DESIGN OF A MOBILE ROBOTIC PLATFORM FOR RESEARCH IN GROUP ROBOTICS

JAGAN N. SESHADRI

DESIGN OF A MOBILE ROBOTIC PLATFORM FOR RESEARCH IN GROUP ROBOTICS

by

©JAGAN N. SESHADRI

A thesis submitted to the School of Graduate Studies
in partial fulfillment of the requirements for the
degree of Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland
2002

St. John's Newfoundland Canada

Abstract

This thesis presents the design and development of the "SAMBUCA" (Semi Autonomous Mobile Base and Utile Control Architecture) mobile robot base for the purpose of advancement of research in group robotics. The base is intended to be used as part of a multi-robot system whereby a number of robotic vehicles, including ones derived from this design, will be deployed and coordinated to perform tasks semi-autonomously in order to reach a goal.

Research in group robotics is being conducted at Memorial University of Newfoundland, in the Intelligent Systems laboratory of C-CORE, and the outcomes of these endeavors will be applied to industrial problems in harsh environments. The initial application of this research will be toward the mining industry, and specifically toward the automation of underground ore mines in Canada. Future applications include, but are not limited to, space exploration, toxic waste management, and automated farming.

The work done to date in the Intelligent Systems laboratory has included the initial development of a vehicle route planner and a discrete-event based traffic controller that has been successfully interfaced with groups of robotic vehicles to carry out specified tasks in relatively structured, static environments. Currently, these systems break down goals into tasks that are coordinated and dispatched to toy robots that operate in a model mine. While the use of toys is sufficient for proof-of-concept demonstrations of the discrete-event control and planning systems, toys must be adapted and coerced to work under conditions that they were not intended to experience. These adaptation efforts are not central to the research goals, and not only demand unnecessary efforts from the research group, but indeed limit the progress that can be made in developing an automated system.

While it is too early to incorporate the current research results into full-sized industrial robotic vehicles, specially designed small-scale robotic vehicles can be used in mock system trials instead of toy-based robots, thereby providing more accurate representations of the challenges encountered by full-sized robotic automation systems.

This thesis presents the development of an indoor/outdoor mobile platform that will be used to advance the group's research in automated mining by providing a more realistic experience of tele-operation, remote sensing, and semi-autonomous robot behavior than what currently exists in the Intelligent System laboratory.

The work completed to date acts as a starting point from which improvements and extensions can be easily made and incorporated. In this regard, suggestions for future work are also presented.

Acknowledgments

I will begin by thanking my supervisor, Dr. Ray Gosine, who suggested this as my research topic, provided me complete freedom to try ideas and be essentially self-directed, and gave support when it was needed. Working with fellow graduate students in the Intelligent Systems Laboratory at C-CORE allowed for the exchange of ideas as well as many educational discussions and debates. The mobile robot is undoubtedly better because of these meetings. Acknowledgement must also be given to Dr. Siu O'Young for inviting me into the INCA (Instrumentation, Controls, and Automation) laboratory meetings and seminars. These events have been tremendous for the cross-pollination of ideas, and for networking with other talented engineers and scientists.

Personal acknowledgements are necessary to express how much interdependence there is even in a self-directed project like mine. I wish to thank Raymond Pretty for providing tips about software design methodologies and design patterns, Mark Simms for lending his expertise in software troubleshooting, Lloyd Smith for permitting me to use his RAD-card circuit in the early stages of the robot design, Baxter Smith for showing me the essentials of PIC assembly programming and for helping test the robot base, and Sheldon Andrews for providing a very solid underlying code skeleton for the RAD-card. As well, I wish to thank Peter Woodman for listening to my various troubleshooting theories semester after semester and for being an all-around helpful and talented fellow, Dylan da Silva for offering the code base for the force-feedback joystick, Andrew Sinclair for offering the RT200 manipulator interface code and for helping to design the safety motion watchdog circuit, Jamie King for critically questioning decisions throughout the entire design and for advocating neat handiwork, Melanie Campbell for offering to do detailed tests on the infrared proximity sensors, and Terry Hussey for helping to build the safety motion circuit.

Many thanks to C-CORE engineers Jerry English and Ian Durdle for helping me out of many a PC/104 bind. I also wish to show my appreciation to technician Karl Tuff for suggesting appropriate tools and techniques for integrating SAMBUCA's systems, and to Geoff Liggins for suggesting a myriad of mechanical design ideas.

I also wish to thank the Faculty of Engineering and Applied Science and C-CORE, Memorial University of Newfoundland, for providing much needed laboratory space, equipment, and technical services to take this project from idea to implementation.

This work would not have been completed without the support from my parents and my paternal grandmother who were always there to accentuate the positive side of post-graduate studies. They were right, of course!

Thanks also goes to my roommates (fellow electrical engineering graduate students themselves) Michael Wrinch and Lloyd Smith, who shared the experience of living in an "Empire", working 80-hour weeks starting up a engineering company while earning post-graduate degrees, and understanding the balance between fun, relaxation, and discipline.

Financial support from the National Sciences and Engineering Research Council of Canada (NSERC) and from the Government of Newfoundland and Labrador (Atlantic Accord Career Development Award) was greatly appreciated.

List of Figures

Contents

Chapter 1

Introduction

The prospect of living a carefree life with all work done by machines has long captured the imagination of people since times well before modern science. Although time and experience has tempered these expectations somewhat since the onset of the widespread automation during the Industrial Revolution of the 1700s, automation remains an exciting and rewarding pursuit. Understood now as a set of helper technologies rather than a complete replacement for human effort, automation has made a positive impact in areas such as manufacturing, shipping, and communications. While automation in the form of mechanized systems has existed for centuries, epitomized by the Jacquard loom, it is the recent synergistic combination of mechanical systems with computer systems that has resulted in modern robotics, which has added the important ability for a system to monitor and optimize its own processes, further reducing the need for human operators, and thereby increasing process efficiency.

While the manufacturing industries have benefitted from today's robotic automation systems (especially the automobile industry, whose use of industrial robotic arms has arguably provided the greatest manufacturing improvement since Henry Ford devised the assembly line), the state of technology has limited the use of robotics in automated systems to operating in fixed positions, dealing with well-constrained and patterned problems (e.g. welding, basic visual inspections). These limitations have thus far precluded the effective use of automation in less structured environments, such as those encountered in underground or surface mining or other resource development applications.

The continuing push on automated technologies will make great use of robotics as computer technology continues to advance, and with the establishment of the Internet communications infrastructure. Given the current technological conditions, it makes sense for engineering research and development laboratories, both in the private sector and in academia, to pursue the area of robotic automation, a subset of intelligent systems. The Intelligent Systems laboratory at C-CORE, Memorial University of Newfoundland, Canada, benefits from close ties with industry, and is a fitting place to carry out applied, industrially-related research in robotic automation.

The research and development necessary to implement ideas is effort-intensive, and requires the interplay of many systems. The work presented in this thesis describes one of the subsystems needed for an intelligent system to work effectively, namely a mobile robotic platform. This introductory chapter will define an intelligent system and will provide a brief review of the work that has preceded this current project in order to provide a context for the author's research.

1.1 Background of Intelligent Systems Laboratory

1.1.1 Definition of Intelligent Systems

This section will begin by first presenting the laboratory's modified Precarn[1] definition of intelligent systems, which serves as an overall guide for the kind of R&D conducted there.

> *Intelligent Systems*: systems that perceive, reason and act in ways that are similar in function to humans[19].

The work carried out in the Intelligent Systems laboratory is therefore focused in areas of sensor fusion, process modeling, system automation, and remote monitoring.

1.1.2 History of the Intelligent Systems Laboratory

It is often the case that research in intelligent systems begins as a logical extension from the computer science subfield of artificial intelligence (AI). Interestingly, this

[1]Precarn is a consortium of Canadian corporations, research institutes, and government partners working within the intelligent systems industry.

was not the evolutionary path taken by the Intelligent Systems group; its beginnings were rooted in the area of machine vision rather than AI, and the group had experience developing automated visual inspection systems for applications such as fish processing, automated aerial image analysis, acoustic image processing and analysis, and telerobotics.

Participation in a mining company's automated ore fragmentation image analysis project led the Intelligent Systems group to pursue further work in mining automation, and to participate in the SMART[2] project, a joint research effort between a number of Canadian research laboratories [20]. The next subsection will give a description of the technological challenges facing the mining industry, and the subsection after that will give more detail about the SMART initiative.

1.1.3 The Industrial Problem

The North American mining industry, like many other industries, is interested in increasing its efficiency at a time when labor costs in North America are higher than in other parts of the world, and worker safety issues are paramount. A significant portion of the operational costs is associated with providing adequate infrastructure for workers to be able to mine at depths of up to one mile beneath the surface where heat, air quality, and structural integrity of mining tunnels are all major safety issues. Even when these issues are addressed, underground mines remain harsh environments in which to work, an ideal focus for applied R&D in intelligent systems.

INCO, a Canadian mining company, has begun evaluating and incorporating automated mining machines to help improve the efficiency of their mining processes. According to their proposed Mining Automation Program (MAP), INCO desires to move workers out of the harsh environment by completely automating the mining process, and permitting the workers to supervise the machinery from a safe and comfortable supervisory console on the surface [21]. This will improve worker safety and mining efficiency, both of which help INCO remain competitive.

Some of the systems needed for mining automation to occur, as noted by INCO [22], are:

1. Underground telecommunications

[2]Sensori-Motor Augmented Reality for Telerobotics

2. Vehicle positioning (localization) and navigation

3. Mining equipment automation

4. Process engineering and control (i.e. operations research)

INCO has begun addressing the issues of underground telecommunications, and has been collaborating with mining equipment manufacturers to deal with converting standard mining vehicles into automation-ready vehicles. The remaining systems (i.e. vehicle positioning/navigation and process engineering/control) have been the focus of research laboratories, and the SMART initiative has focused work on these remaining subsystems.

1.1.4 Sensori-Motor Augmented Reality for Telerobotics - SMART

The overall goal of SMART was to take the existing concept of direct teleoperation in underground mines (i.e. remote control of mining vehicles by a human operator without the aid of automation) and extend it to provide mining vehicles with a level of automation, thereby overcoming some of the following drawbacks of direct teleoperation:

1. The human operator must remotely control a machine that interacts with the environment in a complex, detailed, and realtime manner. Therefore the operator must devote full attention to the control of the machine for the duration of the task.

2. Transmission delays experienced by remote signal propagation time can disrupt the synchronization required for the operator to control the vehicle, resulting in decreased motion accuracy.The communication infrastructure[3] used by INCO in the past, however, "showed no noticeable delay to the operation of the controls on the [remote vehicle] from surface approximately 1300 metres away" [23].

[3]The communications infrastructure in INCO's Copper Cliff, Ontario 'North Mine' consisted of radio frequency coaxial cable feeds whereby analog video was transmitted on certain channels, and computer data was transmitted on other channels, with each channel taking up approximately 6 MHz of dedicated bandwidth.

Delays, jitter, and lost data are more likely in packet-switched communication infrastructures, especially those that do not have guaranteed quality-of-service or real-time transmission protocols in place [24].

3. Directly teleoperated vehicles do not partake in intervehicle communications, thereby decreasing the chances of avoiding multivehicle conflicts that may arise during operations.

By enhancing vehicles with safety systems, a degree of built-in automation, and by presenting the human operator with an augmented projection of the remote situation, SMART intends to decouple task planning from task execution. Rather than have the human operator do both the planning and execution, the operator would be able to create a plan for the vehicle and then set the vehicle on auto-pilot to execute the task.

As a side benefit of this planning/execution decoupling, the human operator could possibly control a fleet of vehicles, thereby increasing the efficiency of the entire operation.

1.1.5 The Laboratory's Approach

The Intelligent Systems laboratory (ISLab) originally planned to contribute to the SMART project in the areas of intelligent control of robotic equipment, and intelligent mediation between the system's automatic control mode and human-operator control mode. As it was difficult to work on these aspects in isolation from the rest of the SMART system components, the ISLab devised its own "General Framework for Group Robotics". This framework is discussed in more detail in the next major section. ISLab first focused on intelligent control of vehicles to complete a task, which included plan generation of subtasks, autonomous completionof the subtasks, and monitoring of subtasks to alert the human operator to any troubles during task execution. The following subsections outline the incremental work that was done prior to the start of the work presented by this thesis.

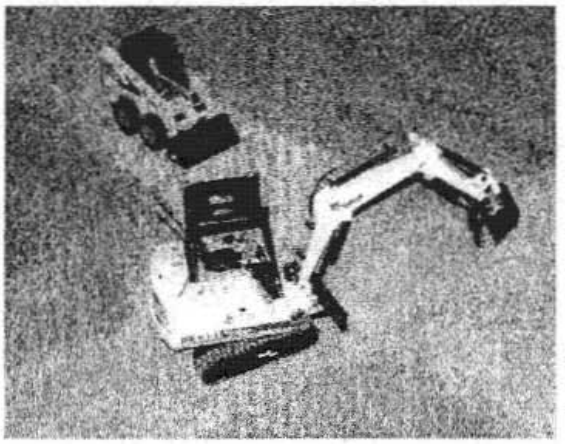


Figure 1.1: Two-robot Collision Avoidance Demonstration [1]

Demonstration #1

The first demonstration of the ISLab system was to show how two mobile robots could drive along separately specified but intersecting paths and avoid collision (see figure 1.1). This experiment was carried out using two miniature Bobcat differential-drive toys that were computer controlled using altered radio-frequency remote controls. This demonstration, which was used to test Hwang's Petri-net controller [1], made use of an overhead camera system to provide robot localization to the Petri-net controller. The Bobcat toys had no on-board intelligence, and simply reacted to issued commands. The toys also lacked self-monitoring abilities.

Demonstration #2

The next demonstration was carried out by Jamie King [25] in order to coordinate three robot vehicles with a rewritten Petri-net controller (renamed as the Discrete-Event Controller) using a more complex roadway as well as a route planner. This time, the Bobcat toy chassis were outfitted with MIT-designed MiniBoards [26] and 'UniLink' wireless RS-232 serial communications (manufactured by Wireless Mountain Laboratories). By tracking transitions between reflective and non-reflective floor tape, the central coordination system kept track of robot localization and dispatched

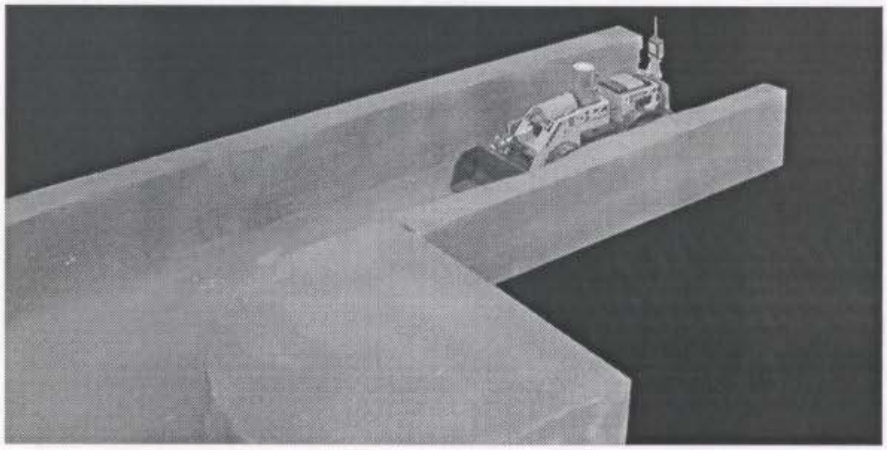


Figure 1.2: Styrofoam Mine with Tonka Toy Robot

commands accordingly. However, sensor noise, communications noise, and battery drain all posed reliability problems during the experimental trials.

Demonstration #3

The third demonstration was essentially the same in scope as the second demonstration, except that it was done in a scale-sized mine constructed of styrofoam, and made use of larger, more powerful toy vehicles controlled by locally-designed RAD cards [27] rather than the MiniBoards. This demonstration avoided most sensor misreadings by not relying on reflective tape, but rather using proximity readings between the toy vehicles and the walls of the scale mine (see figure 1.2). Reliability problems with serial communications still remained.

Shortcomings of Existing Robots

As has been presented so far, as the demonstrations of the general framework for group robotics became more ambitious and technically complex, a larger portion of the problems came from the limitations of the toys being used as robots. The problems caused by the use of toys were consuming more development time than was deemed acceptable by the group, and it was agreed that work would begin on specifying design requirements for a more useful and universal robotic base with

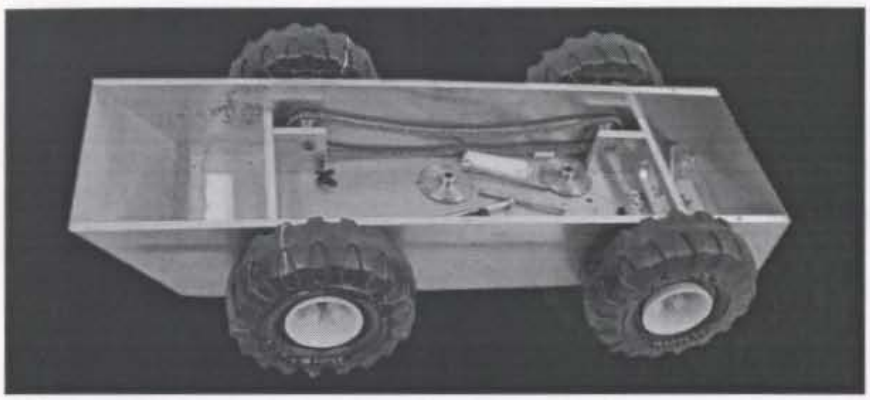


Figure 1.3: SmallBot

which to advance the group's research.

SmallBot

The SmallBot project was the first attempt at creating a more heavily-featured miniature robot that was rugged enough to operate on challenging non-laboratory terrains, and flexible enough to make use of a wide variety of sensor and actuator configurations. In essence, the SmallBot was to be a full-fledged personal computer on wheels in miniature form so that it could be used in the scale-model of an underground mine. A miniature personal computer was specified, based on the PC/104 standard used in industry. PC/104 systems are comprised of computer cards that all have a form factor of approximately 4" by 4". These cards are stacked to provide a completely operational personal computer that takes up very little space and uses very little power compared to a desktop or laptop PC. The SmallBot base was designed around this form factor, resulting in a narrow boat-like shape, and requiring miniature motors, miniature sprockets and chains, miniature wheels, and three small gel-cell 12-Volt batteries.

The project soon was halted as it became apparent that the costs of miniature gears, sprockets, and custom machining were all exceeding the anticipated costs. The problem with creating a small yet rugged robot was that the parts were not in high demand worldwide, and therefore were rare and expensive to procure. The

cancellation of the SmallBot effort led way to the robot base presented in this thesis.

1.1.6 Objective of Research

From the outset of this project, the design of the MRP was to be general enough to be used for a variety of robotic research projects which might deal with telemanipulation, mapping, path planning, navigation, object recognition and research, surveillance, and object transport, to name a few possibilities.

The objective of this project, therefore, was to design and implement a mobile robot platform (MRP) that would:

- operate within the ISLab's General Framework for Group Robotics
- operate in both fabricated and natural environments
- be modular and extensible to ease future development
- provide a more realistic portrayal of controlling a robotic vehicle than what was currently being experienced with miniature toy robots

1.1.7 Organization of Thesis

Chapter 2 presents the Intelligent System group's "General Framework for Group Robotics" to explain the context in which the mobile robot base was developed. Chapter 3 is a review of the mobile robotics research literature, as pertains to this project. Chapter 4 gives details on the specifications and designs of the mobile robot base hardware and the associated control software. Chapter 5 describes the implementation and testing conducted on the base to date and presents the experimental results. Conclusions and suggestions for future work are presented in Chapter 6.

Chapter 2

General Framework for Group Robotics

The job of automating a system as complex as an underground mine is best accomplished in stages, from first working with computer simulations of vehicles, to using incrementally better physical models, to finally incorporating the honed designs into full-size machinery. The work in the ISLab has been primarily directed at using and controlling physical models of mining robots to achieve high-level goals. Design of such a system benefits from a divide-and-conquer approach, and as such, requires definitions of work scope and of connecting interfaces between these work modules.

Rather than use an *ad hoc* approach to factoring the complex problem into manageable segments, it was decided that a general framework for group robots (i.e. not a mining-specific implementation) be devised. This framework, based on the warehouse security robot architecture of Everett and others [28], is presented in figure 2.4 and more details can be found in [25] and [1].

2.1 Overview of Framework

The General Framework for Group Robotics is an architecture for controlling multiple robotic vehicles in dynamically changing unstructured or semi-structured environments. This framework was formed based on the following observations:

1. Complete automation of complex semi-structured environments is not yet technically feasible, but partial automation (making use of human intervention) is feasible.

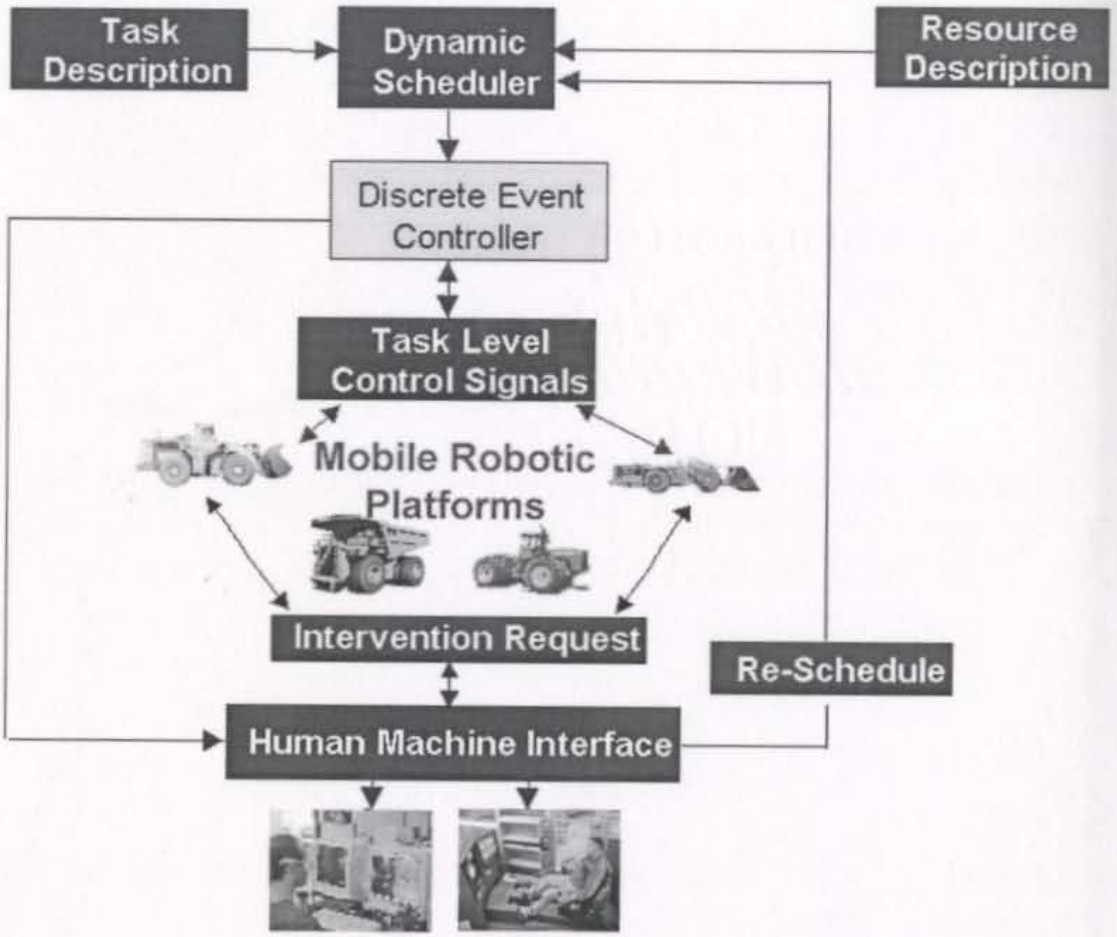


Figure 2.4: General Framework for Group Robotics

2. The environment in which the system operates is assumed to dynamically change at a rate slow enough to permit effective control system adaptation.

3. The system is to control a fleet of heterogeneous vehicles, and therefore should not depend on individual robot architectures.

2.1.1 High-Level Task Description

The framework for controlling a team of robotic vehicles incorporates subsystems that operate at varying levels of task abstraction. At the highest level of task abstraction, human operators can use command constructs similar to those used in high-level computer languages to issue tasks for the system to execute. For example, an operator can give the system a task of moving ore from a blast site to a crusher machine repeatedly until the blast site is empty.

2.1.2 Resource Description

Automation requires an understanding of the resources available in the system, so that path planning, coordination, and optimization of effort can take place. In this case, resources (followed by pertinent parameters) are: vehicles (vehicle type, size, fuel consumption, time to maintenance), roadways (traversal costs), intersections (maximum number of vehicles allowed at one time), ore piles (quantity, grade), and crushers (capacity).

2.1.3 Dynamic Scheduling

For high-level task descriptions to be carried out, co-operation is often required among a fleet of robotic vehicles. Each vehicle's individual duties are determined by a scheduler, whose plans are the result of constraints provided by the resource description applied to the high-level task descriptions. The scheduler can be thought of as a coach that formulates the plays for the team to carry out. The work schedules may need to be altered if something goes wrong with the robots or with the surroundings (e.g. a blocked pathway), which effectively changes the resource description. Work re-allocation is to be done dynamically (as the task is underway), to

prevent all of the dispatched vehicles from having to be recalled. Dynamic Scheduling is a major feature of the general framework, and is paramount when dealing with real-world automation problems.

2.1.4 Discrete Event Control

Once the work duties and paths have been allocated to each vehicle, the vehicles need their plans coordinated, and this is done using the discrete event controller (DEC). The DEC is a centralized controller that minimizes inter-robot communications, and therefore increases the scalability of the control framework. By using Petri-net formalisms to model resources and tasks, multi-vehicle coordination can be achieved so that traffic deadlock and vehicle collisions can be avoided. The DEC monitors task completions and sends signals for the execution of the next task. Task-level control signals tell the vehicles what task to perform and when to perform it, but does not give details of how the task is carried out, as that information is part of the vehicle's on-board intelligence. This abstraction reduces the number of discrete events that need to be coordinated and gives flexibility to the designer of the robotic vehicle control system on how to carry out specified tasks. Each vehicle used in this framework must be able to accept these task-level control signals and carry them out.

2.1.5 Mobile Robots

For industrial purposes, these mobile robots would actually be mining vehicles retrofitted with onboard machine intelligence and sensors, but are abstracted in this framework simply as mobile devices that can accept task-level signals from the DEC and return progress signals. The work presented in this thesis fits into this category by providing a mobile robotic platform (MRP) onto which additions and modifications can be made to create a mobile robot suited for the tasks desired by the system designer.

2.1.6 Human-Machine Interface

One key assumption in this group robotic system is that complete automation is not a reasonable expectation given the current level of technology. However, it is fair to expect the system to run most of the time in automatic mode, but in case the system cannot resolve a situation then a human operator will be alerted and will manually remotely operate (teleoperate) the vehicle or vehicles until the system can carry on in automatic mode. Therefore, a human-machine interface (HMI) is an integral part of the general framework. The HMI will convey pertinent environmental information to the human operator, including camera views, compass headings, vehicle tilt readings, fuel levels, speeds, vehicular distance from obstacles, and force feedback responses. The HMI could also include input mechanisms such as joysticks, voice recognition, gaze tracking, and standard keyboard and mouse inputs.

2.1.7 Chapter Summary

Automation of complex operations, such as those in an underground mine, are solved by using a control architecture that divides the automation problem into more manageable subproblems. The General Framework for Group Robotics presents a design pattern by which the overall automation problem can be divided and conquered. In addition to outlining subsystems for pure automation[1], the framework allows human intervention to occur in scenarios where pure automation may fail. This group robotic control architecture is also meant to be largely independent of individual robot design specifics, thus giving latitude to the kinds of mobile robot designs that can be used. Examples of mobile robot designs are presented in the next chapter, with the intent of incorporating beneficial ideas into the design of a mobile robot platform for the ISLab's research endeavors.

[1] Pure automation means automation that does not require human intervention.

Chapter 3

Review of Related Work

Every mobile robotic platform is comprised of two major subsystems: the physical mobile base and the associated control system. Since the motivation of this project was to create an MRP that would operate indoors and outdoors and be able to operate in natural and artificial terrains, it was decided that the literature search for this project be focused on all-terrain MRPs. This chapter presents a brief overview of commercially available mobile robotic platforms as well as non-commercial platforms developed in other research laboratories, with the focus being kept on the physical mobile base. Since the details of robotic control systems vary from implementation to implementation, a background on robot control architectures will be presented instead in order to reveal common ideas and techniques used to develop mobile robotic platforms.

3.1 Commercial Mobile Robotic Platforms

Modern robotics research requires computer systems to perform environmental data processing, motion control, and communication functions, all of which can be computer-intensive. In the past it was not uncommon to see multiple computers interconnected on the robot itself, with each computer dedicated to a specific function. Systems created in this fashion were task-specific by nature, requiring specialized hardware configurations depending on what the robot design was meant to accomplish. Therefore, researchers devoted much of their time toward building specialized computer configurations, resulting in designs of one-of-a-kind robots. More

recently, as computational power has continually become less expensive, and as computers themselves have become more of a commodity item than a specialty item, an individual off-the-shelf computer system is now capable of performing all necessary robotic computational duties, and configurations can be adjusted in software. With this advancement in computer technology, some companies have identified and filled a business niche by creating generic mobile robotic platforms specially targeted toward robotics researchers who can then tailor these systems to their liking and needs.

This section will describe some of the all-terrain MRPs that are commercially available, as well as name the research institutions who have used them.

3.1.1 K-Team

K-Team is a company that grew out of the Swiss Federal Institute of Technology of Lausanne's Research Centre in Mobile Robotics. The company designs and manufactures small mobile robotic platforms used for research, education, and even entertainment purposes. K-Team makes a number of different types of mobile robotic platforms, including the Koala all-terrain platform [29]. The Koala is touted as a mid-sized robot able to perform complex tasks in real-world environments. It has a footprint of only 30cm by 30 cm, essentially keeping within the 'toy-robot' category, but it is modular by design, and is capable of interfacing with six ultrasonic sensors, a 500- by 582-pixel pan/tilt camera, a wireless RS-232 modem, a 4000mAh battery, and can be used with several software development environments such as C, LabView, and MATLAB. Modular hardware design and ready-made software development kits are both selling points for the K-Team robots, and for the Koala in particular.

3.1.2 ActivMedia Robotics

ActivMedia Robotics describes itself as "a robotics manufacturer, systems integrator and (producer) of affordable, useable intelligent mobile robots" [30]. Specializing in robots both larger and more rugged than the K-Team products, robotics researchers find ActivMedia robots quite appealing, especially its Pioneer 2-AT platform[31].

Figure 3.5: Pioneer 2-AT [2]

Marketed as an all-terrain robot, the Pioneer 2-AT is a four-wheel differentially-driven robot having over 250 Watt-hours of battery power, that can operate autonomously using its PC/104-compatible on-board computer, and that can carry payloads of up to 30kg.

3.1.3 iRobot

Founded by Rodney Brooks (MIT Artificial Intelligence Laboratory) and two of his former students, Colin Angle and Helen Greiner, iRobot recently combined with the creators of research robotic platforms, Real World Interfaces (RWI). The all-terrain robots made by iRobot are the ATRV, ATRV-Jr., and ATRV-Mini, and are intended to enable robotics research to move from the lab out into the field [32]. These platforms cost tens of thousands of dollars, but are rugged and well-equipped.

Before RWI joined with iRobot, RWI sold their all-terrain robot as model ATRV-2. Since the late 1990's, the Perception and Robotics Laboratory at Ecole Polytechnique has used an ATRV-2 robot as their experimental robotics platform, which they use to study path planning and robotic control in dynamic environments [33][34]. It was their use of this ATRV-2 platform which motivated the Intelligent Systems group

to commission the creation of its own, but less costly, mobile robotic platform.

3.1.4 Applied AI Systems

Based out of Ottawa, Canada, Applied AI Systems creates their own line of mobile robotic platforms as well as distributes K-Team and iRobot platforms [35]. Among AAI's robots is the GAIA-1a Autonomous Indoor/Outdoor Mobile Platform, and the LABO-2 large-payload platform, both of which run on a Motorola 68332 32-bit microcontroller. As well, AAI sells its TAO-6 Intelligent Wheelchair Base (which also uses a Motorola 68332 microcontroller), targeted toward research in autonomous wheelchair design.

3.2 Research Laboratory Robot Vehicles

The robotic vehicles presented in this section predate the commercial platforms presented in the last section, but that is not to say that commercial robot platforms are a new phenomenon. In the 1980's, the now-defunct Denning Mobile Robotics company provided platforms that were used by researchers at Carnegie Mellon University, Georgia Tech, and even the United States Military[36]. However, these were not all-terrain vehicles, and therefore researchers who required robots with all-terrain handling had to create these themselves.

3.2.1 SARGE

Developed for the United States' Department of Defense by Sandia National Laboratories, the Surveillance and Reconnaissance Ground Equipment (SARGE) robot was originally designed for direct teleoperation on the battlefield, but has since included the use of vision systems and navigational algorithms, permitting a level of robot autonomy [37] (see figure 3.6).

Figure 3.6: SARGE [3]

3.2.2 CyberATV Platform

As part of their CyberScout project, researchers at Carnegie-Mellon University (CMU) created an all-terrain robot to enhance their research in perception, navigation, path planning, vehicular control, and distributed agent-based collaboration, within the context of land-based military mobile robotics [38]. The CyberATV was made by retrofitting a commercially-available ATV with electromechanical control actuators and a gasoline-powered electric generator, and by adding a two-tiered computer system; the first tier being a PC/104 computer for vehicle control, and the second tier being a group of three PC's responsible for communications, planning, and perception. A full sensor suite, including Global Positioning System (GPS) receivers and pan/tilt cameras were part of each vehicle. Two of these CyberATV platforms were built (named Lewis and Clark, after the famous American explorers).

