MICRO ENGINEERING DEPARTMENT
MICROPROCESSOR SYSTEMS LAB (LAMI)
PROFESSOR D. FLOREANO & J-D. NICOUD
ASSISTANT JOSEBA URZELAI

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

PATRICK RAMER · JEAN-CHRISTOPHE ZUFFEREY

8TH SEMESTER, MICROENGINEERING

NAVIGATION WITH THE ROBOT ZEPHYR

LAUSANNE, JUNE 2000

CONTENTS

1 INTRODUCTION

1.1 The story of the robot Zephyr

The story of Zephyr is a long story. Already in autumn 1998, we decided to take part in the fire-fighting contest organized by Jean-Daniel Nicoud at the EPFL. We were a team of three people (with Sébastien Schlup for the mechanics).

Our choice for the main processor was influenced by the possibilities offered by the LAMI. Thus a Rok11 board based on a Motorola HC11 processor was mounted and the code was entirely written in CALM assembler. The only sensor that was necessary for this first contest (find and extinguish 6 candles without obstacles) was a linear camera. The speed controller was implemented with an analog electronic board. The great novelty was our extinction system based on a Kisag gas cylinder (see 3.1.2), which is still in action on the robot.

With this configuration (see the first image on the right) our robot won the first prize in the contest in April 1999.

After that, the decision has been taken not to stay on this little win but to pursue in this good direction. During the summer 1999, we negotiated with K-Team, a small enterprise that produces autonomous robots, in order to obtain their robotic board, the Kameleon. This powerful equipment rapidly took the place of the Rok11 (see the second image on the right) in order to carry out the first tests and to become familiar with the 68'376 processor.

Our objective was to compete in the "Fire Fighting Home Robot Contest" in Hartford, Connecticut, USA, in April 2000. The purpose of this contest is to induce autonomous robots to fight a fire (represented by a candle) in a home structure embodied in a scaled down house model containing rooms (see 2.1).

Due to the strong charge of such an experience, our team has been extended with two more people: Cyril Halter (for the ultrasonic sensors) and Marc Vettovaglia (for the sponsoring).

The most important matter was the **navigation** in the house model. Therefore Zephyr has been improved with three ultrasonic sensors and two flame detectors.

This version of the robot is shown in the third photography (in Figure 1).

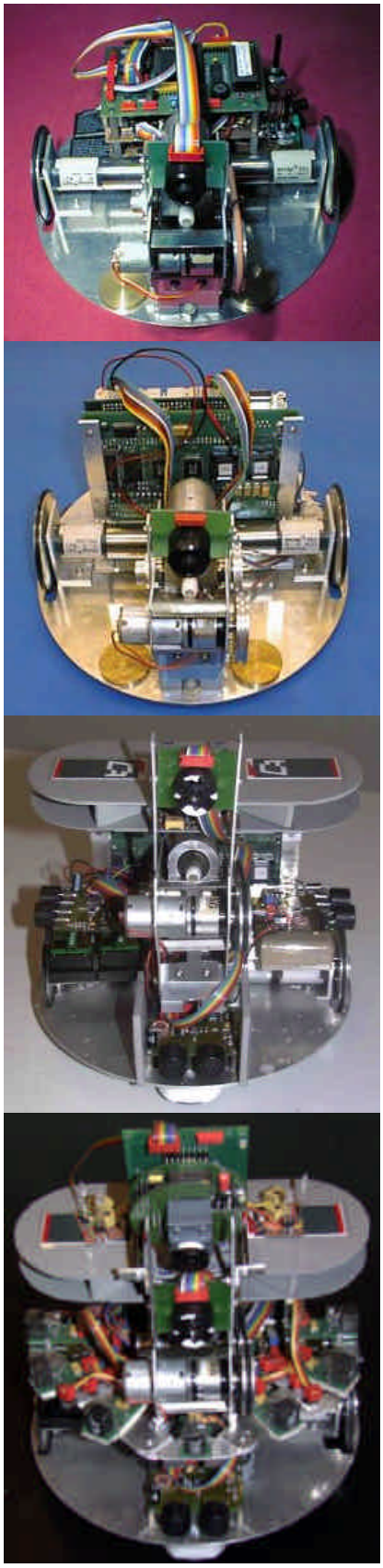

Figure 1 - Zephyr from 99 to 2000

After some discussions with Dario Floreano, we chose a standard method for the navigation in the maze. Some behavior-based solutions were taken into consideration but the high speed of the robot and the very well defined environment lead us to implement very simple, **model based** functions.

