circuit protection of power inputs from the MODBUS. The various power levels are then routed to the other components on the module via a terminal strip. Standard voltages of +5V/+8V DC, +12V DC, and +24V DC are available. The PDN also has the capability of receiving control inputs from the ICN or the MPU for power down of sensors or actuators while not in use.

Component selection (i.e., circuit breaker size, regulator type, etc.) determines the current carrying capabilities of each PDN and can be modified so that a module may be supplied with an appropriate amount of power. The MRA specifies limits on power consumption for each module so that a power budget for the entire platform is achieved.

2.2.2 Intelligent Communications Node

The primary function of the Intelligent Communications Node (ICN) is inter-module communications. The ICN (figure 4) is designed using CMOS technology to facilitate the low power consumption desired in autonomous systems. The ICN, driven by an Intel 80C152 Universal Communications Controller, typically requires less than 100mA at 12 volts. This 8-bit microcontroller communicates with other ICNs via an internal medium-speed global communications channel. An ICN can transmit to one, several, or all of the 255 other possible nodes. The microcontroller communicates with the MPU through a local communications channel which can utilize either a standard UART or an 8-bit parallel port.

The global communications channel is configured to transfer data using the CSMA/CD protocol at a data rate of 921.6 KBPS (data rates of 1.8 MBPS are achievable). Under the CSMA/CD protocol the microcontroller transmits data through DMA channels whenever data is available and it determines that the line is idle. After the microcontroller has transmitted its information, it waits for an acknowledge from the receiving station (indicating a valid reception). If no acknowledge is received, the transmitting ICN will automatically re-transmit. This feature aids in recovery from data collisions or bus noise.

The local communications channel provides a means for the ICN to communicate with the MPU. The interface is user configurable and can be set up for either parallel or serial data transfer. The parallel protocol can optionally be interrupt driven, and the serial link is standard RS-232 with eight user selectable baud rates between 300 baud and 38.4K baud.

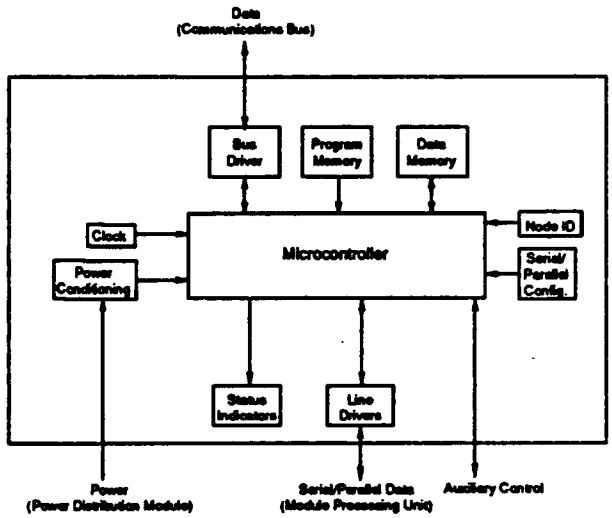

Figure 4. Intelligent Communications Node (ICN).

2.2.3 Module Processing Unit

The Module Processing Unit (MPU) is responsible for carrying out the intended function of the module. The MPU processor can be selected by the user to give the speed and processing power as required by the module's intended application. The only restriction on this processor is that it be capable of communicating with the ICN via one of the local communications channel protocols previously described.

In addition to maintaining transmission with the ICN, the MPU is responsible for processing sensor information and sending commands to actuators. Once sensor information is processed it must be stored and passed on to any other module requesting that information. The application software will also per-

form any high bandwidth servo control loops required by actuators.

The function of an MPU can best be seen by looking in more detail at one of MOSER's modules. The Collision Avoidance Sonar Module of MOSER (figure 1) is a collection of twelve ultrasonic sonar sensors covering the front 180 degree view of the robot. When the robot is commanded to move by another module (the High Level Processing Module, for example) that module will send a command to the collision avoidance module telling it to monitor its sonar sensors and report back if there is an object within two feet of the robot. The collision avoidance module then continually fires its sensors and calculates the distance to objects based on the time-of-flight of the sonar "ping" and the air temperature. The collision avoidance module will then report back to the processing module, via the ICN, only when it finds an object in the path of the robot (within two feet).