3.2.3 Tin Man

Designed and build by the KISS Institute for Practical Robotics (KIPR), the Tin Man and Tin Man II are both power wheelchairs that have been modified in order to solve problems in assistive robotics [39]. These modified wheelchairs make it easier

Figure 3.7: "Lewis" the CyberATV [4]

for seriously disabled people to move around in the wheelchair since there is a level of navigational autonomy and assistive steering built in.

3.2.4 Wheelesley

Researcher Holly Yanco worked on creating the Wheelesley robotic power wheelchair as her Ph.D. topic. Wheelesley (named after Wellesley College, where the project originated) used KIPR's Tin Man modified wheelchair as its base (see figure 3.8), and added specialized user interfaces suited for people with motor skill disabilities. Designed to operate semi-autonomously in indoor and outdoor environments, Wheelesley operates in three modes: the first mode is direct joystick control (i.e. normal power wheelchair operation), the second mode is joystick control with automatic obstacle avoidance, and the third mode is control through specialized user interfaces [40].

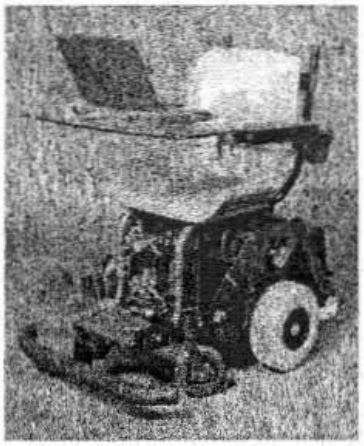


Figure 3.8: Tin Man Power Wheelchair Base [5]

3.2.5 The Wheelchair Project

Carnegie-Mellon University's "Wheelchair Project", led by Dr. Illah Nourbakhsh, is a robot based on a power-wheelchair, whose overall goal is to navigate autonomously in both indoor and outdoor environments. The power wheelchair used for this project was the Tin Man wheelchair, acquired from KIPR (see figure 3.8), and incorporates additional sensors into the wheelchair, with a vision system being the primary sensor used in navigation [41]. The project is meant to be solved in stages, from initially having a wheelchair that can be given a general direction of travel and automatically avoid obstacles, to a wheelchair that can be given a high-level goal (i.e. 'go to the coffee room') and take care of the navigation completely on its own.

3.2.6 NavChair

University of Michigan researcher Dr. Johann Borenstein, well-known for his work on mobile robot navigation systems, is involved in an assistive robotics project called NavChair. Using a power wheelchair as a base, and additionally equipped with a ring of ten Polaroid ultrasonic sensors, the NavChair uses an obstacle-avoidance technique called the "Vector Field Histogram" (VFH) to assist severely disabled people with the wheelchair navigation. Borenstein has noted that dead-reckoning techniques on

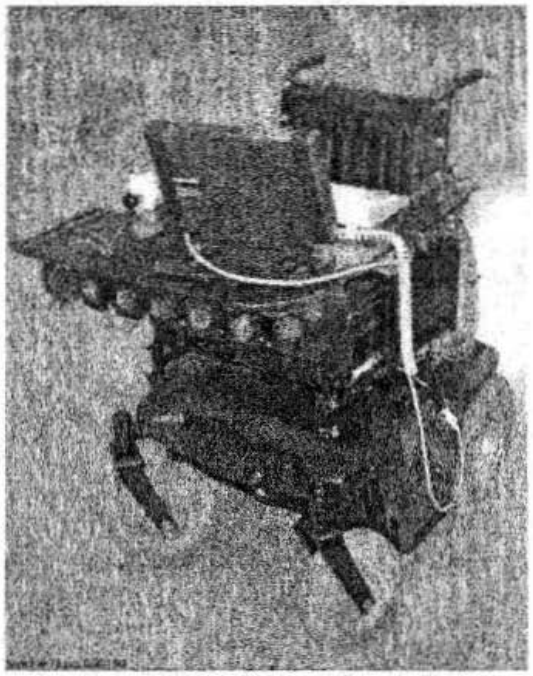


Figure 3.9: NavChair [6]

power wheelchair bases are not very effective, and while this may pose problems for robot localization, it is not problematic for obstacle-avoidance [42] (see figure 3.9).

3.3 Mobile Robot Control Architectures

The hardware aspects of robot design are important, but they only affect the brawn and agility of the physical system. The importance of a robot's intelligence, though less obvious to the casual onlooker, is equally important to the overall design. Having already reviewed examples of physical designs in the previous section, this section discusses some robot control architectures that have been suggested by researchers in robotics and artificial intelligence.

> *Robot control architecture*: Pattern of design that guides the construction of robot control systems using a collection of common hardware and

software building blocks and techniques.

Before deciding on a robot control architecture, existing architectures were examined for their benefits and drawbacks. An overview of the major architectural groups are described below.

3.3.1 Robotic Control Spectrum

Mobile robotics began as the physical manifestation of artificial intelligence (AI). Since the previous section discussed some of the physical designs done to date, this section will address the intelligent portion of MRPs. Traditionally, AI researchers believed that computers could be used to perform decision-making tasks normally performed by humans. This approach became known as deliberative control, and was the basis for machine intelligence throughout the 1960s and 1970s [7]. It has only been since the mid 1980s that an alternate control scheme came into being, namely reactive control, which was designed to combine multiple sensors together and as a side-effect, produce the correct actions for the given situation [43] [44]. Each of the above control paradigms sit at extreme ends of what can be considered to be a robotic control spectrum, and hybrid control architectures for mobile robots fall somewhere between these two extremes.

3.3.2 Deliberative Control

Also known by the acronyms SCR (Sense-Calculate-React), SPA (Sense-Plan-Act), or SMPA (Sense-Model-Plan-Act), deliberative systems usually first sense the environment of operation, create or refine an internal model using new sensor readings, compute a plan based on the sensor readings possibly combined with a performance cost function, and finally embark on the planned journey within the environment. For many years, this was the way that mobile robot control was approached, as was showcased by pioneering mobile robots Shakey[1] [45] and the Stanford Cart[2] [46]. While this approach was successful in research laboratories, it had its failings when applied to real-world situations. The key assumption when using deliberative control

[1]Shakey was developed by Nils J. Nilsson in the late 1960s at Stanford Research Institute (SRI).
[2]The Stanford Cart was developed by Hans P. Moravec.

is that the environment is static in terms of the robots sense-plan-act cycle. That is, the environmental characteristics sensed are assumed be the same when the plan is enacted some time later. The grander the plan, or the greater the number of inputs to process, the more the elapsed time between sensing and acting, and the less likely the original assumption would hold. Further complicating matters for the deliberative controller is that real-world robots are non-holonomic. The robot has dynamic limitations, and its movement through the environment can affect the dynamics of the environment that it is trying to measure and base plans upon. Even if the above problems can be avoided, the computational resources, both in term of CPU cycles and memory resources in which to calculate and store the model of the environment (often called the world model) can be very demanding, and suffers from the so-called curse of dimensionality whereby each new feature added to the world model exponentially puts more demand on computational resources. Deliberative control, however, computes answers to situational problems, and this computation enables more efficient plans than might otherwise be possible. The benefits and drawbacks of deliberative control include:

Benefits:

- Useful for structured predictable environments
- Sees the big picture and can optimize plans
- Flexible and adaptable in strategy

Drawbacks:

- Not suited for dynamic, unpredictable environments
- Computationally intensive, not suited for real-time control

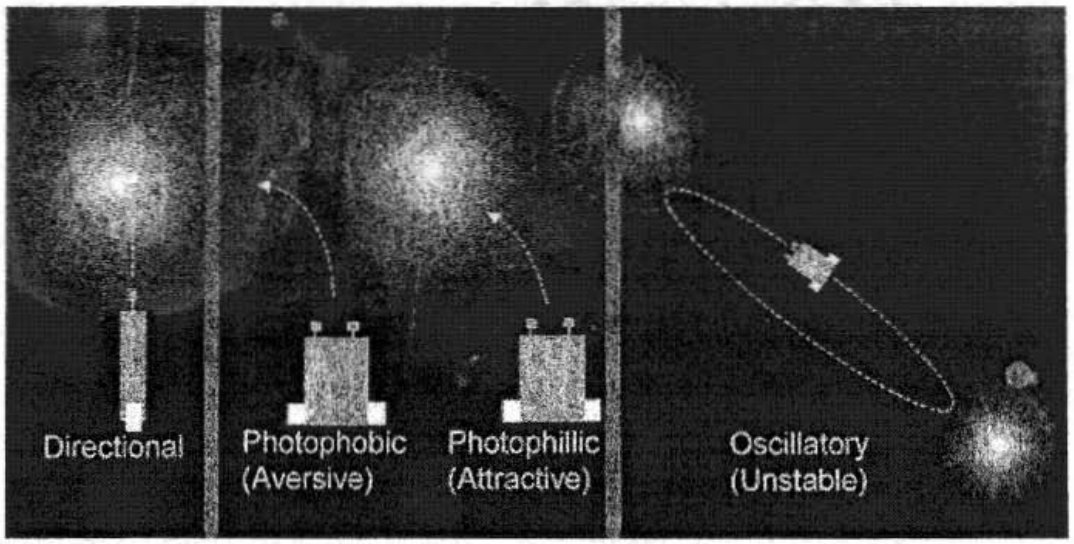


Figure 3.10: Reactive "Braitenberg" Vehicles in Different Configurations. Figure adapted from [7].

3.3.3 Reactive Control

This type of control is an outgrowth of traditional control system engineering whereby the sensors are coupled very closely with actuators, foregoing any explicit computation stage but resulting in very fast reactions to the dynamics of the environment. Early examples of robots that used reactive control are W. Grey Walter's tortoise (1953), and Valentino Braitenburg's collection of vehicles (1984), dubbed by researchers at MIT as Braitenburg Vehicles (see figure 3.10).

These robots were designed with multiple concurrently-running motor control systems that, when combined, produced a seemingly synergistic behavior known as an 'emergent behavior'. A hallmark of reactive systems is that they do not build up an internal model of the environment in order to act. While this aspect results in system inflexibility once a robot is built, the payback is that the robot does not spend its time in a computational daydream when faced with sometimes-perilous changes in the environment where it is situated. Before the mid 1980s, robots that used reactive control often were difficult to build in a way that would produce a desired

and somewhat predictable emergent behavior. This was because of the complex interplay of the many concurrent control system dynamics that caused this emergent behavior. MIT researcher Rodney Brooks provided a way of sensibly and intuitively assembling these multiple control systems so that they would do the bidding of their designer. Published as the Subsumption Architecture (1986), Brooks prioritized sensory inputs, whereby the most highly prioritized input would take precedence and dictate what action the robot carried out [44]. Using a system of prioritized inputs and circuitry that could inhibit competing actuation signals or suppress (and replace) signals, the subsumption architecture provided a way of designing reactive robot control to achieve desired behaviors. The benefits and drawbacks of reactive control include:

Benefits:

- Responds quickly to avoid problems when operating in an unstructured, dynamic environment

Drawbacks:

- Unaware of the overall problem or goal
- Not well-suited to operate in structured environments
- Can get caught in oscillatory actions
- Neither flexible nor adaptable in strategy

3.3.4 Hybrid Control

Both deliberative control and reactive control have advantages worth exploiting when designing mobile robots. Ideally, a mobile robot would be intelligent enough to create and follow through with a long-term plan while remaining quick and nimble enough to react to any unforeseen changes in its environment. Nature provides many examples

of the varying behavioral mixes in animal behavior, ranging from predominantly reactive to predominantly deliberative. Flies, for example, are very reactive in behavior, while higher animals are not as nimble or quick to react, but are more deliberative. It is from the biologically-based observations and theories put forth by ethologists that have inspired robotics researchers to adapt these findings to the creation of hybrid robot control architectures. These architectures are similar in that they have both reactive and deliberative control components, as well as a structure to mediate between the two control extremes. The challenge of developing a hybrid architecture is to strike the right balance between reactive and deliberative components. Arkin's 'Autonomous Robot Architecture' (AuRA) [47], Gat's 'A Three-Layer Architecture for Navigating Through Intricate Situations' (ATLANTIS) architecture [48], and Connell's 'Servo-Subsumption-Symbolic' (SSS) architecture [49] are all examples of hybrid robot control architectures.

3.3.5 The Intelligent Systems Laboratory's Hybrid Architecture

The ISLab at C-CORE, Memorial University, has been developed with the focus of providing semi-autonomous control of underground mining vehicles. The architecture has been given the working name, General Framework for Group Robotics and falls under the classification of a Hybrid Control Architecture. As with most hybrid architectures, the ISLab architecture has three major layers:

- Deliberative Layer (High-Level Task Description / Resource Description / Dynamic Scheduler)
- Sequencing Layer (Discrete Event Controller using Petri Nets)
- Controlling Layer (Mobile Robotic Platform's Schema-Based Controller)

The reactive layer is the schema-based controller (discussed in more detail in the next subsection), which makes use of potential field calculations and vector summations to negotiate paths in dynamic environments. This layer is responsible for carrying out actions issued by the sequencing layer and reporting any failures. The sequencing layer takes a set of road sections, treats each section as a resource that is

shared by mining robots, and ensures that traffic flows smoothly. This layer coordinates the motion of multiple robots by issuing high-level commands to each robots controller layer, and any sequencing problems that occur (e.g. deadlock) are brought to the attention of the deliberative layer. The deliberative layer is given a map of the environment and a goal to be reached. From this information, efficient paths for the mining robots are planned and issued to the sequencing layer. In case of sequencing trouble (which encompasses controller trouble as well), the deliberative layer can re-plan paths. In the case where the re-planning algorithms fail, the deliberative layer alerts a human operator who then attempts to correct the problem manually, using teleoperation if needed. Further discussion about the advantages of three-layer architectures can be found in Erann Gat's paper titled "On Three-Layer Architectures" [43].

3.3.6 Behavior-Based Control Architectures

The lowest level of the "General Framework for Group Robotics", i.e. the "Controlling Layer" was described in the previous subsection as 'reactive' since that is the naming scheme used in discussions about three-layered architectures. However, this lowest level need not be a strictly reactive controller; after all, such controllers are by definition limited to be strategically inflexible. From the perspective of the "General Framework for Group Robotics", organizationally equivalent yet not strictly reactive controlling-layer architectures are the *behavior-based* control architectures. As stated by robotics researcher Maja Matarić, "The designer of a behavior based system, rather than design the system itself, performs the mapping between the conditions and the actions" [50]. In other words, instead of being restricted to hard-wiring (or hard-coding) stimulus-reaction pairs and blindly running them in hopes of achieving a useful emergent behavior (as would be done with purely reactive architectures), behaviors can be developed by abstracting the connections between perception and action into meaningful behavioral *primitives*. Behavior-based architectures group reactive perception-action connections into better-understood modules. As a result, the mystery of 'emergent' behavior is replaced by a more intelligent and deliberate design of the robot controller without sacrificing the quickness of traditional reactive architectures. Behavior-based architectures have the following traits:

- Coupling between perception and action is tight
- The complexities of forming and using symbolic knowledge are avoided
- Control is divided into meaningful behavioral units
- Designer has the latitude to decide on the granularity of behavioral decomposition, on the methods of behavioral fusion, and between discrete or continuous responses

The concept of behavior-based architectures has been adapted from research in neuroscience (the physiology of the nervous system), psychology (the study of the 'mind' and behavior), and ethology (the study of animal behavior in natural conditions), which is fitting since animals are living proof that intelligent motor control systems exist. These fields of study have propounded theories stating that the brain is divided into functional modules, with each module being responsible for a specific behavior. The challenge is that although behaviors are easily observed, they are not easily attributed to computational structures. Researchers tend to explain animal behavior in two different ways: through neural networks, and through schema theory.

Neural Networks

The neural network approach models behaviors as the responses generated by highly interconnected, highly distributed groups of individually simple computation devices (i.e. neurons). This approach essentially decomposes behavior with very fine granularity. Each simple computation device (called a 'node') can take in multiple input signals, aggregate this sensory evidence using weighted additions, and make a decision (or classification) using threshold curves. This technique mimics the way nerve cells are operate.

The benefit of this approach is that the topology of a neural network lends itself to heuristic or fuzzy reasoning. Roboticists and researchers in *connectionist* AI try to synthesize behaviors in software agents and in robots by using artificial neural networks (ANNs), which are modeled after the connections of nerve cells in lower animals, and can be trained (in some cases) to learn correct responses to a situation (i.e. learn *behaviors*). An example of ANNs in mobile robotics is Carnegie Mellon University's 'Autonomous Land Vehicle In a Neural Network' (ALVINN)

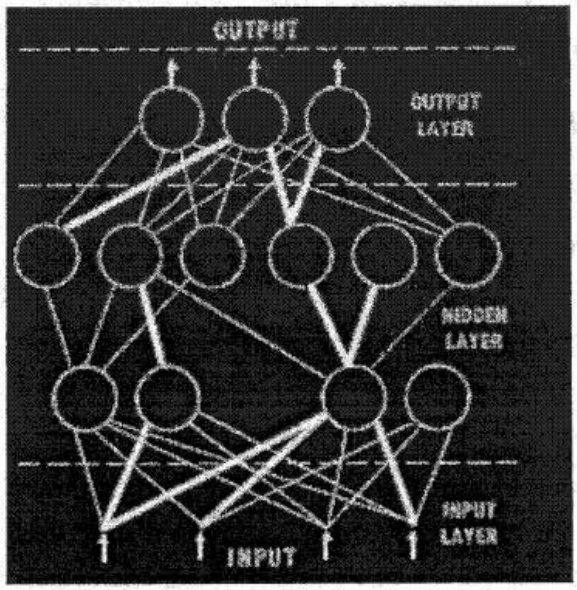


Figure 3.11: Artificial Neural Network [8]

project[3][51]. Weitzenfeld and others have also used ANNs to implement a robotic toad's prey acquisition model, which was then incorporated into non-ANN control system [52]. While artificial neural networks can be useful for robot learning, they have the following drawbacks:

- Can be complex
- Not suited for direct hardware implementation, so must be calculated iteratively in software
- Computationally expensive (as a result of the above point)
- Effective training of ANNs is a research area unto itself
- 'Opaque': The experience of such an ANN is cryptically embodied in its topology, structure, weights, and thresholds. They compute reactions, but not necessarily in an intuitive manner.

[3]The ALVINN project is essentially a trained machine vision system that is given a 30x32 pixel image and outputs a steering direction graded from left to right in thirty discrete increments.

Schema Theory

An alternative to the fine-grained connectionist AI approach is to use schema theory. Decomposing behaviors into modules with a far coarser granularity than ANNs, schema theory categorizes sensory perception as inputs into well-defined behavioral modules called schemas, with each module running concurrent to each other.

> *Schema*: the basic unit of behavior from which complex actions can be constructed [7].

Schema theory has the benefit of giving the robot designer control over how to subdivide robotic behaviors into schemas, and puts no restriction on how the modules should be implemented or combined. In fact, some researchers have implemented behavioral modules using simple ANNs [52] [53] [54], while others have used more traditional programming methods for schema implementation [44].

Subsumption Architecture

The first widely recognized move towards behavior-based control architectures in robotics was the 'Subsumption Architecture' proposed by Rodney A. Brooks of MIT, in 1986 [44]. Subsumption controllers made use of behaviorally-specific reactive modules, assigned a fixed priority level to each module in terms of overall behavioral importance, and fused the output behaviors of these modules by having the highest-priority behavior become the only output behavior through suppressing lesser behaviors and even inhibiting certain system inputs. This type of behavioral fusion used makes the subsumption architecture a behaviorally competitive architecture.

Motor Schema Architecture

Motor Schemas proposed by Ronald C. Arkin [7] also made use of behaviorally-specific modules but, unlike the subsumption architecture, made no mention of the modules being only reactive, and took a cooperative rather than competitive approach to behavioral fusion. Since the behavioral modules (i.e the schemas) output vectors corresponding to the strength and type of actions to be taken, cooperative

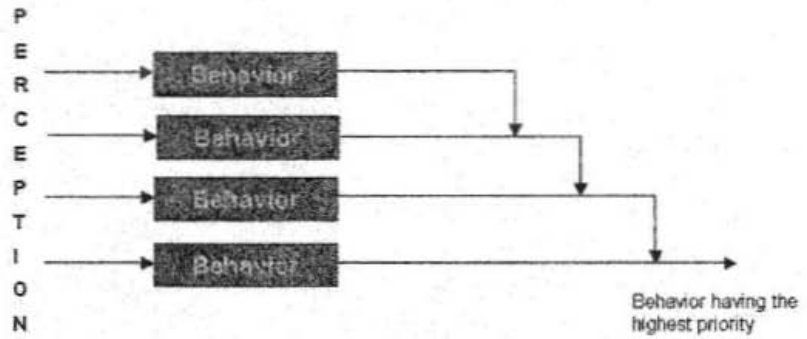


Figure 3.12: Subsumption Structure. Figure adapted from [7].

fusion was achieved by summing and normalizing these vectors to create actions for the robot to execute.

Motor schemas have the following properties:

- Perceptions (perceptual schemas) can be derived from a single sensor or from the fusion of multiple sensors
- Perceptual schemas can be recursively defined over other perceptual schemas
- Behaviors are computed by motor schemas, each of which requires at least one perceptual input
- Motor schemas run concurrent to each other
- Behavioral fusion is achieved through vector addition and normalization

The fact that the outputs of the motor schemas are vectors permits the robot designer to model behaviors in terms of potential fields. The motor schema controller does not internally represent or calculate potential fields, but field diagrams help designers visualize the effects of combined motor behaviors. Examples of this visualization technique are shown in figures 3.14, 3.15, 3.16, 3.17, and 3.18.

One well-documented drawback of using summed potential fields when fusing behaviors together is that there can exist local minima in the summed field that can possibly trap the robot at such locations, either causing an overdamped robot to halt,

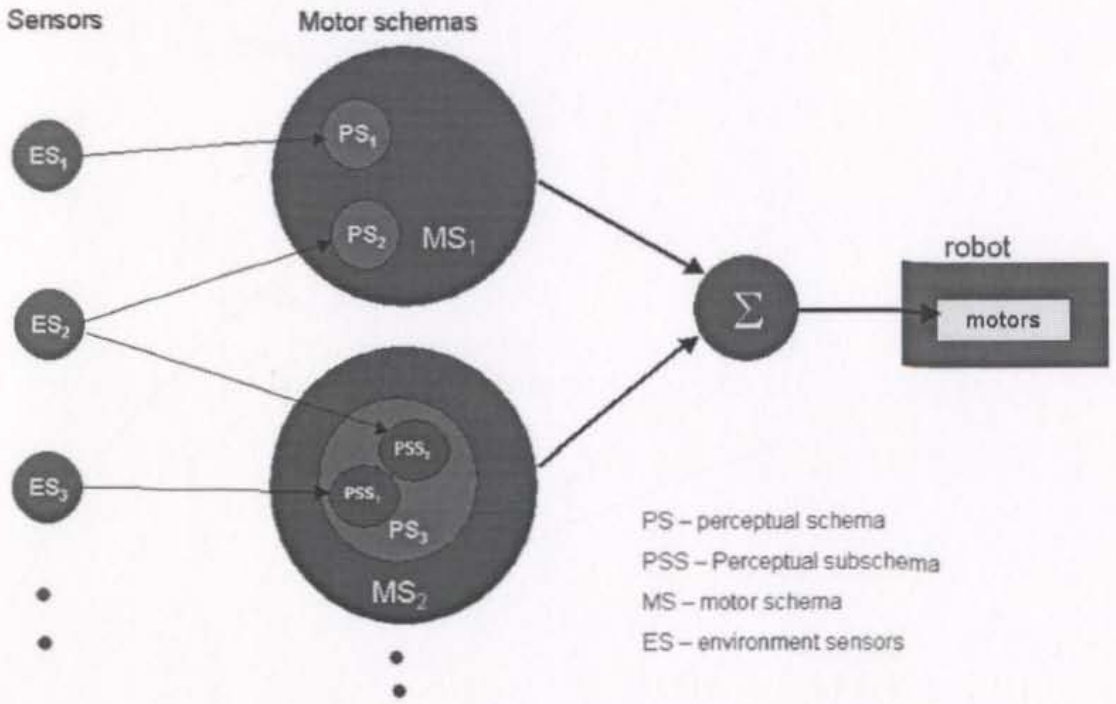


Figure 3.13: Motor Schema Structure. Figure adapted from [7].

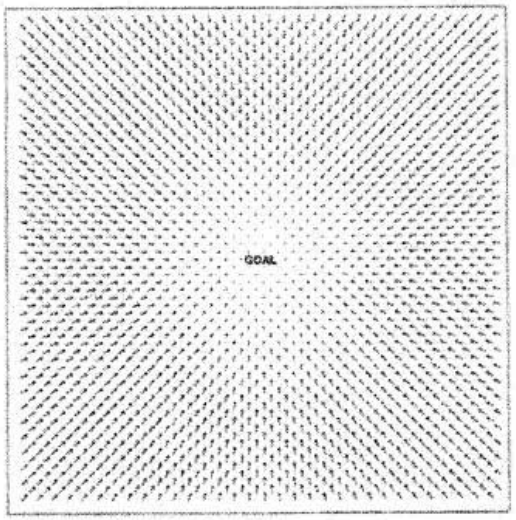


Figure 3.14: Attractant Field - Go Towards Goal [7]

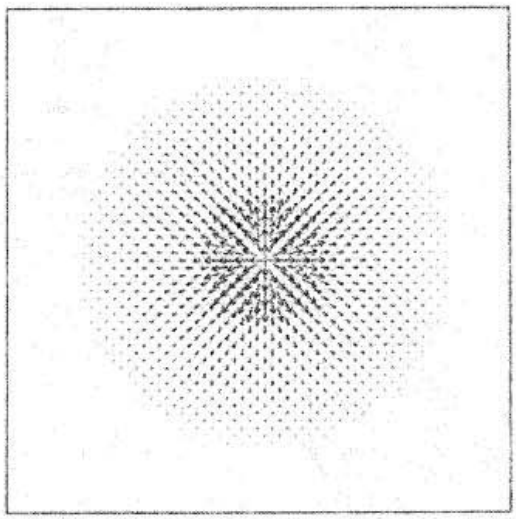


Figure 3.15: Avoidance Field - Avoid Obstacle [7]

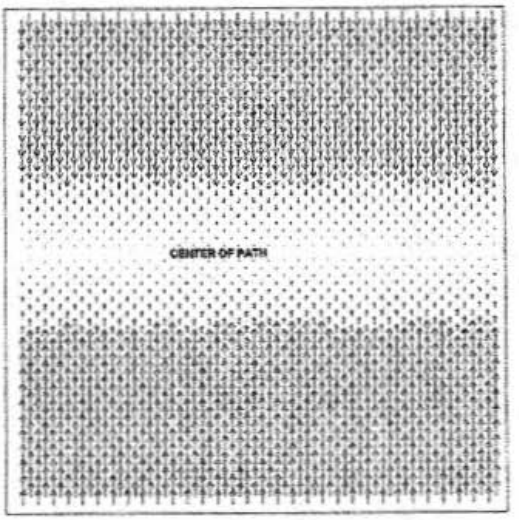


Figure 3.16: Path Field - Stay on Path [7]

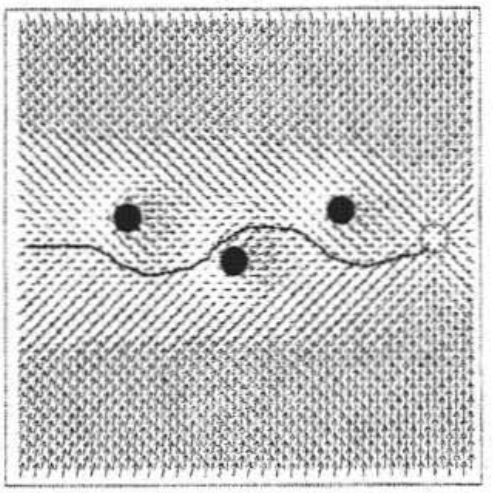


Figure 3.17: Summed Fields - Overall Motion Guidance Field [7]

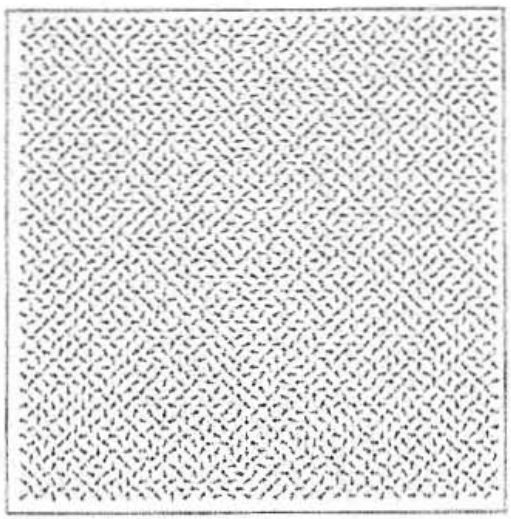


Figure 3.18: Random Noise Field [7]

or causing an underdamped robot to get caught in an oscillatory path. To avoid this "local minimum trap" problem, Borenstein [55] proposed to monitor the difference between the robot's instantaneous heading and the bearing towards the goal, and if these differed by more than ninety degrees, then the potential fields would be ignored and a wall-following behavior would take over. Arkin [7] proposed another approach whereby a random-process field be added into the summed potential field so as to break up the formation of local minima in fixed places. The random-process approach fits better into motor schema theory, but Borenstein's trap detection test could be used to modulate behaviors, as will be discussed in the next subsection.

Behaviors Modulated by Intention

While subsumption architectures fuse behaviors by using a *pure arbitration* (winner-takes-all) approach, and motor schema architectures fuse behaviors by using normalized summation of vectors, these fusions are determined ahead of time by the designer and are expected to operate in various environments with equal ease. Yet

it stands to reason that one behavioral 'mix' may work better than another behavioral mix in a given situation. Instead of designing a robot with a fixed behavioral mix, using different behavioral mixes for different situations could result in better performance over a wider range of situations.

This concept can be stated as the modulation of behavior by intention. In other words, depending on what the robot's intentions are (e.g. to get to a goal, or to go along a corridor), the setting of behavioral gains (in the case of motor schemas) can be made to best suit the run-time requirements of the robot.

As mentioned in the previous subsection, Borenstein's method of avoiding local minima traps is to monitor the disparity between instantaneous heading and bearing towards the goal. When the disparity exceeds a set amount, the behavioral mix is changed (i.e. the behaviors are modulated) to follow walls instead of remain with a problematic behavioral mix. Once the local minimum trap is avoided, the behavioral mix can revert to the original settings to best carry on the robot's intentions. Unlike the subsumption architecture which has a predefined behavioral hierarchy, motor schemas modulated by intentions can provide a very flexible architecture more likely to be successful over a wide set of situations.

3.4 Chapter Summary

Mobile robots for research were historically made by researchers themselves, due to the specialized nature of computers in the past. With the commoditization of computer systems and the reduced cost of computational power, commercially available robots have appeared, ranging from small toy-like devices to rugged and powerful bases equipped with state-of-the-art sensor technologies. Despite the availability of ready-made robots, researchers still choose to build their own robots, often to automate existing vehicles or to avoid paying premiums on prefabricated robots. Among those who make their own rugged mobile robots, powered wheelchairs are a popular base on which to build them.

Robot intelligence and control is accomplished in software more often than by using hard-wired circuitry, and much work has been done in artificial intelligence as applied to robotics. A range of robot control architectures exist, from reflex-based *reactive* architectures to highly computational *deliberative* architectures and hybrid

architectures in between.

The available options for both a robot's physical system and its control system present a wide array of choices to the prospective robot designer. To make a wise choice among all the possibilities, design specifications first must be established. The specifications for the ISLab's mobile robotic platform are therefore presented in the next chapter.

Chapter 4

Specification and Design

The reasons for developing a mobile robot base were that the existing toy chassis robots were insufficiently equipped due to their small size to carry sizeable payloads, to operate for long periods of time, or to be equipped with all the sensors and manipulators desired on a mobile robot. This project's design has always been referred to as a 'base' because it is meant to be augmented with additional sensors and manipulators in a modular fashion to be useful for a number of robotic research pursuits, especially research in autonomous control of mobile robots.

Although the robot has been developed to fit into a group robotic system, design of the robot itself is not simply concerned with artificial intelligence programming, but is an exercise in system engineering, drawing upon aspects of mechanical engineering, electrical engineering, software design, as well as artificial intelligence programming.

4.1 Motivation

From working on aspects of the SMART robotics project, the ISLab was aware of the research being conducted in the Perception and Robotics laboratory at École Polytechnique in Montreal, Canada. As well as using miniature robots, the Montreal researchers used a commercially available all-terrain robot for conducting research in planning for and controlling autonomous robotic vehicles. The robot that the École Polytechnique researchers used was an ATRV-2 wheeled robot with differential steering, costing over sixty thousand Canadian dollars [56].

The ISLab wanted two robots with similar capabilities to the ATRV-2 but having a target cost of between five thousand and seven thousand Canadian dollars each. Since no rugged robots were available at this price it was decided to build the robots in-house. The overall goal of the project was to develop a mobile robot / sensory platform that would be used to advance the SMART project by providing a more realistic experience of tele-operation and remote sensing, as well as to provide means for developing and testing semi-autonomous robot behavior.

4.2 Requirements

The original requirements of the MRP focused on providing a platform that would move a suite of sensors around an environment in order to feed back data to a remote human operator. The ability to change the collection of sensors for different measuring purposes led to an early decision to make the MRP modular so these changes could be made simply. Since the design focus was toward robotic motion and perception, design considerations regarding remote handling capabilities, navigation, and mapping became secondary issues.

The requirements were divided into six categories: performance, electromechanical, sensory, communicational, computational, and safety.

4.2.1 Performance Requirements

The following design guidelines were meant to provide an overall performance expectation range for the MRP. The MRP was required to perform

- Indoor operation (e.g. along flat, open surfaces and connecting corridors[1]
- Outdoor operation (e.g. over uneven terrain, over small rocks and curbs, maximum slope of 30° for the base without payload)[2]
- Speeds from 0 to 3 m/s [3]

[1]Corridors should be at least 1.5m wide, excluding doorways which are typically 1m wide.

[2]The maximum slope of 30° was chosen as a guide to ensure sufficient motor power, not to indicate maximum slope before tipping (which would vary depending on how payloads would affect the base's center of gravity).

