Dipartimento di Elettronica - Politecnico di Milano

BARCS: A NEW WAY OF BUILDING ROBOTS

R. Cassinis

Report n. 87-036

September, 1987

Title: BARCS: A NEW WAY OF BUILDING ROBOTS

Author: R. Cassinis

Report n.: 87-036

Abstract: *As robot technology advances, and robotics become an independent discipline, one should start thinking wether the methods used so far are still up-to-date, and if robots will eventually become those really autonomous machines they should be. An analysis shows that, if this goal has to be achieved, robot philosophy should be radically changed.*

The paper describes a completely new method of building robot control systems, which is specially suitable for mobile machines. The most important issues of this method are that traditional programming is completely abandoned, and a system based on a behavioral architecture and on smart sensor data handling is implemented.

The paper also shows a practical implementation of the proposed system, that is currently being built; the first experimental results are presented.

Keywords: Control Systems, Data Fusion, Mobile Robots, Robots, Sensors.

Name:	____
Address:	____
	____

TABLE OF CONTENTS

1. - INTRODUCTION

Although it describes an experimental implementation, this paper mainly deals with the philosophy of robot control and programming. The author is principally concerned with the severe limits of nowadays robot technology, and is fairly convinced that no significant improvements can be obtained unless the global robot philosophy is radically changed. A support to this theory comes from computer technology (specially software technology), that has undergone a similar process during the last ten or fifteen years. The transformation from algorithmic programming languages to symbolic programming tools was conceptually more significant for computer science than all the advances of semiconductor technology, although they were of course strictly related. Now, the next generation of computers seems to be quite far away, and today's tools are not as powerful as they should be for most practical applications, but the research direction is clearly towards "intelligent" systems, whose main characteristic is that they do not need any *algorithm* description, but only a *problem's* description.

After an initial stage, where no formal methods had been developed yet, industrial robots have always been programmed exactly as computers were, with a description of the actions to be performed in order to achieve a predefined task. In traditional (explicit) programming, the process of finding out the operations to be performed and their correct sequence is left to the programmer, that must develop an algorithm that solves the problem [2, 4]. In implicit programming systems this task is (or should) be automatically performed by the machine, but the result is always an explicit program, that contains full information about each elementary action the robot must perform [1]. In other words, implicit programming systems are planners that substitute the human programmer as far as the process of finding the solution algorithm is concerned. No changes (or very small changes) are required on the robots and on their control systems.

It could now be said that robots should undergo the same transformation that is characterizing computers, but some important differences exist, and should be made clear.

Computers are essentially *data* processors: this means that they work on entities that are very easily transformed into electric signals, and that are processed using a comparatively small set of primitives. Moreover, all the data to be processed are *inside* the machine, no matter how complex it can be. Therefore, a computer can be considered *insensitive* to the environment: the computer with which this paper was written works equally well in a mountain chalet (where it is now) or a town apartment (where it was yesterday): no special adaptments were required to transfer it from one place to the other one.

Conversely, robots are *object* processors: they operate on entities that do not belong to their interior, and must therefore be adapted (or adapt themselves) to the environment they are in.

Industrial robots are somewhat midway between the two extremes: if considered as units, they belong to the category of object processors, but, if all the tooling that usually is around them is considered as a part of the robot system, their behavior is similar to the behavior of computers.

So, while algorithmic methods are well suited for programming classical industrial robots, many problems arise if they have to be applied to other robots, and in particular to mobile robots. The ultimate aim of this paper is to show how mobile robots should be built and programmed using a totally different philosophy, that would greatly enhance their capabilities.

2. - BASIC CONCEPTS

The main difference between a traditional industrial robot and a mobile one is, of course, the fact that the former machine is fixed to its supporting structure, while the latter one has freedom to move around. This has a number of important side effects, that can be summarized as follows:

- The traditional robot works in a much more structured environment than the mobile one[1];
- The position of the traditional robot is always known with good accuracy, while the position of the mobile one can only be estimated, sometimes with coarse errors, unless very expensive systems are employed, or very special situations occur;
- The traditional robot has only one possible way of reaching each point of its working space (in some cases, two or four configurations are possible for some points, but the ambiguity can be eliminated applying some general restrictions). A mobile robot with an arm has virtually infinite possibilities, since its kinematic structure is redundant;
- Traditional robots usually execute repetitive jobs for long periods. This means that some time can be spent for programming the robot, without increasing the cost of the whole system over an unacceptable threshold. Mobile robots on the other hand are often executing each task only once in their life, and this means that programming times must be kept as short as possible.

[1]In fact, all attempts of operating industrial robots in a non-structured environment (the bin-picking problem is a good example), have had poor results so far.

Looking deeply into the previous points, one realizes that nor explicit nor implicit programming techniques are really suitable for mobile robots, and that a completely new programming philosophy must be designed.

2.1. - THE NEED FOR A BEHAVIORAL APPROACH

If a traditional robot system is examined with the aim of identifying where the main functions of the system are executed, a scheme such as the one shown in Fig. 1 will result. This scheme clearly shows that the machine does not know what it is doing, because the four modules *purpose*, *strategy*, *safety* and *behavior* are not located inside it, but in the brain of the programmer. Such a machine is completely unable of withstanding unforeseen situations, since, not knowing the *final goal* it has to reach, it will never be able to produce alternate plans for reaching it.

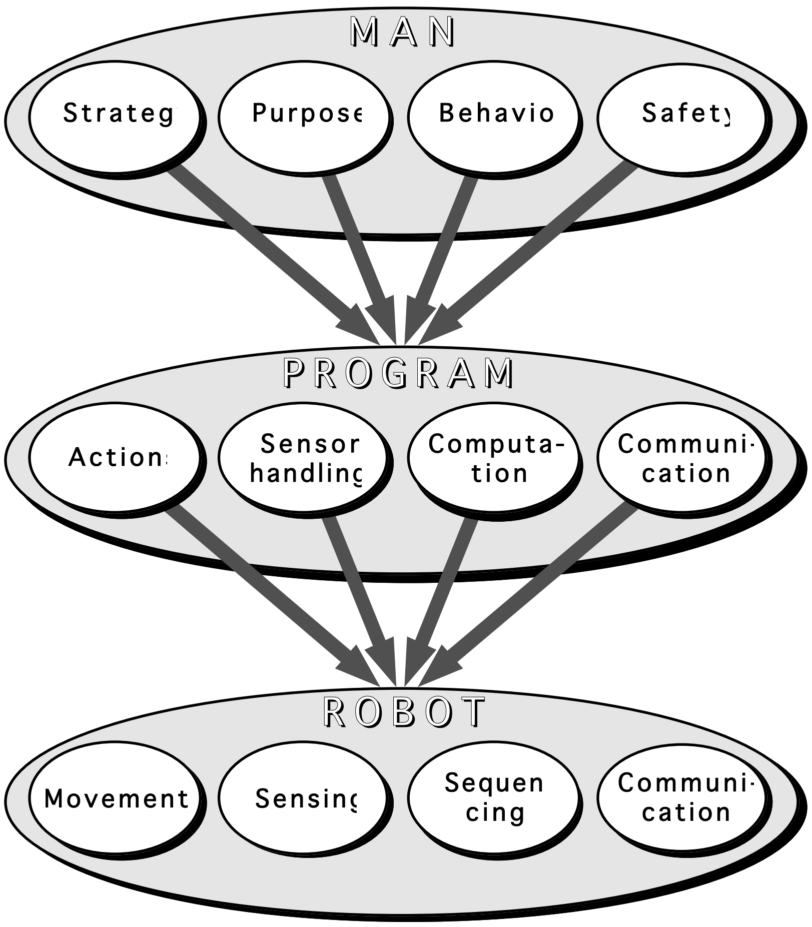

Figure 1 - Traditional robot structure.