2.2.4 Platform Power Conditioning Unit

The Platform Power Conditioning Unit (PPCU) provides a means for supplying relatively clean power to the robot modules via the MODBUS. The PPCU resides in the base of the MODBOT and must be interfaced to the power system available on the application-specific platform. The PPCU first protects and conditions the power supplied by the base with circuit breakers and surge suppressors, and then converts the power to the one 24 volt and two 12 volt supplies required by the modules. The three supplies are isolated from each other to avoid grounding problems. The only requirement the PPCU places upon the platform is that it be capable of providing +24V DC at sufficient amperage.

In addition to its power conditioning function, the PPCU provides a standard interface between the robot platform and the power distribution portion of the MODBUS. The PPCU makes the MODBUS base-independent, and decouples the power system of the base from the rest of the MODBOT.

2.3 Software

The MRA software systems are organized as layers in an $N,N\text{-}k$ architecture where one software subsystem may have views (access) to multiple subsystems below it in the hierarchy (figure 5). MODBOT software application developers are able to use subsystems as desired and are not required to use any particular subsystem. However, applications must provide specific functionality that is established by the MRA interface specifications and is otherwise obtained through the use of these subsystems. A standard software "library" is provided with functions available for inter-module communications, simple task control, and management of local and system module function execution. Software developers need only supply a minimal set of device-specific software "drivers" to implement the entire MRA software architecture, i.e., the software subsystems are portable between hardware implementations with only few modifications necessary.

Each robot module can be thought of as an object that responds to a variety of commands represented by the object's methods and whose internal state is maintained by instance variables. Robot modules are of the class *Module* and possess certain capabilities common to all objects of that class. Modules also inherit the capabilities of their superclass (*Object*). Through classification and other properties of object-oriented programming, robot modules are given (by definition) common functionality.

The software subsystems of the MRA directly support object-oriented design and implementation of distributed, highly modular control systems for use

on mobile (and non-mobile) robots. An object-oriented approach promotes both re-usable software components and modularity at several levels. Additional capabilities can easily be added to a module simply by adding new methods to the object's class.

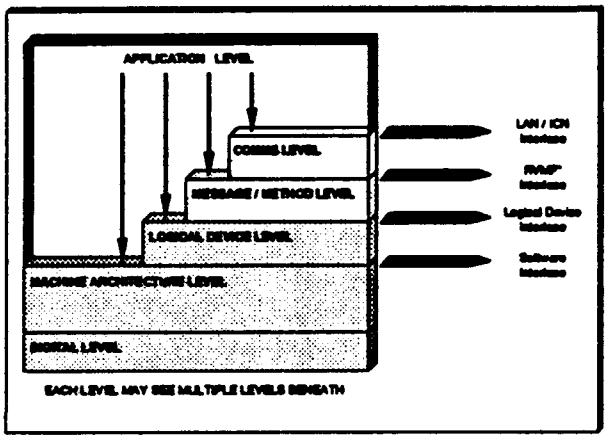

Figure 5. MRA System Image (*N,N-k* architecture).

Below is a functional decomposition of the MRA software systems as implemented on the ICN and the MPU. (The ICN software is a subset of the MPU software with the exception of the Global Communications Subsystem which is unique to the ICN.) Figure 6 is a block diagram of the MRA software subsystems as implemented on the MPU.

2.3.1 ICN Software System

The ICN software system is responsible for managing medium speed communication between the MODBOT Local Area Network (LAN) and the MPU on board each module. The ICN provides a standard interface between the LAN and the MPU, and is the cornerstone of the MRA in that it provides for distributed MODBOT module communication and control.

The ICN software supports a 1.8 MBPS (maximum), CSMA/CD ("peer-to-peer") communications network configured as a bus with deterministic access to a maximum of 255 modules (slots). There is no central or master communications controller. Each node has equal access to the network and is responsible for managing its own resources.

The ICN software system is composed of the following five functional units:

1) Global Comms Device Handler.
2) Global Comms Interface.
3) Local Comms Device Handler.
4) Local Comms Interface.
5) Intelligent Comms Controller.

The Global Communications Subsystem and the Intelligent Communications Controller (not shown in figure 6) are unique to the ICN software system. The Local Communications Subsystem is common to both the ICN and MPU processing systems.