[3]The 3 m/s speed limit was set as a precaution to permit sufficient deceleration time / braking

4.2.2 Electromechanical Requirements

These requirements outline the necessities for the bulk of the MRP's construction: the chassis, motor, gearing, and steering system. The term *electromechanical* was used since electrical motors were more appropriate for indoor use than fume-emitting combustion engines. Requirements for this category were that the MRP:

- Be able to withstand a 20cm vertical drop (e.g. a drop off of a large sidewalk curb) without encountering major damage
- Have a turning radius of at most 50cm to be able to pass unimpeded through most indoor corridors while travelling in the forward direction.
- Have battery-powered electric drive motors for untethered and emission-free operation
- Prevent from moving when not powered, unless the preventative mechanism has been willfully disengaged.
- Have batteries that can be recharged using standard AC mains sources.
- Have motor controllers capable of producing speed variations in gradations of at least 1% of the maximum vehicle speed, to respond fluidly to teleoperative signals.

4.2.3 Sensory Requirements

The sensors are of great importance to the development of the MRP since they capture information necessary for robot navigation and map-building, but also collect data to aid in remote exploration and robotic process automation, such as the kinds mentioned in the SMART proposal [20].

The sensors related to the final robotic applications will change depending on the application, but the supporting sensors that allow for MRP navigation and teleoperation must be developed first. The supportive sensors therefore take precedence over the application-specific sensors in this design.

distance of a 100kg vehicle, given that the range and reaction time of obstacle sensors and processing was to be determined. This conservative upper speed limit was *not* based on power limitations of the drive motors.

The following sensory requirements include:

- Proximity sensors to detect obstacles and path boundaries
- Inclinometers to warn of tipping hazards
- Digitally-interfaced battery level monitor
- Camera-based vision system to aid in robot navigation

4.2.4 Communicative Requirements

For the MRP to operate within the General Framework for Group Robotics in an untethered manner the following communication requirements were set:

- Wireless Ethernet with a range of at least 300 meters (Ethernet being the most common and well-supported network transport layer, and 300 meters being the longest point-to-point distance likely to be encountered in an indoor laboratory setting).
- Minimum bandwidth of 1 Mbps for teleoperation and sensory feedback, excluding live video streaming
- Preferable bandwidth of 11 Mbps to allow for live video streaming
- Optional but preferable PDA (hand-held personal digital assistant) interface for quick vehicle diagnostics and simple offline vehicular control

4.2.5 Computational Requirements

For the MRP's onboard computer to be able to execute motion control software (including sensory data processing and motor control processing), application-specific software (including video image processing), and communication processes, a PC-based system was desired. This system was to be implemented as a module which could be easily upgraded or replaced in future MRP revisions, and be easily powered from electric battery sources. The requirements therefore were:

- Miniature and modular PC-compatible computer system (Pentium grade or higher) with a high-speed bus available for live image capture
- Video capture hardware (also called a *Frame Grabber Card*)
- Data acquisition hardware for collecting sensor inputs and providing motion control outputs

4.2.6 Safety Requirements

The battery-powered rugged MRP was expected to have an approximate mass of 75kg, and such a massive platform travelling upward of 3 m/s would have sufficient momentum to incur physical damage if it lost control and collided with an object. To avoid such a scenario these safety requirements were established:

- Large onboard push-button kill switch
- Electronic kill switch triggered automatically in case of a software fault
- Visual beacon to alert bystanders of the MRP being in operation
- Switchable beeper which could be enabled as an extra level of precaution

4.3 Specifications

Based on the MRP requirements identified in the previous sections, a further level of detail was specified. The MRP specifications were split into two parts: electromechanical design and controller design. Since the specification of the electromechanical portion was deemed as the most difficult part, it was specified first and was followed by the controller specification, and this section is presented in the same order.

4.3.1 Electromechanical Design

The task of specifying the electromechanical portion of the MRP required electric drive components to be sized, which further required estimates of vehicle weight and frictional forces to calculate power requirements for MRP motion. These requirements in turn would refine the vehicle estimates, and this estimation process would

iterate until the estimates seemed to match available electromechanical component descriptions.

Elementary vehicle designs were done in order to get a better sense of the specifications needed for the actual electromechanical design. A few vehicle configurations were proposed early in the design cycle. One important property was that the configurations be modular since modularity was considered essential to the robot's future usefulness.

Custom In-House Design

There always was the option to build a mechanical base rather than buy one commercially. This had the advantage of being able to concentrate costs on what was deemed important, rather than be bound by the choices of commercial designers. This alternative was more engineering-intensive and since there was no local expertise on designing such mechanical systems, educated guesses would need to be made regarding performance specifications and project timelines.

An elementary design was completed with analyses of electric motor requirements, mechanical frame and suspension properties and components, and vehicular layout, with consultation from an electric vehicle specialist over telephone and fax [57]. In brief, the following requirements were established:

- 24 volt direct current electrical system (safe terminal voltage)
- Two low-voltage DC motors, each having a maximum rating of at least 750W
- Rubber drive wheels, 40 cm in diameter
- Fixed-ratio motor gearbox, delivering high-torque (over 150 N-m) low revolution (under 50 RPM) performance
- Anticipated vehicle mass of 70 kg
- Maximum payload mass of 45 kg

Figure 4.19: ARGO Bigfoot Amphibious Vehicle [9]

ARGO Platform

During a review of rugged robot designs, it was found that the Intelligent Robotics Research Centre at Monash University, Australia, was using a rugged six-wheeled amphibious vehicle called an ARGO as the basis of an all-terrain intelligent autonomous vehicle [58]. The advantage of using an ARGO was that is was solidly constructed, with sealed axles and large tires for operating over rough terrains, and perhaps even in mining tunnels. The drawbacks were that new ARGO vehicles cost over $10000 new and the conversion from a gasoline-powered engine to an electric motor drive would introduce unwanted complexity to the MRP project. Furthermore, the size of an ARGO would prevent it from being used indoors during prototype development.

Golf Cart Platform

Electric-drive golf carts were another option when choosing an electromechanical base. Its advantage was that it had electric motor drives but it used conventional yoke steering as opposed to the preferred differential steering, and yoke steering would make computer-controlled motion very difficult to implement.

Electrically-Powered Wheelchair

A number of electrically powered wheelchairs (powerchairs) and scooters were considered as potential electromechanical bases, partially because their power-to-weight ratios rivalled those of electric golf carts. Scooters were generally less expensive than powerchairs, but often only had three wheels and used preferred manual handlebar steering. The powerchairs were all four-wheeled models using powered differential steering that could be electrically controlled. Furthermore, powerchairs were more rugged than scooters, and were rated to carry loads as massive as 135 kg, making it more than able to handle the payloads expected to be placed on an MRP. Both scooters and powerchairs ran off of deep-cycle rechargeable batteries. The cost of a scooter was around $3500 while the cost of a powerchair was closer to $5000.

4.3.2 Electromechanical Design Decisions

By process of elimination, the choice was made to go with the electrically-powered wheelchair, since it was locally available, locally serviceable, modular in its design, emission-free, useable both indoors and outdoors, rugged, affordable, and straightforward to adapt for robotic purposes.

The other alternatives were eliminated for reasons noted in the following brief subsections.

Custom In-House Design

After beginning with some educated guesses of component masses and terrain demands, price estimates were made, and electric drive controllers (power electronics) were searched for. By the end of this design alternative investigation, the cost of the electric drive components (batteries, motors, motor controllers) was expected to cost around $3500, without including the price of parts and labor for the vehicle's proposed metal frame construction. It was concluded that the time and effort needed to custom-build a platform outweighed the value gained in doing so. However, the benefit in analyzing this design alternative was that it helped in understanding the design requirements and practical limitations of an MRP. The estimated design details are presented in the appendix for completeness. Coincidentally, this early design

Figure 4.20: CMU Terregator in Underground Mine [10]

alternative bore striking resemblance (both in terms of specifications and looks) to the *Terregator* (see figure 4.20) developed at Carnegie-Mellon University in 1984 [10].

ARGO Platform

This design alternative was ultimately rejected because it would have required a formidable electromechanical retrofit in order to meet the proposed MRP vehicular specifications. Furthermore, these platforms were markedly more expensive than the other design alternatives and were too large and heavy to be worked on in a non-industrial university laboratory setting.

Golf Cart Platform

Preferable to both the previous design alternatives, the golf cart platform had the major drawback of using conventional steering, as opposed to the differential-drive steering used by the electrically-powered wheelchair. As well as having this handicap, the golf cart was more expensive than the electrically-powered wheelchair, and less accepting of electrical or mechanical modifications.

4.4 Controller Design

This section of the design was twofold: the design of the robot controller (including integration of sensors and actuators), and the design of the power electronics to drive the electric motors in the electromechanical base. As the robot controller design was less dependent on the electromechanical base design than was the power electronics design, the robot controller was the first portion of the electrical design that was considered.

4.4.1 Robot Controller Design Alternatives

The requirements for this MRP demanded the following from the controller: ability for high communication capacity, on-board sensor and video processing, and motion control. The communication and video requirements pointed to the use of a PC-compatible computer system. Since the above requirements were also the requirements for the SmallBot robot, this problem had already been discussed, and since the controller was to be place aboard the SmallBot, size and weight were further constraints to be looked at. During a brief period of time both the SmallBot project and the design of the rugged MRP were underway, and it was anticipated that the controller could be easily integrated into either platform. Therefore, the size and weight constraints were dictated by the SmallBot project. As a result, the PC/104 miniature personal computer standard was adopted for use as the "high-level" controller, i.e. the portion which calculated the robot's heading and speed. Since this controller was to be used with two different robots, the portion responsible for converting speed and heading coordinates into appropriate motor control signals, i.e. the "low-level" motion controller, was separated from the high-level controller and was designed differently for each type of robot platforms.

4.4.2 PC/104: High-Level Control

The PC/104 computer standard was functionally equivalent to a desktop personal computer except for its miniature size, modular architecture, and low power consumption. Comprised of small circuit boards, PC/104 components were made by a number of manufacturers and were used for embedded control primarily in heavy

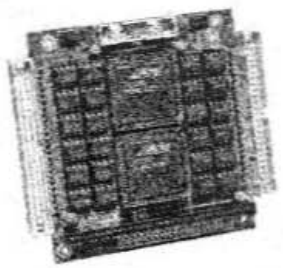


Figure 4.21: Typical PC/104 Module [11]

industries. The advent of the PC/104 "Plus" system improved bus speed, enabling smoother video capture and processing. Some features of PC/104 include:

- Self stacking modules (no separate backplane needed)
- Small form factor (3.6 in. x 3.8 in., per module)
- Low power consumption (typically 1-2 Watts per module)
- PC/104-Plus technology is compatible with PC/104 and supports 32-bit PCI interconnect, which allows for smooth video capture
- Over 150 companies are members of the PC/104 consortium

Another advantage of using the PC/104 standard is that many companies offer similar products, which means that modules are priced competitively. The modules, however, are usually two to three times more expensive than functionally comparable desktop PC cards since there is a higher level of miniaturization and power budgeting that goes into PC/104 modules and there is less of a customer base to distribute the R&D costs incurred by the PC/104 companies.

4.4.3 MiniBoard / HandyBoard: Low-Level Control

Already having been used by the ISLab for control of the toy robots (i.e. miniature bobcat vehicles), the MIT-developed MiniBoard was considered as a possible low-level controller for the MRP. The MiniBoard had the following features:

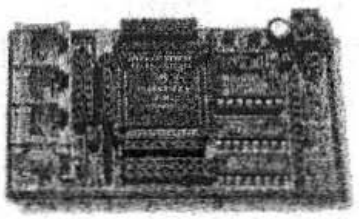


Figure 4.22: MiniBoard [12]

- Based on a Motorola MC68HC11E2 8-bit processor
- 2 Kb of EEPROM
- 256 bytes of RAM on chip
- 4 motor outputs using the LM293D bipolar H-bridge chip
- LEDs indicating motor state
- 8 A/D inputs (8 bit)
- 8 Digital Input/Output lines
- 8 General Input/Output lines

A significant drawback of the MiniBoard is the lack of RAM on which to keep control programs. The more endowed HandyBoard is based around the Motorola M68HC11 8-bit processor, and has 32KB of battery backed up RAM and 512KB of EEPROM. Both the MiniBoard and the HandyBoard make use of CPU chips that are no longer being sold, so choosing either of those solutions would be unwise from the perspective of robot maintenance and repair.

4.4.4 RAD Board: Low-Level Control

Initially envisioned as a cost-effective data acquisition card for teaching control systems to engineering undergraduate students, the RAD (Robotic Analog & Digital)

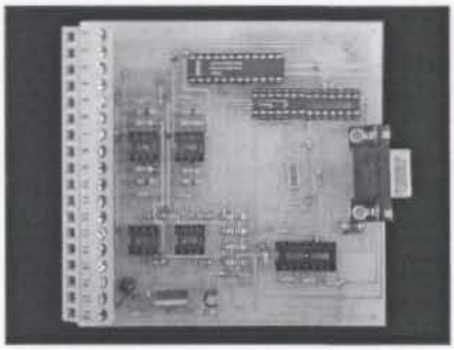


Figure 4.23: RAD Board

card was chosen as the hardware for carrying out low-level motion control on the MRP. The locally-designed RAD card [27] has the following features:

- Designed around a PIC 16F873 8-bit microcontroller
- 4 A/D inputs (8 bit)
- 4 D/A outputs (8 bit)
- Typical input scan speed of 1 kHz
- Serial RS-232 communication capabilities

The RAD card had enough features to be successfully used as an interfacing circuit between the PC-104 controller and the MRP's motor drive power electronics. Since some RAD card expertise already existed at the INCA (Instrumentation, Controls, and Automation) laboratory, basic existing control programs could be utilized, thereby speeding up the design cycle.

4.4.5 Power Electronics Design Alternatives

Preliminary research on power electronics for electrical motor locomotion was done in case the choice of mechanical base required separate motor drive circuitry to be procured.

Calculations that estimated the mechanical base motor requirements suggested the use of two DC motors (to permit differential steering) with an operational rating of one horsepower (approximately 750W) each. It was anticipated that the electric drive system would operate at a nominal 24VDC.

It was decided that the power electronics be purchased rather than designed, for reasons of inexperience in power electronics design, and for reasons of reliability. The commercially available DC motor controllers that met the proposed specifications above were the Curtis PMC MOSFET controller (model 1204), and the Dart DC motor speed controller (model 65E), both of which would control the speed of a single motor given a low power control voltage.

4.5 Controller Design

As was mentioned in section 3.3.5, the overall control system for the ISLab's group robotics system was a three-layered architecture that was comprised of a deliberative planning layer, a coordinative action-sequencing layer, and a layer responsible for the control of an individual robot – the layer that this section will now discuss[4].

According to the General Framework for Group Robotics, each robot in the system gets issued commands to carry out, but the robot is not told precisely how to carry out those subtasks. For example, the robot may be given the following commands:

- Navigate to the next waypoint
- Search for an object
- Follow a path

A number of architectures exist for controlling individual robots, but the one that was chosen to interpret and act upon the given commands was the *motor schema* architecture. This architecture was chosen because it allowed behaviors to be implemented individually, and combined in an intuitive additive fashion. Based on animal behavior theories, motor schemas work by subdividing the robot controller into a set

[4]Figures 5.26 and 5.46 illustrate the way the controller was implemented, and may provide insight to this section.

of concurrent subsystems that each react in specific ways to specific stimuli. The reactions of the subsystems are then summed to create the robot's behavior given its circumstances. Examples of combined behaviors are presented in figure 4.24.

This architecture was to be implemented on a PC-compatible computer system in software so as to provide the developer the flexibility and quick development turnaround that would not be afforded if a strict hardware approach was taken. As is illustrated in figure 3.13, the motor schema architecture was comprised of objects of types *Environmental Sensor*, *Perceptual Schema*, and *Motor Schema*. Ronald C. Arkin, proponent of this control architecture, stated in [7] that "Schemas can be instantiated or deinstantiated [sic] at any time based on perceptual events", which logically led this specification to be stated in terms of object-oriented design.

The specification of the motor schema software is presented here in terms of base classes and derived classes, as well as their proposed interconnections. The terminology used is based on *Hungarian Notation* commonly used when programming with Microsoft Foundation Classes (MFC), as has historically been done in the ISLab.

4.5.1 Concurrency

As was mentioned in section 3.3.6, behavior-based control architectures are premised on distributed modules running concurrent to one another. As a result, many objects used in the software implementation were to have the ability to run concurrent to other objects, while following the well-known tenets of concurrent programming (e.g. mutual exclusion, use of invariants, avoidance of deadlock). Thus, the ultimate base class, *CThread*, would provide future derived classes with the mechanisms necessary to operate concurrently. The purely virtual method *ThreadHandler()* would be overloaded by derived classes to encapsulate the instructions meant to be run in a concurrent manner.

4.5.2 Sensor Objects

Each physical sensor was to be abstracted as a sensor object, derived from the class *CSensorObject*, which itself would be derived from CThread. Each CSensorObject would contain an identification field, an adjustable sensor polling period, and a data accessor function which would pull data from a shared memory structure. The

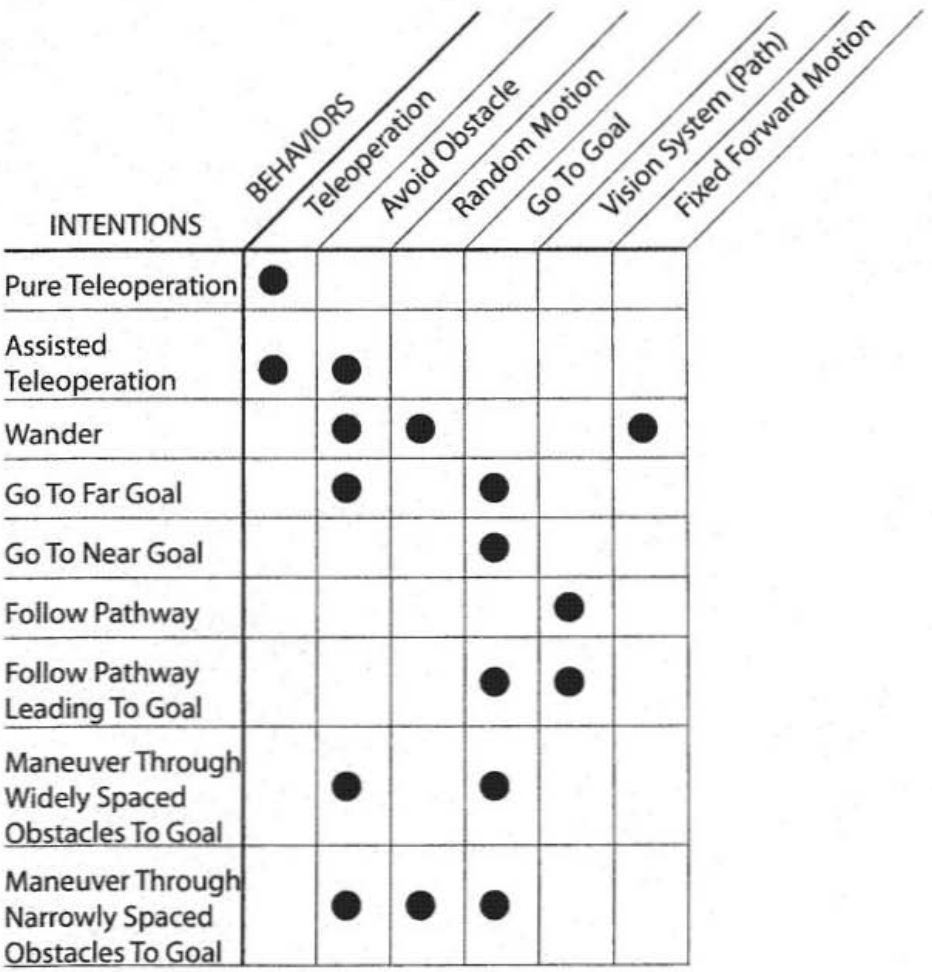


Figure 4.24: Behavioral Combinations for Various Intentions

CSensorObject would collect sensor data through a data acquisition (DAQ) member object, instantiated using the *Singleton* design pattern (since only one DAQ unit would exist in hardware, shared among multiple CSensorObject instances) [59].

4.5.3 Perception Objects

While the motor schema architecture states that perceptual schema be used to perform sensor fusion, which can then be fed into motor schema objects which will produce motion vectors[5] to be summed and finally sent to the robot motors, a slightly different approach was taken for the software specification. Rather than have so many layers of abstraction, perceptual schema objects were made to compute motion vectors given sensor object inputs; any sensor fusions would be implicitly performed. Thus, separate motor schema objects became superfluous and were therefore removed from the specification.

The perceptual schema object *CPerceptionObject* was meant to compute behavioral motion vectors, such as *avoid obstacle*, *random motion*, and even relay remote operator commands as *teleoperation*.

4.5.4 Motor Object

Labeled *CMotors*, this class was meant to be instantiated once for the purpose of providing an abstraction to the robot's motors. It accepts the motion vectors from perception objects, calculates the weighted sum of the vectors which are then scaled and shifted for output through a data acquisition card to the motor power electronics. The main method is the *MotionVector()* method which carries out the above specification, and the *Stop()* method which acts as an emergency stop in software. It is in this motor object where the behavioral mix could be altered depending on the intention of the system, e.g. depending on communication received from the discrete-event controller.

[5]Motion vectors were specified as $\vec{v} = \{(x, y) | x \epsilon \mathbb{Z}, y \epsilon \mathbb{Z}, [-100 \leq x \leq 100], [-100 \leq y \leq 100]\}$

4.6 Communications

Being a key part of the "General Framework for Group Robotics", communications needed to be included in the controller specifications. In terms of the "General Framework", what needs to be communicated to and from the robot is:

- Discrete-Event Task Commands (providing the robot with intentions)
- Robot Feedback (task completion status)
- Teleoperation commands
- Sensory Feedback (video, force, proprioceptive information)

All these communications needs could be met by using conventional local area network (LAN) technologies made very common and inexpensive due to the popularity of the Internet. While early communication development could be achieved by tethering an ethernet twisted-pair wire to the MRP, ultimately wireless communication could be achieved by using a wireless ethernet module (IEEE 802.11b) that would be transparent to all other devices on the network. Further discussion on the advantages of wireless ethernet for mobile robotics can be found in the literature [60].

While the higher-level implementation details (e.g. DCOM[6], CORBA[7], Java RMI[8]) were still being specified by other members of the ISLab, it was safe to make use of the well-known Transmission Control Protocol/Internet Protocol (TCP/IP) communication layer by means of a Windows *Socket*, for the following reasons:

- Windows was already decided to be the operating system on which to develop the software
- Sockets had been successfully used in applications within the ISLab
- Socket APIs were readily available as public domain software

[6]Microsoft's Distributed Component Object Model (DCOM) is a protocol which allows software modules to interoperate across a network.

[7]The Common Object Request Broker Architecture (CORBA) is an open-source protocol that allows software to interoperate across a network.

[8]Sun Microsystems' Java language allows Remote Method Invocation (RMI) whereby parts of a program are executed on networked remote computers.

Sockets abstracted communications programming over a data network, and a single instance of a socket could provide bidirectional communications. If needed, multiple sockets could be used to send and receive different types of information without necessitating any special parsing routines to sort between information types.

4.6.1 Commands and Responses

The commands issued to the robot by the discrete-event controller and the associated responses were not clearly defined at the time this specification was written, as it was still unclear whether message-passing (i.e. forming and parsing of text strings) would be used to communicate information, or if instead the communication software objects aboard the robot would be commanded using remote procedure calls (RPC's, as is done with various distributed object software architectures). It was anticipated that the commands issued to the MRP's behavioral controller, regardless of protocol, would have a low information rate. Examples of such commands are: go down corridor, go to detected goal, stop, and dead-reckon path.

Whereas the communication to the robot would have a low information rate, the communication from the robot back to the centralized control system would have a higher information rate, mostly due to the requirements demanded by teleoperation.

The most bandwidth-intensive aspect of the returning information sources is the video stream, which could be made to work by utilizing off-the-shelf video streaming software with built-in compression routines suited for WLAN information capacities or lower [61]. The remaining sources of information could be transmitted at a comparatively low data rate (likely 2 kbps or lower) while still achieving the demands of a teleoperation workstation.

4.7 Chapter Summary

This chapter presented a specification of the proposed ISLab mobile robotic platform, in terms of both the physical design and the control system design.

The physical portion of the MRP was specified to be a rugged indoor and outdoor modular platform. Safety features, sensor types, as well as computer and communications hardware were discussed. Design alternatives for the electromechanical base

were considered, ranging from custom designs to retrofits of commercially available vehicles. Ultimately an electrically powered wheelchair base was chosen.

The specifications of the MRP's computational design was presented, specifically the motor schema control architecture and the software implementation requirements (see figure 4.25).

All of the specifications provided in this chapter served as a guide for the implementation of the mobile robotic platform, which is discussed further in the next chapter.

Design Aspect	Specification Listing
Performance	Indoor operation: widths: 1.5m corridor, 1m doorway Outdoor operation: 30° max. slope, 3m/s max. speed
Electromechanical	20cm vertical drop Turning radius $\leq$ 50cm 24VDC, 750W permanent magnet motors Gearbox output of $\geq$ 150N-m at $\leq$ 50RPM Anti-roll mechanism Batteries rechargeable from standard AC source Motor controller with $\leq$ 1% speed control precision
Sensory	Proximity sensors for obstacle avoidance and boundary detection Inclinometer to indicate tipping hazards Battery voltage-level monitor
Communicative	Wireless Ethernet, range $\geq$ 300m, preferred bandwidth of 11Mbps Communication using TCP/IP sockets
Computational	Miniature and modular PC-compatible system (Pentium of higher) Video capture hardware Data acquisition / signal output (DAQ) Hardware
Safety	Large on-board pushbutton kill switch Electronic kill switch triggered in case of a software fault Visual beacon Switchable beeper
Controller	Object-oriented software design to achieve concurrency Motor schema approach Concurrency class (mutual exclusion of shared resources) Sensor class (ID field, adj. sensor polling period, data accessor) Perception class (computes behavioral motion vectors) Motor class (Converts motion vectors to motor signals)

Figure 4.25: Summary of Specifications

Chapter 5

Implementation and Demonstration Results

This chapter will present the design modifications that were made during construction, and the first demonstration results.

5.1 Robot Implementation

The implementation phase of the mobile robotic platform began with the choice of its name: SAMBUCA[1]. The name, which refers to the popular Italian black licorice liqueur, was chosen because the robot was predominantly black in color (reminiscent of black licorice), and because the name provided a fitting acronym.

SAMBUCA was designed around an electromechanical base, with motor control, sensors, onboard intelligence, and communication systems added afterward. The description of SAMBUCA's implementation will hereto follow the same sequence. A diagram of SAMBUCA's system interconnections is provided in figure 5.26.

5.1.1 Electromechanical Base

The commercially-available powerchair that was chosen as the electromechanical base was the "Invacare Pronto R2" without the chair portion, at a cost of $3600. This powerchair came equipped with two deep-cycle sealed lead-acid batteries, a battery

[1]Semi Autonomous Mobile Base and Utile Control Architecture, pronounced *zam-boo-kah*.

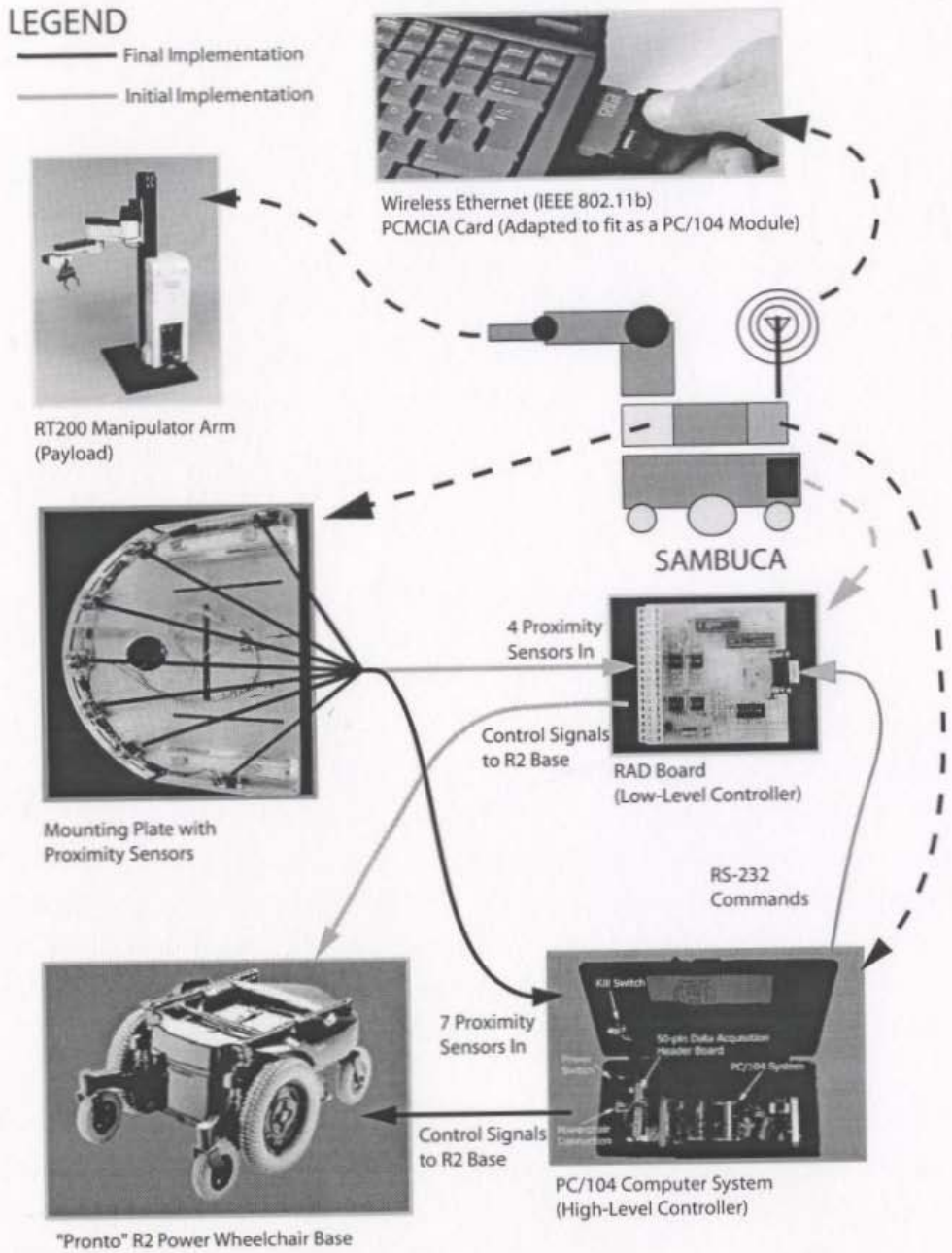


Figure 5.26: SAMBUCA System Interconnection

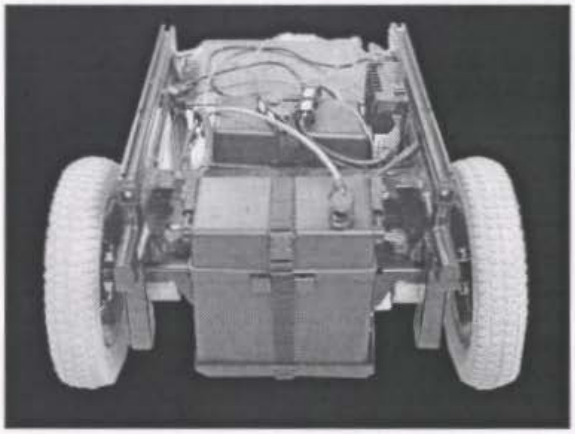


Figure 5.27: Invacare Pronto R2 Base

recharger, a motor controller, and a control joystick. Each battery produced a nominal EMF of 12 Volts, and the two were wired in series. The motor controller (which gives the powerchair its "R2" designation), was manufactured by Dynamic Mobility, New Zealand, and is an older analog controller (as opposed to the newer digital motor controllers). The analog (R2) controller was chosen because it was easier to interface to external robot controller electronics. The R2 controller is factory-preprogrammed to exhibit smooth forward and reverse motor accelerations. These preprogrammed accelerations can be altered using an optional $800 device, which merely communicates the settings to the controller via a digital serial line. This fine-tuning feature was deemed non-essential to the operation of the robot, and so the motor controller remained with its original factory settings intact.

The Pronto R2 powerchair was designed to be steered through a joystick interface (see figure 5.28) connected to the powerchair's built-in motor control circuitry. The joystick acted as a two-axis variable voltage divider, and the orientation of the joystick determined the two control voltages that were sent to the motor control circuitry, with the centered joystick position outputting 2.5V on both control lines. There were fifteen control lines overall that connected the joystick to the powerchair, and these are presented in figure 5.29.

Joystick Control	Robot Motion
Full forward direction	Maximum forward motion
Full reverse direction	Maximum reverse motion
Full right direction	Maximum clockwise rotation
Full left direction	Maximum counterclockwise rotation
No deflection (centered joystick)	Stopped
Throttle fully counterclockwise	Maximum speed set to lowest value
Throttle fully clockwise	Maximum speed set to highest value

Figure 5.28: Steering Response from the Pronto R2 Joystick

Pin	Signal	Voltage Levels (VDC)
1	Switched DC Power	24 (Switched)
2	Unswitched DC Power	24 (Unswitched)
3	Signal "Rail" Voltage	5
4	SPDM (unused)	Not Applicable
5	SPD	Max Reverse: 0.75 Stop: 2.50 Max Forward: 4.75
6	DIRM (unused)	Not Applicable
7	DIR	Max CCW: 0.75 Straight: 2.50 Max CW: 4.75
8	S-POT (maximum speed knob)	Slow: 0 Intermediate (0 to 5) Fast: 5
9	Green LED	5
10	Red LED	5
11	BATT-	GND
12	BATT-	GND
13	BATT+	24
14	BATT+	24
15	Serial Digital Communications	**Not used.** Meant for factory tuning of the motor drive circuitry parameters

Figure 5.29: Control Lines from the Pronto R2 Joystick [13]

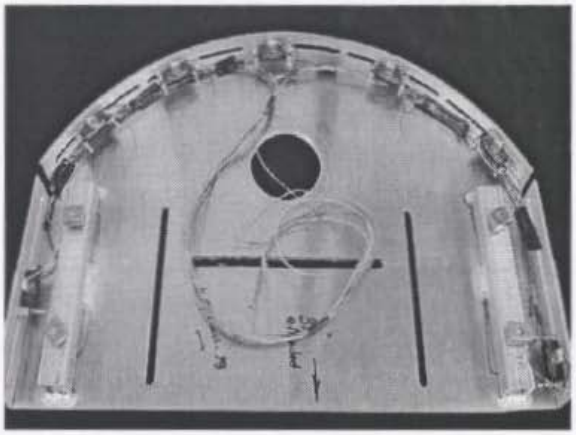


Figure 5.30: Underside of Mounting Plate with Infrared Proximity Sensors

Mounting Plate

Since the powerchair was to be unmanned, instead of having a chair atop the base, a mounting plate was designed and built in order to seat on-board sensors, manipulators, controllers, and computers. The mounting plate was designed to slide into the slots where the chair would normally go. The first plate design was made to hold the RT200 robotic arm fairly low to the ground, and was fabricated by the Technical Services division of Memorial University using aluminum sheet metal. A second mounting plate design was made to house infrared sensors as well as provide a mount for the robot arm higher off the ground (see figure 5.30). This second mounting plate has an accompanying hand-drawn design sketch which is included in the appendix, figure A.59.