In some more sophisticated programming systems (*implicit* programming systems) some of the functions here depicted as pertaining to the human programmer are moved inside the machine [5, 6, 10, 13], but the problem of solving emergency situations still remains. In some attempts to make robots capable of generating emergency recovery plans the structure of to-day's programming systems requires that the source program be analyzed in order to extract semantic information from it [14]. This is exactly the same information the programmer had in mind when he wrote the program, and that was lost during the coding process. This is obviously absurd: the purpose of the actions being executed must be *inside* the robot, so that the process just described (which, by the way, cannot be solved in most practical cases) becomes unnecessary.

As far as mobile robots are concerned, similar problems can be found. Most "toy" robots have a control system and a programming structure that closely resembles the one currently used in industrial robots. But, since their working conditions are very different, their behavior is not at all satisfactory. Research robots (and some advanced industrial models) are on the other hand much more sophisticated, and exhibit very good skills. Nevertheless, in order to allow inferential systems to plan their actions, they often require a huge amount of information to be processed before they can even start the planning phase. For instance, they usually need a complete knowledge of the environment they are in, in order to be able to calculate the path to be followed to reach the goal (navigation problem) [12, 15, 16].

The human approach to a navigation problem is completely different: in order to reach a position that has already been identified, a human will only roughly map the environment in the area that will supposedly be interested by his movements, and will produce an approximate plan (path to be followed and actions to be done). This plan will be refined as the job proceeds, and alternate plans will be generated only if difficulties arise.

Now, if (as it happens with mobile robots) each job will be executed only once[2], it is not worth spending time for generating a complete and accurate plan before starting the execution.

[2]This also applies to repetitive tasks, since in an unstructured environment each iteration of the same task may require a completely different set of actions, and can therefore be considered as the execution of a new program.

3. - *BARCS* STRUCTURE

In order to overcome the previously mentioned problems, a research was started around the end of 1986 whose program included the definition of a control system architecture that could withstand unstructured environments and unpredictable situations. Since the beginning of the research it was clear that such machine should contain all information inside itself, to avoid cumbersome reconstruction of information lost during the programming phase: in other words, this means that the machine should only be programmed by telling it the *final goal* to reach, and by giving it some hints about how this goal could be reached. The machine should know how to *behave* in any possible situation. This assumption led to the name of the project: *BARCS* is an acronym for Behavioral Architecture Robot Control System. The work was mainly inspired from the ongoing research on autonomous and mobile robots, which seems to be going in a similar direction [3, 17, 18, 19].

BARCS architecture is based on a number of cooperating modules (Fig. 2), each one having a specific function. The main principles of this structure are:

- All modules are conceptually at the same level: no fixed hierarchies are established;
- All information in the system is public, i.e. every module can access all information. This was done because the needs of each module cannot be foreseen: a suitable data structure must therefore be implemented;

- At any time, each module knows what the other modules are doing . This is perhaps the most important difference between *BARCS* and traditional robots: if the latter can be seen as a society of persons working together (they cooperate for reaching a common goal, but they may have secret thoughts and purposes) [7], *BARCS* resembles a single brain (several parts of the brain work in parallel, but they do not have secrets for each other[3]);
- Module's functions are quite unconventional: they will be discussed in the sequel.

[3]Ego, Super-Ego and other unconscious processes are obviously excluded from this theory: the brain is far too complex to be emulated!

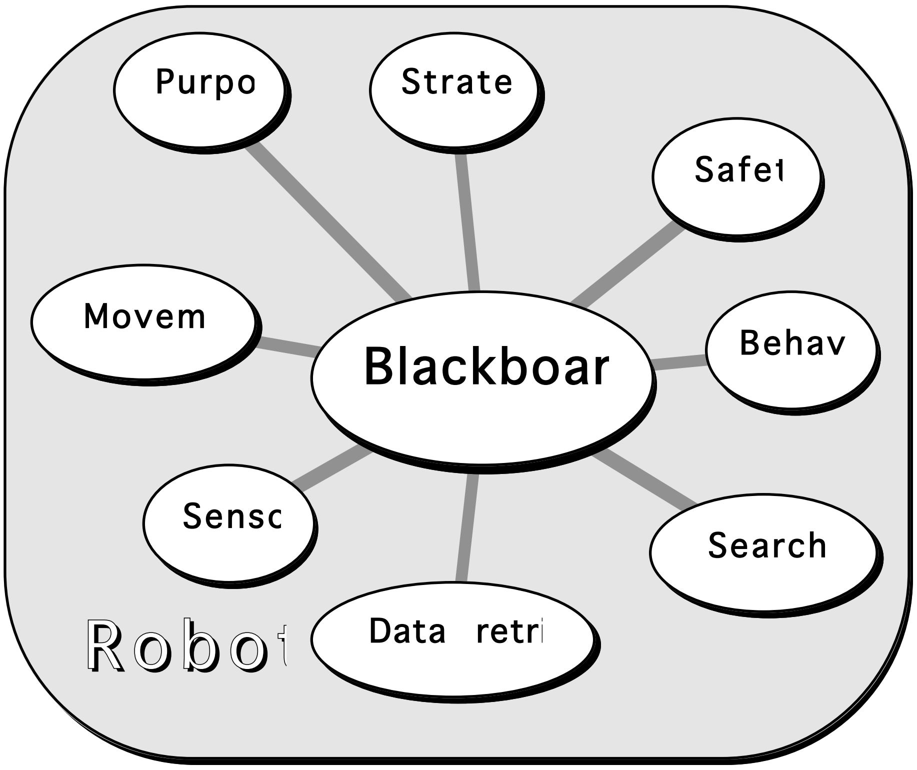

Figure 2 - *BARCS* structure.

3.1. - *BARCS* MODULES

Each *BARCS* module equally contributes to the generation of plans that allow the robot reaching its goals. The following paragraphs contain a detailed description of these functions.

For clarity purposes, the description of some of the tasks pertaining to the motion module was moved to other modules. Given the flexibility of the system however this is not important: in future versions of *BARCS* several other changes will surely have to be done [9, 11].

3.1.1. - The purpose module

This module contains the final goal the robot must reach. It is a very simple module: given a number of goals the robot can accomplish, it only selects the one to be pursued, according to the instructions received from the user. In other words, the purpose module is the user-machine interface.

3.1.2. - The strategy module

Given the goal to pursue, the machine has to develop a strategy for achieving it. Strategy here means a series of subgoals whose sequential achievement will eventually lead to the accomplishment of the global task. For instance, if the goal is "open the window", the strategy for reaching it could be the following sequence of steps:

- Locate the window;
- Reach near the window;
- Locate the handle;
- Reach the handle;
- Turn the handle;
- Pull the handle.

In the first development stages, it was decided that this module should not have any intelligence in it: it only handles a number of tables, one for each executable job. It is also important to note that, during task accomplishment, the strategy can change: for instance, if, with reference to the previous example, an obstacle is found between the actual robot location and the window, a new subgoal will have to be generated and inserted in the list: the new subgoal will be "pass by the obstacle".

3.1.3. - The safety module

It is obvious that a robot that develops its own program while executing it will often incur in potentially dangerous situations. The safety module continuously monitors all parameters, such as the distance from obstacles, which could indicate danger and, if these parameters are found to be outside the allowable ranges, it takes full control of the machine, stopping what it is doing and inserting new subgoals in the list. It could be objected that if the planner were good enough these situations should never occur, but, as a counter example, we should consider the case of dynamically changing environments: if a moving obstacle gets in the robot's way, the original plan should be changed. It is then the safety module's task to keep an eye on moving objects around the robot.

3.1.4. - The behavior module

As it was said before, a mobile robot is mechanically redundant, i.e. it has many different alternatives for doing the same thing. The behavior module has the task of establishing general guidelines to help the robot solving these ambiguities: it will issue and enforce rules that will have general value. It is interesting to note that this module can have a great importance in the elimination of global controllers that now heavily condition systems where multiple mobile robots are used (as in automatic storage systems): the robots could be given some "circulation rules" that would automatically solve possible deadlocks when two or more machines interfere with each other. This is exactly the same thing that happens with road circulation: if no behavior rules existed, road circulation would be impossible, but, with a few of them, traffic on the roads goes fairly well.