2.3.1.1 Global Communications Subsystem

The Global Communications Subsystem implements a standard set of functions on the target LAN hardware. These functions are used by higher-level software to access the LAN. The subsystem is broken down into the Global Communications Device Handler (responsible for low-level control of the LAN hardware) and the Global Communications Interface (which implements the standard network access functions available to other software subsystems).

The Global Communications Device Handler interacts directly with the ICN hardware to provide external communications between the LAN and the ICN. The functions at this level are device-specific, and must be modified for the target (LAN) hardware. The functions provided by the GSC and the device handler implement the Physical Link Layer and the Data Link Layer of the ISO open

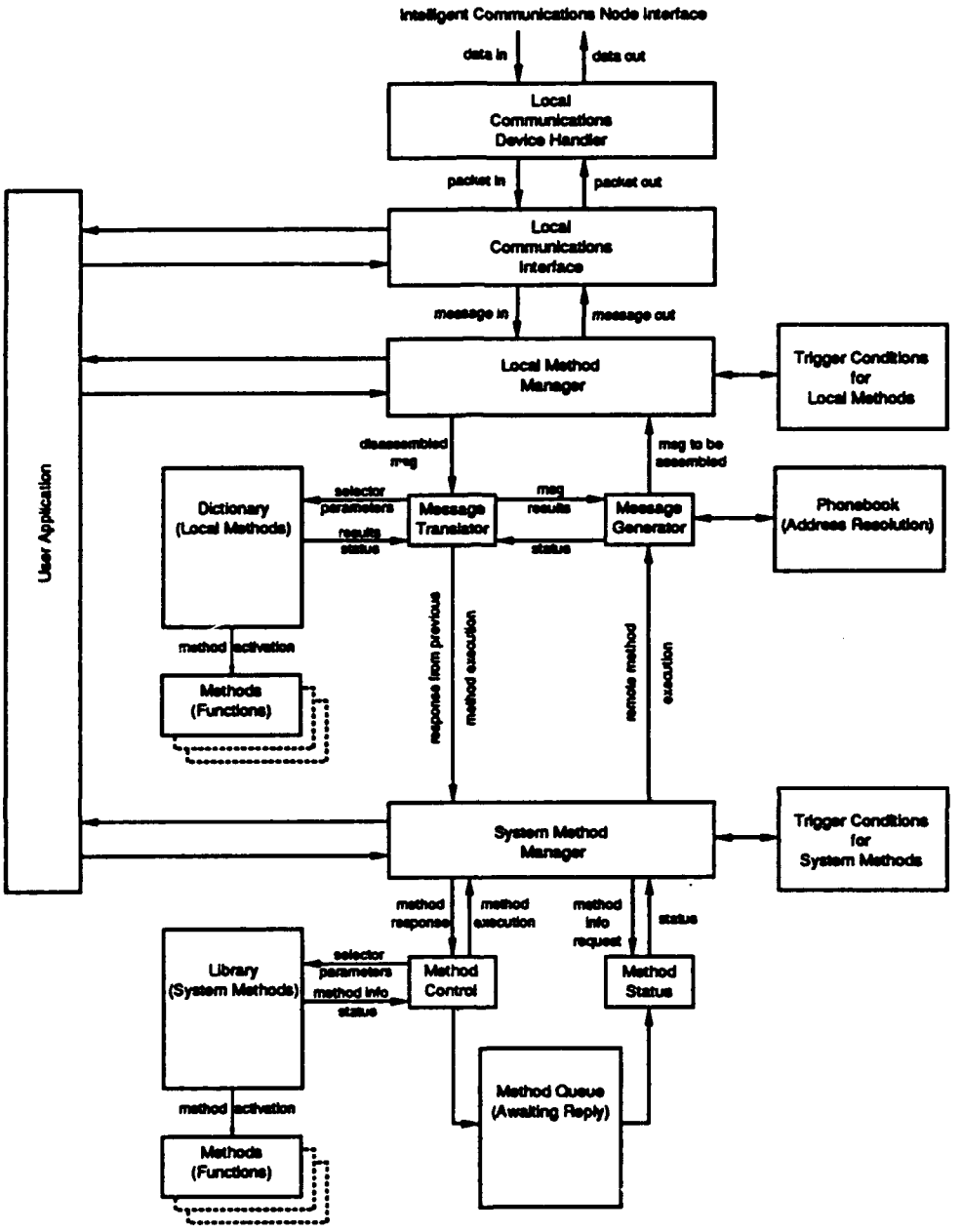

Figure 6. MRA Software Block Diagram (Global Communications and Logical Device Subsystems not shown).

systems communication model (International Standards Organization, 1980).