5.1.2 Sensors

Proximity Sensors

Methods of obstacle detection were considered early in the design of SAMBUCA, with the primary concern being collision avoidance for safety reasons. The first sensor tested was the 6500-Series Sonar Ranging Module manufactured by the Polaroid Corporation [62] (see figures 5.31 and 5.32. Although this sensor detected obstacles

as far away as 35 feet, each sensor required numerous interface wires[2] to be connected to the RAD card unit, which used up scarce I/O lines. Furthermore, the ranging module required the RAD card unit to dedicate one of its few internal timer units to measure the time between the 'INIT' and 'ECHO' signals being asserted, since the ranging module did not directly provide a distance measurement value.

Since using more than one sonar ranging module would require more I/O and timer resources than were available, it was decided that more simple sensors be used for safety-based obstacle avoidance. The sensors that were chosen were the Sharp GP2D12 infrared sensors since they simply output an analog voltage in non-linear proportion to the proximity of an obstacle. A calibration formula could then be applied to this voltage level to resolve distance with centimeter accuracy. The infrared sensors did not have as much range as the sonar sensors, but still had a usable range of 50cm, which was suitable for safety obstacle avoidance purposes at low speeds (1.5 m/s and slower).

Wheel Encoders

For robots to be able to map and navigate within an environment, it is necessary to have a localization system on board. It was decided that SAMBUCA use electro-optical wheel shaft encoders to read the angular displacement of both drive wheels. Such readings can be processed to provide estimated path information, as well as speed and heading information. The sensors used were the Agilent Technologies HEDS-9100-G00 "Two Channel High Resolution Optical Incremental Encoder Modules", in conjunction with 11mm-radius codewheels having an angular resolution of 360 counts per revolution. These sensors provided digital pulses in quadrature with

[2]The interface wires to the Polaroid 6500 Ranging Module were:

- V+ (+5VDC)
- GND
- OSC (Oscillator signal input)
- INIT (Begin measurement)
- BLNK (Blanking signal)
- BINH (Blanking inhibit)
- ECHO (The returning signal)

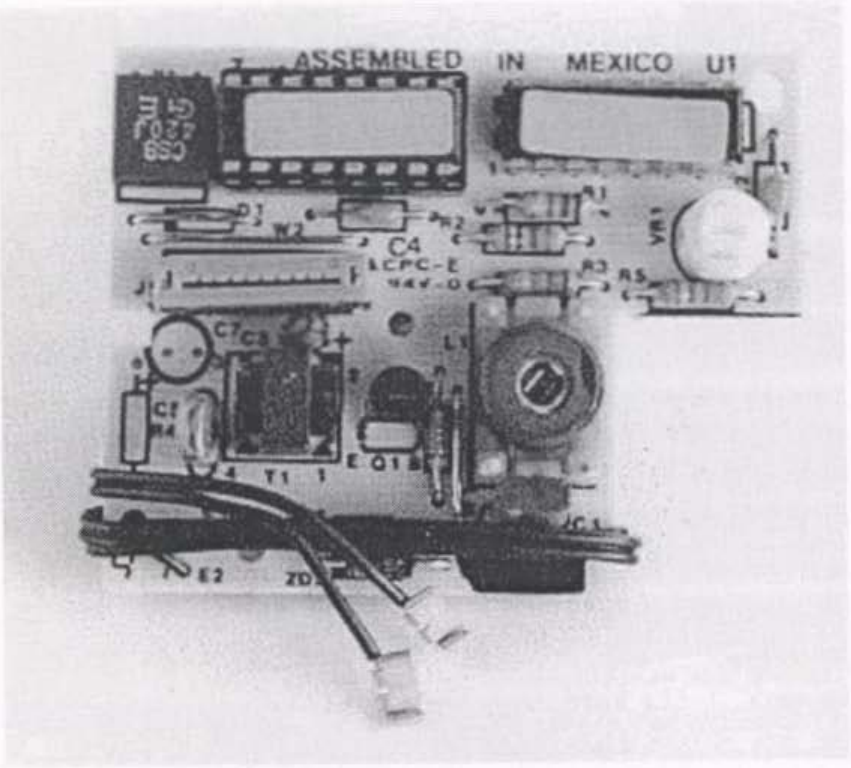


Figure 5.31: Polaroid 6500 Rangefinding Module [14]

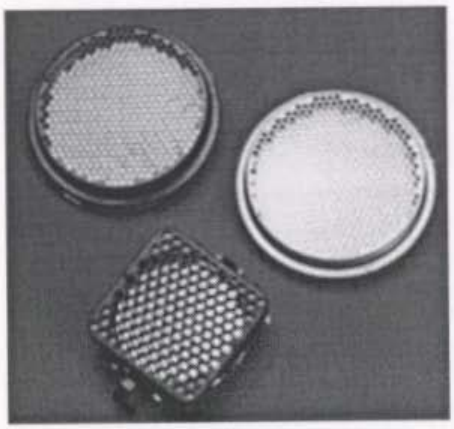


Figure 5.32: Sonar Transducers [15]

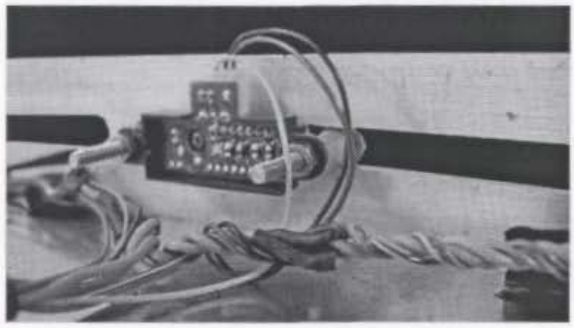


Figure 5.33: Infrared Proximity Sensor Mounted onto Plate

each other, permitting angular displacement and direction to be deduced using a straightforward digital state machine.

These wheel encoders were mounted on the each of SAMBUCA's drive shafts, on the end of the shaft opposite the tires. The codewheel was connected to the drive shaft by tapping into the drive shaft and then screwing in a mounting post upon which the codewheel sat. While there were concerns that such a mount would cause the codewheel to be off-center and cause sensor errors due to rotational wobble, the sensor which read the codewheel was sufficiently robust to mitigate this issue. Since less than 15mm of clearance between the drive shaft and the robot's batteries, detailed hand drawings were made for the technicians to follow (see appendix figure A.60). Furthermore, the encoders has to be sealed from dust, water, and other debris that the underside of SAMBUCA would likely be exposed to. A metal enclosure was created for this purpose, and was temporarily friction-fitted into place, with the intent of using epoxy sealant as a more permanent solution.

The both the left-wheel and right-wheel encoders were wired in the following manner:

Function	Wire Color
$+5V(V_{cc})$	Orange
GND	Grey
Channel A	Brown
Channel B	Purple

5.1.3 Controller Hardware

PC/104

As was mentioned in section 4.4.1, the computing hardware was divided into two categories: the low-level motion control and the high-level intelligent control. It was decided that a PC/104 miniature computer system be used to carry out the high-level intelligent control because of its low power consumption, its ability to perform nearly as well as a desktop PC, and its modular card structure. The following cards were purchased from various vendors, and assembled as SAMBUCA's first high-level control hardware.

- CPU: Ampro CM3-P5e, 266 MHz Pentium II, 64Mb RAM
- Video Capture: Imagenation PXC200 Color Frame Grabber
- VGA Display Card: Advanced Digital Logic MSMVGA104
- Power Supply: Tri-M Engineering HE104, 50 Watts
- Data Acquisition Card: Diamond MM-16-XT, 16-bit Analog I/O, 100000 samples per second over DMA
- Hard Drive: RTDUSA CMT106 6.0Gb hard drive module
- PCMCIA conversion card: M.K. Hansen dual-slot adapter

While each individual card met the specifications laid out for the high-level controller hardware, problems arose when the components were integrated with each other. A critical problem encountered was getting the operating system installed on system since the Ampro CM3-P5e central processing unit (CPU) module's basic input/output system (BIOS) did not permit the CD-ROM and hard drive from being utilized simultaneously, thereby making it impossible to install a Windows operating system from CD onto the hardware. The operating system was eventually loaded onto the hard drive by connecting the hard drive to a second PC/104 system[3] which was not affected with this BIOS limitation[4].

[3]The second PC/104 system belonged to a separate development group.

[4]This second PC/104 system was built around a CPU module made by the Parvus Corporation.

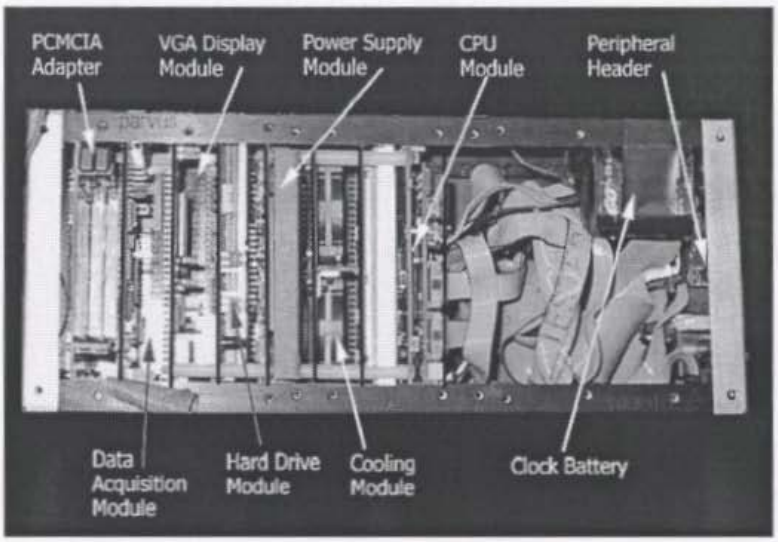


Figure 5.34: PC/104 Hardware

After the operating system was successfully installed the PCMCIA adapter card, which provided a means of networking the PC/104 system using wither wired or wireless ethernet cards, failed, therefore cutting out an essential data communication pathway. As well, the VGA card did not work under Windows 98, so Windows 95 had to be used[5]. There were reliability problems with the PC/104's power supply also, so it was decided that with this list of problems, an alternative be found for the high-level controller hardware.

The second PC/104 system that was used to help load the operating system became the new main controller for the MRP. The Parvus-made CPU card had both built-in ethernet connectivity and VGA support but was still limited to running Windows 95[6]. Parts of this second PC/104 system were matched with parts from the first PC/104 system, resulting in a system that worked properly (see figure 5.34).

[5]Windows 98 was the most current operating system available at that time.

[6]The Windows 95 limitation was not due to technical shortcomings of the hardware, but because the other development group needed that OS to remain installed.

RAD Pin	RAD Description	R2 Pin	R2 Description
18	BATT+	1	Switched 24VDC Power
10	Analog Out 0	8	Maximum Speed Setting
9	Analog Out 1	5	Fwd/Rev Speed
8	Analog Out 2	7	Turning (CW/CCW) Speed

Figure 5.35: Interconnect between the RAD Board and the R2 Power Wheelchair Motor Drive Circuit

Low-Level Controller

During periods of uncertainty troubleshooting the PC/104 system, efforts were often redirected toward getting the low-level control hardware working. The low-level controller on the SAMBUCA mobile robot base was designed to implement motion commands received from an RS-232 serial communication source. This controller was based on the locally-designed RAD (Robotic Analog & Digital) board [27], which itself was based around a PIC16F873 RISC microcontroller. The RAD board provided:

- an RS-232 serial communication interface
- an analog voltage interface to the R2 motor drive circuitry of the power wheelchair base[7](see figure 5.35)

The RAD board was programmed using PIC assembler code to have the following functions:

- Motion Control

  Motion control of the powerchair mechanical base was the main function of the low-level controller. The base's forward and backward motion, its rotational motion, and its setting for maximum allowable speed were controlled.

- Motion Inhibition

  The low-level controller monitored up to four Sharp GP2D12 infrared proximity sensors to detect obstacles and therefore inhibit hazardous motion commands.

[7]The R2 circuit should be thought of as a motor driver circuit rather than a robotic controller. The R2 circuit takes in low-voltage control signals from the RAD card and provides the appropriate high-power signals to the motors.

Pin	Schematic Name	Function
1	DIO3	Not Used
2	DIO2	Not Used
3	DIO1	Not Used
4	DIO0	Not Used
5	No Connection	Not Used
6	No Connection	Not Used
7	Analog Out 3 (DAC4)	Not Used
8	Analog Out 2 (DAC3)	Rotational Control
9	Analog Out 1 (DAC2)	Speed Control
10	Analog Out 0 (DAC1)	Set Maximum Speed
11	Analog In 3	Sharp IR proximity sensor
12	Analog In 2	Sharp IR proximity sensor
13	Analog In 1	Sharp IR proximity sensor
14	Analog In 0	Sharp IR proximity sensor
15	No Connection	Not Used
16	+5VDC (regulated)	For powering the IR sensors
17	GND	
18	BATT+ (typically +24VDC)	For powering the RAD board

Figure 5.36: RAD Board Connection Descriptions

These sensors were optional, and polling of each and every one of these sensors could be switched on or off.

- Temporary Validity of Commands

  An additional safety feature of the low-level controller was the "temporary validity" of all motion commands (except for the stop command which always had immediate and overriding precedence). The "temporary validity" safety feature allowed an issued command a lifespan of 1 second or less. If the same command was re-issued within that validity-timer span, the validity-timer was reset and the 1-second countdown restarted. The reason for implementing this behavior was to shield the robot base from navigational system crashes and communication lapses. A functional navigation system would pump out commands periodically in order to keep the robot base moving. In the case where these commands would not get re-issued in time, the robot would come to a halt in 1 second or less.

As can be seen in figure 4.23, the RAD Board had eighteen connections off of its header. Each connection is described in 5.36, as it pertained to the operation of the low-level controller.

The original low-level controller software accepted ASCII command strings via asynchronous RS-232 (serial) data transmission, and executed those commands. The

Command	Description
b, (ASCII 0-127)	Moves robot backward
f, (ASCII 0-127)	Moves robot forward
l, (ASCII 0-127)	Spin counter-clockwise (left)
r, (ASCII 0-127)	Spin clockwise (right)
s, (ASCII 0-255)	Set maximum allowable speed
x, *no argument necessary*	**STOP**

Figure 5.37: RAD Board Motion Command List

RAD card output control voltages, which were then fed into the powerchair base. The card was also configured to monitor proximity sensors so that the base did not collide with obstacles.

Commands could be issued to the low-level controller via an RS-232 cable whose bitrate was adjustable, but was typically set at 9600 bps, no parity, 8 data bits, 1 stop bit. The syntax of the commands were $< commandbyte, argumentbyte >$ where the argument byte was an ASCII character interpreted as a level from 0 to 255.

Each command was active for *at most* 1 second except for the STOP command.

Commands were as follows:

The RAD-based low level controller was successfully used as part of a senior term project by Baxter Smith in the winter of 2001. While discussion of this test is deferred for a later section of this thesis, it should be mentioned here that the following results were observed:

- Safety timeouts worked as planned
- Proximity sensor ring was unreliable due to insufficent sensor coverage
- Occasional commands would be ignored by the low-level controller

As a result of these tests, it was recommended that:

- Proximity sensors report a measurement rather than an alarm
- More proximity sensors be used
- Motion commands be augmented beyond just the four compass-points.

- Command reliability be improved
- PC/104-based vision system not be pursued further

The reason for not pursuing a PC/104-based vision system was because an easier method for implementing visual capture & analysis was discovered in the interim. As reported in [63], a webcam could send video to a PC or laptop running Intel's OpenCV vision libraries, resulting in acceptable performance at a fraction of the cost (in terms of money, effort, and time) of proposed PC/104 solutions. Due to this observation and the need to upgrade the low-level controller, members of the ISLab group were in favor of retiring the RAD-Board altogether and using the PC/104 system (specifically its data acquisition board) as a low-level controller. The advantages of doing so would be:

- Increased number of sensors accessible to the low-level controller
- Finer control of vehicular motion
- Programmable in C instead of PIC assembler

The drawbacks of switching to a PC/104-based low-level controller would be:

- No command timeout safeguard built-in
- Not likely to have PC/104 low-level controllers on future MRPs, due to excessive cost
- Lower reliability than RAD-Board

In the end, since the ISLab group would be using the SAMBUCA base in the future, their suggestions were implemented and the RAD-Board was retired. This change in hardware architecture required the software to be rethought as well.

PC/104 Enclosure

Since SAMBUCA made use of a number of electronics and sensors, there were many wires that needed to be disconnected and reconnected during development and regular use. To prevent electrical accidents (e.g. short-circuits, mismatched wires), a

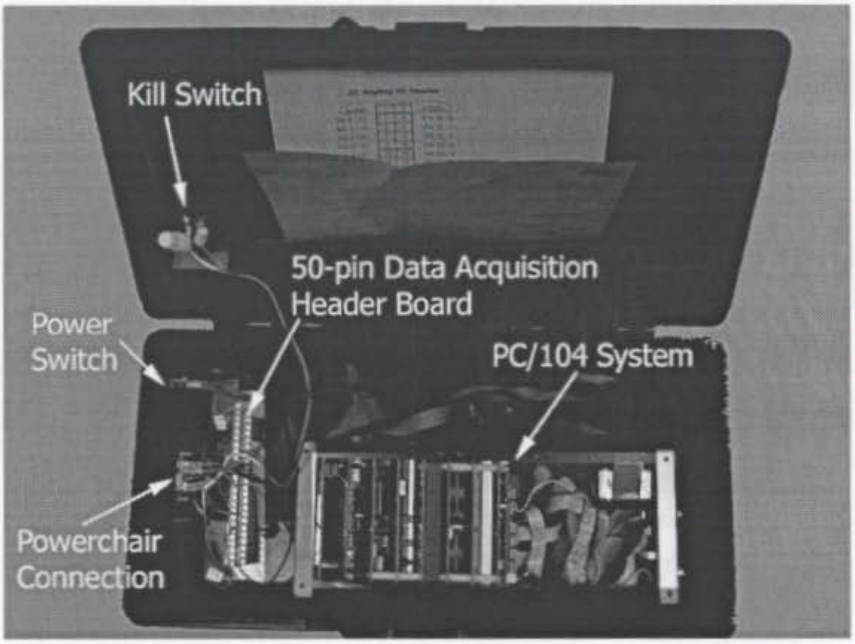


Figure 5.38: Opened PC/104 Enclosure - Top View

junction box was created using DB-9 and DB-25 connectors, switches, and a break-out board for a 50-pin IDC[8] ribbon cable, all enclosed in a black plastic toolbox (see figures 5.38 and 5.39).

5.1.4 Controller Software

The decision to not use the RAD board resulted in adding extra functionality to the PC/104 controller software. The PC/104 controller had previously been responsible for sensory processing (including video processing), wireless communication, robot intelligence, and abstracted motion control. Not having the RAD board to convert the abstract motion control commands into vehicle-specific control signals, the PC/104 system was given this responsibility and the control software was revised accordingly.

The software running on the PC/104 system aboard the SAMBUCA MRP was

[8]Insulation Displacement Connector, used to connect devices with ribbon cabling.

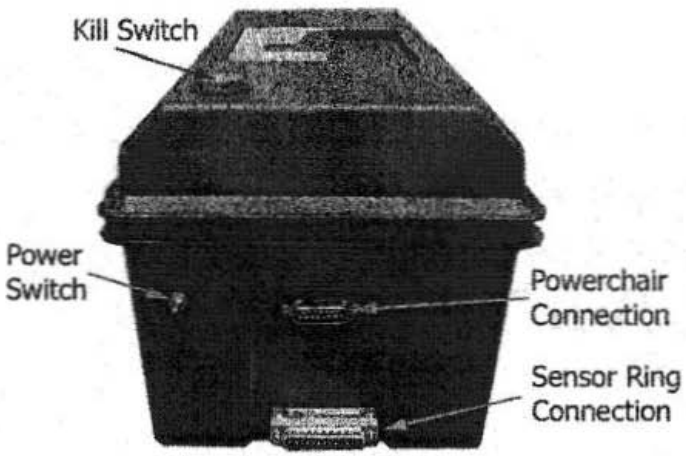


Figure 5.39: Closed PC/104 Enclosure - Side View

called "SchemaController" and provided a motor schema implementation similar to the one presented in section 3.3.6. Since teleoperation of the robot was a requested feature, a separate remotely-operated program called "TeleOp" was written and treated as a sensory input by the SchemaController software. The overall system is diagrammed for clarity in figure 5.40.

The SchemaController software focused on implementing the obstacle avoidance behavior, teleoperative control ability, an prevention of the "local minimum trap"[9] problem by having in place a random-motion behavior to 'kick' the robot out of the trap. These three behavior modules would be fed into a behavior-mixing block which could be used to change the gains of each behavior to suit a specific intention. From there, the resultant motion vector would be sent to the robot's motor controller circuit, and the behavior-based controller could be tested.

[9]The *local minimum trap* problem is where the desire to move is exactly opposed by a virtual repulsive force created by the obstacle avoidance routine, creating an inability for the robot to progress with its intentions.

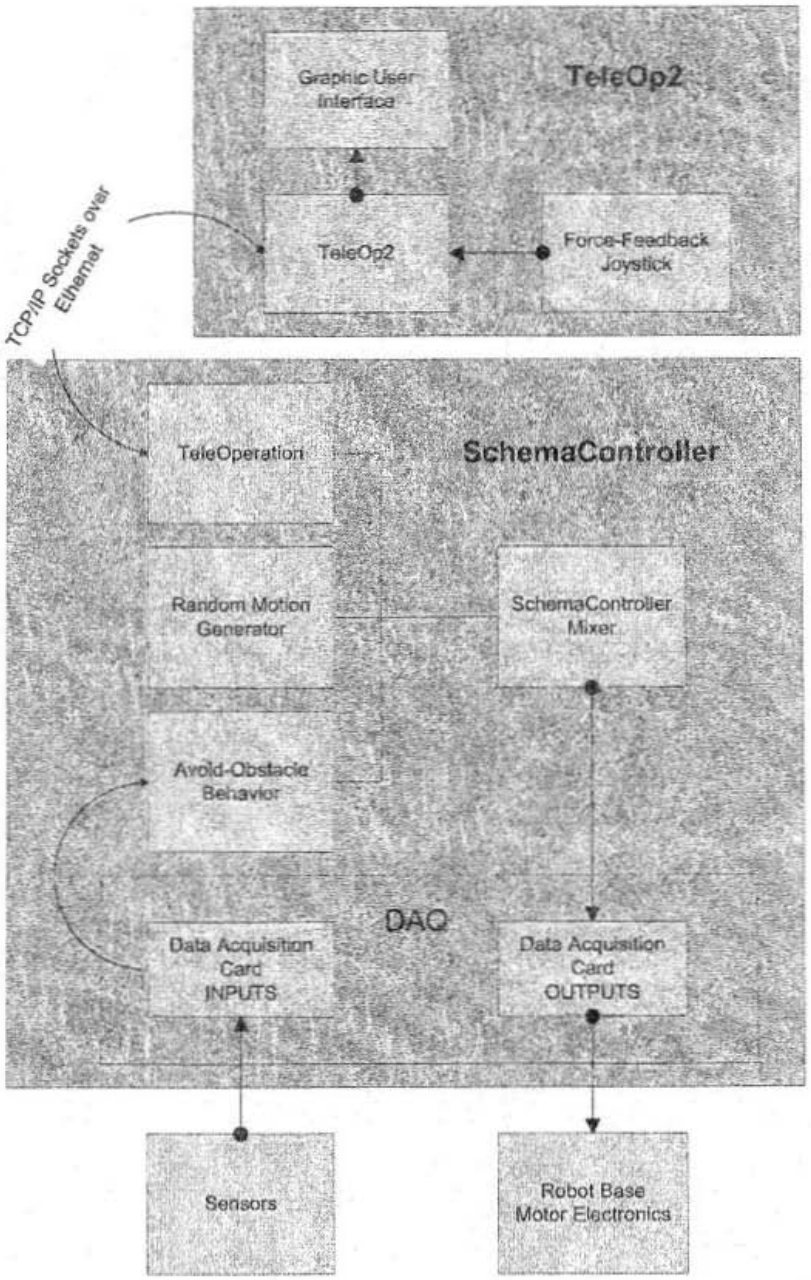


Figure 5.40: System-Level Controller Overview

SchemaController Features

This software, which ran onboard the SAMBUCA MRP, was divided into four main functional groups: sensory data acquisition, behavioral algorithm calculation, mixing of behaviors, and motor control output. SchemaController also used network sockets to communicate with its sibling application, TeleOp.

The first of these functional groups, sensory data acquisition, was designed to represent each sensor as an object (in *object-oriented programming* sense), derived from a common sensory object parent class. This parent class would provide the following features to each sensor:

- The ability to carry metadata[10], which could be useful to behavioral algorithms and system troubleshooting.
- The ability for sensors to be polled at separate polling rates, to make the best use of the data acquisition circuitry's sensor scan capacity.

The second of these groups, behavioral algorithm calculation, was designed to represent each robotic behavior as a concurrently-executable[11] object, so that the behavioral algorithms would simultaneously produce behaviors given the sensory inputs. The software architecture made it possible for any sensor to be read by any of the behavioral objects. Each behavior would have a standard two-dimensional vector output in the range of $< [-100, 100], [-100, 100] >$.

The SchemaController's third functional group, mixing of behaviors, would be responsible for calculating the overall behavior of the robot given the robot's intentions[12]. The behavioral mixer would work as all linear signal mixers work, by having signal enable lines to toggle, gain blocks to adjust, and a summation block as illustrated in figure 5.46. Like the output of the individual behavioral blocks, the output of the behavioral mixer would be in the range $< [-100, 100], [-100, 100] >$.

The fourth functional group, motor control output, would be responsible for converting the behavioral mixer's output into appropriate signals to be sent from the DAQ card to the motor control circuitry on the MRP.

[10] *Metadata* refers to the information about the sensor's own properties as opposed to the information that the sensor measures from its environment.

[11] Somewhat of a fallacy, since no parallel processing hardware was actually used, but rather the operating system's task scheduler provided a satisfactory facsimile of operational concurrency.

[12] *Intentions* refer to the preferred mix of behaviors used to accomplish a specific task.

Finally, the SchemaController application would make use of network sockets (as discussed in section 4.6) to broadcast and receive communications to and from the remotely-located TeleOp application.

TeleOp Features

An important facet of semi-autonomous systems is the ability to remotely operate a vehicle when the computerized controller is faced with a situation requiring a human operator's expertise. Within the SchemaController program, the teleoperation module (which masquerades as a behavioral algorithm calculation module) acts as a command receiver, shares the same output interface as other behavioral modules (i.e. vector output), and can be fed into the behavioral mixer to be combined with the avoid-obstacle behavior, for instance, to create a teleoperative experience safeguarded from collisions.

The commands that are received by the SchemaController's teleoperation module are issued from a separate application called TeleOp, which accepts human operator control through a joystick, performs smoothing on the input, and issues the commands over a network socket to the SchemaController application. Since network sockets operate using a server/client approach, with the server being the system always 'on' and the client being the system that occasionally connects to the server, the SchemaController application aboard the mobile robot platform was set up as the server, and the TeleOp application was set up as the client.

SchemaController Implementation

General Implementation Remarks

This software was written in C++, under the Microsoft Visual C++ 6.0 integrated development environment (IDE), primarily because this was the tradition of the ISLab and therefore troubleshooting and integration with other systems would be less troublesome than if another implementation language was chosen.

While it is true that motor schema control architectures would benefit from the inherent concurrency capabilities of an object-oriented programming language (OOP) such as Java™, the need to access hardware (namely data acquisition inputs and outputs) was a higher priority, and C++ provided the best means of accomplishing

the goal. A Unified Modeling Language (UML) diagram of the SchemaController software is shown in figures 5.41 and 5.42, and will be referenced throughout this description. As well, a flowchart of the SchemaController has been provided in figure 5.43.

The SchemaController application was based on skeletal "Dialog Application" code generated by the Visual C++ IDE. This was done to present the end-user with a single-window interface containing familiar Windows buttons and indicators. The central class in this application was *CSchemaControllerDlg* ('Dlg' referring to the 'dialog'-style of application), and was where the graphic user interface (GUI) was developed, GUI message handlers were written, where data input and output were coordinated, where network communication was established, and where schema objects were located and interconnected.

Sensory Data Acquisiton Implementation

Early in the design of this software, individual sensors were to be represented as objects of type *CSensorObject*, with the primary objective of setting individual sensor polling rates, and with the secondary objective of providing metadata to perceptual schema objects. It was not possible, however, to achieve individual sensor polling rates, as the data acquisition driver software did not permit concurrent methods to be called on it. The proposed solution to this was for shared-memory to be used, whereby all of the sensor signals would be periodically scanned by the DAQ and the readings would be placed in the PC/104's memory so that schema objects could access the data from memory at whatever rate was convenient for those objects.

These ideas were implemented using the classes *DAQ_Singleton* and *Memory_Singleton*. The reasons for using the 'singleton' design pattern on these classes was to achieve the effect of having globally-accessible resources across the application, while retaining the benefits of object-oriented software design. The singleton design pattern ensured that all instances of DAQ_Singleton and of Memory_Singleton actually referred to the singular DAQ resource, or memory resource, respectively, while still exhibiting OOP niceties such as information hiding and functional encapsulation.

