Restore operations for Spaceship Computational Systems

Ronald E. Oribio, Rebeca Monasterios, Eduardo Fernández, Mª Gabriela Vicioso, Antonio Rodríguez

Abstract—**This is the Team Description Paper of the Caerus team participating in the SEK category of the seventh Latin American IEEE Robotics Competition. While returning to Earth from an exploration mission to Mars, an electromagnetic storm caused a spaceship's Main Computer to crash and lose its course. The project consists of building two robots, working in an efficient and cooperative way. Together, they must reestablish the spaceship's computational system, in less than five minutes, and prevent the spaceship from colliding with a nearby planet. This paper describes the hardware and software approaches used by the team as well as the difficulties encountered during the development of the robots.**

I. Introduction

THE Latin American Student Robotics Competition of the Institute of Electrical and Electronics Engineers, IEEE [1], is an event which on its seventh occasion, students and investigators, get to share their knowledge and experiences in both robotics and artificial intelligence fields.

Five students, from *Universidad Católica Andrés Bello*[2] (UCAB) located at Caracas -Venezuela, have developed the required prototypes to be able to participate in the SEK category and complete the task proposed in this year's competition, share their ideas as well as learn from other participants.

The main objective of the SEK Category 2008's edition, is to use autonomous robots constructed utilizing only official parts from one of the many popular education kits available on the market to quickly accomplish a system restore operation of a spaceship's computational system which has lost its course from its safe journey back to Earth. To complete this mission, the robots must navigate through the different rooms of the spaceship and retrieve the most recent backup tape along with the corresponding password key. After both robots have arrived to the Main Computer's inputs, the system can be accurately restored, only if the backup tape and the password key are inserted into the correct inputs. This task must be accomplished in a synchronized manner, within a time frame not greater than five seconds.

The spaceship was modeled as an arena made from MDF wood (see Fig. 1) of a fixed size (2 m x 3 m x 10 cm) and divided into two sections to represent the ship's different levels. The backup tape and password key are represented as two cubes 2 cm in size and colored black.

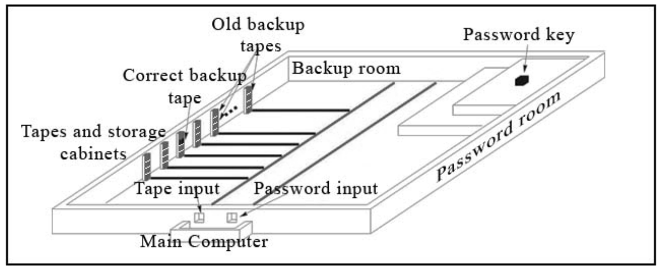

Fig. 1. Diagram of the arena

Taking into account all the rules governing this year's challenge [2], possible scenarios of the problem, as well as available resources, the Caerus team implemented several ideas to solve the problem at hand. As a result, two prototypes, both autonomous and fully operational robots that are able to work cooperatively to restore the spaceship's system, were constructed.

The two robots are not equal, and each of their designs intends to optimize the characteristics they must possess for them to accomplish the challenge efficiently. Nonetheless, during their construction many of the ideas and principles on which both robots are based, as well as several characteristics that make up each one, were shared. The goal was to capitalize on the design decisions which we found to be most effective during our experiments with Robot A. With this approach, we were able to develop Robot B faster and optimize its design making it lighter, simpler and more agile so as to accomplish the harder task faster.

II. Hardware Design

All the hardware used for robots are official LEGO parts, from MINDSTORMS educational kits #979786 & #979797 [3], this includes not only wheels and body, but also engines, sensors and processing units.

The restrictions defined by the competition for each robot are:

- It must fit in a 25 cm cube at all times, even with its moving parts completely opened or extended
- It may use any number of processing units
- It may use no more than six sensors
- It may use no more than six actuators

The mechanical design of the robots took into account the

[2] The authors/ Caerus Team members are part of the Grupo de Robótica y Automatización, ROA UCAB (e-mail: roaucab@gmail.com).

specifications and restrictions of the competition as well as other factors, such as available resources and time frame to complete the project. The aspects that were considered necessary for the completion of the task are the following:

- A structure that allows ease of movement and capable of turning in place
- The robots must be equipped with some type of mechanism that enables them to grab and deposit the cubes
- A safety method must be applied to ensure all cubes will remain vertically stacked when removing the correct backup tape
- Both robots must implement a way of aligning themselves to the storage cabinets and Main Computer inputs
- The mechanism used to grab the cubes must be able to ascend and descend vertically
- It must have light sensors that enable it to follow the lines that lead back to the deposit location
- Capability to detect the stairs and to go up and down the steps without difficulty
- The robots must be able to withstand the collisions that the robots can suffer during the competition and not lose any essential pieces
- The robot must also have additional light sensors to allow for cube differentiation and recognition
- In case of the existence of multiple, and possibly different types of processing units, a method of internal communication between them must be implemented and guaranteed
- A method of external communication between robots must also be guaranteed so they can monitor the progress of the overall task and be alerted when both are in position to deposit the cubes and be able to do so in a synchronized manner

In order to fulfill the requirements for each robot, the team came up with a model that is based on the principle of *skid steering*, which consists of two wheels located at the back of the robot, to provide mobility, and ball casters located up in front, to allow for minimal friction when turning (see Fig. 2).

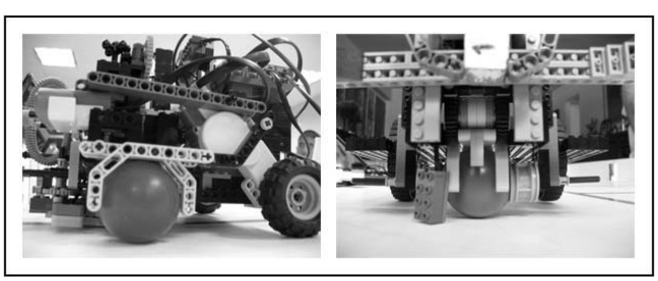

Fig. 2. Ball caster implementation on both Robot A (left) and Robot B (right)

Both robots have a two-wheel drive (2WD) mechanism; different types of wheels were used depending on the tasks to be carried out by each one. One of the robots would be in charge of retrieving the backup tape, referred to as Robot A, and the other of retrieving the password key, referred to as Robot B. Since Robot A would always travel on level ground, small and wide wheels were used for speed and accuracy. On the other hand, Robot B needed to climb two steps, so bigger wheels with treads were used for better traction.

To retrieve and recognize the cubes, two different mechanisms were developed. Robot A needed higher precision and control, so, and adaptation of the traditional lift-arm mechanism was made with a rack/worm-gear combination. As part of the lift-arm, a compact grabber was mounted along with a light sensor that allowed for exact positioning and precise retrieval of the cubes. Additionally, we devised a special bumper connected to the front of the lift-arm and attached a spring between them (see Fig. 3). When the robot moved forward to grab the cube, the bumper would compress to allow the robot to come within grabbing distance of the cube. As the robot moved backwards, after grabbing the cube, and the bumper decompressed, the remaining cubes in the stack would be pushed, keeping them from falling and maintaining their positions in the storage cabinets.

Robot B, on the other hand, had less demanding grabbing requirements, so the lift-arm implemented followed a more traditional approach, with some slight modifications. Two touch sensors were mounted on the front corners, a bar along its bottom for sweeping the cube, and positioned a grabber vertically on its center, for retrieving the cube.

Fig. 3. Detail of the adapted lift-arm with Grabber, light sensor and special bumper (left) and demonstration (right)

Light sensors were used, facing towards the floor, positioned on the bottom of the body of both units. Robot A utilizes two light sensors, for alignment and positioning in front of the storage cabinets, as well as for finding its way through the storage room and back to the Main Computer. A similar approach was used on Robot B, but with only one light sensor.

To improve and offer error correction in Robot A’s alignment algorithm, an Ultrasonic sensor was put on its left side. This sensor, combined with both of the light sensors on its bottom, allowed the robot to determine its relative position to walls and storage cabinets. This way, it would always be able to align itself, while maintaining the correct distance at the same time.

III. SOFTWARE DESIGN

Depending on the task that would be carried out, different types of hardware were needed, but due to the available resources, as well as restrictions put in place for the competition, only a combination of RCX and NXT hardware was possible. Therefore, each robot was equipped with one RCX (v2.0) and one NXT (v1.0) processing units (see Table I).

By design, RCX and NXT versions of the MINDSTORMS educational kits are incompatible, due to their technical characteristics. This made working with both kits somewhat of a challenge, and presented the team with compatibility issues, not only among types of hardware, but software as well.