However we didn't have enough time to choose the higher level in the contest with the presence of furniture (embodied by cylinders) in the rooms. This level introduces a perturbation in the well-defined environment of the maze. For this reason we wanted to try some **behavior based** modules. After the contest we mounted a 2D camera and 7 proximity sensors (see the last photography in Figure 1) in order to provide the useful input values to the modules.

1.2 Official description of our semester project

The aim of this project is to evaluate the possibilities and performances of a behavior-based approach on a particular application for an autonomous mobile robot. The task is to find a lit candle in a model floor plan structure of a house and then extinguish it in the shortest time. Our robot "Zephyr", which already exists, will be used as the basic tool.

During the first part of the project the robot has to be developed (including the use of classical approaches) in order to participate at the "Fire Fighting Home Robot Contest" organized by the Trinity College in Hartford (April 2000).

During the second part, different modules and behavior-based techniques will be applied and tested on the robot. Some specific methods like neural networks will be used. A final evaluation of the different possibilities has to be made.

1.3 How to read this document

As seen in the preceding paragraphs, other people that did not take part in this semester project have also worked on the robot Zephyr.

However this document essentially contains information about our part of the work during the semester project, that is sensors, software implementation and navigation strategy. Other domains such as mechanics and the contest description are mentioned in a summarized form to complete the description of the entire project "Zephyr".

Since we used two completely different approaches for the navigation, we also make a difference between the descriptions of the two in this report. All **paragraphs marked with a star *** contain information about devices or part of code that have been added after the contest for our behavior based experiences.

2 THE CONTEST

2.1 Philosophy

The robot Zephyr was built to participate at a particular robot contest in the first place. For the second part of this project, the robot and the idea of the contest served us to execute further experiences. The concept, construction and features of Zephyr were chosen in order to match the contest's challenge. Therefore the contest principle and rules as well as some important technical features will be outlined here.

The aim of the contest is to promote the development of robotics. The suggested application is the fight against fire. This fight can be considerably simplified by using robots. Their advantages are the speed of intervention and the local action. Above all, their autonomy limits the risks of human losses. Those robots would be able to act in houses, halls or storerooms where they can also be stored. When fire breaks out, they can immediately be put into action.

Such an environment is simulated, at the contest, with a scaled down house model containing 4 rooms:

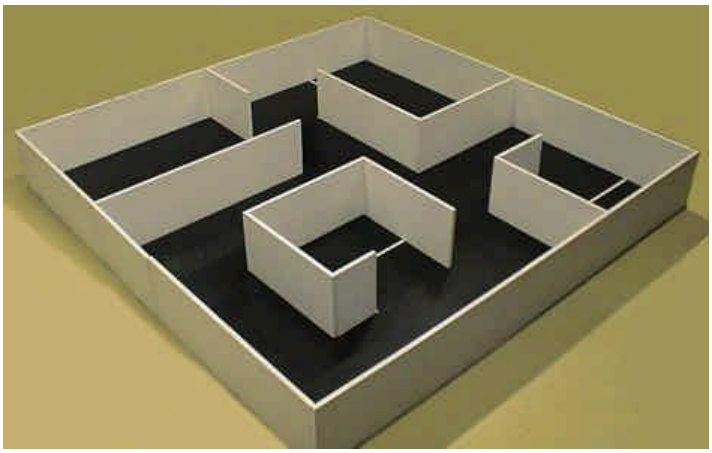

Figure 2 - Model floor plan structure of a house

Figure 3 – A ramp in a corridor of the maze

This makes it look a bit like a **maze of 2.5 by 2.5 m**. In one of the rooms is placed a candle representing a fire. **The robot starts at a well-known starting position and has to find and extinguish the candle in the shortest time.** There are **several levels of difficulty**, which will be considered to determine the final score of a robot's run. For instance there may be **obstacles in the rooms** (highest level) and **ramps in the corridors** to make the environment more realistic. Ramps (see Figure 3) help to prove that the robot works in a **non-dead-reckoning mode**.