Although *CSensorObject*'s planned use of metadata did not pose any implementation problems, it was not paramount to the operations and was removed from the

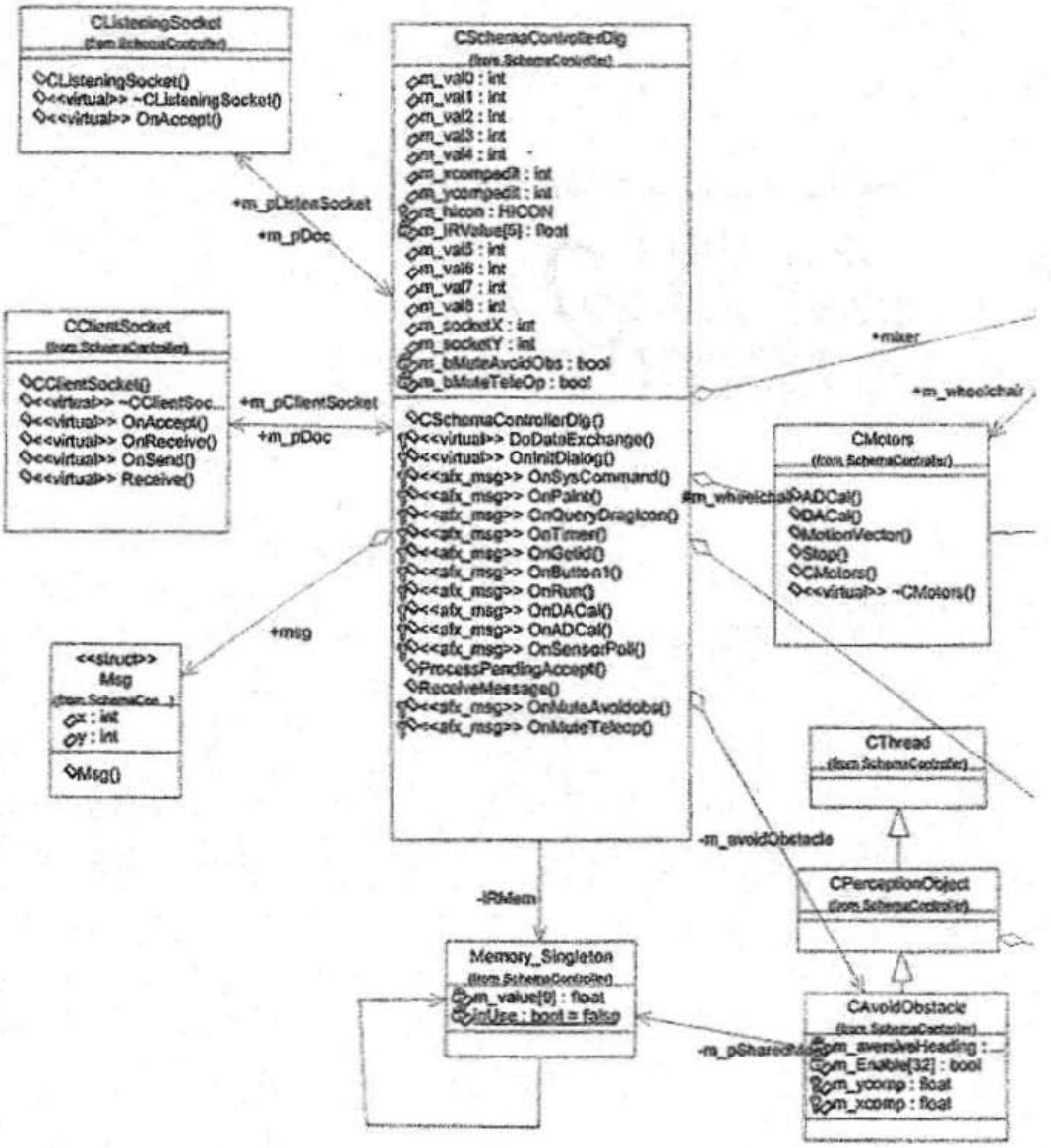


Figure 5.41: UML Diagram of SchemaController, Part 1 of 2

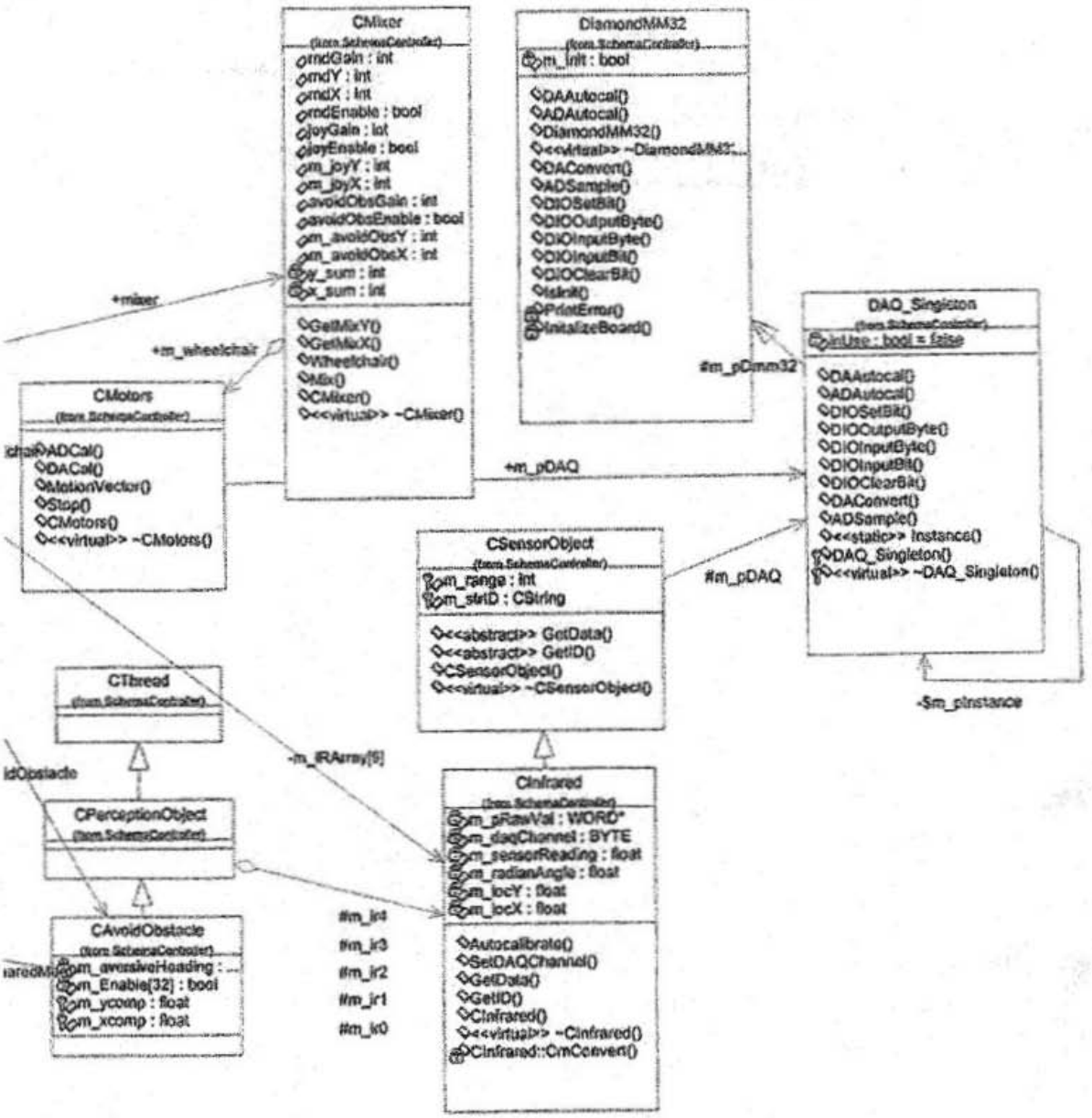


Figure 5.42: UML Diagram of SchemaController, Part 2 of 2

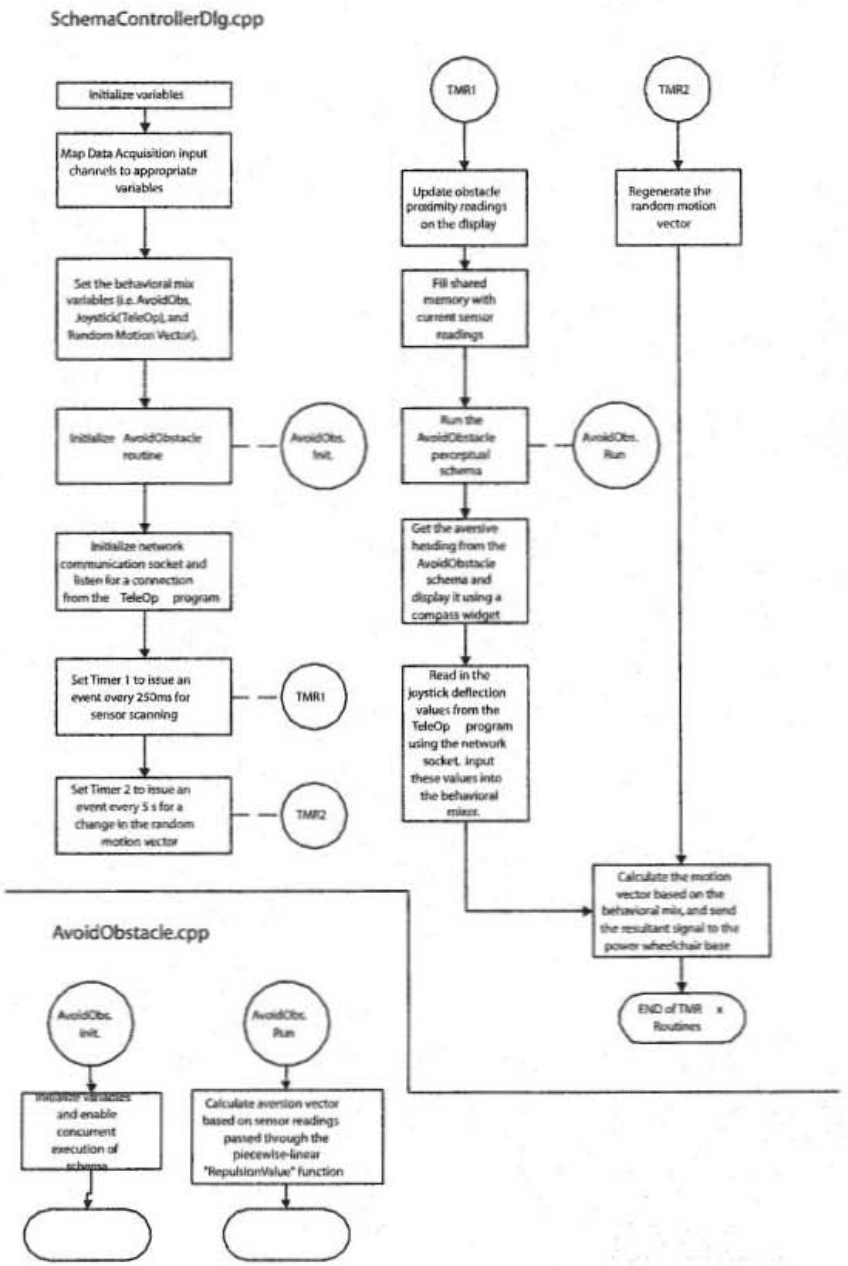


Figure 5.43: SchemaController Flowchart

design, causing *CSensorObject* to be relegated as a wrapper class of *DAQ_Singleton* and also providing virtual methods such as "GetData()" and "GetID()" to its derived classes, to be implemented there.

The *DAQ_Singleton* class was created by applying the singleton design pattern to the *DiamondMM32* class, which itself was a C++ wrapper class around the C-based application programming interface (API) methods, which in turn were provided with the DMM-32 card when purchased. Therefore, the functionality of *DAQ_Singleton* is documented within the larger scope of the Diamond MM-32 driver manual [64].

The *Memory_Singleton* class contained nine variables of type *float* to act as shared memory between the DAQ and the schema objects requiring sensory data. There was no special significance to having nine variables, merely that it provided sufficient memory for the number of sensors that were used in early testing.

As the only type of sensor used in initial testing was the Sharp infrared proximity sensor, the only class derived from *CSensorObject* was *CInfrared*, whose 'GetData()' method would access the appropriate DAQ input port for that sensor's value.

Behavioral Algorithm Calculation Implementation

Despite the lack of concurrency in the objects derived from *CSensorObject*, concurrency was achieved in the more important *CPerceptionObject* class, which was designed as the parent class for the behavioral schemas themselves. As is shown in figure 5.42, this class was derived from *CThread*[13] which provided its child classes with the multithreading ability necessary for concurrent operation.

The class *CPerceptionObject* had no virtual methods to be implemented, nor did it have member variables common to all perceptual schema objects, but *CPerceptionObject* was included in the architecture as a provision for future functionally-grouped additions, as well as to help in any future re-factorizations of the software.

Implementation of the Obstacle Avoidance Behavior

One of the important behaviors for SAMBUCA was the 'avoid-obstacle' behavior,

[13]CThread was written by Dominik Filipp as free public-domain software [65]. It provides a more programmer-friendly encapsulation of worker-threads for the Windows operating system than the native MFC framework for multithreading.

which was implemented in the class *CAvoidObstacle*. This dealt with obstacle avoidance in 2D but not 3D, since the robot only moved over surfaces and not in free space. The essence of this class in written within the 'ThreadHandler()' method, in order to be run in its own worker thread, according to the dictates of the CThread parent class. The algorithm reads in infrared proximity sensor readings from shared memory, and calculates an aversive motion vector to move the robot clear of perceived obstacles. The raw sensor data from shared memory is process in three steps:

1. Converting the non-linear raw data into linear distance readings
2. Passing linearized readings through a function which focuses sensitivity in the middle range of the sensor
3. Using the linearized, re-sensitized readings to geometrically calculate the aversive motion vector

The first step of linearizing the sensor response was necessary for the Sharp infrared proximity sensors. The non-linear response of one of the sensors (figure 5.44) was computed using the PC application "CurveExpert, version 1.3" by Daniel G. Hyams [66] and the resulting curve fit is shown below in figure 5.45. The power law function which best fit the response points was

$$y = 1548374.9x^{-1.2418576}$$

and was used to convert raw readings (denoted by x) to the corresponding centimeter proximities (denoted by y). The number of significant digits in the higher-order coefficients was necessarily high in order to perform the conversion satisfactorily, and therefore 'doubles' were used to store coefficient values.

Since the extreme ranges of the infrared sensors were not as accurate as had been hoped for (they were oversensitive around 10cm, noisy around 80cm), a piecewise function was applied to the sensed distances in order to squelch the effects in these end ranges. Through some experimentation, it was determined that all readings greater than 50cm be treated as if the perceived object is out of range, readings less than 20cm be treated as if the obstacle presents maximum hazard, and all values in between 20cm and 50cm take on a linear range of 'repulsion values' between 100 (maximum hazard) and 0 (minimum hazard/out-of-range).

Raw Data	cm
15000	10
11500	15
9000	20
7500	25
6300	30
5500	35
5000	40
4500	45
4100	50
3800	55
3500	60
2500	90
2100	120
1700	150

Figure 5.44: Non-Linear Infrared Sensor Response

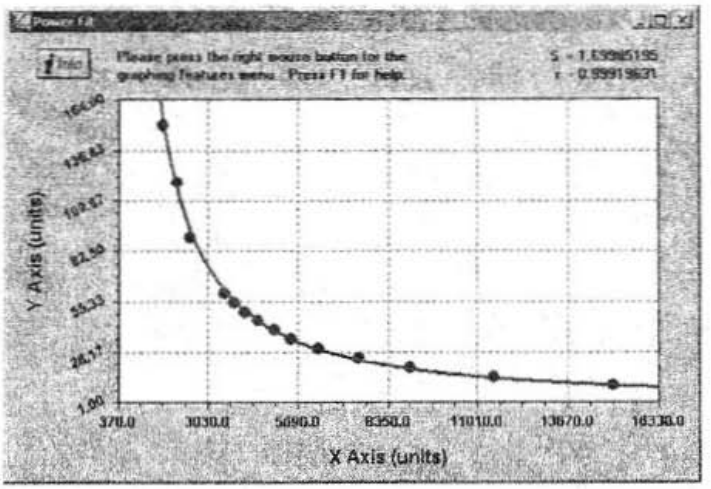


Figure 5.45: Power Law Fit of Infrared Sensor Response

$$RepulsionValue = \begin{cases} 0 & \text{if } cmVal > 50 \\ 100 & \text{if } cmVal < 20 \\ 100 - \frac{10}{3}(cmVal - 20) & \text{otherwise} \end{cases}$$

After all proximity sensors have had their repulsion values calculated, the overall aversion vector is calculated based on the locations of the sensors placed along the front ring of the aluminum mounting plate. Originally, a ray-tracing approach was proposed in order to mathematically refer the repulsion values toward the center of rotation of the base, since the arrangement of the infrared sensors was along an elliptical arc rather than purely circular arc. However, since the accuracy of the sensor placement was low, the ray-tracing approach was not worth the complexity, and so was replaced by a simpler calculation which neglected the eccentricity of the mounting plate's arc, by summing the x-axis and y-axis projections of each sensor's repulsion value:

$$F_x = \frac{\sum_{i=0}^{n} RepulsionValue_i cos(\theta_i)}{n+1}$$

and

$$F_y = \frac{\sum_{i=0}^{n} RepulsionValue_i sin(\theta_i)}{n+1}$$

The overall repulsion magnitude presented to the robot base by the perceived obstacles is calculated in the conventional way as:

$$F = \sqrt{F_x^2 + F_y^2}$$

and the aversive heading is calculated as:

$$arg(F) = -tan^{-1}\left(\frac{F_y}{F_x}\right)$$

For development purposes, the aversive heading was displayed in the GUI of

SchemaController using a commercially-available ActiveX widget. This gave immediate feedback as to whether or not the avoid-obstacle behavior was correctly calculating the virtual repulsion vector presented by obstacles in front if the infrared sensor array.

Behavioral Mixer Implementation

The avoid-obstacle behavior was just one module which was to affect the robotic platform's motion; others being the teleoperation inputs, the random vector generation input, and even behavioral modules not yet implemented. Therefore, a mixing block was introduced to the system, implemented as the class *CMixer*. As shown in figure 5.46[14], each behavior outputs a motion vector which, once sent to CMixer, is matched with a gain variable $G_{behavior}$ that sets the respective behavior's importance to the overall system. The behaviors are linearly combined.

$$\langle x_{sum}, y_{sum} \rangle =$$
$$G_{avoid\_obs}\langle x_{avoid\_obs}, y_{avoid\_obs} \rangle + G_{random}\langle x_{random}, y_{random} \rangle + G_{teleop}\langle x_{teleop}, y_{teleop} \rangle$$

The summation is then hard-limited to keep within the range of the motion vector.

$$x_{sum} = \begin{cases} x_{sum}, -100 \leq x_{sum} \leq 100 \\ 100 \times abs(x_{sum}), \text{otherwise} \end{cases} \quad y_{sum} = \begin{cases} y_{sum}, -100 \leq y_{sum} \leq 100 \\ 100 \times abs(y_{sum}), \text{otherwise} \end{cases}$$

The motion vector is then scaled and shifted from a $\langle -100, 100 \rangle$ scale to a $\langle 0, 4096 \rangle$ scale corresponding to the 12-bit analog output of the DAQ card.[15]

$$x_{DAC} = [20.48 \times x_{sum}] + 2048$$
$$y_{DAC} = [20.48 \times y_{sum}] + 2048$$

The resulting voltage output from the DAQ to the R2 power wheelchair electronics is expressed as

[14] Although figure 5.46 shows *two* "Future Behavior" blocks, fewer or more behavior blocks may be added to the system.

[15] Although 12-bit numbers range from 0 to 4095, the value 4096 was used in the calculations with no ill effects.

$$x_{voltage} = \frac{x_{DAC}}{4096} \times 5\text{V}$$
$$y_{voltage} = \frac{y_{DAC}}{4096} \times 5\text{V}$$

While not currently implemented, it would be straightforward to create mixer presets or allow for other system components to vary the gain and enable variables to adapt the behavioral mix according to the robot's intentions. For instance, one mix may be suitable for operating in open spaces while another mix could be better for following narrow paths.

Motor Control Output Implementation

The mixing of the vectors provided by the behavioral modules results in the overall direction and speed of intended travel for the robot base. The CMotors class takes care of the conversion from the vector number range to the DAQ output range, by performing a straightforward shift-and-scale operation. Both x and y vector components range from [-100, 100] but must be scaled to DAQ values ranging over [0, 4096], with 2048 corresponding to no motion. In addition to performing the conversion, CMotors ensures that the digital-to-analog converter is in calibration so that a vector value of $< 0, 0 >$ will indeed correspond with zero motion of the robot base. In case the SchemaController software detects an emergency, the CMotors::Stop() method can be invoked to immediately cease any robot base movement.

TeleOp Implementation

A UML diagram of the TeleOp application is shown in figure 5.47. This application was also written in MFC C++ using the dialog-style, and therefore the main class *CTeleOp2Dlg* is derived from *CDialog*. The Microsoft Sidewinder force-feedback joystick used by the human operator is accessed through the class *CJoystick*. The Sidewinder is capable of steering two-dimensionally (controlling the x- and y- values) and adjusting a throttle value via its throttle dial (correspondingly displayed in the TeleOp GUI using the *CProgressCtrl* class). The Sidewinder was also equipped with a torsional input (controlling the z-value,) trigger buttons, a 'point of view hat'[16] controller, and force-feedback, but integration of these extra inputs was deferred for

[16]The *point-of-view hat* controller is itself a mini-joystick atop the main joystick, is controlled by the thumb, and is used in first-person games to control the projected view of a virtual world.

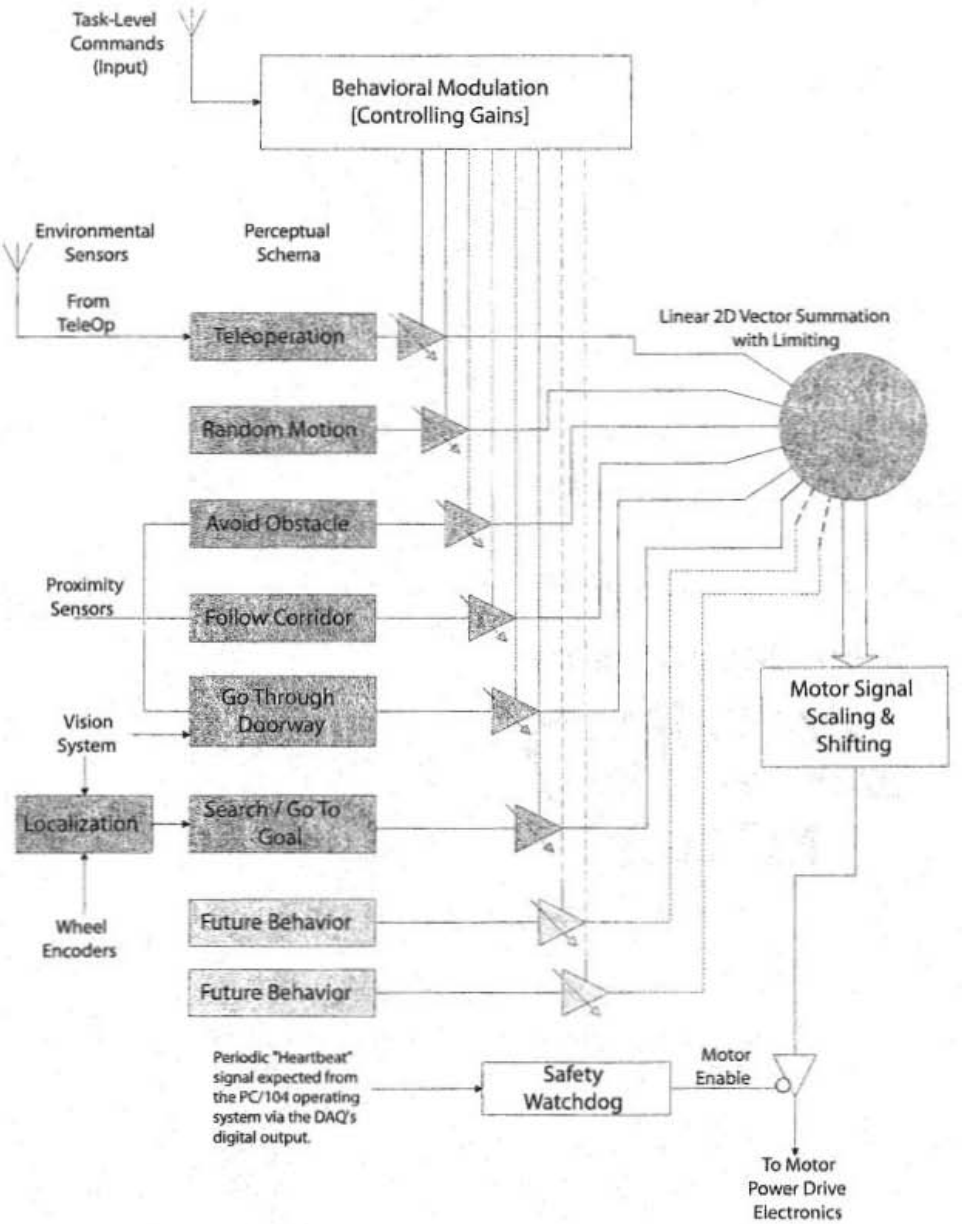


Figure 5.46: Behavioral Mixer

future work on the proposed human machine interface. The joystick's x- and y-values were read from the joystick every 100 milliseconds, sent through a five-valued running average computation to smooth out the input, encapsulated in a C-style *struct* and sent over the network using the *SendingSocket* class.

Back in the SchemaController application, the teleoperation module made use of a *CListeningSocket* to await a network connection by the TeleOp application. Once established, SchemaController would set up a *CClientSocket* to read information from the TeleOp application, convert the readings into vector form to be sent to the behavioral mixer. The advantage to using network sockets was that the protocols and physical-layer means used were transparent to both the SchemaController and the TeleOp applications, which meant that switching to a wireless connection from a wired connection would not entail any changes in the software.

The TeleOp joystick convention mimicked the joystick that was on the original Pronto R2 powerchair base, as shown in figure 5.28. As expected, any in-between position of the joystick would result in partial combinations of the respective behaviors listed.

5.1.5 Safety Measures

Due to the PC/104 system's problems with its VGA module, in order to see what was happening onscreen Windows 95 had to be used instead of the more stable Windows 98 or Windows 2000 operating systems. While an operating system failure (known in colloquial Windows parlance as a *Blue Screen Of Death*, or BSOD) is merely an annoyance on a desktop PC system, when a BSOD occurs aboard a powerful motorized platform, a hazardous situation emerges. It was verified that the DAQ card continued outputting voltage after a BSOD, which could possibly result in a runaway robot with sizeable momentum.

In order to prevent such accidents, a watchdog timer circuit was proposed whereby the DAQ card would be periodically instructed to toggle an binary output at some minimum rate, and this output would be monitored by the watchdog circuit. In the case of a BSOD, the instructions to toggle the binary output would cease, and the output would become stuck at some level, causing the watchdog circuit to react by cutting power to the motors.

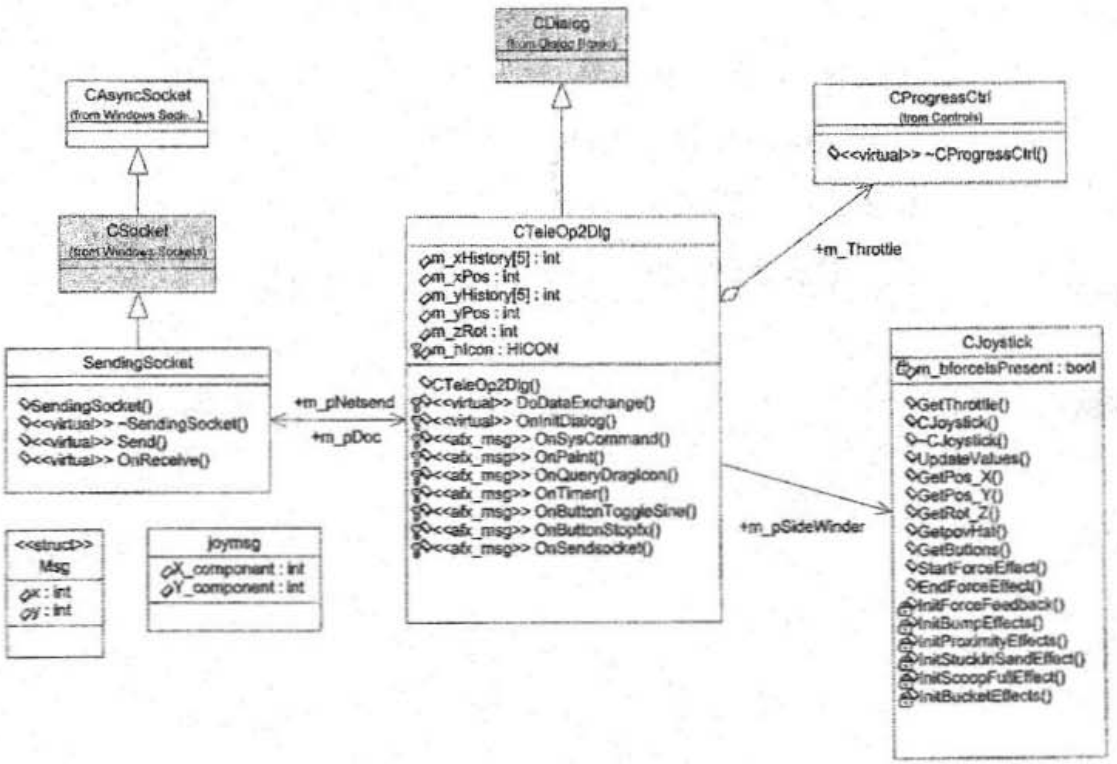


Figure 5.47: UML Diagram of TeleOp

The watchdog circuit was designed around a UCC3946 IC, described as a "Microprocessor Supervisor with Watchdog Timer" (see appendix figure B.3).An external capacitor of $1nF$ was used to set a watchdog period of 25 milliseconds, corresponding to a minimum binary output rate of 40Hz from the DAQ card. If the DAQ's binary output falls below this rate, the watchdog output is asserted, current-amplified by a discrete FET, and the amplified signal is used to energize the coil of a 4-pole/double-throw (4PDT) relay, which then cuts out the signal to the motors. As soon as the DAQ's binary output resumes its toggling above 40Hz, the watchdog circuit reconnects the signals to the motors.

5.1.6 Communications

Robotics projects such as the one being carried out by the ISLab cannot be effectively done by one developer writing a single piece of code to control everything. Rather, a number of different modules, some physically distant from the others, must interoperate. Effective interprocess communication techniques therefore must be used to combine disparate modules into a cohesive operational robotic system. These communications are achieved across two types of interfaces: the software interface and the hardware interface.

Hardware Interface

The SAMBUCA platform is a prime example of a system module physically removed from the other modules. Therefore there has to be a way of communicating data to and from the platform's on-board computer system. As was mentioned in section 4.6, wireless ethernet was the desired approach to take. The system that was chosen was a WaveLan system by Lucent Technologies (since renamed the 'Orinoco', made by Agere Systems). The system consisted of a Wavepoint *II* Bridge (which extended the existing wired ethernet network to the wireless domain), two 11 Mbps PCMCIA cards with 'wired equivalent privacy' (i.e. data encryption), and one range extender antenna. Operating in the licence-free 2.4GHz frequency band using direct sequence spread spectrum (DSSS) modulation, the wireless system did not interfere with other wireless equipment operating in the lab, and was designed so as to not interfere with other wireless ethernet systems nearby. PCMCIA cards are typically used as

peripheral devices on laptop computers, but could also be used in PC/104 systems by means of a PC/104 - PCMCIA adapter module. The antenna was purchased with the expectation that the SAMBUCA base be able to operate outside of the laboratory (and even outside the building) over distances of hundreds of meters, while being coordinated from within the ISLab.

Software Interface

While the SchemaController and TeleOp applications made effective use of elementary 'homemade' message-based communications over sockets, it was felt that as the ISLab group robotics system developed, ad hoc approaches to software communication interfaces would be a poor use of development time, considering that solutions to distributed computer communications already existed. Gowdy's report on interprocess communication toolkits provided a background on the topic, stating that these interfaces would enable software applications to control remotely-located objects over a network in the same way that traditional local objects could be controlled, regardless of the remote system's programming language or operating system [67]. While not a panacea for all software communication challenges, it was felt that these toolkits could provide tremendous value to the ISLab group robotics project, both in terms of development time and in terms of functionality.

The three toolkits that the ISLab decided to investigate were CORBA (Common Object Resource Broker Architecture), DCOM (Distributed Component Object Model), and Jini (see section 4.6 for footnotes on these toolkits). In short, CORBA (by the Object Management Group) was designed for use on UNIX systems but also was said to operate on systems running Microsoft Windows. DCOM (by Microsoft) was designed to run on systems using Microsoft Windows. Jini (by Sun Microsystems) was an extension of Java that would operate on any system that would run Java applications. Ultimately, the Jini approach was favored since it would more standardized than CORBA and more open than Microsoft's DCOM.

Currently, the Jini approach is being tested in the Discrete Event Controller portion of the 'General Framework' and will likely be implemented in the SAMBUCA platform's software as well, resulting in seamless communication across the entire framework.

5.1.7 Problems

The development of the SAMBUCA platform encountered some implementation difficulties, which are discussed here.

Infrared Proximity Sensor Switching Noise

Early tests using the Sharp GP2D12 infrared proximity sensors revealed an easy-to-use sensor that returned an analog voltage related (albeit non-linearly related) to the proximity of the closest object in front of the sensor, measurable to within one centimeter over a range from 10cm to 80cm (corresponding to 0 to 5 Volts of signal). The difficulty ensued when an array of these sensors were used, and signal noise became so bad that sampled readings could no longer be trusted. Since the GP2D12's were active sensors, cable impedance issues were ruled out, and individually the sensors worked correctly. It turned out to be a power issue: while the steady-state consumption of the array was within the capabilities of the PC/104 power supply (itself rated for 5V, 50W), the sensors actually drew extra current every millisecond, and since the sensors were not synchronized in this respect, one sensor's current-drawing phase would affect the other items on the power bus (including other sensors and the DAQ system itself) manifesting as signal noise. The more sensors that were added to the array, the worse the noise became. The simple remedy to this was to place $100\mu F$ electrolytic capacitors across the power terminals of each of the infrared sensors.

It should also be noted that inter-sensor interference (i.e. the infrared beam from one sensor affecting another nearby sensor's receptor) was not found to be an issue.

Pronto R2 Motor Controller

The process of feeding control signals from the PC/104 system into the existing powerchair motor controller circuitry was presumed to be as simple as applying control voltages at will from the DAQ system. As it turned out, the powerchair's original motor controller was equipped with a safety feature whereby the motors would not turn if the powerchair's joystick was not centered perfectly at power-up. This meant that the DAQ had to put out precise 2.5V levels on its translational and rotational

control lines upon startup. To combat this, auto-calibration routines were run periodically on the DAQ system, since the PC/104 power supply's effective internal resistance could cause the DAQ to fall out of calibration when other loads were presented to the power supply. Another problem with the powerchair motor controller was that the motors would cease to spin if the control voltages from the DAQ were jumped suddenly (skipping over intermediate voltages), as this phenomenon would not happen with the standard analog joystick that originally came with the powerchair. To combat this, the TeleOp application incorporated a smoothing filter to the joystick inputs. Failure to meet the powerchair's safety conditions resulted in the motor controller being unresponsive for upward of ten to fifteen minutes.

Non-Holonomicity

Mobile robot navigation is always easier to implement and study in simulation where robots are typically holonomic (i.e. a massless point capable of instantaneous changes in its motion). On SAMBUCA, the single largest contributor to its status as a non-holonomic robot is its front caster wheels. Caster wheels (the kind found on the front of shopping carts) are most problematic when changing directions between forward and reverse, since they can momentarily bind and cause the vehicle to deflect from its intended heading. This problem can be mitigated by using corrective feedback (e.g. through wheel encoders), or by operating in teleoperation mode when the robot is in a tight spot.

Wheel Encoder

Wheel encoders are important sensors to have when carrying out robotic navigation, but they also require adequate signal processing (e.g. quadrature decoding) to result in meaningful readings. The encoders on SAMBUCA were originally intended to be fed into the DAQ system where they could be processed, and the processed result could be used by the software controller. Unfortunately, the DAQ was not capable of performing any processing, so this would have to be accomplished by the software. The problem was that each encoder wheels would have to be sampled at minimum 1000 samples per second in order to meet the Nyquist sampling criterion. Furthermore, all of these samples would have to be processed in real-time by the

software in order to deduce the amount of travel experienced by one wheel. This scenario would likely cause the system to be I/O-bound just to keep up with the signals coming off the wheel encoders. A solution to this would be to implement the quadrature decoding using dedicated hardware (e.g. a RAD-board) containing 'tick' count registers for each wheel that could be polled occasionally by the DAQ for navigational purposes. The drawback of this approach is that in order to convey the count to the software, a number of data input lines would need to be consumed at the expense of other sensors. Alternatively, the count could be expressed serially and sent to the PC/104 system over RS-232 or some other serial protocol.

Robot Drift

The SAMBUCA platform, not yet having an operational wheel encoder feedback system in place, exhibited drift to one side when traveling forward. This was reduced by adding an offset to one of the motor speed values within the SchemaController application. While it is preferred to use feedback to correct for this, the offset proved to be an acceptable temporary measure while initial testing of the behavior-based controller was carried out.