The device handler logically decouples the standard functions of the Global Communications Interface from the hardware implementation. To use a device other than the GSC as the MODBOT LAN, only the Global Communications Device Handler functions must be rewritten.

The Global Communications Interface is an abstraction of the lower-level functions and represents the user interface to the LAN. Functions at this level are completely hardware independent, and

provide services for initializing and transmitting messages to/from the LAN.

2.3.1.2 Local Communications Subsystem

The Local Communications Subsystem implements a standard set of functions on the target processor communications hardware. These functions are duplicated on both the ICN and MPU, and are used by higher-level software to communicate between the two systems. The subsystem is broken down into the Local Communications Device Handler (responsible for low-level control of the serial or parallel communications hardware) and the Local Communications Interface (which implements the standard communications port access functions available to other software subsystems).

The Local Communications Device Handler is responsible for low-level data exchange between the MPU and the ICN. Functions at this level deal directly with the target hardware and are device-specific, so their implementation will change as the hardware changes. The device handler functions are distributed between both the MPU and ICN, providing the communications link between the two systems.

The device handler logically decouples the higher-level functions provided by the Local Communications Interface from the specific hardware implementation. Changing serial communications controllers, for example, requires only that certain portions of the physical interface be modified in order to maintain consistency and compatibility at higher levels.

The Local Communications Interface is an abstraction of the lower-level functions and represents the user interface to the inter-processor (local) communications port. Functions at this level are completely hardware independent, and provide services for initializing and transmitting packets to/from the serial or parallel port.

2.3.1.3 Intelligent Comms Controller

The "main" program of the ICN is the Intelligent Communications Controller whose MPU counterpart is the User Application. The ICN controller is very simple: it initializes the Local and Global Communications Subsystems, and then coordinates transmission of information (data packets which represent module messages) between the ICN and the MPU. Messages are received from the network and passed along to the MPU. Messages are also received from the MPU and then sent on to the network for distribution as appropriate.

Currently, the communications controller is a simple message buffer between the MPU and the MODBOT LAN. Future plans include adding the message recognition and response capabilities of the Message Manager to the ICN for more sophisticated control. This would make the ICN software nearly identical to that of the MPU.

2.3.2 MPU Software System

The MPU software system is responsible for execution of both local and system methods (functions) as directed by the application module. The MPU provides the User Application with standard interfaces to the message passing, function execution, and logical device control facilities of the MRA.

The MPU software system is divided into two distinct components: the MRA MPU standard software services, and the MPU application program which is the main routine supplied by the developer. A default controller is supplied by the MRA (for simple applications) that replaces the main program.

The MPU software system is composed of the following six functional units:

1) Local Comms Device Handler.
2) Local Comms Interface.
3) Local Method Manager.
4) System Method Manager.
5) Logical Device Interface.
6) Application Controller.

The Message Manager, Logical Device Subsystem, and the Application Controller are unique to the MPU. The Local Communications Subsystem is common to both the MPU and ICN, and was described previously in section 2.3.1.2.

2.3.2.1 Message Manager

The Message Manager is responsible for translating and executing incoming messages, and for generating messages in response to external and internal requests. The Message Manager is also responsible for external message address resolution which relies on the ability to automatically determine the status of a module on the MODBOT network.

Functions that describe the behavior of a module are called *local methods* and are referred to as "internal". The functions available to all modules are called *system methods* and are referred to as "external". A *dictionary* contains compiled versions of the local methods that are executed upon receipt of the appropriate message. A *library* holds references to all of the system methods that an MPU needs for its application.

The Local Method Manager coordinates receipt of incoming messages and their interpretation. Messages that request action or information are activated as local methods contained in the dictionary, while messages that are responses from previous external requests are passed on to the System Method Manager.

The System Method Manager coordinates activation of and response to system methods. A *method queue* is maintained by the System Method Manager for external commands or data requests that require a response. Upon receipt of the required information, the method queue is searched according to message address and sequence number, and the external reference is resolved with the response (result) being passed back to the calling function (method). The queue allows the application program to activate several methods sequentially and then coordinates receipt of responses, allowing the main program to continue execution until it is ready to process the incoming data. Services are available to the application program for examining the status of queued method activations.