3.1.5. - The searching module

Since one of the most common subgoals the robot has to pursue is the identification of objects and of their location, a module has been devoted to this special purpose. Given an object to search, the module issues strategies for identifying it, and uses sensory information to locate it.

As it happens for the strategy module, no intelligence is contained in the searching module: all objects to be recognized should be listed by the user, and appropriate searching strategies should be linked to each object.

It is interesting to note that the *meaning* of objects depends on the task the robot is pursuing: for instance, if the robot (now thought as an office machine) can alternately (*a*) perform the job of emptying waste paper baskets and (*b*) collect mail from paper trays, it will obviously have to be able to recognize both kinds of objects. But, when performing task *b*, there will be no need of *recognizing* waste paper baskets: they will only have to be considered as any other obstacle, since they do not have anything to do with task *b*.

3.1.6. - The motion module

This module controls all movements of the robot, according to the requests of the other modules. This is another critical point in most mobile robots. Usually, motion is considered to be exact (motors are equipped with position and velocity sensors, and robot movements are commanded in terms of exact quantities). Now, we know by experience that the absolute position of a mobile robot is very difficult to compute, unless very expensive systems are used[4]. But, in the general philosophy of *BARCS*, no exact quantities are known in the system: everything is approximate and, in any case, all quantities are related to the goal to be reached, and not to an absolute reference system. Therefore, since even a small error would make absolute calculations impossible or meaningless, it was decided to eliminate all position and velocity systems in *BARCS*, and to use existing sensors to correct the robot's movements as errors are discovered.

3.1.7. - The sensing module

The sensing module takes care of handling sensors, and of coordinating and processing the information they provide. Its main task is to understand what kind of information the other modules need, and to gather this

[4]Some "tricks" may be employed for precise positioning of the robot, as will be shown later: however, the problem does not change during normal robot navigation.

information from the various sensors the robot is equipped with. In other words, the blackboard should always contain up-to-date sensory data that can be readily used by the other modules. The kind of these data depends on the particular goal (or subgoal) being pursued at each moment.

3.1.8. - The data retrieval module and the blackboard

This probably is the most critical part of the system, since its efficiency affects the overall efficiency of the robot. Conceptually, the blackboard is a multi-port memory where all modules can store and retrieve information. Practically, it will be a complex data structure handled by the data retrieval module, that will have to queue all accesses to the blackboard and to organize the information stored in it.

3.2. - THE BEHAVIORAL PROGRAMMING

As it was said before, in order to accomplish any task in a non-structured environment, a traditional program cannot be written, mainly because it would be too long and complex, and because it would require an a-priori knowledge of the environment, which is absent in the present case.

The modular structure explained before suggests that the robot should be *pre-programmed*, giving it a number (possibly quite small) of tasks it can accomplish, and the correspondent solution strategies.

Each module should contain programs capable of executing the tasks that are assigned to the module itself. The single programs are quite simple, and can be better understood with reference to a practical example, that will be given in the next paragraph.

All information exchanges between modules take place using the blackboard. It must be noted that, in order to increase the efficiency of the system, only sensory information that are really needed are acquired from sensors: for instance, if the robot is moving in a straight direction, the sonar will only span through a small angle in front of the machine, and only these data will be stored.

Although all modules normally have the same priority, in some cases the priority is altered. For instance, if the robot is pursuing a goal, the motion and strategy modules will have a virtual "control" of the machine. But, if some danger situation arises (imminent collision, for example), the safety module will raise its own priority to prevent any action of the motion module until the danger situation is eliminated.

The structure of *BARCS* calls for a very unconventional programming environment (user interface). In fact, no program needs to be written, but a number of purposes to be listed, together with some strategic hints to the machine. This can be easily done taking advantage of the Macintosh operating system, that offers several tools for managing the screen.

In this phase, programming is only done by inserting data structures that represent strategies in the appropriate modules. In the next future, provisions for allowing the user to add and to alter strategies will be included. Some ideas for the programming environment will be taken from the CHIPWITS program [20] that, for some respect, uses a similar philosophy.

3.3. - THE ROLE OF ARTIFICIAL INTELLIGENCE IN *BARCS*

Readers will have noticed that, in the previous description of *BARCS*, the words "Artificial Intelligence" were never used. Now, specially in the last years, the terms *Robotics* and *Artificial Intelligence* have often been used together, and in some cases they are considered as synonyms. The author whishes to firmly reject this concept of robotics depending upon the developments of AI. In fact, robotics have undergone a very deep development, and can now be considered an autonomous discipline, that takes advantage of the development of AI exactly as it does with many other sciences.

Now, considering the structure of *BARCS*, it is easy to see that all modules can be built using standard programming techniques or, in some cases, approximate techniques (fuzzy logic, etc.). Nevertheless, the machine has some characteristics that allow its classification among "intelligent" machines. The reason is that, as it happens in AI, the structure of the system makes it capable of solving problems in such a way that some "intelligence" might show up. It could be said that this is just a new way of building AI systems (after all, expert systems, that are nowadays the most promising development of AI, are not intelligent at all: they are just, sometimes, smart), but the author prefers to use the term *smart* that better suits the system being described.

Nevertheless, AI techniques could be employed in the system with some advantages [8, 21], provided they met speed and cost requirements that still seem to be quite far away. For instance, the strategy, sensing and data retrieval modules could be made smarter if they were implemented with expert systems, rather than with traditional programs. It should be noted however that in many practical cases the functions of each module are so simple, that implementing them with AI techniques would only decrease their efficiency and increase system's cost.

4. - EXPERIMENTAL ASPECTS

As it was said before, the theory is being tested on an experimental model, a very simple mobile robot, whose main task is to demonstrate the feasibility of these concepts. Therefore, it does not have at this moment any special tooling. The robot (Fig. 3) was named *RISK* (Robot In Shape of Kettle), because of its quite odd appearance.

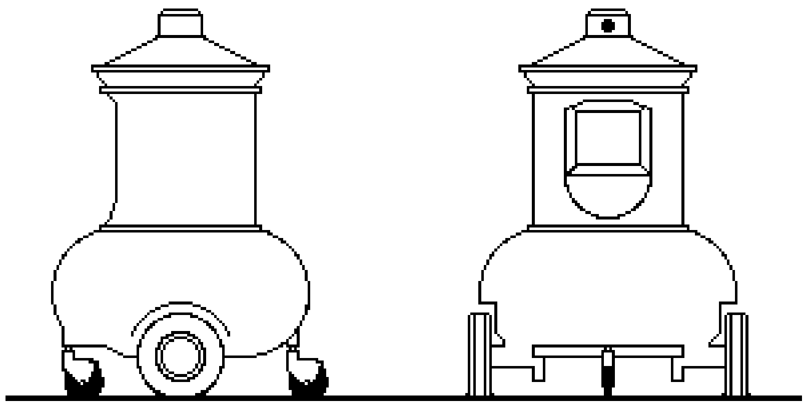

Figure 3 - *RISK* robot.

From the description of *RISK* that will be given in the following paragraphs it will become clear that the whole machine was built collecting scratch components from various sources. Although this makes *RISK* quite inefficient (and expensive, if it had to be built on an industrial scale), one of the goals was to demonstrate that nor sophisticated mechanical components and materials, nor highly advanced electronic parts will be necessary, until the control system philosophy will be fully assessed. In other words, it could be said that what makes a good robot is its brain, and not its muscles. Nevertheless, several ideas that were included in *RISK* could be readily engineered for industrialization of the machine.

4.1. - MECHANICS

The mechanical structure of *RISK* is very simple, and is one of the most commonly used in indoor mobile robots. It basically consists of a platform with two large wheels, driven by two D.C. motors (Fig. 4). The wheels are independent from each other, and steering is accomplished by varying their relative speed. The other two wheels are idle, and are only used to stabilize the platform. The front wheel is higher than the other three, so that, under normal conditions, the robot stands on three wheels only[5]. The front wheel prevents the robot from falling during abrupt stops and direction changes. The steering system allows rotation of the robot around its vertical axis, if the two motors are driven in opposite directions at the same speed. This allows *RISK* to move in narrow areas.