The contest is held at Trinity College in Hartford, Connecticut, USA every year in April. The responsible organizer is Jake Mendelsohn. This year it took place on the 15th and 16th of April 2000. There were about 100 teams from all over the world taking part. Registration is open for everyone (see [II]).

2.2 Important features

It is useful to mention here also the most important rules and features of the contest, which limit the choice of sensors and navigation techniques.

The numbers, which are given here, may appear a little strange. But since the contest is organized in the USA, the original dimensions are defined in inches and feet. We will directly give the dimensions in the metric reference system including possible rounding errors due to the conversion.

Each robot has got **three trials** to find and extinguish the candle. The two best runs are taken to evaluate the final score. However if the third run also is successful, this will add a little 10% to the final score.

The plan of the maze is known before hand with a precision of ±2.5 cm for all dimensions. The hallways are 46 cm wide. The same is true for all four doorways that lead into the rooms. There are no doors in the doorways, just a 46 cm wide opening. The walls are 33 cm high. The floor is painted in black; the walls are painted in white. The floor may have discontinuities of up to 3.2 mm. Every room entrance is marked by a white stripe of 2.5 cm wide.

The candle may be put anywhere in a room except in the entrance area. This candle-free entrance zone is 33 cm deep into the room. **It is not known before hand in which room the candle is being placed.** It will be placed at random in one of the rooms in the arena. The candle has an equal chance of being in any of the 4 rooms in each of the 3 trials that a robot has. However the number of the room in which the robot finds the candle will influence the score in order not to favor a robot that finds the candle in the first visited room to another one that finds it only in the fourth room. The candle must not be touched and has to be extinguished from a distance of at most 30 cm. A white stripe marks this 30 cm radius around the candle.

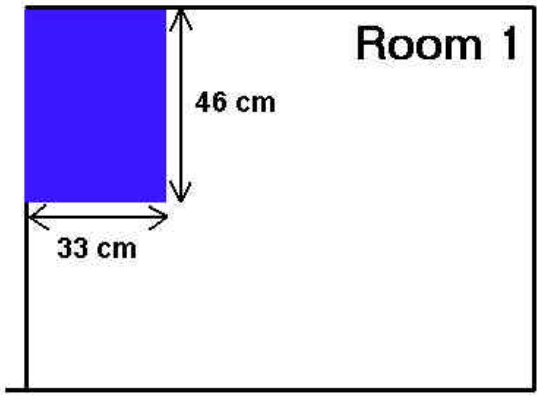

Figure 4 - Free zone of room 1

The height of the candle's flame base is between 15 and 20 cm. The height of the flame is not specified and may vary as well as long as it can be considered as a "normal" flame. The candle is mounted on a wooden base (7.6 cm x 7.6 cm x 3.2 cm) painted semi-gloss yellow.

Penalties are given for touching or bumping into walls or touching the candle.

The maximum size of the robot shall be 31 cm in all 3 dimensions. It is not allowed to look over the walls, which are 33 cm high. An external computer may also control the robot.