Multiple Sensor-Polling Rates

As was briefly mentioned in section 4.5.2, the original design of the SchemaController was to feature adjustable polling rates for each sensor input, so that fast-changing inputs could be scanned more often than slow-changing inputs, and that input scanning be done in an efficient manner. However the Windows 95 operating system was unable to run the DAQ board in a multithreaded fashion so as to poll different inputs at different rates. An alternative was to use a number of 'waitable timer' objects, each with different alarm rates, and each timer alarm corresponding to the polling of a specific sensor input. Unfortunately waitable timers were not supported under Windows 95, so in the end all sensor inputs were polled at one uniform scan rate on the DAQ board, and input efficiencies were disregarded.

PC/104 - PCMCIA Adapter

The originally specified PC/104 system came with an adapter card which enabled PC-Cards (PCMCIA credit-card sized peripheral devices) to be used in a PC/104 computer system. Accordingly, network connectivity was to be established using a wired ethernet PC-Card, with the prospect of upgrading to a wireless PC-Card at a later date, while not causing any changed to the underlying PC/104 hardware. Since the original PC/104 system had problems from the start in getting operating systems loaded, the PC-Card Ethernet became the main communication interface into and out of the PC/104. Unfortunately the PC/104 - PCMCIA adapter stopped working with no obvious explanation, thereby cutting the original PC/104 system from the network and from effective development. Rather than order a new adapter card for an already problematic system, work was transferred onto a more developer-friendly PC/104 system with built-in wired ethernet.

Since the addition of wireless ethernet was deferred, development of the SAMBUCA base was done over a wired network, with all software being developed on a desktop PC, and the software transferred to SAMBUCA. From there, a VGA monitor, keyboard, and mouse were attached to the PC/104, the behavioral software was set running, the monitor, keyboard, and mouse were unplugged once again, the robot's motors were engaged, and tests were conducted without being able to see the SchemaController GUI on a monitor.

This process of tethering for development and untethering and testing without the benefit of a GUI was tolerable for initial tests, but suffered from the "running researcher syndrome" whereby if anything went wrong, the developer would have to chase down SAMBUCA and stop it, instead of being able to conveniently stop it from a remote PC.

By purchasing a new PC/104 - PCMCIA adapter and incorporating wireless ethernet, remote desktop applications such as the popular 'pcAnywhere' application will make future development and testing far more effective. Furthermore, wireless ethernet is necessary for the ultimate goal of interfacing to the rest of the general framework for group robotics.

Power Distribution

Since the SAMBUCA platform was intended to operate in an untethered fashion, there had to be provisions for powering all payload and actuators from the onboard battery supply. The PC/104, which was able to operate off of DC supplies ranging from 6V to 40V, was able to be powered directly off of the battery supply. Likewise, the RAD-board system has an on-board regulator which could be connected to a 12V potential and drop the voltage down to the required 5V for the board. However, while all payload ultimately ran off of DC supplies, it was cumbersome to create separate DC supplies for each module from the 12V/24V battery supply. Many payload modules all had 120VAC power plugs, so it made sense to include a power inverter unit to convert the onboard DC battery supply to a 120VAC supply which could be distributed on a conventional power bar and sent to the needy payload modules. A *Portawattz* power inverter (see figures 5.48 and 5.49) was used when trying this approach to power distribution.

The 300-Watt rating proved barely sufficient to power the RT200 robot arm, as any useful loading of the arm would increase its power demand and cause the low-voltage alarm to sound. While this problem could be rectified by using a higher-capacity power inverter, another more hazardous problem existed. On two occasions, there were high-current surges towards the RAD board that entered through its RS-232 communications interface, causing (fortunately) only a pull-up resistor on the RAD-Board to burn out. Since communication signals and power lines had nothing electrically in common except for shared ground lines, it was postulated that a grounding problem existed when the inverter was used. Although these two incidents were not well documented, it is believed that they happened when the 'floating' (i.e. non-referenced) ground of the SAMBUCA base (which in turn was the ground on the RAD-Board-side of the RS-232 connection) was connected to a desktop PC's RS-232 port, whose ground was ultimately tied to the laboratory's electrical system's ground. While this connection should have merely caused the SAMBUCA supply to take on the laboratory's ground reference, the fact that a power inverter was in operation, likely using power transistor switching and filtering to create its modified sine wave AC output, the simple DC circuit assumptions stated above may have not actually held. Despite these incidents, it is believed that the

Figure 5.48: Power Inverter [16]

use of a power inverter is a viable way to power payload modules off of the battery supply, as long as care is taken to avoid grounding issues. With the implementation of wireless communications on SAMBUCA, there should be no need to use signal tethers, so the cause of this problem will disappear.

5.2 Demonstration

5.2.1 Vehicle Performance

The performance of the Pronto R2 Base controlled by its original joystick (i.e. the vehicular performance independent of the SchemaController-PC/104 or RAD control systems) was as follows:

- Maximum speed of slightly over 1 m/s (thus improving overall motion safety)
- Mid-body differentially-steered drive wheels (a *zero-turning radius* vehicle)
- 35.5cm diameter rubber drive wheels

Max. Long-term Power Output	300 W
Surge Capability (Peak Power)	500 W
Optimum Efficiency	$\approx 90\%$
No Load Current Draw	0.18 A
Output Waveform	Modified Sine Wave
Input Voltage Range	10 - 15 VDC
Low Voltage Alarm	10.6 V
Low Voltage Cut-Out	10 V
AC Receptacles	Dual (Grounded)
DC Fuse	30 A

Figure 5.49: Specifications for the Power Inverter

- Tested indoors (in the ISLab and the neighboring engineering building)
- Tested outdoors (along gravel pathways, over 15cm curb obstacles, on roughly 30° slopes)

5.2.2 Motion Control

The earliest demonstration of the SAMBUCA base tested the motion control interface using the RAD-Board, before the PC/104 had taken over the responsibility of low-level motion control. The RAD-Board controller's outputs were connected to the powerchair's power electronics module, and the RAD controller was input serial communication commands[17] from a desktop PC.

The attached PC was running vision system software developed by Smith [63], which detected and distinguished between bottles and cans as seen by the system's webcam[18]. The vision system would send the appropriate commands to the RAD controller so that SAMBUCA would drive to the target object while avoiding collisions with nearby objects. Collision avoidance was achieved through the use of an infrared proximity sensor ring connected to the RAD controller. Once SAMBUCA was driven into place, the vision system PC would command the robotic arm aboard

[17]See figure 5.37 for the list of RAD-Board commands accepted over the RS-232 connection.

[18]The webcam (network video camera) that was used was a 3Com HomeConnect model, chosen for its superior imaging quality among webcams.

SAMBUCA to pick up the target object and then place that object into either a bin for bottles or the a bin for cans, depending on the type of object that the target was.

The demonstration was carried out successfully, and is documented further in Smith's own report [63].

5.2.3 Motor Schema

The implementation of SAMBUCA's motor schema controller was done incrementally, as is the preferred approach in behavior-based controller development. The SchemaController application was the first application that was written, followed by the TeleOp application. The levels of demonstration were:

- Pure Teleoperation
- Augmented Teleoperation (Teleoperation with Obstacle Avoidance)
- Goalless Autonomous Operation

The fourth level of demonstration, Goal-driven autonomous operation, was not able to be tested since it required higher-level systems such as a vision system or a supervisory control system to be available and working. At the time of SAMBUCA's testing, such systems were not ready for operation. Pictograms illustrating SAMBUCA's typical actions are presented in figure 5.51.

Pure Teleoperation

The first demonstration of pure teleoperation of the robot platform was done using the early RAD-Board controller system hooked up to a laptop PC, wirelessly commanded using commercially-available *pcAnywhere* remote desktop software, which was also used to send back live video over wireless ethernet to the remote operator's console (see figure 5.52). Although this early demonstration afforded only very crude motor control, it demonstrated that effective teleoperation could be carried out over a WLAN.

The second demonstration of pure teleoperation was done within the framework of the motor schema architecture, by having only the teleoperation behavioral module enabled (refer to figure 5.46). It was demonstrated that the SAMBUCA platform

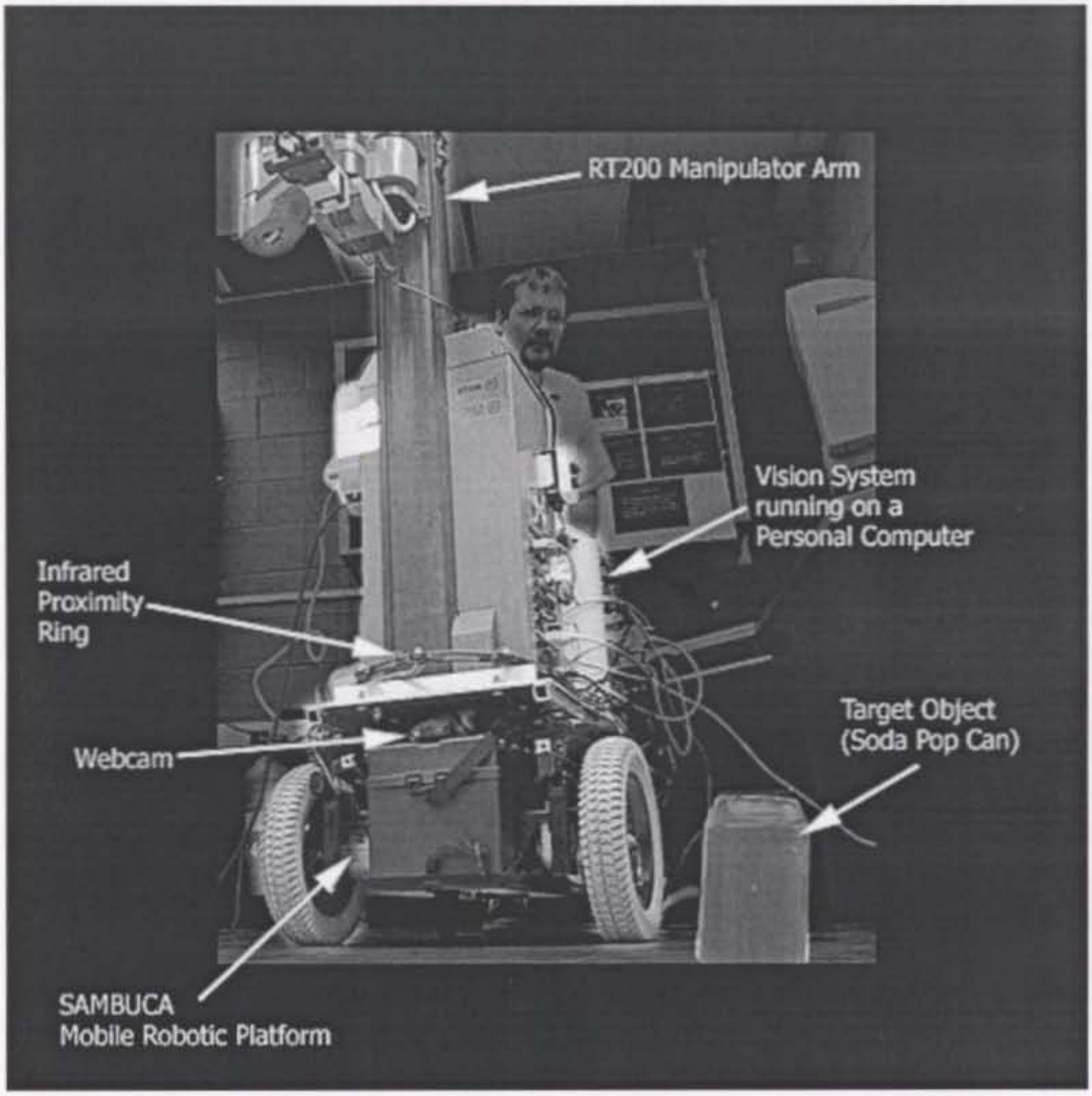


Figure 5.50: SAMBUCA: Motion Control Testing

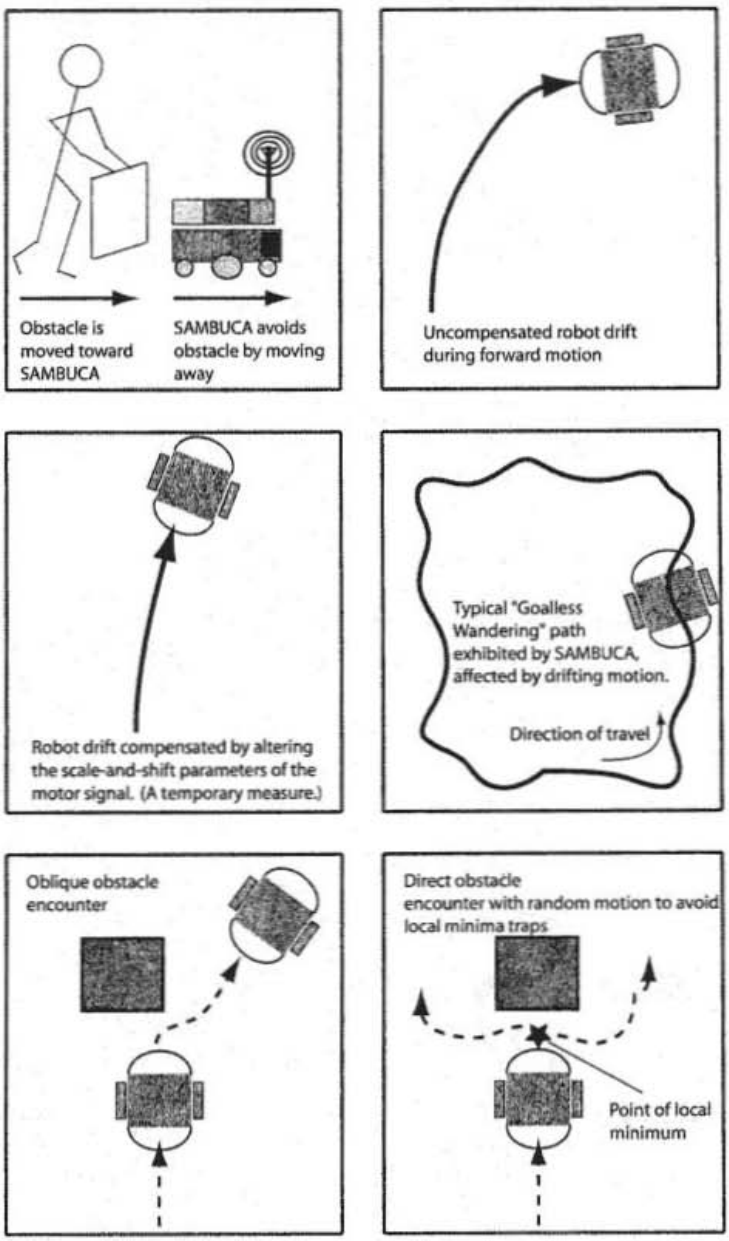


Figure 5.51: Typical Actions of SAMBUCA

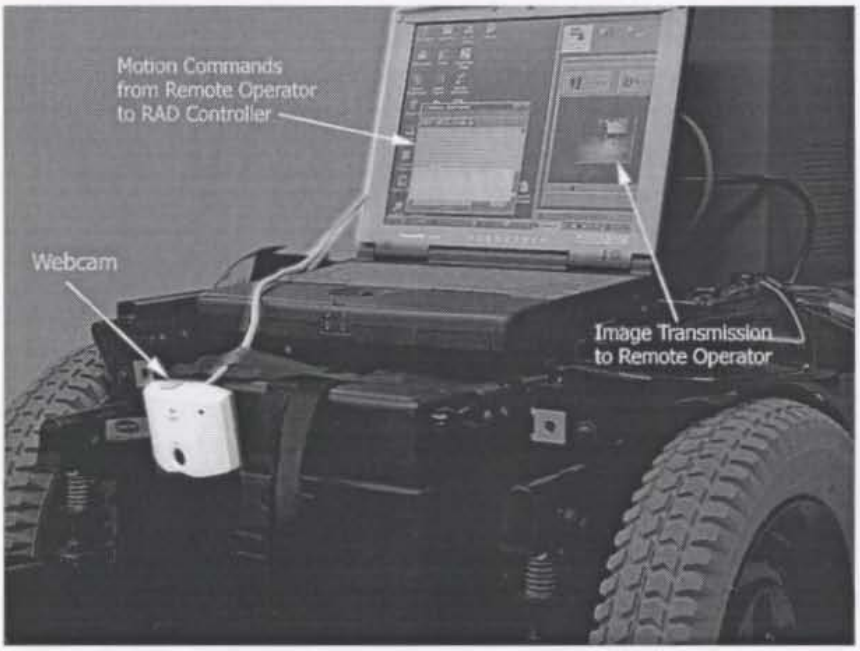


Figure 5.52: Teleoperation with a RAD-Board, Laptop, and Webcam

could be controlled over wired ethernet from a desktop PC workstation connected to a joystick. Compared to the earlier demonstration, while the motor schema teleoperation was not done wirelessly (as the PC/104 adapter card was broken by this time), the motor control was finer than the RAD-Board would permit, and was comparable to the Pronto R2 powerchair's original joystick interface.

Augmented Teleoperation

Pure Teleoperation expects that the remote operator has a sufficiently rich[19] interface to safely maneuver a mobile robot through its surroundings. While this assumption greatly simplifies the design of a telerobotic system, it also is often a poor assumption to make since it is very difficult to create a synthetic human-machine interface (HMI) that gives a remote operator the same level of information that a traditional operator would have sitting atop the vehicle. Even if all the requisite environmental information is conveyed remotely, it may not be equally intuitive to different human operators. To compensate for these likely shortcomings of HMI's, one can implement features that stabilize the vehicle's motion and that avoid hazards automatically.

SAMBUCA's augmented teleoperation consisted of the TeleOp application providing mathematically smoothed steering signals into the SchemaController teleoperation module, which then was mixed with steering signals from the avoid-obstacle module. In this way, SAMBUCA was able to avoid obstacles despite a remote human operator steering toward the hazard. The demonstrations were again conducted under tethered conditions since the wireless functionality of the PC/104 was experiencing troubles at the time.

Goalless Autonomous Behavior

In terms of motor schema control, autonomous behavior occurs when the system recognizes some goal to achieve. In the test plan for SAMBUCA, the goal was to be a visual target that would be acquired by a vision system, which in turn would create an attraction for the platform to drive towards the goal. However, no vision system had been implemented at the time of testing, so a simulated attractive force

[19]The kinds of information that are most useful in teleoperation are often visual, but force-feedback control and other interfaces are being investigated by human-machine interface developers.

was added to the SchemaController software.

The simulated attractive force was implemented as a fixed-value forward motion vector that, if not mixed with other behavioral vectors, would cause SAMBUCA to travel forward at a fixed speed in a straight line until it collided with something in the way. When combined with the obstacle avoidance behavior vector, however, the resulting motion is primarily forward with obstacles causing a redirection in the robot's heading. A true attractive force vector would re-orient the robot toward the goal after an obstacle was avoided.

As expected, SAMBUCA encountered scenarios whereby its forward-motion vector was perfectly opposed by its obstacle avoidance behavior, resulting in a net motion vector of zero and SAMBUCA getting stuck[20]. To help combat this problem, the random-motion behavior was enabled and mixed in with the existing behaviors. The random motion vector works by periodically generating a motion vector within a predetermined range of values in order to move SAMBUCA out of any local minima traps it encounters.

While the random-motion broke SAMBUCA out of most local-minimum traps, it worked using a fixed range of allowable vector values, with each random vector being held constant for five seconds before being replaced. This approach was acceptable for getting SAMBUCA out of traps within open spaces, but resulted in crude maneuvers in tight quarters where finer random motion vectors would have been preferred. These observations made apparent the importance of a robot knowing its surrounding environment. Had SAMBUCA been aware that it was in a tight space, the random distribution used to break out of local minima could be more centered about the mean, resulting in control with the needed finesse. Despite this lack of finesse in tight spaces, the vector-field approach to navigation was shown to work, given the limited accuracy and range of the sensors used, and the absence of a move-to-goal behavior.

[20]This is known as a local minima trap, and is also mentioned in section 3.3.6 .

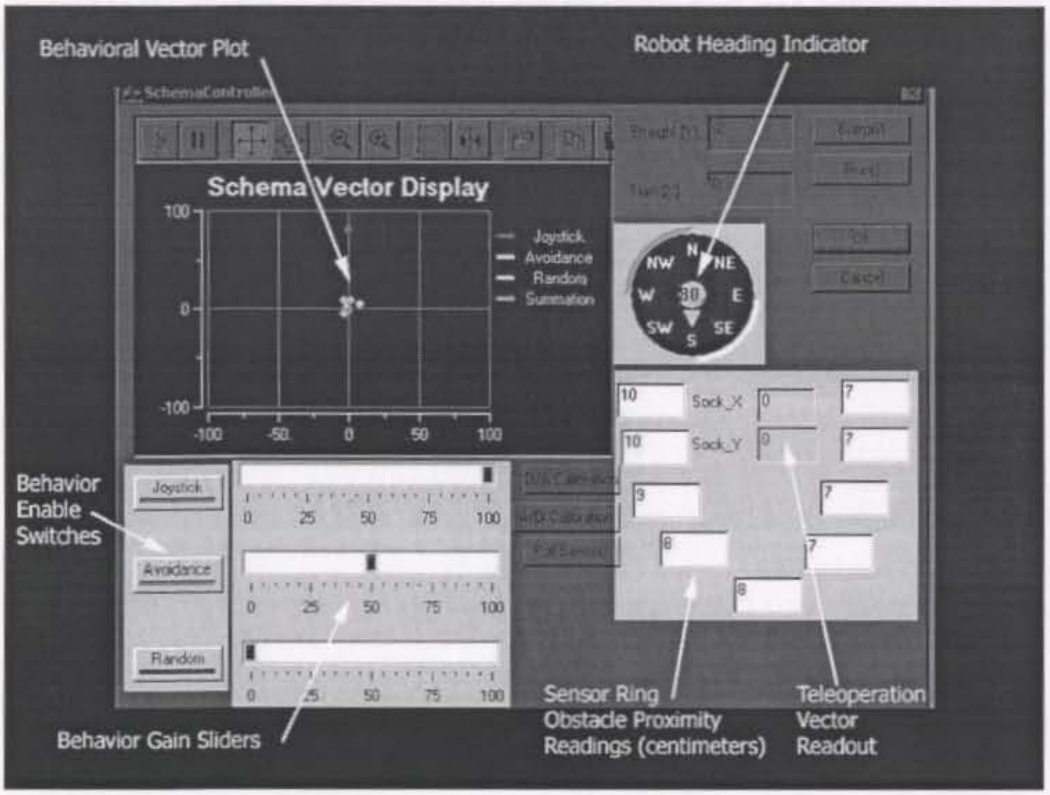


Figure 5.53: Screenshot of the SchemaController software interface

5.3 Chapter Summary

This chapter presented the implementation of the ISLab's mobile robotic platform, SAMBUCA. The electromechanical base, mounting plates, and sensors, which constituted the physical base, were each explained. Descriptions of the control system hardware followed, with focus on the PC/104 components and the RAD Board low-level controller. This was followed with a discussion of the software implementation of motor schema behavioral control, showcasing the designs of the SchemaController and TeleOp applications. The section on SAMBUCA's controller was concluded with a description of the wireless communication system used to communicate within the General Framework for Group Robotics.

The implementation of SAMBUCA illuminated some problems, including sensor noise, powerchair electronics peculiarities, power distribution concerns, and motion drift. The demonstration section presented testing of the SAMBUCA platform in conjunction with both PC/104 and RAD Board controller systems. Pure teleoperation, tethered augmented teleoperation, and goalless autonomous behavior tests were all carried out. A summary of SAMBUCA's properties and preliminary test results are shown in figure 5.54.

The performance tests and implementation activities both revealed some areas for future work. These recommendations are presented along with the conclusions of this thesis, in the following final chapter.

Property	Comments
Test Locations	Indoors at the ISLab
Speed Range	0 m/s to approx. 1.5 m/s
Turning Radius	Zero-Turning-Radius robot
Obstacle Avoidance	Widely-spaced obstacles easily avoided Narrowly-spaced obstacles often hit Oblique encounters most successful Direct encounters require added random motion
Pure Teleoperation	Behavior similar to using the original R2 joystick
Augmented Teleoperation	Prevents human operator from steering robot into obstacles
Goalless Wandering	Sometimes spent 5 to 20 seconds in local minima traps before escaping
Repeatability	Exact robot paths not repeatable due to random motion components(when enabled), and due to force-resistive effects of the caster wheels

Figure 5.54: Summary of Preliminary Test Results

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

Industrial automation is a cornerstone of intelligent systems research, and the automation of equipment in hazardous environments presents significant research opportunities and challenges. While many industries could benefit from using a team of intelligent machines to carry out tasks, it was in specific response to the needs of the mining industry that the ISLab embarked on its group robotic system. The high-level architecture of the system, called the "General Framework for Group Robotics" partitioned the project into following subsystems:

- Human-Machine Interface [Human supervision and intervention]
- Dynamic Scheduler [Task-robot allocations]
- Petri-Net Controller [Task coordination]
- Mobile Robotic Platform [Task execution by individual robots]

These subsystems inter-operated using established communication protocols and toolkits, as mentioned in section 4.6. The Petri-net controller was implemented by Hwang and was tested using modified remote-control toys [1]. While the use of toys proved the concept of robotic task coordination, future research demanded a more rugged and versatile mobile robot platform that would receive task-level dispatches

from the Petri-net controller, and carry out those tasks semi-autonomously. In addition to these demands, the platform was to be able to operate indoors in a created laboratory environment as well as outdoors in a less structured natural environment.

The intent for building this platform was to provide an adaptable low-cost robotic vehicle for testing ideas in group robotic research, while representing the control challenges posed to robot automation systems and remote human operators that would normally be experienced with industrial-grade intelligent machines. The resulting mobile robot platform, SAMBUCA, is a rugged and operationally flexible machine with the following attributes:

- Modular design: easy to modify and repair
- Electrically Powered: operable indoors, no emissions
- Autonomous or Remotely-Controlled Operation
- Payload capacity of over 100 kg
- Self-contained power supply for the base and payload
- Rugged enough to operate outdoors in good weather
- PC-based onboard computer
- Networked using wireless LAN technology
- Anti-Runaway Safety Feature

SAMBUCA has been tested with the RT200 manipulator arm as payload, in conjunction with a webcam-based vision system, with the arm being powered from the platform's own power supply. Furthermore, SAMBUCA has been tested in indoor environments with a motor-schema control architecture and has shown its ability to be remotely controlled, and to wander autonomously while avoiding obstacles detected by an infrared proximity sensor ring.

The work done to date has shown the utility of the physical platform and of its schema-based controller. Specifically, the implementation of motor schemas is central to providing the platform its ability to effectively maneuver in a number of

situations, and the addition of wireless teleoperation and behavioral mixing helps in the development of effective autonomous robotic modes of operation.

SAMBUCA, as described in this thesis, is a working mobile robot platform, developed primarily for use in the "General Framework for Group Robotics" but flexible enough to be used as a test platform for other robotics research in the future (e.g. sensor fusion, advanced control systems research, schema theory). The platform possesses elementary semiautonomous navigational capabilities, however these capabilities can be improved upon by trying different sensors and schemas, and by adding mapping and machine-learning modules to the controller. The focus of future work needs to be in further tuning of the current navigation system, and the integration of this platform into the rest of the "General Framework" using an appropriate interprocess communication toolkit, such as Jini™. Development of an effective Human-machine interface should complement the inclusion of this platform into the "General Framework", incorporating the features of the existing TeleOp application.

The current implementation of SAMBUCA has provided the ISLab with a higher caliber of machine with which to carry out future research in group robotics, but is an initial version that should be improved upon as the group robotics project continues.

6.2 Recommendations

This section outlines some improvements on SAMBUCA that merit future investigation.

6.2.1 Motor Drive Power Electronics

Nearing the end of development on SAMBUCA, the R2 power electronics (see figure 6.55) that came with the powerchair failed. A cursory investigation of these electronics revealed no apparent damage to the power transistors on either the left-motor or right-motor H-Bridges. It was decided that further troubleshooting not be pursued since the complexity of the circuit board[1] made hand-testing ineffective. Although these electronics worked in conjunction with the PC/104 system and the

[1]The motor drive power electronics were manufactured using a four-layered printed circuit board with many unreachable signal paths and few test points.

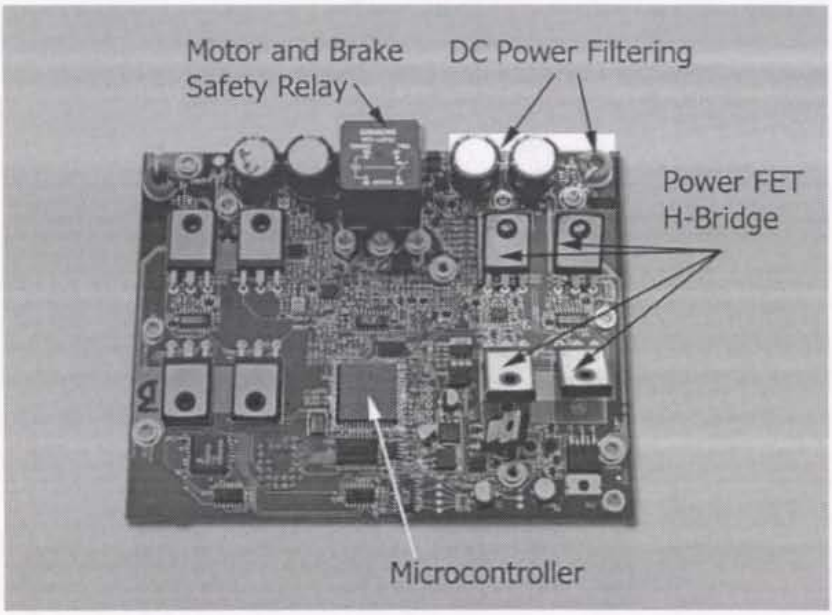


Figure 6.55: Motor Drive Power Electronics

SchemaController software without incident for over six months, it is possible that an overvoltage from the PC/104's DAQ card may have caused the failure of the power electronics. The cost of a new module was estimated to be between $700 and $1000 Canadian dollars, which was deemed too expensive to replace.

Rather than replace the power electronics with a factory replacement, an investigation was made into using a newer model that could be digitally interfaced directly with the PC/104 rather than using an analog voltage interface as was previously done. The writings of Bourhis *et al.* suggested the use of a *DX* controller manufactured by Dynamic Controls[2] [68]. Further investigation revealed that the DX system development kit was no longer supported, and the interface specifications were not available to interested developers [69], [70].

A third alternative to the replacement of the power electronics would be to design the electronics in-house. The design would not need to use a microcontroller[3] or so

[2]Dynamic Controls, New Zealand, was the company that manufactured the R2 power electronics module under licence from Invacare Corporation.

[3]A Motorola microcontroller was used on the R2 power electronics in order to control motor

many supporting chips, but could be made similar to Blanchard's H-Bridge design[4] (see appendix figure B.4) [71] except with properly-rated components[5].

6.2.2 Onboard Computer

The computer system onboard SAMBUCA is a PC/104 system that was initially chosen because it was to be shared with a second, smaller robot which has since been canceled[6]. While PC/104 systems offer PC-compatible performance in a small form factor, and draw little power, they have the following drawbacks that seriously impede development:

- A state-of-the-art PC/104 system is roughly 4 generations of improvement behind a state-of-the-art desktop/laptop PC. Example: 266 MHz Pentium II was the best CPU available on a PC/104 when the best CPU for desktop/laptop systems was a 1 GHz Pentium IV.

- The PC/104 and PC/104-Plus standards are not as standard as advertised. Often one module will not be able to stack on top of another module because of a blocked pin receptacle. Other times, a module will exceed the 'standard PC/104' module dimensions, preventing the use of supporting rails or ruggedized PC/104 containers.

- PC/104 system setup may be possible only with expensive vendor-supplied setup hardware. Example: The CPU module's BIOS is incapable of being simultaneously connected to a CD-ROM drive and a hard disk drive, thereby making installation of an operating system literally impossible without the use of auxiliary hardware (as mentioned in section 5.1.3).

- PC/104 modules are difficult to source, service, and replace. Since the market for PC/104 is markedly smaller than for conventional desktop/laptop systems,

---

acceleration profiles and to implement safety features such as the one mentioned in section 5.1.7.

[4]The H-Bridge circuit design is presented in the appendix with the permission of its author.

[5]The transistors used in the R2 power electronics modules were power MOSFETs rated for 70A, 60V in order to safely supply the powerchair motors

[6]SmallBot (see section 1.1.5) and SAMBUCA were originally to share the PC-compatible portable computer, and SmallBot imposed a size constraint, hence why the small PC/104 was chosen instead of a laptop

it is harder to find suppliers, to find out which modules are compatible with each other, and to have service warranties on individual modules or on the system as a whole.

Once a PC/104 is operational, it behaves like a laptop machine, but lacking the mouse, keyboard, and display. Thus, future developers should consider replacing the PC/104 computer system with a well-equipped laptop system. Laptops, as well as not experiencing the PC/104 symptoms presented above, provide the following features to developers:

- Laptop computers are readily and locally available.
- Laptops are small and portable, especially in comparison with the SAMBUCA platform.
- Laptop systems are less expensive than their equivalently-functional PC/104 counterparts.
- Laptops are equipped with PCMCIA slots (and therefore can be connected to wireless Ethernet and certain data acquisition units), a parallel port, one or two serial ports, typically two USB ports (which can be further split into as many as 128 ports sharing the overall USB bandwidth), as well as outputs for external monitors and user-interface control devices.
- Laptops have a more resilient construction than PC/104 systems.

6.2.3 Proximity Sensors

Currently, SAMBUCA uses infrared proximity sensors which have a detection range from 10 cm to 80 cm. Polaroid's ultrasonic range sensors, for sake of comparison, have an obstacle detection range from 15 cm to 1065 cm, at the expense of a wider detection beam and more post-processing than the infrared sensors. As was stated in section 5.1.2, ultrasonic rangefinders were not used on SAMBUCA because they required more control lines and processing to operate using a DAQ card. However, having a dedicated circuit to pre-process readings from a ring of sonar sensors would result in a system that could bypass the DAQ card entirely and be fed as a serial 'results' stream to the PC.

By closely spacing ultrasonic rangefinders so that their wide pickup patterns overlap, processing could be done to more precisely locate obstacles in the collective field of sensor view. As well, ultrasonic and infrared proximity sensors could be combined so that their complementary obstacle detection properties could be fused to aid robot navigation behaviors. This could be part of a larger project utilizing SAMBUCA for research into *sensor fusion* [72].