Initialization of the Message Manager includes resolving system method address references. A *phone book* is maintained that contains the names of all externally referenced modules. Upon start-up, each module whose name appears in the phone book is searched for on the MODBOT network. If found, the module's address is entered and subsequent references to that module can be resolved. If the module can't be located, then system methods referencing that module will fail.

2.3.2.2 Logical Device Subsystem

The Logical Device Subsystem consists of a Logical Device Interface and an associated *blackboard* data structure. The blackboard is used as a global module data storage and retrieval mechanism, and provides a convenient and consistent means of maintaining local variables (Aviles, Laird, and Myers, 1988).

The blackboard is typically based upon *logical devices* that have an abstracted real-world implementation. Logical sensors and actuators, for example, are used

to represent devices whose state is maintained in the blackboard. Functions attached to the logical devices update their physical counterpart as information is requested from or entered into the blackboard. Devices that have no hard implementation are called *virtual*, and can be used to simulate an actual entity.

The Logical Device Interface provides software services for adding items to the blackboard, and for updating and retrieving the various data fields of those items. Activation of the functions attached to the items is automatic depending upon the device interface function used.

2.3.2.3 Application Controller

For applications that require no special processing, e.g., simple sensor or actuator modules, a Default Application Controller is provided by the MRA MPU. The default controller takes the place of the User Application main program, and is responsible for initializing and coordinating the other subsystems of the MPU.

For specialized modules such as high-level path planning or distributed task controllers, the user must supply the main application program (i.e., the User Application). In this case, the application program is responsible for initializing the MPU subsystems and for managing the MPU software resources as required (see sections 2.2.3 and 3.2.2).

2.3.3 Inter-Module Message Format

The MODBOT network is based upon the ISO open systems communication model, and implements the physical, data link, presentation, and application layers. The message format used at the presentation and application levels is based upon the Robotic Vehicle Message Format (RVMF) developed by TACOM (Brendle, 1990). The primary differences between the RVMF and that used under the MRA is in the placement and bit requirements for the unit destination and source address fields as well as the message length and sequence number. These fields were modified to optimize message acknowledgement and function execution (only three bytes are required to ACK a message and only five bytes needed to execute a module function). The RVMF block address and unit ID correspond to the MODBOT address and module ID respectively.

The modified RVMF message format (RVMF*) is basically maintained from layer to layer, that is, very few overhead bytes are added as the message is passed between the data link and application layers. This greatly increases data throughput and simplifies the MRA communications software interfaces.

3. THE MOBILE SECURITY ROBOT (MOSER)

As previously stated, MOSER is the first application of the MRA. The mobile security robot is made up of two physically separate subsystems. First to be explained is the control station followed by the remote platform (figure 1). Though functions of a mobile security robot are numerous (Everett, 1988), a typical scenario might include a guard positioned at the control station who would instruct one or more robots to patrol specific areas and to alert him/her when an intruder is detected.

3.1 The Control Station

The purpose of the control station is to provide various ways for an operator to receive sensor information from the MODBOT, to process that information, and to give commands as to future actions to be taken.

The control station is viewed conceptually as a MODBOT module whose main processing unit is located remotely from its associated module on the remote platform. The telemetry link connects the control station processor to the remote platform portion of the control station. The hardware architecture does not specify the actual processor requirements for the control station nor any part of the control station hardware except that it be capable of supporting the MRA standard software systems previously described.

For our application, the control station's main processing unit is a Macintosh IIx computer which communicates with the MODBOT via the RS-232 port. The main control station program (written in THINK C) running on the Macintosh will allow the operator to receive and send information to the robot. This program will also allow the operator to activate or deactivate specific robot modules and to turn on or off a MODBOT's video display by use of a pop-up window on the Macintosh monitor. The operator will be capable of communicating with the computer via the standard keyboard and mouse, or via speech and joystick inputs.

3.2 The Remote Platform

The remote platform for MOSER is made up of a detachable base with accompanying power source and various sensor, actuator, and processing modules.