There are five **possible operating modes** at the contest. Each of them can be independently chosen and can be combined with any other mode. The choice of any supplementary mode will add factors to improve the final score:

- **Standard operation**
- **Sound activation** (gain of 5% on the final score): instead of being manually activated by the press of a button on the robot, the robot activates itself when it detects a 3.5 kHz sound signal.
- **Return trip** (gain of 20% on the final score): after extinguishing the candle, the robot returns to the starting position.
- **Non-dead-reckoning** (gain of 40% on the final score): one or more inclined sections will be placed in the hallways of the arena, which will have the effect of changing the distance to the rooms. Thus odometry cannot be used alone for navigation.
- **Furniture** (gain of 50% on the final score): there will be one piece of furniture in each room. The furniture will be placed at random in the rooms. The robot may touch the furniture, but it cannot push it out of the way. The furniture will be made of 11.5 cm diameter steel cylinders, painted semi-gloss yellow. The cylinders are 30.5 cm high and weigh 2.25 kg.

For the official, detailed rules, you can take a look at the "**Trinity College Fire Fighting Home Robot Contest**" web site (see [II]).

2.3 Our results as competitor

The first place has been reached in both contests in Lausanne (see [I]): in April 1999 (no navigation) and in April 2000 (in the maze). For this last participation, we decided not to use ramps (see Figure 3) because they were much more difficult to drive over than the ones used in the USA (they were ours, therefore they were built according to the worst case). Moreover, we didn't know where they could be placed officially. The most important problem was the settings of the flame detectors (see § 3.3.4), which are sensitive to the type of light in the contest room.

For the contest in Hartford, we opted for the **"sound activation"** and **"return trip"** modes. The **"non-dead reckoning"** mode (with ramps) was foreseen if possible. The "furniture" mode (with cylinders in the rooms) was not implemented due to its too high complexity. See § 2.2 for more precision about the mode description.

Arrived in Hartford (see [II]), we had to recalibrate all the distance constants because of the allowed imprecision of the maze dimensions. The ramps were really much more easy to drive over but it was impossible to obtain a good information about their potential places. More problematic, the front ultrasonic sensor was sometimes sensitive to these American ramps. The front range finder detected the ramp instead of giving the distance to the next wall. However, during the night that preceded the contest day, we decided to modify Zephyr in order to make it be able to perform the runs with the ramps. **The front sensor was mounted just above the extinction system**. Certain bends were reprogrammed in order to handle the situation if there would be a ramp.

The contest day arrived. For the two first runs, one ramp has been placed at an unforeseen location in a bend. However Zephyr performed well the first trial. But the second one was a disaster. The third trial was pretty good, but the flame detectors had not been well adjusted. As a result, Zephyr entered each room for a complete scanning

what led to an important loss of time. Our final ranking was the 12th place out of 81 participants.

See § 7.2.1 if you want to learn some further details about the encountered main difficulties.

In conclusion, Zephyr was among the most efficient robots (high speed, flame detection from the corridors, non-dead reckoning mode, etc.) but it was **very sensitive to the unforeseen events** (like the location of the ramps or the imprecision in the maze's dimensions). It was also **totally unable to find his way after an important error** like bumping into a wall. Therefore an excursion in the behavior-based approach (see chapter 6) could be a very good thing to improve this robot.

3 THE ROBOT ZEPHYR

3.1 Mechanics

3.1.1 Platform

Zephyr is based on a standard architecture. It is equipped with two motor wheels and two free balls in ball-bearing housing working as fulcrums. The motor wheels are directly mounted on the axis of their own DC motors (Portescap 22N48, 12V, 3.8W).

Figure 5 – DC motor with reduction gear

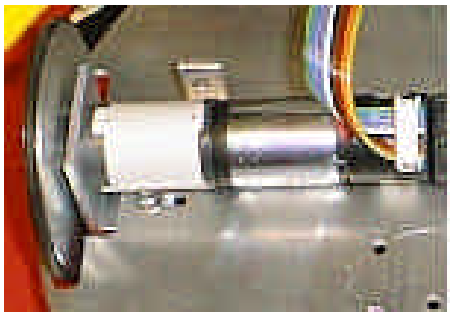

Figure 6 – DC motor with gear and encoder

These motors have a planetary reduction gear (Portescap R22, reduction ratio: 16.2) and an optical encoder (Portescap A22, 100 lines/revolution).