[5]The batteries, that are the heaviest component, have been placed on the rear part of the platform, so that the robot's center of gravity falls near the middle of the triangle formed by the two main wheels and the rear idler wheel.

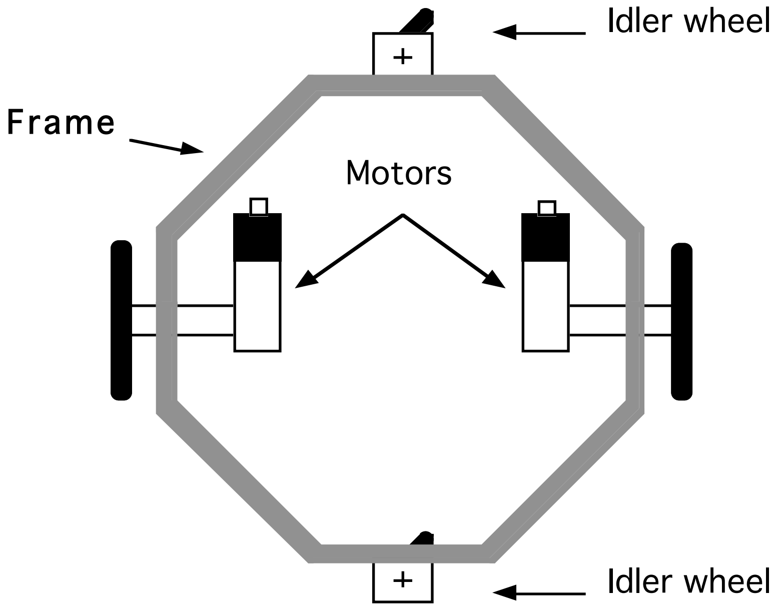

Figure 4 - *RISK* platform structure.

Motion transmission from motors to wheels is accomplished with two gears, that are enclosed in the motors assembly (Fig. 5). Motors and gears are of the kind used in truck windshield wipers, and yield a very high torque at a maximum speed of about 30 Rpm. With the wheels that were employed, that have a diameter of 250 mm, the maximum speed of the robot is around .4 m/s.

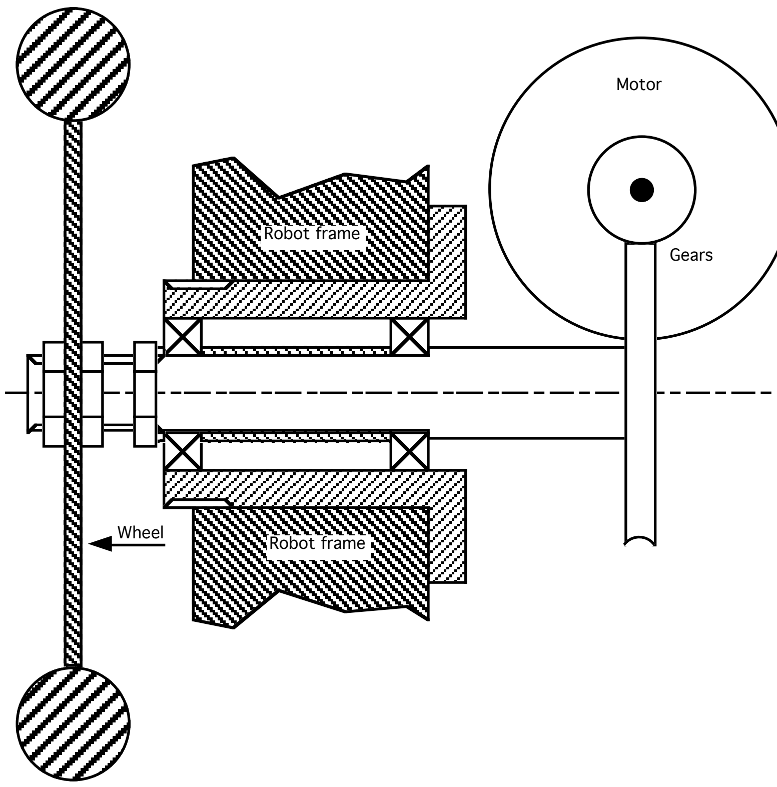

Figure 5 - Simplified diagram of motion mechanism.

The platform carries motors and related electronic circuits, batteries and some of the power regulators. Over the platform, a vertical frame supports the rest of the equipment (Fig. 6).

Four screws connect the platform to the vertical frame, so that the whole machine can be quickly divided in two parts for ease of transportation.

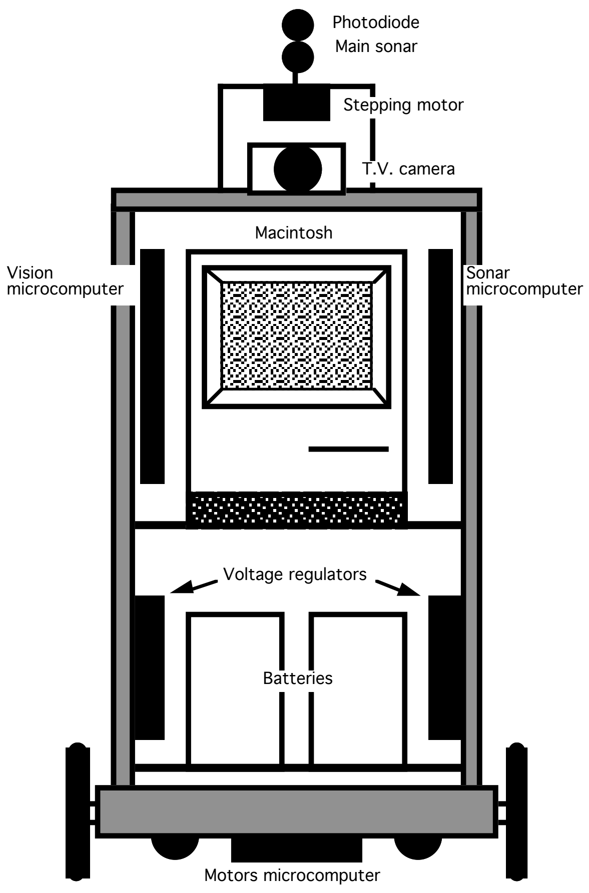

Figure 6 - Main components allocation.

An auxiliary device, the *RISK* Energy Tank, contains a battery charger and a power supply to be used when independent motion of the robot is not required.

4.2. - ELECTRONICS

The conceptual structure of *BARCS*, with its independent but strictly connected modules, would normally require ad-hoc electronics. Since the requirements of *RISK* are not very demanding, it was decided to simulate this structure, rather than actually building it. The overall block diagram of the machine is shown in Fig. 7.