3.2.1 Mobile Base

The mobile base for MOSER is currently a modified version of the TRC Labmate. The Labmate is a stand alone, battery powered base which has RS-232 interface capabilities to talk to the mobility module (not shown). The mobility module issues high level commands to the base such as "go forward 10 meters" or "turn left 90 degrees". The processing unit on-board the base uses these commands to control the drive system.

The base is also responsible for supplying power to MOSER's modules via the PPCU and MODBUS. The 24 volt battery supply is input to the PPCU which then distributes the conditioned power to the rest of the bus.

3.2.2 Remote Platform Modules

Each robot module enables the robot to obtain and process different information about its surroundings. Once processed, the information is used for purposes such as collision avoidance, navigation, and intrusion detection. In addition to use on-board the robot, pertinent information is sent back to the control station via the communications link. The Collision Avoidance Sonar Module and the Mobility Module have been previously described. Other modules shown in figure 1 are described below.

The High Level Processing Module, housing a WinSystems, Inc. AT286 computer mounted in a card cage, receives higher level commands from the control station, for example "patrol the third wing of building 1". This module then uses its internal room and building map information, as well as information from other modules, to navigate to its new patrol location. The Near-Infrared Proximity Sensor Module is another means of determining if objects are in close proximity to the robot. This ring contains eleven sensors in its front 180 degrees, each one having a range of approximately 3 feet. The Near-Infrared Module can also be used as a substitute for the Collision Avoidance Module. The Intrusion Detection Module, yet to be constructed, will include heat, light, and motion detecting sensors whose information will be fused together to determine the probability of an intruder. When the probability is above a predefined threshold, the operator at the

control station will be alerted. The Control Station Module includes a communications link which is used to transfer data between the control station and the remote platform. The remote platform portion of the module will include an ICN, a PDN, and the electronics necessary to drive the telemetry link. Initially an RS-232 tethered link will be used, while future plans are to convert to a more complex RF or Infrared link. The Navigational Sonar Module is a 360 degree version of the Collision Avoidance Module. The range information gathered by these 24 sensors is used by the high level processor for position estimation and navigation. The Stereoscopic Pan and Tilt Module will be used to return vision and acoustical information on a separate link to the control station.

4. SUMMARY

With the NOSC Modular Robotic Architecture, a robot control system with evolutionary growth potential can be realized. Adding and replacing robot modules, thus altering the robots functionality and complexity, can be accomplished with a minimum of integration overhead. Standardized hardware interfaces achieved with the Intelligent Communications and Power Distribution Nodes, used in conjunction with simple, flexible software interfaces allow for parallel development of various robot capabilities. Though the architecture is specific in its interface requirements, its design is general enough to be suitable for use in all autonomous and semi-autonomous applications be they air, sea, or land based.

The Mobile Security Robot is the first implementation of a robot being developed using the Modular Robotic Architecture. The MODBOT will be capable of patrolling its designated area and alerting a security guard positioned at the control station when an intruder is detected. This capability allows one guard to monitor and command several robots at various locations throughout the secured area.

5. REFERENCES

Aviles, W.A., Laird, R.T., Myers, M.E., November 1988. "Towards a Modular Robotic Architecture", *Proceedings SPIE Mobile Robots III*, pp. 271 - 278, Cambridge, MA.

Brendle, B.E. Jr., July 1990. "Robotic Vehicle Communications Interoperability Protocols", *Proceedings AUVS-90*, pp. 267 - 279, Dayton, OH.

Everett, H.R., March 1988. "Security and Sentry Robots", International Encyclopedia of Robotics Applications and Automation, John Wiley, NY.

Everett, H.R., Gilbreath, G.A., Tran, T.T., August 1990. *Modeling the Environment of a Mobile Security Robot*, NOSC Technical Document 1835, Naval Ocean Systems Center, San Diego, CA.

Intel Corporation, January 1989. "83C152 Hardware Description and Data Sheets", *Intel 8-Bit Embedded Controller Handbook*, No. 270645-002, pp. 9-1 - 9-87.

International Standards Organization (ISO), April 1980. *Reference Model for Open Systems Interconnection Architecture*, ISO/TC97/SC16 N309.

Laird, R.T., Smurlo, R.P., April 1990. *System Specification for a Modular Robotic Architecture*, NOSC Technical Document Draft, Naval Ocean Systems Center, San Diego, CA.