Figure 7 – Both motors on Zephyr

This configuration allows reaching a velocity of about 1.5 m/s after acceleration on 2 m. However, in the maze, the velocity has been maintained at about 60 cm/s in order to reduce the acceleration distances.

The rear free ball is mounted on a suspension device to let the robot roll over ramps. The front free ball is fixed in order to maintain the correct attitude when the robot brakes in front of a candle.

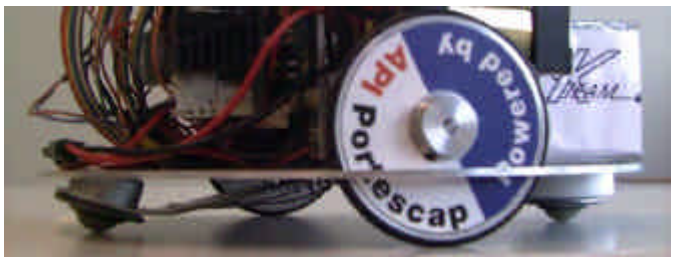

Figure 8 – Wheel arrangement

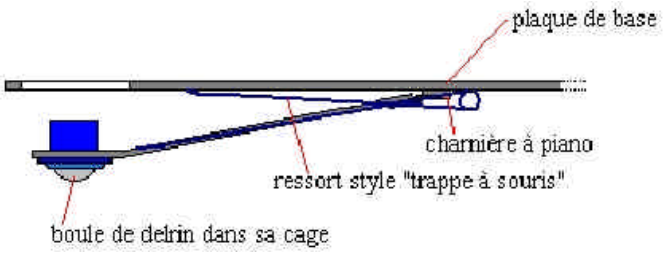

Figure 9 – The rear free ball system

Completely made of aluminum, the basis platform is a perfect circle in order to avoid any problems when the robot turns on itself. No on-board device passes over the boundaries of this circle.

Figure 10 – Circle basis made of aluminum

3.1.2 Extinction system

Our system is based on a gas stream (N_2O) provided by a "Kisag" cylinder (normally used to make Chantilly cream). This gas cylinder is placed in an aluminum cylinder, which is mounted on two spring sheets. A little DC motor drives a camshaft to which it is connected by a plastic belt. The cam is able to pull the cylinder in its armed position. At this point, if the motor begins to turn, it immediately releases the gas cylinder, which will strike an awl and free a part of the gas.

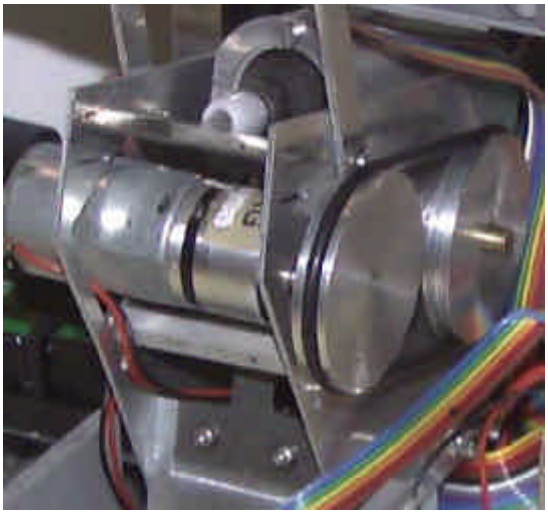

Figure 11 – Extinction system

Figure 12 – Kisag gas cylinder

This system is able to extinguish the flame of a candle from more than 50 cm away in a range of 15 to 20 cm above the floor. One gas cylinder allows about 50 extinctions.

3.2 Kameleon board

The central unit of the robot Zephyr is a Kameleon board, developed by K-Team (see [VI]). This board has been specifically conceived for robotic applications, which makes it very powerful and flexible for our application. On this board is a **Motorola MC68376 processor**, which is clocked at a frequency of 22 MHz. This processor manages the whole system.