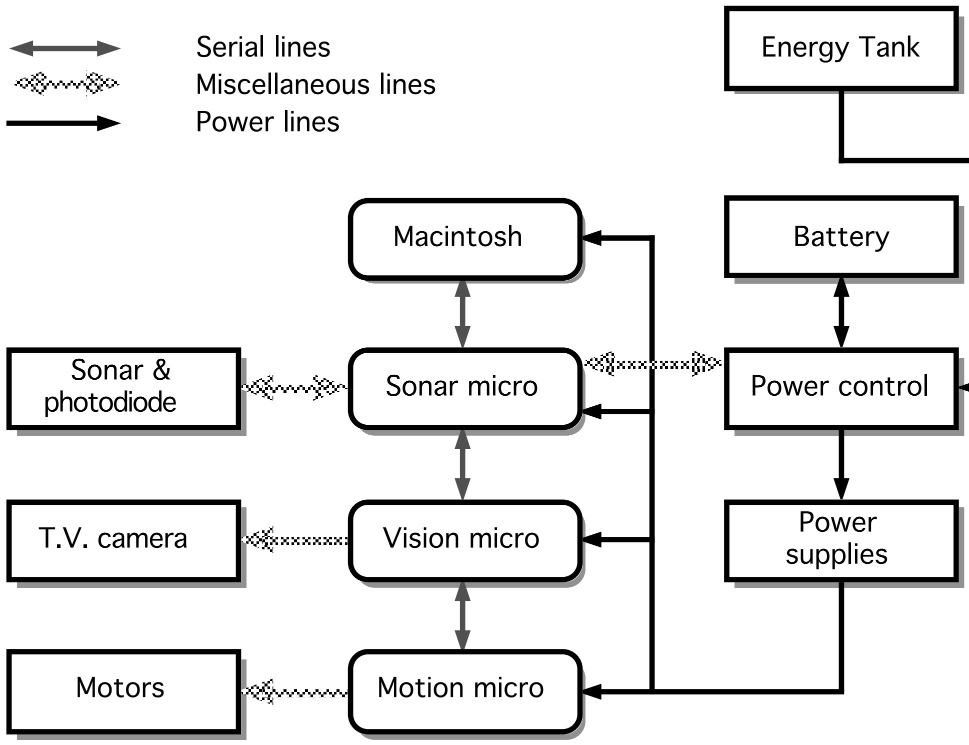

Figure 7- *RISK* electronics block diagram.

As it may be seen, the main block is a Macintosh Plus computer. This machine is linked to the rest of the system by a bidirectional serial line, that allows communication among all modules. Due to some hardware problems, the serial line is presently run at 4800 baud, except for the lowest part (connection between vision and motors microcomputers), that runs at 9600 baud.

Each microcomputer (except the motors microcomputer) maintains two queues of messages, one for each transmission direction. Therefore, all messages issued from the Macintosh are retransmitted by all microcomputers, except the motors microcomputer that, being the last in the chain, does not have to retransmit anything.

There are no contentions on the serial lines, since they are all point-to-point connections, and each microcomputer inserts its own messages between messages coming from other microcomputers and transmits them in an orderly fashion.

4.2.1. - Sonar microcomputer

This microcomputer (Fig. 8) is mainly devoted to the handling of sonars, that are the main sensors used for environment mapping and recognition. The main sonar sensor is mounted on the top on the robot, and can be rotated by means of a small stepping motor, in order to obtain polar maps of the environment.

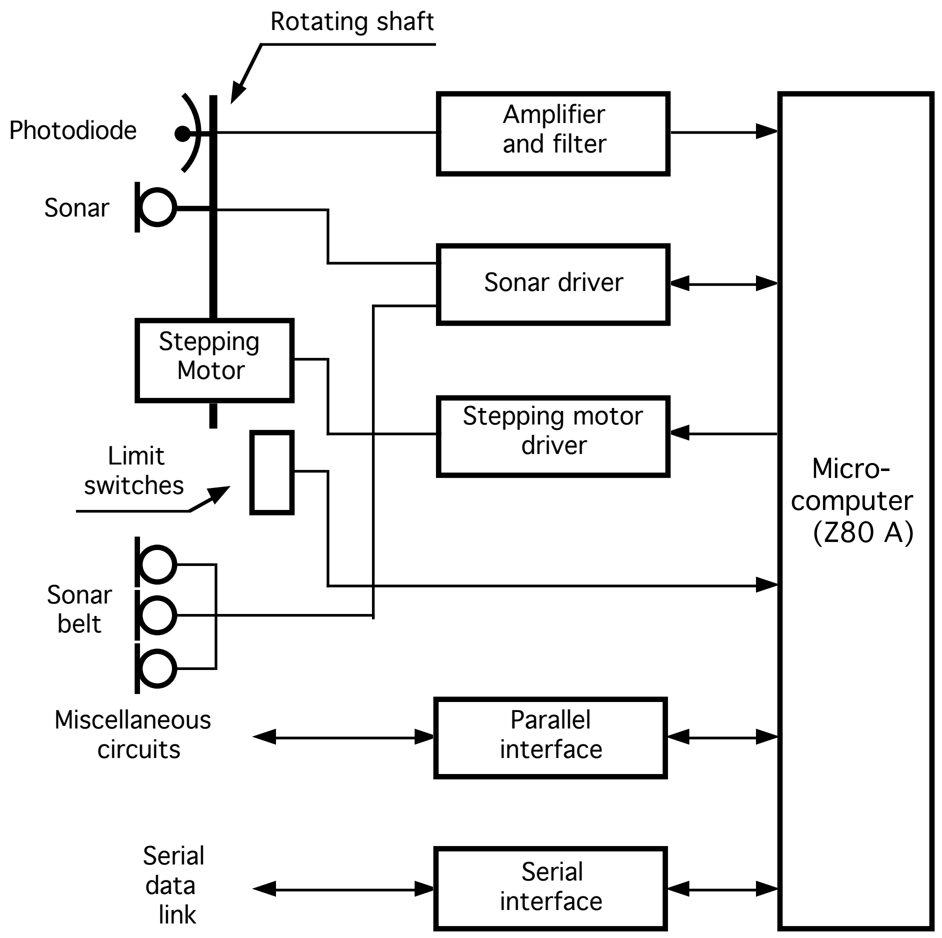

Figure 8 - Sonar microcomputer block diagram.

Other sonar transducers, placed as a belt around the robot, are mainly used for safety purposes and for detecting low objects that cannot be seen from the main one. They are all driven by the same hardware that drives the main sonar. The sonar technology was derived from the well known Polaroid ranging system, and the block diagram of the related circuits is shown in Fig. 9.

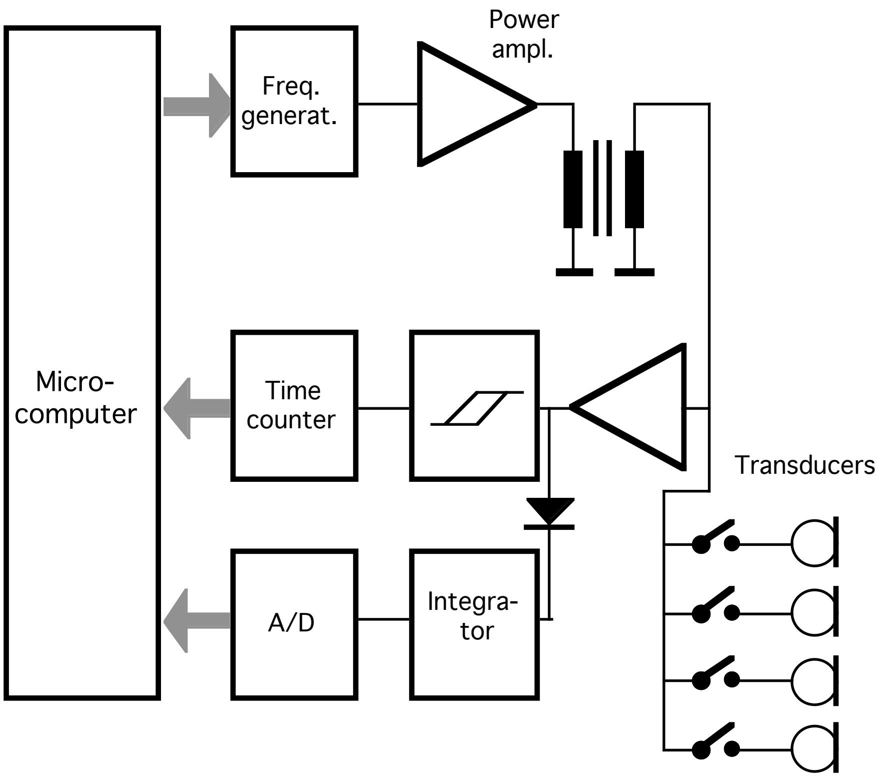

Figure 9 - Diagram of the sonar circuits.