6.2.4 Wheel Encoders

While motion encoders were added to both of SAMBUCA's drive motors (as mentioned in sections 5.1.2 and 5.1.7), they were never read directly by the DAQ card because of the excessive proportion of DAQ resources it would require, compared to other DAQ inputs and outputs. Again, offloading the processing of quadrature encoder signals from the DAQ onto a dedicated circuit is one possible solution. The dedicated circuit could deduce wheel travel either by synchronously sampling the quadrature signals (as the DAQ would otherwise do) or by *asynchronously* triggering using the quadrature signals' edges. In both cases, the perceived encoder transitions would feed into a state machine which would increment or decrement a counter with an adequately-sized register until the PC-compatible computer requested the count. This would free PC system processor power for more important things such as video processing, and running the motor schemas.

6.2.5 Additional Sensors

Tilt Sensor

Depending on the type of payload aboard the mobile platform, tilt readings may be needed to ensure that tipping does not occur on certain terrain. This precaution is more necessary when, for example, the RT200 manipulator arm is used with the platform, since the center of gravity rises resulting in lower stability. One sensor that was reviewed was the Model 900-series biaxial clinometer (see figure 6.56, manufactured by Applied Geomechanics Incorporated) which can measure tilt about two orthogonal axes, and thus can be mounted to measure the roll and pitch of the platform. The sensor outputs analog voltages for each tilt reading, which can be read

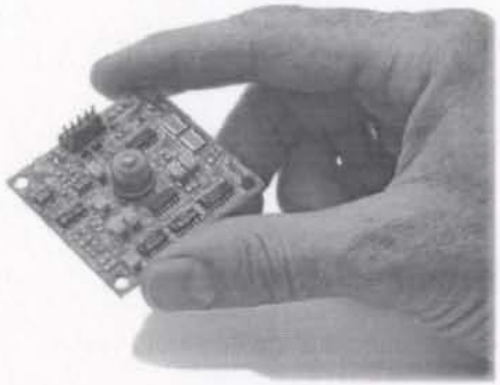


Figure 6.56: Biaxial Clinometer [17]

and processed by the existing DAQ card, included in the motor schema controller, and sent back to a remote human operator if needed.

Force Feedback Signals

Currently, the input device used with the TeleOp application is a Microsoft *Sidewinder* (see figure 6.57) which has the capability of providing force feedback to the human operator. While these joysticks are typically used in video games to simulate jarring, bumping, and mechanical resistance, this feature can enhance telepresence during remote operation of the platform. While it is possible to affix accelerometers to the platform to measure terrain-induced vibrations that can then be expressed remotely using the joystick, it is also possible for the *avoid-obstacle* behavior's virtual force field to be manifested as a real force that can help guide the human operator during teleoperation.

Battery Meter

Perhaps one of the most important proprioceptive sensors aboard a robot is its fuel meter. In the case of the electrically-powered SAMBUCA platform, this can be accomplished by using commercially available battery monitor integrated circuits which use a battery chemistry model to determine the percentage of charge remaining. A

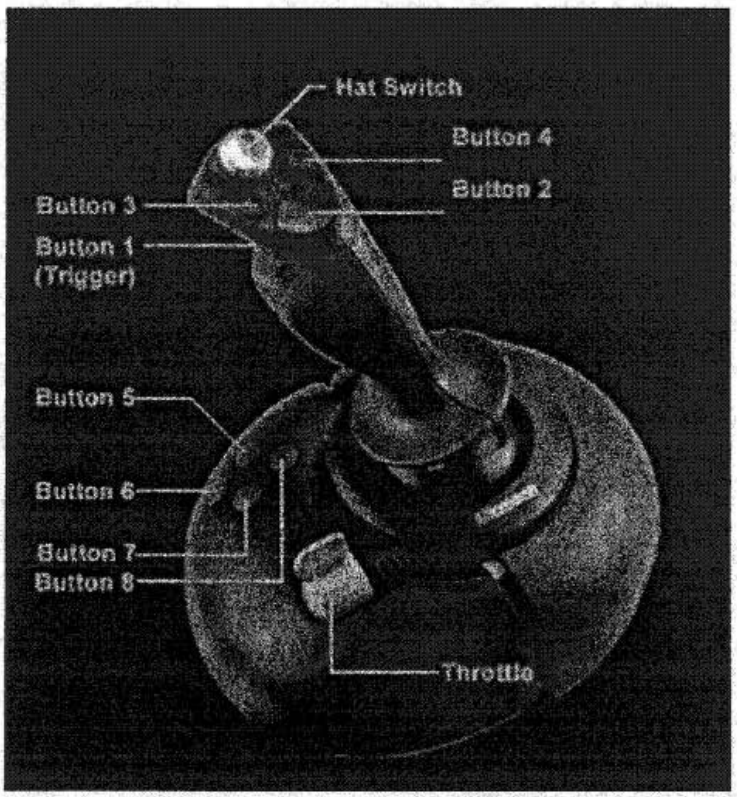


Figure 6.57: Force Feedback Joystick [18]

simpler 'homemade' method (see figure 6.58) is to monitor the terminal voltage by comparing fractions of it (using an adjustable-range voltage-divider ladder) to a reference voltage, likely provided by a Zener diode or a regulator chip. The battery status then can be interpreted by reading the outputs of the comparators. To prevent electrical overvoltages from damaging the digital electronics which read the status levels, the comparator outputs could be passed through opto-isolator units before reaching the computer system. While the PC/104 system's DAQ card could read in this value, it was not considered prudent to read in the voltage of a high-current source (ei.e. the battery) alongside low-current signals since possible short-circuits would be catastrophic. The simple battery meter circuit in essence was a sacrificeable level converter that could provide battery level information.

6.2.6 Intermodule Hardware Communication

Sections 6.2.3 and 6.2.4 both hint toward the decentralization of sensor processing from the PC system to peripheral circuitry. As with any peripheral device, there has to be an effective and preferably standard way of communicating information to and from the core PC system. While devices such as the Miniboard and the RAD-Board have used serial RS-232 communication as their standard, there are newer standards such as USB which have bandwidth and connectivity advantages over RS-232 (see section 6.2.2). Another approach worth investigating is use of the Control Area Network Bus architecture, commonly known as the 'CAN' bus. According to the specification, this serial communication protocol "efficiently supports distributed realtime control," which essentially is the requirement here [73]. Furthermore, the INCA Laboratory[7] has begun using the CAN bus, so local support and working examples would be available.

6.2.7 Safety Measures

In addition to the existing anti-runaway watchdog circuit which shuts down the motors in case of an operating system failure (see section 5.1.5), a physical kill switch could be implemented in case of any general emergency. Furthermore, conventional

[7]The Center for Instrumentation, Control and Automation, Faculty of Engineering and Applied Science, Memorial University of Newfoundland.

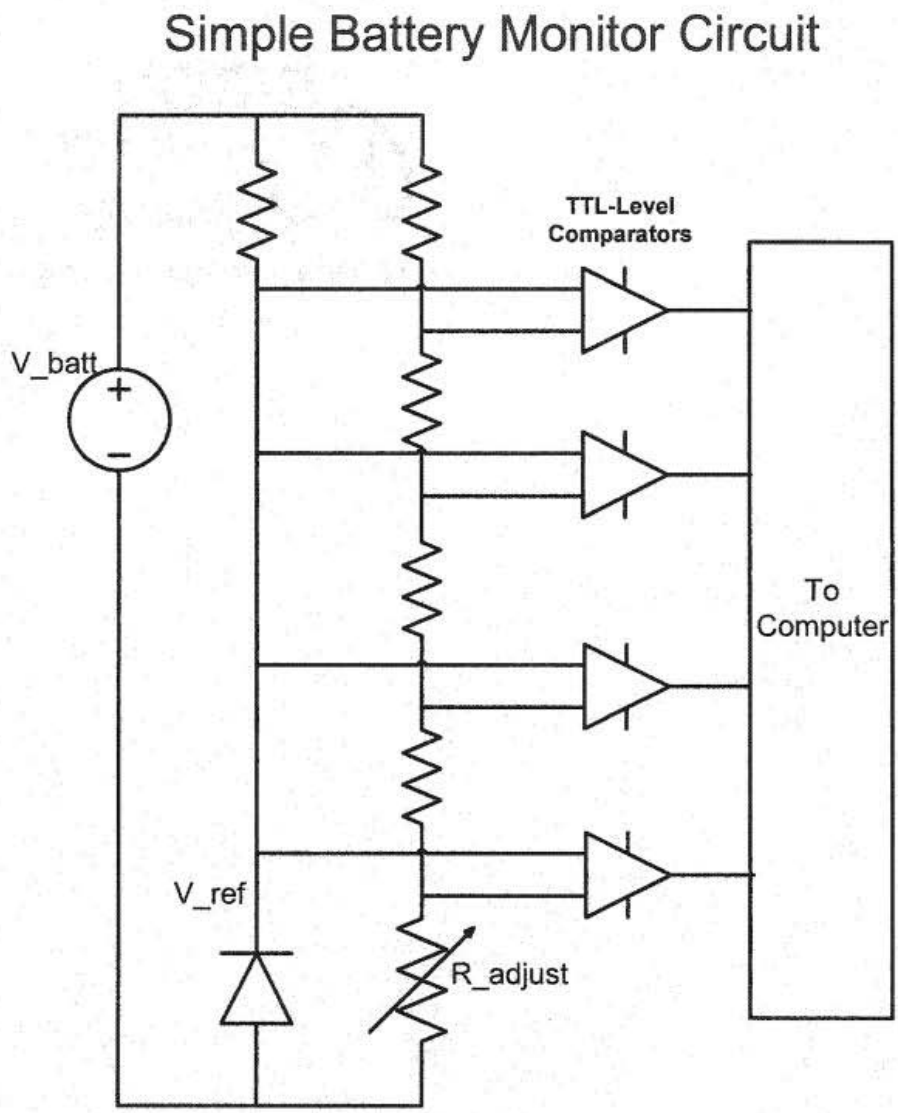


Figure 6.58: Simple Battery Monitor Circuit

radio remote control (i.e. 27 MHz or 49 MHz R/C) could be integrated in over to override the onboard computer, however it is more effective to simply re-connect the Pronto R2 wired joystick to the platform if such a case should arise. If deemed necessary, warning lights and buzzers can be added to alert bystanders that the platform is operating.

6.2.8 Goal Driven Autonomous Behavior

The motor schema controller implemented to date has demonstrated the ability to avoid obstacles while wandering or while being teleoperated. A key ingredient to making the base semi-autonomous is to add goal-detection and goal-attraction capacities to the controller. By far, the most information-laden sensory system is the vision system, and the incorporation of a vision system that can detect goals, obstacles, wall edges, or other important environmental features will be an important addition to the motor schema controller, as it will permit more 'intelligent' behaviors to emerge.

Vision System Hardware

As became apparent in the course of the platform's design, a vision system based on PC/104 image-capturing (frame-grabbing) modules, while providing good quality images, suffers from the following drawbacks:

- Image processing limited by slow PC/104 CPU module
- Requires expensive camera hardware
- Vision system development restricted to PC/104 system (which often has substandard video display capabilities anyway)

In comparison *webcams* offer an acceptable though lower-quality image capturing capability. In addition to being inexpensive, they are easily connected to PC-compatible systems often via USB. The vision system successfully tested by Baxter Smith used a webcam and was processed using a desktop PC [63]. The benefit of going this route in the future is that webcams can be connected to laptops (whose benefits are listed in section 6.2.2) can be purchased at any local computer store,

and can provide surprisingly good results given the price point. Furthermore, development of the vision system can be done on a desktop PC system with the said webcam attached, which often is more convenient for the developer.

Vision System Software

Regardless of the specific vision system hardware that is used, the choice of software for processing captured images can greatly affect the ease and speed of vision system development. There are a number of imaging APIs (application programming interfaces) available to the developer, but many effective APIs require the developer to pay a licensing fee in order to use them. Alternatively, OpenCV (Open-source Computer Vision libraries, overseen by the Intel Corporation) offers a freely-distributed set of image analysis algorithms that can be incorporated into a future vision system aboard SAMBUCA. The advantage of going with an open imaging library is that there tend to be more developers worldwide using the libraries (and some who have even written portions themselves), who can then provide support-in-kind to other developers.

6.3 Chapter Summary

This concluding chapter recapped the defining features of SAMBUCA as well as provided recommendations for future work. The suggestions to replace the motor drive power electronics, to move away from PC/104 hardware, and to redesign the wheel encoder electronics were based on experiential knowledge gained from building and using SAMBUCA. Other recommendations presented in this chapter are meant as future additions to the platform, and these include the incorporation of a navigational vision system, proprioceptive sensors, and the use of the CAN bus for SAMBUCA's subsystem communications.

SAMBUCA was designed to sense its environment and be controlled either semi-autonomously or by human teleoperation.

Although the failure of the motor drive circuitry prevented further testing of SAMBUCA, tests conducted beforehand showed encouraging results regarding the operational qualities of the platform and its control software.

The SAMBUCA design has shown early successes and can be easily built upon due to its modular construction. This mobile robotic platform has achieved its mail goals of being a low-cost platform on which to base future rugged semi-autonomous indoor/outdoor robots. Having begun from using computer simulations to using toy-based simulations, the ISLab can now use the SAMBUCA MRP as an effective research tool for the next stage of group robotics research in natural environments.

References

[1] F. Hwang, "A petri-net on-line controller for the coordination of multiple mobile robots," Master's thesis, Memorial University of Newfoundland, 2000.

[2] "Pioneer 2-AT photograph." `http://www.activrobots.com/PICTURES/ATwCoke.gif`, February 2002.

[3] "Sarge photograph." `http://www.sandia.gov/isrc/SARGE/SARGE.jpg`, February 2002.

[4] "Lewis cyberatv photograph." `http://www-2.cs.cmu.edu/afs/cs/project/cyberscout-12/ATV/images/lewis.jpg`, February 2002.

[5] "Tinman robot photograph." `http://www.ai.mit.edu/people/holly/wheelesley/wheelesley.gif`, February 2002.

[6] "Navchair photograph." `http://www-personal.engin.umich.edu/~johannb/navchair.jpg`, February 2002.

[7] R. C. Arkin, *Behavior-Based Robotics*. MIT Press, 1998.

[8] "AI Glossary, University of Georgia, College of Agricultural and Environmental Sciences.." `http://www.uga.edu/caes/AI/images/neuralnetwork.gif`, February 2002.

[9] "Argo bigfoot photograph." `http://www.argoatv.com/images/bigfoot-02.jpg`, February 2002.

[10] L. Champeny-Bares, L. S. Coppersmith, and K. Dowling, "The terregator mobile robot," Tech. Rep. CMU-RI-TR-93-03, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 1991.

[11] "Pc/104 module photograph, Parvus Corporation." http://www.parvus.com/Datasheet/viewPrimaryPicture.asp?id=PRV-0696X-01&isPrimaryPicture=1&size=large&type=jpg, February 2002.

[12] "MiniBoard photograph, Chuck's Robotic Notebook." http://www.mcmanis.com/chuck/robotics/controllers/images/miniboard.jpg, February 2002.

[13] Invacare Corporation, "R2 powerchair electronics interface. Correspondence with Mr. John Mattes (Invacare engineer) over telephone and fax, July 2000."

[14] "Polaroid 6500 Ranging Module photograph, Acroname, Inc.." http://www.acroname.com/robotics/parts/R11-6500.jpg, February 2002.

[15] "Ultrasonic transducer photograph, Polaroid Corporation." http://www.polaroid-oem.com/images/usp-7.gif, February 2002.

[16] "Portawattz 300 Power Inverter, Xantrax Technology, Inc.." http://www.statpower.com/images/pwtz300.gif, February 2002.

[17] "Applied Geomechanics Model 900 Biaxial Clinometer." http://www.geomechanics.com/pdf/900.pdf, February 2002.

[18] "Buttons - Force Feedback 2 - Microsoft Sidewinder, Microsoft Corporation.." http://www.microsoft.com/hardware/sidewinder/_graphics/Devices/ffb2/swf%f2_callouts.gif, February 2002.

[19] "Precarn incorporated: Intelligent systems technologies." http://www.tomoye.com/simplify/precarn/ev.php?URL_ID=1340&URL_DO=DO_TOPIC&URL_SECTION=201&reload=1008794490, January 2002.

[20] "Precarn incorporated: Sensori-motor augmented reality for telerobotics (smart)." http://www.tomoye.com/simplify/precarn/ev.php?URL_ID=1054&URL_DO=DO_TOPIC&URL_SECTION=201&reload=1008696741, February 2002.

[21] G. Baiden, R. Strom, and G. Preston, "Mining automation program." http://www.inco.com/about/telemining/baiden.htm, 1996.

[22] G. Baiden, R. Strom, and G. Preston, "Mining automation program." http://www.inco.com/research/minesresearch/telemining.asp, 1996.

[23] G. Baiden, *A Study of Underground Mine Automation.* PhD thesis, McGill University, March 1993.

[24] A. Lassó, T. Urbancsek, and A. Helybély, "Microrobot teleoperation through WWW." http://www.hszk.bme.hu/~s7373las/kut/MicroCad01_LassoUrbancsekHelybely.pdf, 2001.

[25] J. King, R. Hale, F. Hwang, J. Seshadri, M. Rokonuzzaman, and R. Gosine, "A general framework for group robotics with applications in mining," in *Proc. SPIE, Mobile Robots XV and Telemanipulator and Telepresence Technologies VII*, vol. 4195, SPIE, November 2000.

[26] F. Martin, *Miniboard 2.0 Technical Reference*, 1992.

[27] L. Smith, "Rad card," tech. rep., Memorial University of Newfoundland, St. John's, NF, April 2000.

[28] H. Everett, G. Gilbreath, T. Heath-Pastore, and R. Laird, "Controlling multiple security robots in a warehouse environment," 1994.

[29] "K-team::koala robot." http://www.k-team.com/robots/koala/index.html, February 2002.

[30] "Activmedia robotics, llc." http://www.activrobots.com/ACTIVMEDIA/index.html, July 2001.

[31] "P2at – the all-terrain robot." http://www.activrobots.com/ROBOTS/p2at.html, February 2002.

[32] "iRobot Corporation." http://www.irobot.com/rwi/default.asp, February 2002.

[33] "Online demos: Obstacle detection." http://www.ai.polymtl.ca/webDemos/quicktime/ObstacleDetection_High.mov, February 2002.

[34] "Online demos: Guideless autonomous navigation." http://www.ai.polymtl.ca/webDemos/quicktime/GuidelessAutonomousNavigation_High.mov, February 2002.

[35] "Applied AI systems, inc. - products - robots." http://www.aai.ca/products/robots/index.html, February 2002.

[36] Cmdr. Bart Everett (Rtd.), "Introduction: Autonomous security robots." http://www.nosc.mil/robots/land/robart/history.html, July 2001.

[37] J. Pletta and J. Sackos, "An advanced unmanned vehicle for remote applications." http://citeseer.nj.nec.com/pletta98advanced.html, 1998.

[38] "CyberATV." http://www-2.cs.cmu.edu/afs/cs/project/cyberscout-12/ATV/, February 2002.

[39] "The tin man wheelchair robots." http://www.kipr.org/robots/tm.html, February 2002.

[40] "Wheelesley: Development of a robotic wheelchair system." http://www.ai.mit.edu/people/holly/wheelesley/, February 2002.

[41] I. Ulrich and I. Nourbakhsh, "Appearance-based obstacle detection with monocular color vision," in *Proceedings of the AAAI National Conference on Artificial Intelligence*, AAAI, July/August 2000.

[42] D. Bell, J. Borenstein, S. Levine, Y. Koren, and A. Jaros, "The navchair: An assistive navigation system for wheelchairs, based on mobile robot obstacle avoidance," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, IEEE, May 1994.

[43] E. Gat, "On three-layer architectures." http://citeseer.nj.nec.com/gat97threelayer.html, 1997.

[44] R. A. Brooks, *Cambrian Intelligence*. MIT Press, 1999.

[45] N. Nilsson, "Shakey the robot," Tech. Rep. 323, SRI International, Menlo Park, CA, 1984.

[46] H. Moravec, "The stanford cart and the cmu rover," in *Autonomous Robot Vehicles* (I. J. Cox and G. T. Wilfong, eds.), pp. 407–41, Springer-Verlag, 1990.

[47] R. C. Arkin and T. R. Balch, "Aura: principles and practice in review," *JETAI*, vol. 9, no. 2-3, pp. 175–189, 1997.

[48] E. Gat, "Integrating planning and reaction in a heterogeneous asynchronous architecture for controlling mobile robots," in *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI), San Jose, California*, pp. 809–815, AAAI, July 1992.

[49] J. Connell, "Sss: A hybrid architecture applied to robot navigation," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, France*, pp. 2719–2724, IEEE, May 1992.

[50] M. J. Matarić, "Behavior-based control: Main properties and implications," in *Proceedings, IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems, Nice, France, May 1992.*, pp. 46–54, 1992.

[51] S. Baluja, "Evolution of an artificial neural network based land vehicle controller," in *IEEE Transactions on Systems, Man, and Cybernetics, Vol. 26, No. 3*, pp. 450–463, IEEE, June 1996.

[52] A. Weitzenfeld, R. Arkin, F. Cervantes-Perez, R. Olivares, and F. Corbacho, "A neural schema architecture for autonomous robots," 1998.

[53] R. Arkin, A. Kahled, A. Weitzenfeld, and F. Cervantes-Perez, "Behavioral models of the praying mantis as a basis for robotic behavior." `http://citeseer.nj.nec.com/arkin00behavioral.html`, 2000.

[54] R. Arkin, F. Cervantes-Perez, and A. Weitzenfeld, "Ecological robotics: A schema-theoretic approach." `http://citeseer.nj.nec.com/arkin98ecological.html`, 1998.

[55] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots.," in *IEEE Transactions on Systems, Man, and Cybernetics, Vol. 19, No. 5*, pp. 1179 – 1187, IEEE, September / October 1989.

[56] Applied AI Systems, Inc., "Atrv-2 price quote. Personal correspondence by email, February 6, 2002."

[57] Electric Vehicles of America, Inc., "Electric vehicle design investigation. Correspondence with Mr. Bob Batson over telephone and fax, June 2000."

[58] "Robotic researchers play the field." `http://www.monash.edu.au/pubs/eureka/Eureka_95/robot.html`, July 2001.

[59] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns*. Reading, MA: Addison Wesley, 1995.

[60] A. Winfield and O. Holland, "The application of wireless local area network technology to the control of mobile robots," 2000.

[61] A. Winfield, "Wireless video tele-operation using internet protocols," in *Proceedings of the 14th International Conference on Unmanned Air Vehicles*, April 1999.

[62] "Technical Specifications for 6500 Series SonarRanging Module, Polaroid Corporation." `http://www.polaroid-oem.com/pdf/6500series.pdf`, February 2002.

[63] B. Smith, "Vision-based robot control," tech. rep., Memorial University of Newfoundland, St. John's, NF, April 2001.

[64] "Diamond-MM-32-AT User Manual, Diamond Systems Corporation." `http://www.diamondsystems.com/files/binaries/dmm32v25.pdf`, February 2002.

[65] D. Filipp, "Cthread - a worker thread wrapper class." `http://www.codeproject.com/threads/cthread.asp`, 1999.

[66] "CurveFit v1.3, Daniel G. Hyams." `http://www.ebicom.net/~dhyams/cvxpt.htm`, February 2002.

[67] J. Gowdy, "A qualitative comparison of interprocess communications toolkits for robotics," Tech. Rep. CMU-RI-TR-00-16, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, June 2000.

[68] G. Bourhis, O. Horn, O. Habert, and A. Pruski, "An autonomous vehicle for people with motor disabilities," *IEEE Robotics & Automation Magazine*, vol. 8, pp. 20 – 28, March 2001.

[69] Université de Metz, France, "DX-system powerchair electronics interface. Correspondence with Mr. Guy Bourhis, (Researcher) over email, January 2002."

[70] Dynamic Controls, New Zealand, "DX-system powerchair electronics interface. Correspondence with Mr. Graham Keen (Commercial Manager) over email, January 2002."

[71] "N & P Channel MOSFET Single Supply H-Bridge, Eugene Blanchard." `http://www.cadvision.com/blanchas/hexfet/np-s.pdf`, February 2002.

[72] T.-S. Jin, J.-M. Lee, B. L. Luk, and S. K. Tso, "A study on multi-sensor fusion for mobile robot navigation in an indoor environment," in *8th IEEE International Conference on Mechatronics and Machine Vision in Practice, Hong Kong, 2001.*, pp. 455–458, 2001.

[73] Robert Bosch, GmbH, "CAN Specification Version 2.0," 1991.

Appendix A

Design Sketches

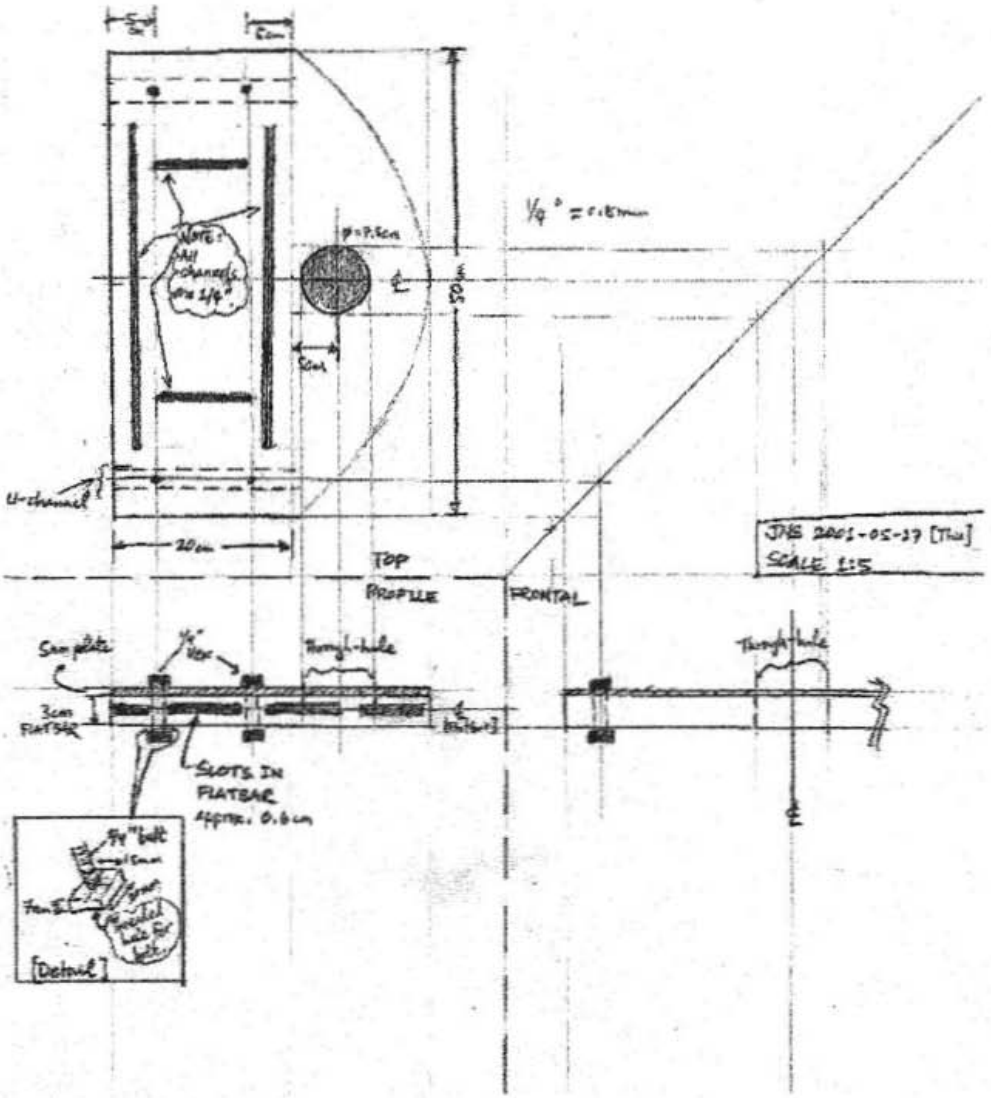


Figure A.59: Design Sketch for Mounting Plate

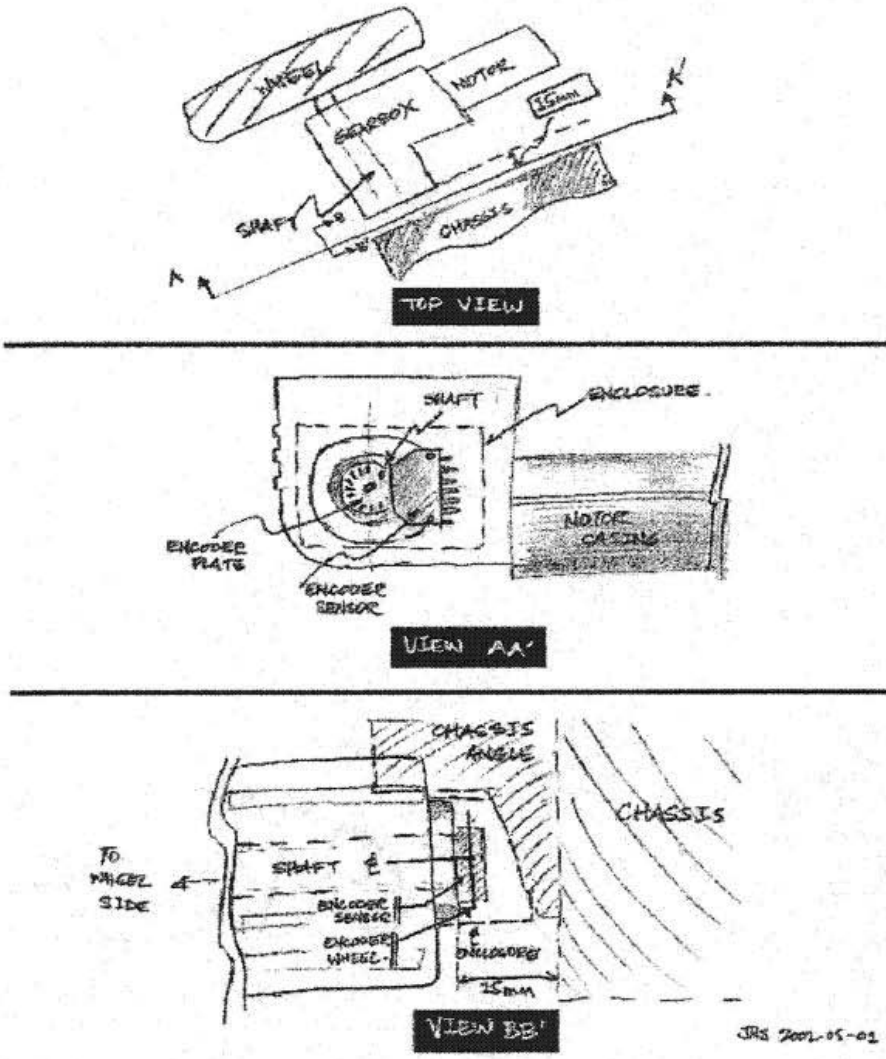


Figure A.60: Design Sketch for Wheel Encoder Mount and Enclosure

Appendix B

Datasheets

B.1 Agilent Wheel Encoder Sensors

Agilent Technologies
Innovating the HP Way

Two Channel Optical Incremental Encoder Modules

Technical Data

HEDS-9000
HEDS-9100

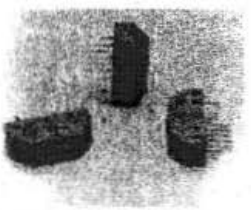
Features

- **High Performance**
- **High Resolution**
- **Low Cost**
- **Easy to Mount**
- **No Signal Adjustment Required**
- **Small Size**
- **-40°C to 100°C Operating Temperature**
- **Two Channel Quadrature Output**
- **TTL Compatible**
- **Single 5 V Supply**

Description

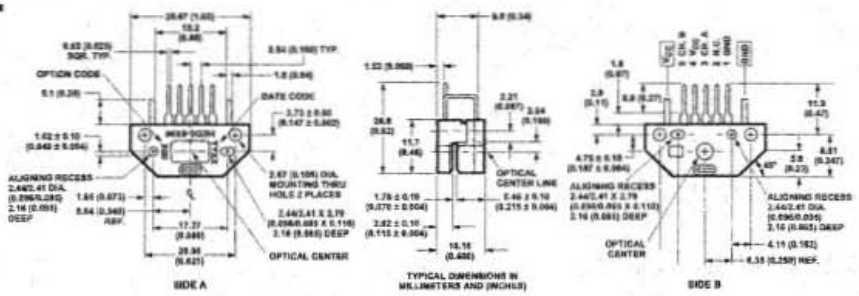
The HEDS-9000 and the HEDS-9100 series are high performance, low cost, optical incremental encoder modules. When used with a codewheel, these modules detect rotary position. The modules consist of a lensed (LED) source and a detector IC enclosed in a small C-shaped plastic package. Due to a highly collimated light source and unique photodetector array, these modules are extremely tolerant to mounting misalignment.

The two channel digital outputs and the single 5 V supply input are accessed through five 0.025 inch square pins located on 0.1 inch centers.



Standard resolutions for the HEDS-9000 are 500 CPR and 1000 CPR for use with a HEDS-6100 codewheel or equivalent.

Package Dimensions



ESD WARNING: NORMAL HANDLING PRECAUTIONS SHOULD BE TAKEN TO AVOID STATIC DISCHARGE.

For the HEDS-9100, standard resolutions between 96 CPR and 512 CPR are available for use with a HEDS-5120 codewheel or equivalent.

Applications

The HEDS-9000 and 9100 provide sophisticated motion detection at a low cost, making them ideal for high volume applications. Typical applications include printers, plotters, tape drives, and factory automation equipment.

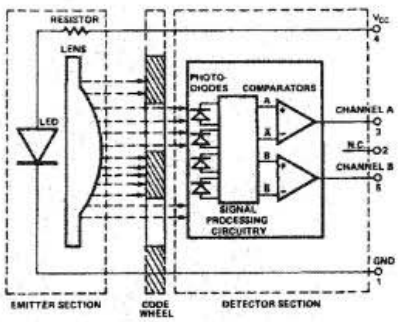
Theory of Operation

The HEDS-9000 and 9100 are C-shaped emitter/detector modules. Coupled with a codewheel, they translate the rotary motion of a shaft into a two-channel digital output.