Fortunately, some backwards compatibility was introduced in the NXT. This enabled the team to connect to any of the four NXT input ports, with the help of a special converter cable, RCX motors or sensors. Although this solved the hardware incompatibility issue to some extent, solving software implementation issues, and establishing RCX to NXT bi-directional communication, still remained a big concern.

TABLE I
ROBOT'S HARDWARE

Robot's Components		*Light Sensors*	*Touch Sensors*	*Ultra sonic Sensors*	*Motors*
Robot A	NXT	1	-	1	3
	RCX	4	-	N/A	1
Robot B	NXT	1	-	-	3
	RCX	3	2	N/A	2

To program the robots, it was found that the programming languages "Not Quite C (NQC)" [4] and "Not Exactly C (NXC)" [5], which relied on both of the processing unit's native firmware, were best suited for the project. These C-like languages are simple, fast and support the RCX and NXT respectively. Another important feature was that NXC introduced backwards compatibility support for the NXT, to control RCX sensors and motors.

While working with this style of combined hardware setup (see Table II), two different approaches were tried. In the first approach, events were triggered depending on the different sensors readings, past or present; in another thread, specific actions for each event were taken. This methodology, although elegant and theoretically sound, resulted to be ineffective and difficult to control during transitions from one state to another.

To solve this problem, the team relied on a second approach, a set of condition-action pairs were used. Every possible situation that the robots could encounter was modeled and matched to a specific action.

TABLE II
ROBOT'S HARDWARE DISTRIBUTION

		Robot A		*Robot B*	
		NXT	RCX	NXT	RCX
Motor Inputs	A	Left Wheel (NXT)	-	Left Wheel (NXT)	Support Lever (RCX)
	B	Right Wheel (NXT)	Grabber (RCX)	Right Wheel (NXT)	Grabber (RCX)
	C	Lift-Arm (NXT)	-	Lift-Arm (NXT)	-
Sensor Inputs	1	Light (NXT)	Light (RCX)	Light (NXT)	Light (RCX)
	2	Ultra-sonic (NXT)	Light (RCX)	Light (RCX)	Light (RCX)
	3	Light (RCX)	-	Touch (NXT)	-
	4	Light (RCX)	N/A	Touch (NXT)	N/A

Bi-directional RCX-NXT Communication

As seen in Table II, the different motors and sensors used by each robot were connected to one of its two processing units, either to an RCX or an NXT. As mentioned before, this implementation presented an incompatibility issue, which made direct bi-directional communication between RCX and NXT impossible.

A workaround, based on a previous proof of concept [6], was developed by using two light sensors facing each other and keeping their LEDs' turned on continuously. By placing them in this manner, both sensors were capable to determine the state of each other's LED, on or off.

By defining arbitrary pauses, during which the LEDs' would be off before turning back on, each processing unit was able to differentiate between both states and assign to each pause a one or a zero. This way, a simple, binary based, communication protocol was able to be put in place.

Since time efficiency was an important factor to complete the mission, and such a small number of commands and responses were required, communication was limited to three bits and pause time thresholds were lowered to a minimum. These three bit, binary, numbers were translated to its decimal equivalents and interpreted as single commands, paired to a particular action. Each action could generate a positive or negative response, indicating the success of failure of such action.

Although both processing units were independent, for them to be able to control the robot correctly, it was necessary for them to work as one cohesive unit. To solve this, a type of Master/Slave approach was utilized. The NXT worked as the master; here all of the program logic for each robot each robot would reside and decisions would be made. The RCX worked as the slave; responding only to commands received from the NXT.

ACKNOWLEDGEMENTS

We would like to thank Professor Alejandro del Mar, Professor Wilmer Pereira and Professor Iñaki Mendizabal, for all their help, dedication and support. We would also like to thank, all of our families, friends and colleagues who have supported us or collaborated and contributed to the project in some way.

REFERENCES

[1] 7th Latin American IEEE Student Robotics Competition
http://jri2008.dca.ufrn.br/LARC/index_en.php

[2] Rules for 2008 edition of the SEK Category
ftp://jri2008:web@users.dca.ufrn.br/regras/SEK2008_en.pdf

[3] LEGO MINDSTORMS, LEGO GROUP
http://www.legoeducation.com/store/

[4] NQC – Not Quite C
Dave Baum
http://bricxcc.sourceforge.net/nqc/

[5] NXC – Not Exactly C
John Hansen
http://bricxcc.sourceforge.net/nbc/

[6] Brickshelf.com – Gary 2511
http://www.brickshelf.com/cgi-bin/gallery.cgi?m=gary2511