Sonar readings are affected by several factors, such as the reflectivity of surfaces, their dimensions, the opening of the ultrasonic beam, and the ambient temperature. The importance of all these parameters is however quite small, except for the opening of the beam, that can lead to important errors, specially when scanning complex environments from a great distance. Figure 10 shows a computer simulation of a sonar scan, obtained with a beam opening of 30°, and with 192 partially overlapping readings. Gray dots represent the environment, and black dots represent simulated readings. The sensor position is indicated by the light gray hexagon. It can be easily seen that far objects are seen in a distorted way[6]. Figure 11 represents the polar diagram of the same scan. An important consideration that can be done is that all measurements, if wrong, yield a distance value

[6]For instance one of the openings, that represent doors, has completely disappeared.

that is smaller than the actual one: this means that the robot has no chance of colliding with an object because it thinks it is farther than it actually is; when approaching the measurements will anyway become more precise.

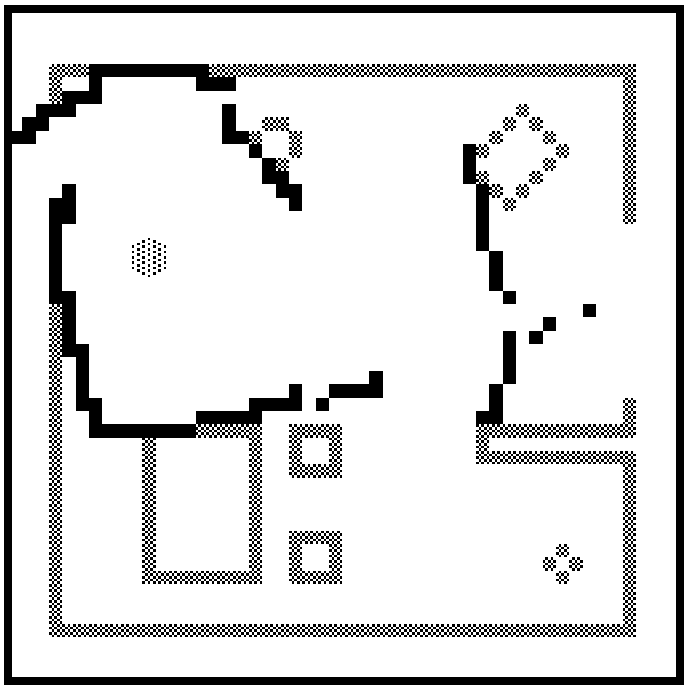

Figure 10 - Simulation of sonar behavior.

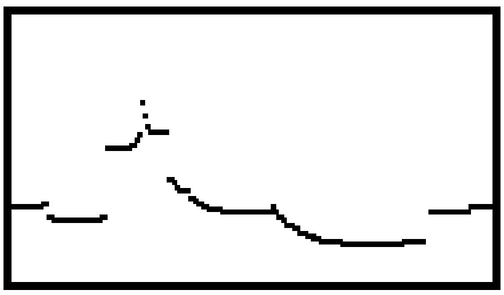

Figure 11 - Polar diagram for the environment of Fig. 10.

Different materials reflect the ultrasonic beam in different ways: in particular, some materials adsorb beams of particular wavelengths. For this reason, the beam frequency can be changed, in order to find the frequency that best suits each particular material.

Since the task of driving the sonar and the stepping motor does not saturate the computing power of this module, some other devices have been added. On the same shaft that carries the main sensor, a directional photodiode is placed and aims at the same direction as the sonar. This sensor can be used for several purposes, the main one being the detection of glowing LED's, that will be used for helping the robot in finding its way and in recognizing the environment. The block diagram of associated circuits is shown in Fig. 12.

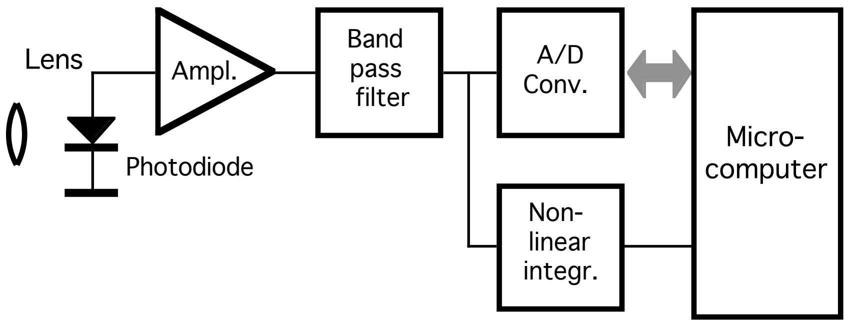

Figure 12 - Photodiode circuits.

After an amplifier and a band pass filter, the photodiode is connected to two different circuits: a level measuring device and a non-linear integrator.

Significant points of the environment will be marked with LED's, emitting modulated light that will not be confused with natural or artificial lighting. By rotating the sensor until the received signal reaches its maximum, the robot will be able to precisely locate the angular position of the LED with respect to itself. The signal emitted from the LED can also be coded, in order to make the robot capable of distinguishing among different LED's. The coding is done by transmitting trains of variable length pulses, that can be detected using the non-linear integrator and decoded with a simple software routine.

4.2.2. - Vision microcomputer

The hardware of this microcomputer is shown in Fig. 13. A Vidicon surveillance camera was used. The only reason for choosing this device instead of a solid-state camera was that it was readily available at no cost. A simple frame grabber was then built to interface the camera to the microcomputer. In order to avoid the need of having a large memory in the microcomputer and to speed up system's operation, it was decided to perform binarization of the image before transferring it to the microcomputer.

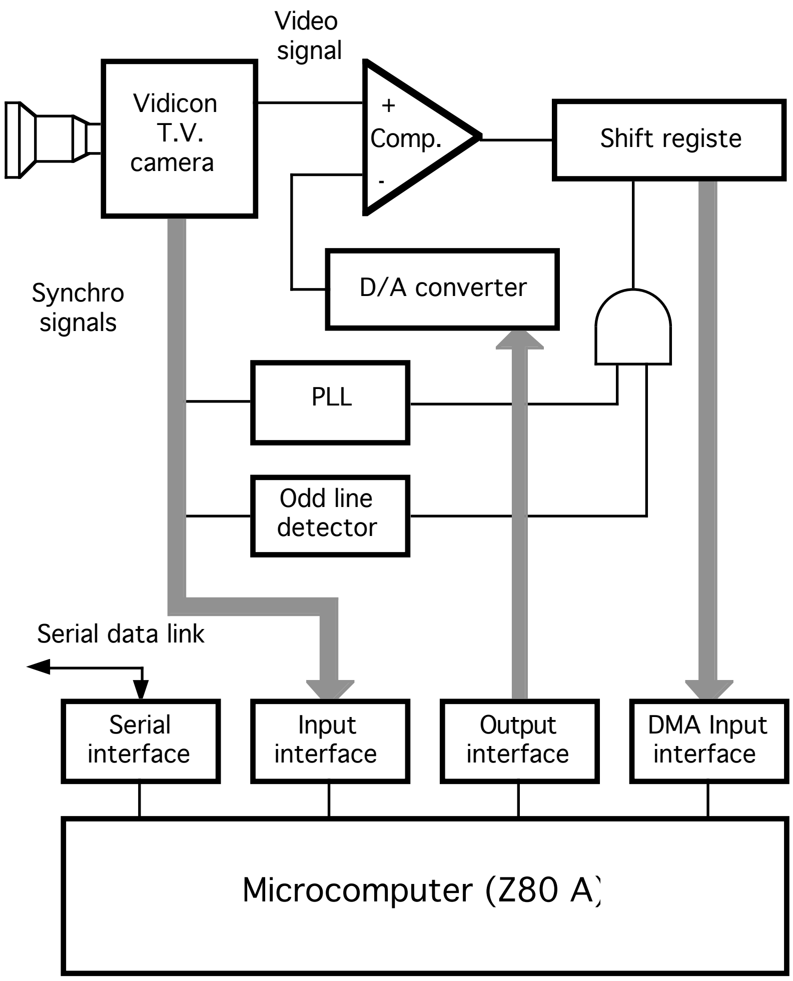

Figure 13 - Vision microcomputer block diagram.