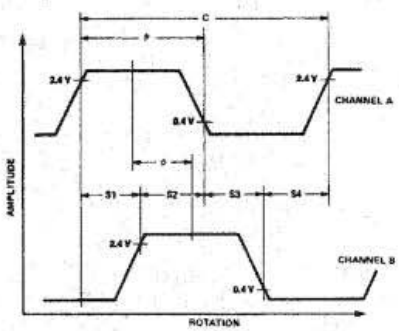
As seen in the block diagram, each module contains a single Light Emitting Diode (LED) as its light source. The light is collimated into a parallel beam by means of a single polycarbonate lens located directly over the LED. Opposite the emitter is the integrated detector circuit. This IC consists of multiple sets of photodetectors and the signal processing circuitry necessary to product the digital waveforms.

The codewheel rotates between the emitter and detector, causing the light beam to be interrupted by the pattern of spaces and bars on the codewheel. The photodiodes which detect these interruptions are arranged in a pattern that corresponds to the radius and design of the ode-wheel. These detectors are also spaced such that a light period on one pair of detectors corresponds to a dark period on the adjacent pair of detectors. The photodiode outputs are then fed through the signal processing circuitry resulting in A, A, B, and B. Two comparators receive these signals and produce the final outputs for channels A and B. Due to this integrated phasing technique, the digital output of channel A is in quadrature with that of channel B (90 degrees out of phase).

Block Diagram



Output Waveforms



Definitions

Count (N): The number of bar and window pairs or counts per revolution (CPR) of the codewheel.

1 Shaft Rotation = 360 mechanical degrees, = N cycles.

1 cycle (C) = 360 electrical degrees (°e), = 1 bar and window pair.

Pulse Width (P): The number of electrical degrees that an output is high during 1 cycle. This value is nominally 180°e or 1/2 cycle.

Pulse Width Error (ΔP): The deviation, in electrical degrees of the pulse width from its ideal value of 180°e.

State Width (S): The number of electrical degrees between a transition in the output of channel A and the neighboring transition in the output of channel B. There are 4 states per cycle, each nominally 90°e.

State Width Error (ΔS): The deviation, in electrical degrees, of each state width from its ideal value of 90°e.

Absolute Maximum Ratings

Storage Temperature, T_S ..... -40°C to 100°C
Operating Temperature, T_A ..... -40°C to 100°C
Supply Voltage, V_{CC} ..... -0.5 V to 7 V
Output Voltage, V_O ..... -0.5 V to V_{CC}
Output Current per Channel, I_{out} ..... -1.0 mA to 5 mA

Phase (ϕ): The number of electrical degrees between the center of the high state of channel A and the center of the high state of channel B. This value is nominally 90°e for quadrature output.

Phase Error (Δϕ): The deviation of the phase from its ideal value of 90°e.

Direction of Rotation: When the codewheel rotates in the direction of the arrow on top of the module, channel A will lead channel B. If the codewheel rotates in the opposite direction, channel B will lead channel A.

Optical Radius (R_{op}): The distance from the codewheel's center of rotation to the optical center (O.C.) of the encoder module.

Recommended Operating Conditions

Parameter	Symbol	Min.	Typ.	Max.	Units	Notes
Temperature	T	-40		100	°C	
Supply Voltage	V_{CC}	4.5		5.5	Volts	Ripple < 100 mV_{p-p}
Load Capacitance	C_L			100	pF	3.3 kW pull-up resistor
Count Frequency	f			100	kHz	$\frac{\text{Velocity (rpm) x N}}{60}$

Note: The module performance is guaranteed to 100 kHz but can operate at higher frequencies.

Encoding Characteristics

Encoding Characteristics over Recommended Operating Range and Recommended Mounting Tolerances. These Characteristics do not include codewheel/codestrip contribution.

Description	Sym.	Typ.	Case 1 Max.	Case 2 Max.	Units	Notes
Pulse Width Error	ΔP	30	40		°e	
Logic State Width Error	ΔS	30	40		°e	
Phase Error	Δϕ	2	10	105	°e	

Case 1: Module mounted on tolerance circle of ± 0.13 mm (± 0.005 in.).
Case 2: HEDS-9000 mounted on tolerances of ± 0.50 mm (0.020").
HEDS-9100 mounted ontolerances of ± 0.38 mm (0.015").

Electrical Characteristics

Electrical Characteristics over Recommended Operating Range, typical at 25°C.

Parameter	Symbol	Min.	Typical	Max.	Units	Notes
Supply Current	I_{CC}		17	40	mA	
High Level Output Voltage	V_{OH}	2.4			Volts	I_{OH} = -40 µA max.
Low Level Output Voltage	V_{OL}			0.4	Volts	I_{OL} = 3.2 mA
Rise Time	t_r		200		ns	C_L = 25 pF R_L = 11 kΩ pull-up
Fall Time	t_f		50		ns	

Recommended Codewheel Characteristics

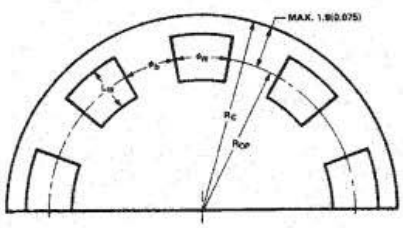


Figure 1. Codestrip Design

Codewheel Options

HEDS Series	CPR (N)	Option	Optical Radius mm (in.)
5120	96	K	11.00 (0.433)
5120	100	C	11.00 (0.433)
5120	192	D	11.00 (0.433)
5120	200	E	11.00 (0.433)
5120	256	F	11.00 (0.433)
5120	360	G	11.00 (0.433)
5120	400	H	11.00 (0.433)
5120	500	A	11.00 (0.433)
5120	512	I	11.00 (0.433)
6100	500	A	23.36 (0.920)
6100	1000	B	23.36 (0.920)

Parameter	Symbol	Minimum	Maximum	Units	Notes
Window/Bar Ratio	ϕ_w/ϕ_b	0.7	1.4		
Window Length	L_w	1.8 (0.071)	2.3 (0.09)	mm (inch)	
Absolute Maximum Codewheel Radius	R_C		R_{OP} + 1.9 (0.0075)	mm (inch)	Includes eccentricity errors

Mounting Considerations

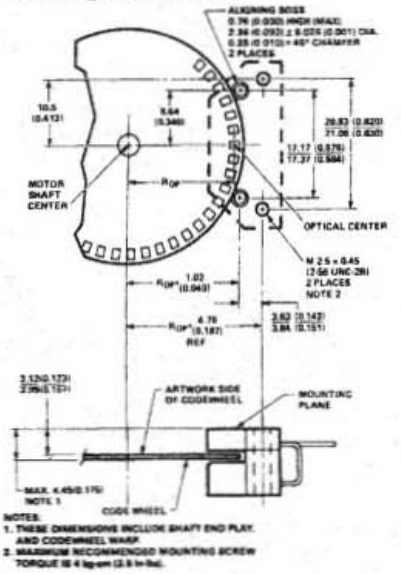


Figure 2. Mounting Plane Side A.

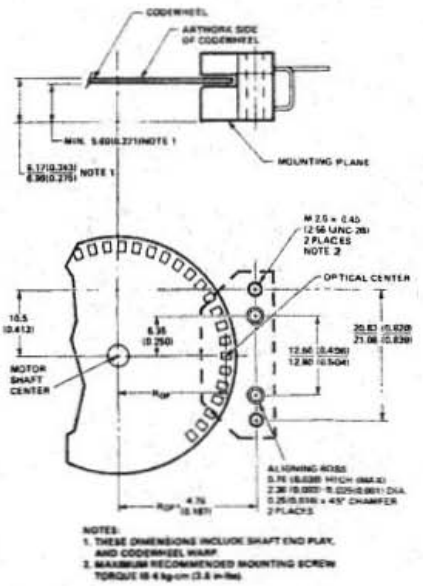


Figure 3. Mounting Plane Side B.

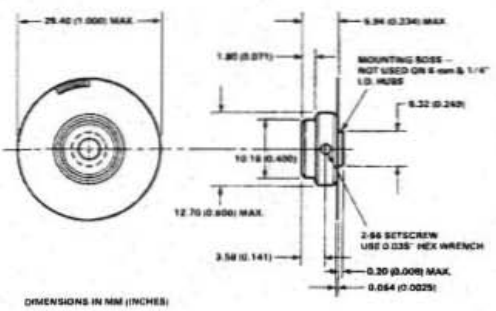


Figure 4. Mounting as Referenced to Side A.

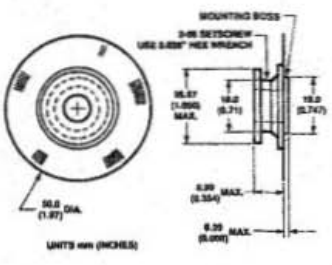


Figure 5. Mounting as Referenced to Side B.

Agilent Technologies
Innovating the HP Way

Connectors

Manufacturer	Part Number	Mounting Surface
AMP	1203686-4 640442-5	Both Side B
DuPont	65039-032 with 4825X-000 term.	Both
Agilent	HEDS-8902 with 4-wire leads	Side B (see Fig. 6)
Molex	2695 series with 2759 series term.	Side B

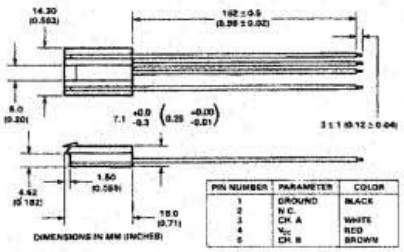


Figure 6. HEDS-8902 Connector.

Ordering Information

HEDS-9000 Option □ 0 0 **HEDS-6100 Option** □ □ □

Resolution (Cycles/Rev)
A - 500 CPR
B - 1000 CPR
T = 2000 CPR*
U = 2048 CPR*

Shaft Diameter	
05 - 3/16 in.	10 - 5/8 in.
06 - 1/4 in.	11 - 4 mm
07 - 5/16 in.	12 - 6 mm
08 - 3/8 in.	13 - 8 mm
09 - 1/2 in.	

HEDS-9100 Option □ 0 0 **HEDS-5120 Option** □ □ □

Resolution (Cycles/Rev)	
S - 50 CPR	G - 360 CPR
K - 96 CPR	H - 400 CPR
C - 100 CPR	A - 500 CPR
D - 192 CPR	I - 512 CPR
E - 200 CPR	B - 1000 CPR*
F - 256 CPR	J - 1024 CPR*

Shaft Diameter	
01 - 2 mm	11 - 4 mm
02 - 3 mm	14 - 5 mm
03 - 1/8 in.	12 - 6 mm
04 - 5/32 in.	13 - 8 mm
05 - 3/16 in.	
06 - 1/4 in.	

HEDS-91 □ □

Resolution (Cycles/Rev)
BA - 250 CPR
BB - 480 CPR
BC - 576 CPR

*Please refer to separate HEDS-9000/9100/9200 Extended Resolution series data sheet for detailed information and Codewheel selection.

www.semiconductor.agilent.com

Obsoletes 5091-8349E (7/93)
5965-5887E (11/99)

B.2 Sharp Infrared Sensors

SHARP

GP2D12/GP2D15 Distance Measuring Sensors

General Purpose Type Distance Measuring Sensors

■ General Description

SHARP's **GP2D12/GP2D15** are general purpose type distance measuring sensors which consist of PSD* and infrared emitting diode and signal processing circuit. It enables to detect objects without any influence on the color of reflective objects, reflectivity, the lights of surroundings.

*PSD:Position Sensitive Detector

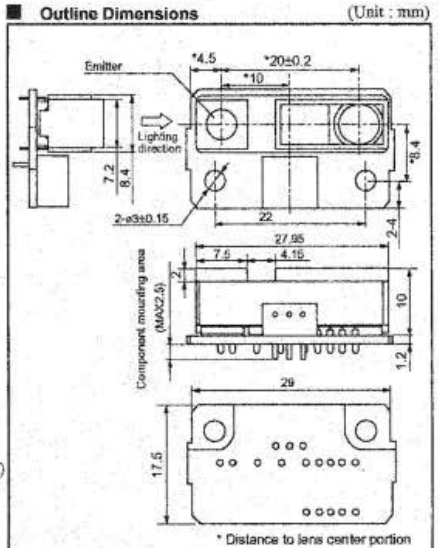
■ Features

(1) Less influence on the color of reflective objects, reflectivity
(2) Line-up of distance output/distance judgement type
Distance output type(analog voltage) : **GP2D12**
Detecting distance : 10 to 80cm
Distance judgement type : **GP2D15**
Judgement distance : 24cm (Adjustable within the range of 10 to 80cm)
(3) External control circuit is unnecessary.
(4) Low cost

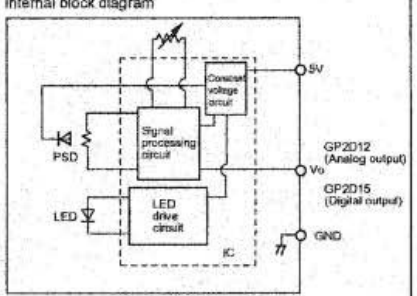
■ Applications

(1) TVs
(2) Personal computers
(3) Cars
(4) Copiers

■ Outline Dimensions (Unit : mm)



Internal block diagram



SHARP

Tec.U970501

SHARP

GP2D12/GP2D15 Distance Measuring Sensors

■ Specifications

GP2D12 (Ta=25°C)

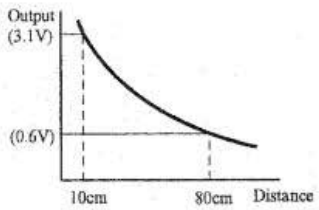
Parameter	Symbol	Rating
Supply voltage	Vcc	4.5 to 5.5V
Dissipation current	Icc	MAX.35mA
Measuring range	L	10 to 80cm
Output type	—	Analog output
Operating temperature	Topr	-10 to +60°C

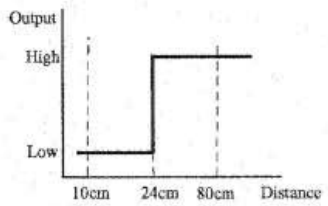
GP2D15 (Ta=25°C)

Parameter	Symbol	Rating
Supply voltage	Vcc	4.5 to 5.5V
Dissipation current	Icc	MAX.35mA
*Judgement distance	L	TYP.24cm
Output type	—	Digital output
Operating temperature	Topr	-10 to +60°C

* Adjustable within the range of 10 to 80cm.<Custom products>

■ Output pattern





B.3 Watchdog Timer IC

UCC1946
UCC2946
UCC3946

Microprocessor Supervisor with Watchdog Timer

FEATURES

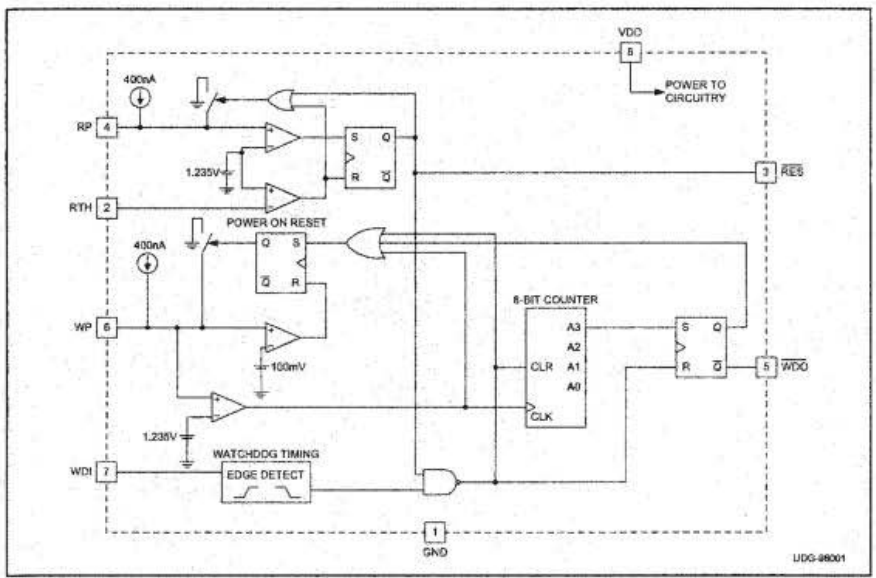
- Fully Programmable Reset Threshold
- Fully Programmable Reset Period
- Fully Programmable Watchdog Period
- 2% Accurate Reset Threshold
- VDD Can Go as Low as 2V
- 18µA Maximum IDD
- Reset Valid Down to 1V

DESCRIPTION

The UCC3946 is designed to provide accurate microprocessor supervision, including reset and watchdog functions. During power up, the IC asserts a reset signal $\overline{RES}$ with VDD as low as 1V. The reset signal remains asserted until the VDD voltage rises and remains above the reset threshold for the reset period. Both reset threshold and reset period are programmable by the user. The IC is also resistant to glitches on the VDD line. Once $\overline{RES}$ has been deasserted, any drops below the threshold voltage need to be of certain time duration and voltage magnitude to generate a reset signal. These values are shown in Figure 1. An I/O line of the microprocessor may be tied to the watchdog input (WDI) for watchdog functions. If the I/O line is not toggled within a set watchdog period, programmable by the user, $\overline{WDO}$ will be asserted. The watchdog function will be disabled during reset conditions.

The UCC3946 is available in 8-pin SOIC(D), 8-pin DIP (N or J) and 8-pin TSSOP(PW) packages to optimize board space.

BLOCK DIAGRAM



Note: Pinout represents the 8-pin TSSOP package.

SLUS247C - FEBRUARY 2000

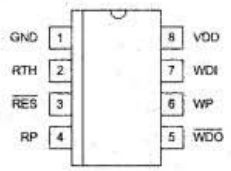
ABSOLUTE MAXIMUM RATINGS

V_{IN} 10V
Storage Temperature . . . . . –65°C to +150°C
Junction Temperature. . . . . –55°C to +150°C
Lead Temperature (Soldering, 10 sec.) . . . . . +300°C

Currents are positive into, negative out of the specified terminal. Consult Packaging Section of the Databook for thermal limitations and considerations of packages.

CONNECTION DIAGRAM

SOIC-8, TSSOP-8, DIL-8 (Top View)
D, PW, N or J Package



ELECTRICAL CHARACTERISTICS: Unless otherwise specified, VDD = 2.1V to 5.5V for UCC1946 and UCC2946; VDD = 2V to 5.5V for UCC3946; TA = 0°C to 70°C for UCC3946, –40°C to 95°C for UCC2946, and –55°C to 125°C for UCC1946; $T_A = T_J$

PARAMETERS	TEST CONDITIONS	MIN	TYP	MAX	MIN	TYP	MAX	UNITS
		UCC3946			UCC1946 & UCC2946			
Operating Voltage		2.0		5.5	2.1		5.5	V
Supply Current			10	18		12	18	µA
Minimum VDD	(Note 1)			1			1.1	V
Reset Section								
Reset Threshold	VDD Rising	1.190	1.235	1.260	1.170	1.235	1.260	V
Threshold Hysteresis			15			15		mV
Input Leakage				5			5	nA
Output High Voltage	I_{SOURCE} = 2mA	V_{DD} – 0.3			V_{DD} – 0.3			V
Output Low Voltage	I_{SINK} = 2mA			0.1			0.1	V
	VDD = 1V, I_{SINK} = 20uA			0.2			0.4	V
VDD to Output Delay	VDD = -1mV/µs (Note 2)		120			120		µs
Reset Period	C_{RP} = 64nF	160	200	260	140	200	320	ms
Watchdog Section								
WDI Input High		0.7· V_{DD}			0.7· V_{DD}			V
WDI Input Low				0.3· V_{DD}			0.3· V_{DD}	V
Watchdog Period	C_{WP} = 64nF	1.12	1.60	2.08	0.96	1.60	2.56	s
Watchdog Pulse Width		50			50			ns
Output High Voltage	I_{SOURCE} = 2mA	V_{DD} – 0.3			V_{DD} – 0.3			V
Output Low Voltage	I_{SINK} = 2mA			0.1			0.1	V

Note 1: This is the minimum supply voltage where $\overline{RES}$ is considered valid.
Note 2: Guaranteed by design. Not 100% tested in production.

PIN DESCRIPTIONS

GND: Ground reference for the IC.

$\overline{RES}$: This pin is high only if the voltage on the RTH has risen above 1.235V. Once RTH rises above the threshold, this pin remains low for the reset period. This pin will also go low and remain low if the RTH voltage dips below 1.235V for an amount of time determined by Figure 1.

RTH: This input compares its voltage to an internal 1.25V reference. By using external resistors, a user can program any reset threshold he wishes to achieve.

RP: This pin allows the user to program the reset period by adjusting an external capacitor.

VDD: Supply voltage for the IC.

WDI: This pin is the input to the watchdog timer. If this pin is not toggled or strobed within the watchdog period, $\overline{WDO}$ is asserted.

$\overline{WDO}$: This pin is the watchdog output. This pin will be asserted low if the WDI pin is not strobed or toggled within the watchdog period.

WP: This pin allows the user to program the watchdog period by adjusting an external capacitor.

APPLICATION INFORMATION

The UCC3946 supervisory circuit provides accurate reset and watchdog functions for a variety of microprocessor applications. The reset circuit prevents the microprocessor from executing code during undervoltage conditions, typically during power-up and power-down. In order to prevent erratic operation in the presence of noise, voltage "glitches" whose voltage amplitude and time duration are less than the values specified in Fig. 1 are ignored.

The watchdog circuit monitors the microprocessor's activity, if the microprocessor does not toggle WDI during the programmable watchdog period $\overline{WDO}$ will go low, alerting the microprocessor's interrupt of a fault. The $\overline{WDO}$ pin is typically connected to the non-maskable input of the microprocessor so that an error recovery routine can be executed.

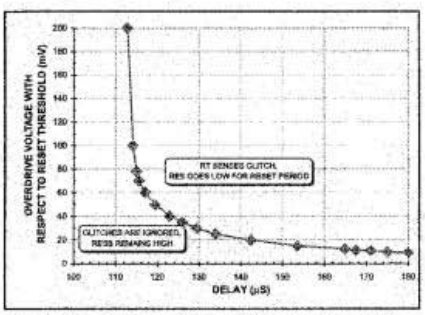


Figure 1. Overdrive voltage vs. delay to output low on RESB.

Slew rate: –1V/mS; monitored voltage = V_{DD}.

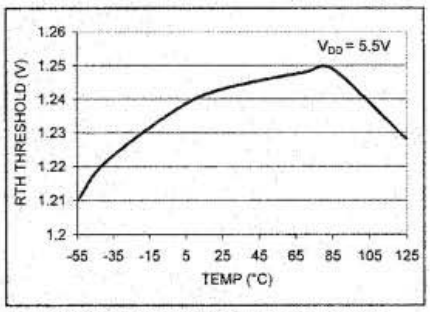


Figure 2. Typical RTH threshold vs. temperature.

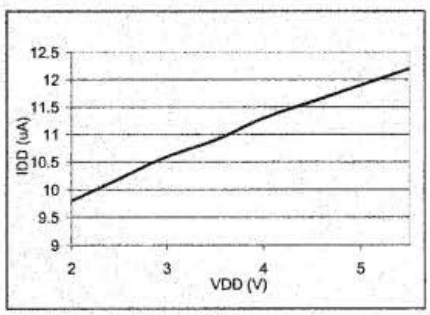
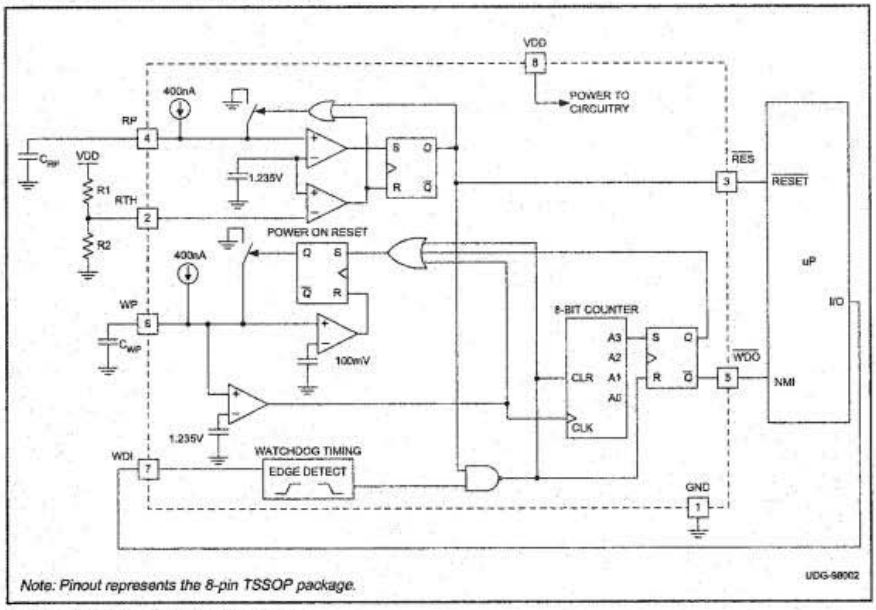


Figure 3. Typical IDD vs VDD.

APPLICATION INFORMATION (cont.)



Note: Pinout represents the 8-pin TSSOP package.

Figure 4. Typical application diagram.

Programming the Reset Voltage and Reset Period

The UCC3946 allows the reset trip voltage to be programmed with two external resistors. In most applications VDD is monitored by the reset circuit, however, the design allows voltages other than VDD to be monitored. Referring to Fig. 4, the voltage below which reset will be asserted is determined by:

$$V_{RESET} = 1.235 \cdot \frac{R1 + R2}{R2}$$

In order to keep quiescent currents low, resistor values in the megaohm range can be used for R1 and R2. A manual reset can be easily implemented by connecting a momentary push switch in parallel with R2. $\overline{RES}$ is guaranteed to be low with VDD voltages as low as 1V.

Once VDD rises above the programmed threshold, $\overline{RES}$ remains low for the reset period defined by:

$$T_{RP} = 3.125 \cdot C_{RP}$$

where T_{RP} is time in milliseconds and C_{RP} is capacitance in nanofarads. C_{RP} is charged with a precision current source of 400nA, a high quality, low leakage capacitor (such as an NPO ceramic) should be used to maintain timing tolerances. Fig. 5 illustrates the voltage levels and timings associated with the reset circuit.

Programming the Watchdog Period

The watchdog period is programmed with C_{WP} as follows:

$$T_{WP} = 25 \cdot C_{WP}$$

where T_{WP} is in milliseconds and C_{WP} is in nanofarads. A high quality, low leakage capacitor should be used for C_{WP}. The watchdog input WDI must be toggled with a high/low or low/high transition within the watchdog period to prevent $\overline{WDO}$ from assuming a logic level low. $\overline{WDO}$ will maintain the low logic level until WDI is toggled or $\overline{RES}$ is asserted. If at any time $\overline{RES}$ is asserted, $\overline{WDO}$ will assume a high logic state and the watchdog period will be reinitiated. Fig. 6 illustrates the timings associated with the watchdog circuit.

APPLICATION INFORMATION (cont.)

Connecting $\overline{WDO}$ to $\overline{RES}$

In order to provide design flexibility, the reset and watchdog circuits in the UCC3946 have separate outputs. Each output will independently drive high or low, depending on circuit conditions explained previously.

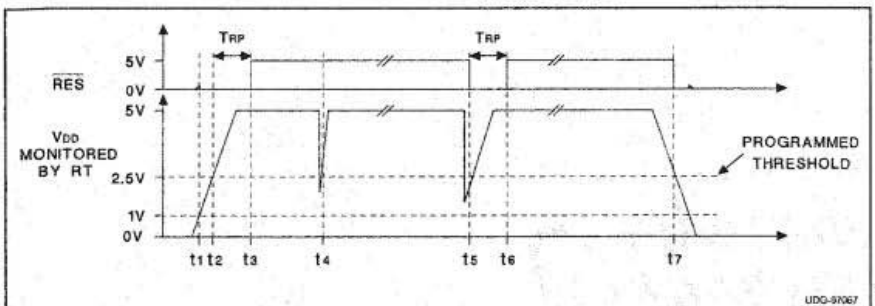
In some applications, it may be desirable for either the $\overline{RES}$ or $\overline{WDO}$ to reset the microprocessor. This can be done by connecting $\overline{WDO}$ to $\overline{RES}$. If the pins try to drive to different output levels, the low output level will dominate. Additional current will flow from VDD to GND during these states. If the application cannot support additional current (during fault conditions), $\overline{RES}$ and $\overline{WDO}$ can be connected to the inputs of an OR gate whose output is connected to the microprocessor's reset pin.

Layout Considerations

A 0.1µF capacitor connected from V_{DD} to GND is recommended to decouple the UCC3946 from switching transients on the V_{DD} supply rail.

Since RP and WP are precision current sources, capacitors C_{RP} and C_{WP} should be connected to these pins with minimal trace length to reduce board capacitance. Care should be taken to route any traces with high voltage potential or high speed digital signals away from these capacitors.

Resistors R1 and R2 generally have a high ohmic value, traces associated with these parts should be kept short in order to prevent any transient producing signals from coupling into the high impedance RTH pin.



t1: V_{DD} > 1V, $\overline{RES}$ is guaranteed low.

t2: V_{DD} > programmed threshold, $\overline{RES}$ remains low for T_{RP}.

t3: T_{RP} expires, $\overline{RES}$ pulls high.

t4: Voltage glitch occurs, but is filtered at the RTH pin, $\overline{RES}$ remains high.

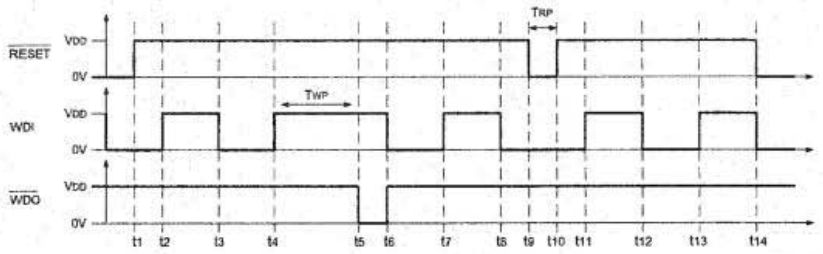
t5: Voltage glitch occurs whose magnitude and duration is greater than the RTH filter, $\overline{RES}$ is asserted for T_{RP}.

t6: On completion of the T_{RP} pulse the R_{TH} voltage has returned and $\overline{RES}$ is pulled high.

t7: V_{DD} dips below threshold (minus hysteresis), $\overline{RES}$ is asserted.

Figure 5. Reset circuit timings.

APPLICATION INFORMATION (cont.)



t1: Microprocessor is reset.

t2: WDI is toggled some time after reset, but before TWP expires.

t3: WDI is toggled before TWP expires.

t4: WDI is toggled before TWP expires.

t5: WDI is not toggled before TWP expires and $\overline{WDO}$ asserts low, triggering the microprocessor to enter an error recovery routine.

t6: The microprocessor's error recovery routine is executed and WDI is toggled, reinitiating the watchdog timer.

t7: WDI is toggled before TWP expires.

t8: WDI is toggled before TWP expires.

t9: $\overline{RES}$ is momentarily triggered, $\overline{RES}$ is asserted low for TRP.

t10: Microprocessor is reset, $\overline{RES}$ pulls high.

t11: WDI is toggled some time after reset, but before TWP expires.

t12: WDI is toggled before TWP expires.

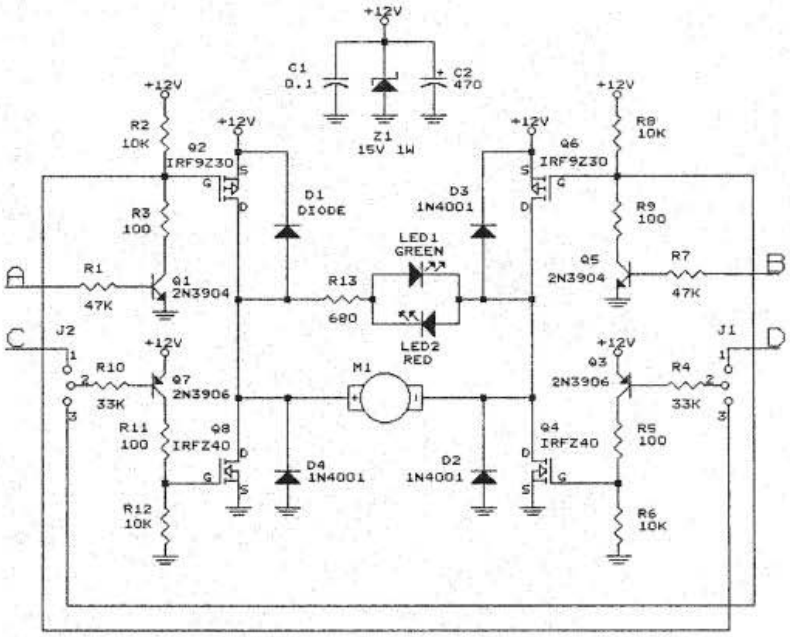
t13: WDI is toggled before TWP expires.

t14: VDD dips below the reset threshold, $\overline{RES}$ is asserted.

Figure 6. Watchdog circuit timings.

UNITRODE CORPORATION
7 CONTINENTAL BLVD. • MERRIMACK, NH 03054
TEL (603) 424-2410 • FAX (603) 424-3460

B.4 Example H-Bridge Circuit for Motor Control



NOTES

1. All capacitors are in ufd and 25V
2. All resistors are 1/4W
3. All connections are indicated by a junction.
4. Inputs C and D are available via J1 and J2 to control Q4 and Q8's operation separately from Q2 and Q6.

For the following table:

J1-2 connected to J1-3
J2-2 connected to J2-3

	A	B
STOP	LOW	LOW
FORWARD	HI	LOW
REVERSE	LOW	HI
NOT ALLOWED	HI	HI

Eugene Blanchard
Copyright April 2001

Title: N & P Ch MOSFET Single Supply H bridge

Size	Document Number	REV
A	np-s.sch	3

Date: April 14, 2001 Sheet 1 of 1