The circuit works as follows: a logic circuit extracts synchronization signals from the camera, and produces, through a PLL, a master clock, whose frequency is 80 times greater than camera's line frequency. This clock drives an 8-bit shift register, that receives at its serial input the result of the comparison of the video signal with an analog signal that represents the chosen threshold, and that is produced from a digital to analog converter driven from the microcomputer. The shift register will thus be filled with sequences of zeros and ones, that represent binarized pixels. A DMA channel loads these bits in memory. The whole image is then grabbed during a single frame. In order to maintain more or less the same resolution along both axes, only even lines are used for image acquisition.

The obtained resolution is therefore 80 x 153 pixels, and requires only 1530 bytes of memory to be stored.

The main purpose of this module is not scene analysis and object recognition (this would be quite difficult with a single Z80), but rather the measurement of position and distance of easily recognizable objects, such as lamps. For this purpose, the camera lens is equipped with a mirror system that doubles the images seen by the camera, and allows measuring their distance with very simple calculations.

The program running on the vision microcomputer includes several functions, among which the most important is a connectivity analysis routine that allows detection of interesting objects against the background. Since the microcomputer was not designed for general-purpose recognition, a special algorithm was developed that does not allow computation of some "classical" parameters (perimeter of blobs, number of holes, centers of gravity, etc.), but that is extremely fast.

4.2.3. - Motion microcomputer

This is the simplest of all microcomputers (Fig. 14). Its only task is to generate two variable duty-cycle waveforms, that feed the motors through power buffers. The speed of the motors under normal load is roughly proportional to the duty-cycle. As it was said before, any discrepancies among the commanded and the actual motors speed should be corrected by the high-level control system. Some additional circuits take care of the sense of rotation and of limiting current in the case of motor blockage.

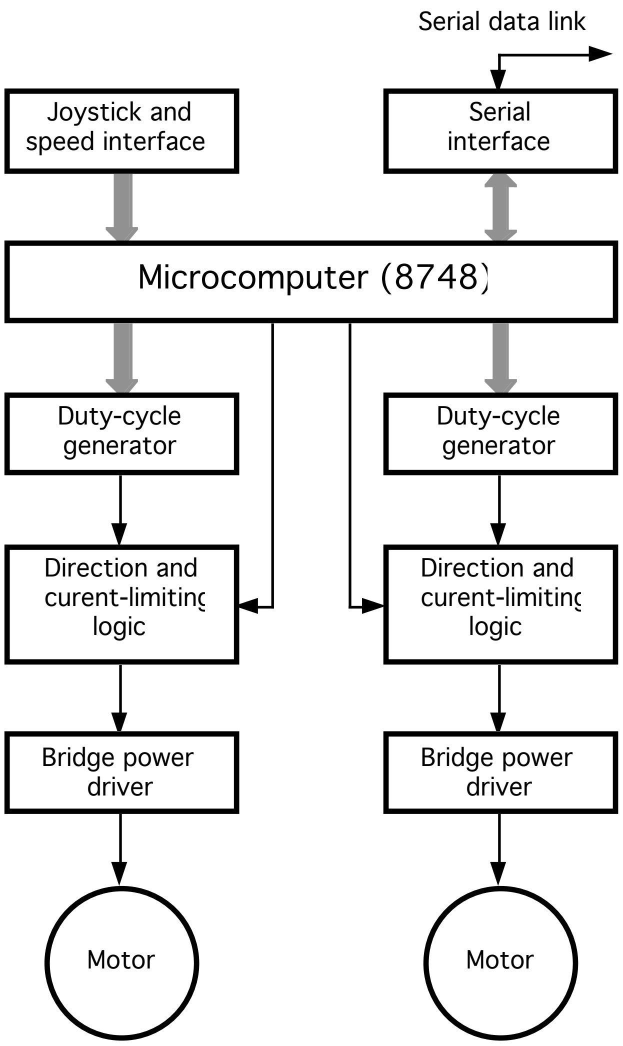

Figure 14 - Motion microcomputer block diagram.

The microcomputer takes also care of generating acceleration and deceleration ramps, in order to avoid excessive stresses on the gears and on the mechanical structure, specially during abrupt direction changes. Since the transmission is unidirectional, and there is no means of moving the robot if the motors are not running[7], a simple joystick circuit was added to manually displace the machine if the control system is not running.

4.2.4. - Power supplies.

Due to the various circuits employed in *RISK*, a number of different voltages were necessary. Primary power comes from two car batteries, that yield 24 V with a capacity of 40 Ah. This power is sufficient for several hours of operation, the power drain ranging from about 4A (when the robot is still) to about 12A (peak consumption during motors startup). In order to minimize power losses, all voltage conversions are done with switching supplies. A simplified diagram of the power supplies is shown in Fig. 15.

[7]The overall weight of the machine (including batteries) exceeds 100 Kg.

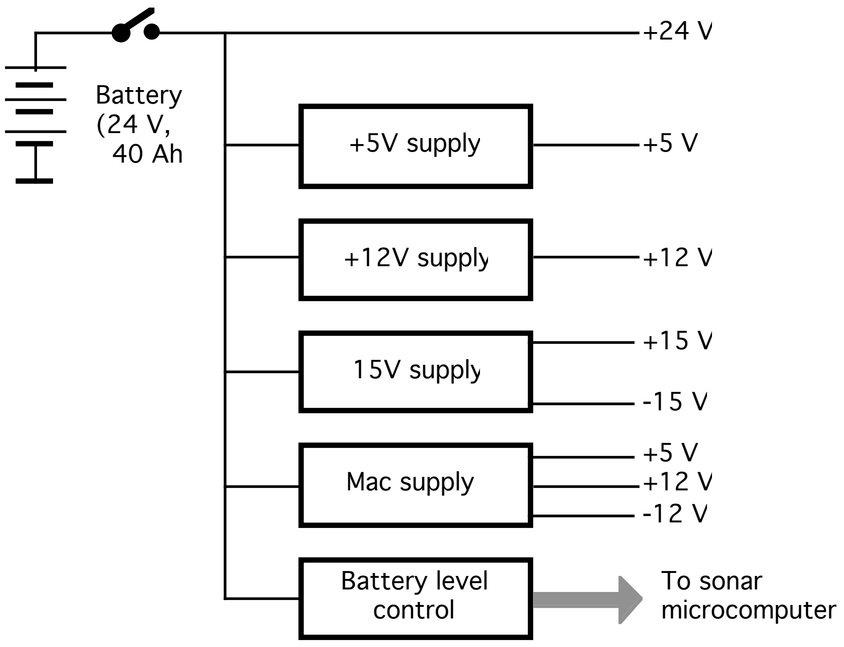

Figure 15 - *RISK* power supplies.

In order to avoid discharging the battery completely, some circuits were added that monitor its output voltage, and that automatically turn off the machine if the voltage is too low. Before doing this, however, the main computer is alerted through the battery level control system, that signals the low battery situation to the sonar microcomputer, which in turn alerts the main computer.

4.2.5. - Future modules

Other modules are planned for inclusion in *RISK* structure. Among them, the most important are two modules for driving two simple arms that will give the robot manipulation capabilities.

Other modules include more sensors, such as passive infrared sensors for detecting humans and animals, structured light beams for precise detection of the position of objects, etc.

4.3. - THE TASK

The first task that was chosen to demonstrate the capabilities of the robot is the following: the robot must move in an a-priori unknown household environment. The only restrictions that apply to this environment are that the whole flat lies on a single floor, and that no down-going stairs are present (the robot is not equipped with sensors that could detect a "hole" in the floor). In this environment, several plants are present, and each one is equipped with a special device, that glows an infrared light emitting diode when the plant needs watering. The robot's task is to wander around, looking for plants that need watering, and when it locates one, to reach it and water it.

The task can be easily described in terms of subgoals, according to the following sequence:

a. Locate a glowing diode;

b. Reach the glowing diode;

c. Pour water over the glowing diode;

d. Repeat from start.

Once a goal of the previous sequence is reached, the following one is "activated", i.e. it is set as the main goal the machine must pursue.

It must be noted that, in order to perform the previously described sequence of actions, it is not necessary for the machine to build a map of the environment.

Purpose *a* only requires that, if from the starting position no glowing diodes are detected[8], the robot moves so that it eventually explores all the

[8]Detecting a glowing LED involves a three-step algorithm: first, the photodiode mounted on the "head" of the robot is rotated until the modulated signal from the LED is detected, and further until it reaches its maximum value. Second, the robot rotates around its vertical axis until it points in the direction of the detected LED; third, the camera sensor is used to obtain a more precise measurement of the direction of the (*cont.*)

environment (the classical algorithm for doing this only requires that the machine follow the perimeter of the environment, and this is easily done using the sonar range finder).

When moving towards the glowing diode (purpose *b*), there is no need for looking around: the only direction the robot has to look at is in front of itself. If an obstacle is detected between the robot and the target, an alternate strategy must be used to overcome the problem. This strategy is easily implemented by changing the direction of the motion, and by observing rules coming from the behavior module about how obstacles should be avoided (minimum distance to maintain, what to do if obstacles seem to move, etc.). As soon as the obstacle is no more between the robot and the target, the previous purpose is resumed.

The process continues iteratively until the final goal is reached. In order to have a simple means of handling this iterative process, a stack of the sub-goals is maintained in the central computer (logically, it is in the blackboard). When a problem is encountered, new sub-goals generated from the strategy module are entered in the stack. When a sub-goal is reached, it is removed from the stack. When the stack is empty, the final goal has been reached.

4.4. - RESEARCH PROGRESS.

As it was said, *RISK* will be the first testbed for *BARCS* architecture. The actual state of the research (August, 1987) is as follows:

- The functions of all *BARCS* logical modules have been defined, and interfaces between the modules and the blackboard are now being layed down;
- Simulation of critical *BARCS* module is being done; first results will be available by the end of 1987;

LED and to add to the data base information pertaining to the distance of the LED from the sensor.

- *RISK* construction is being carried on in parallel with the theoretical research. All mentioned *RISK* modules are already working, except for the sonar module, which is currently being built: *RISK* will therefore be ready by the end of 1987;
- Software for communication among modules and test programs has already been written and tested; manual control of all *RISK* modules from the Macintosh is already possible;
- 1988 will be entirely devoted to the implementation of the software architecture on the Macintosh, and to the refinement of software running on sensory modules.

Figure 16 shows the actual state of *RISK* mechanical and electronic part.

Figure 16 - *RISK* as it is now (August, 1987).

5. - CONCLUSIONS

It was shown how traditional robots are intrinsically limited in their performance, and how a non-traditional control structure may be much better suited to drive robots of the next generation.

A structure was proposed, that should overcome problems of non-structured environments and difficult to describe tasks.

The experimental realization of this structure, and the related implementation problems, were discussed, and the experimental results obtained so far were reported.

6. - ACKNOWLEDGEMENTS

The author wishes to thank Ernesto Biroli, Paola Boselli, Luca Emaldi, Alberto Meregalli and Fabio Scalise, who have carried on most of the theoretical investigation required to establish the principles of *BARCS* structure.

Credit is also due to Patrizia Adamo, Franco Conti, Zoe Mapelli and Daniele Pons for having participated in the project of *RISK* and for cooperation in designing and realizing the hardware.

A particular thank goes to Gaetano Lomazzi, who realized the mechanical components of the robot frame.

Finally, the author wishes to thank Giorgio Zoly, whose continued and enthusiastic cooperation in the design and construction was essential for the realization of the robot.

BIBLIOGRAPHY

[1] ESPRIT Project No. 623: "Operational Control for Robot System Integration into CIM: System Planning, Implicit and Explicit Programming", 4th Interim Report, IPK, Berlin (1987).

[2] Bison,P., Lorenzin,G., and Pagello, E.: "The Formal Definition of VML and a Proposed Portable Implementation", Proc. XI ISIR, Tokyo (1981).

[3] Brooks, R.A.: "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, March, (1986).

[4] Cassani, F., Cassinis, R., Manes, A., and Miracola, G.: "HVML: High Level Virtual Machine Language - Preliminary Definition", ESPRIT P623 Project, Working Paper IP - PdM - 4.87/1, Milano (1987).

[5] Cassinis, R.: "An Application of Automatic Resource Sharing to Robot Programming", Proc. 3rd International Symposium on Robotic Research, MIT Press, Boston, Ma. (1986).

[6] Cassinis, R.: "Automatic Resource Allocation in Industrial Multirobot Systems", Proc. 1986 IEEE Conference on Robotics and Automation, San Francisco, Ca. (1986).

[7] Cassinis, R.: "An Example of a Distributed Intelligence Discrete Process Controller", Digital Systems For Industrial Automation, vol. 1, n. 1, New York (1981).

[8] Cassinis, R., and Guida, G.: "Intelligenza Artificiale e produzione industriale", Produzione e Computer, supplemento a "Sistemi e Automazione", n. 264, Milano (1985) (In Italian).

[9] Cassinis, R.: "Le nuove frontiere della programmazione dei robot", Macchine, Milano (1986) (In Italian).

[10] Cassinis, R.: "Hierarchical Control of Integrated Manufacturing Systems", Proc. XIII International Symposium on Industrial Robots, Chicago (1983).

[11] Cassinis, R., Biroli, E., Meregalli, A., and Scalise, F.: "Behavioral Model Architectures: A New Way Of Doing Real-Time Planning In Intelligent Robots", Proc. SPIE's Cambridge '87 Symposium on Advances in Intelligent Robotics Systems, Cambridge, Mass. (1987) (In the press).

[12] Chatila, R.: "Mobile Robot Navigation: Space Modeling and Decisional Processes", Proc. 1986 IEEE International Conference on Robotics and Automation, San Francisco (1986).

[13] Fox, B. R., and Kempf, K., G.: "A Representation for Opportunistic Scheduling", Proc. 3rd International Symposium on Robotic Research", Faugeras & Giralt, Eds., MIT Press, Cambridge, Ma. (1986).

[14] Gallerini, R., and Somalvico, M.: "General Structure of the Error Recovery Subsystem: Skeleton Extractor, Signal to Symbol Transformer, Sensory Robot and World Perceptor, Error Recovery Module", ESPRIT P623 Working Paper IP - PdM - 12.86/1, Milano (1986).

[15] Harmon, S.Y., Aviles, W.A., and Gage, D.W.: "A Technique for Coordinating Autonomous Robots", Proc. 1986 IEEE International Conference on Robotics and Automation, San Francisco (1986).

[16] Linden, T.A., Marsh, J.P., and Dove, D.L.: "Architecture and Early Experience with Planning for the ALV", Proc. 1986 IEEE International Conference on Robotics and Automation, San Francisco (1986).

[17] Paul, R.P., Durrant-White, H.F., and Mintz, M.: "A Robust, Distributed Sensor and Actuation Robot Control System", Proc. III International Symposium of Robotics Research, MIT Press, Boston (1986).

[18] Rembold, U.: "Programming of Industrial Robot, Today and Future", Proc. NATO Advanced Workshop on Languages for Sensor-based Control in Robotics, Il Ciocco, Castelvecchio Pascoli, Italia (1986).

[19] Saridis, G.N., and Valavanis, K.P.: "Mathematical Formulation of the Organization Level of an Intelligent Machine", Proc. 1986 IEEE International Conference on Robotics and Automation, San Francisco (1986).

[20] Steffin, S., and Cuneo, M.: "CHIPWITS User Manual", Brainworks Inc., Calabasas, Ca. (1984).

[21] Whitney, C.K.: "Building Expert Systems Where no Experts Exist", Proc. 1986 IEEE International Conference on Robotics and Automation, San Francisco (1986).

INDEX OF ILLUSTRATIONS