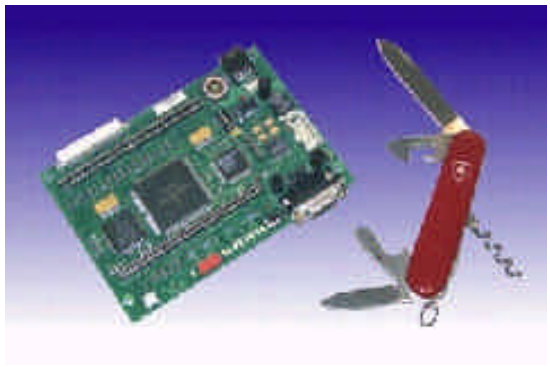

Figure 13 – Kameleon board from K-Team

The processor supports multitasking and can be programmed in assembler and C. **We have entirely programmed it in C.**

A second board, called **Robotics Extension Board or REB**, is mounted on Kameleon. This piggyback contains all motor connectors and allows accessing the different processor ports. On this board are also mounted 4 amplifiers for the motors.

Some specifications of Kameleon: 1MB RAM, 1MB Flash memory, two serial interfaces RS232 (used for the primary communication with the board) and several serial and parallel extensions (for more information see the downloadable documentation on the K-Team website [VI]).

A **BIOS** (Basic Input/Output System) is already implemented on Kameleon. The BIOS manages the low-level functionalities like the control of timing and multitasking, the serial connectors, the A/D converters and the input/output ports.

A very practical tool of the BIOS is the **management of the motors**. PID (Proportional Integral Derivative) controllers for velocity and position of the motors are already implemented and can be configured by the use of simple BIOS functions. The motors are controlled by a PWM (Pulse Width Modulation) above audible frequencies.

3.3 Sensors

3.3.1 Encoders

As basic sensing devices, the optical encoders from Portescap are mounted directly behind the driving motors. Powered by a 5 V supply, they give TTL compatible signals as output. As usual, the two output signals of each device are 90° phase shifted.

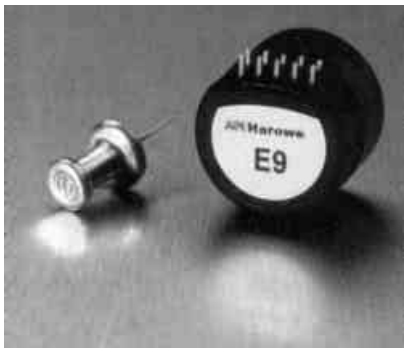

Figure 14 - Optical encoder

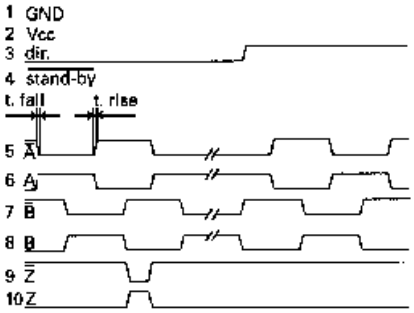

Figure 15 - Optical encoder signals

These sensors allow the Kameleon board and its processor to control the speed of each wheel. They are directly connected to the REB (Robotics Extension Board) such that the processor can apply its integrated **PID controller** (see § 3.2 for more information about the Kameleon board).

Another useful application of these sensors is the **odometry**. There is one counter register per wheel, which is incremented at each impulse from the encoder. A task (see § 5.2) in the main processor is therefore able to calculate the current position (x, y, θ) of the robot.

However, the position given by the odometry is not that precise, especially when Zephyr must drive over the ramps. In order to solve this problem, we decided to mount three distance sensors.

3.3.2 Ultrasonic sensors

We wanted to have at our disposal range finders capable of measuring the distances to walls with a **high reliability and at a high frequency**. For this reason, we chose the ultrasonic sensor that has been developed by Marco Seiz and Unai Viscarret at the Autonomous Systems Lab, EPFL (see appendix 11.1.3 for further information):

Figure 16 - Ultrasonic sensor

As a reminder, the principle of an ultrasonic sensor is to emit a 41kHz signal (burst) and to measure the time of flight (ToF) to its reception after its reflection on an object:

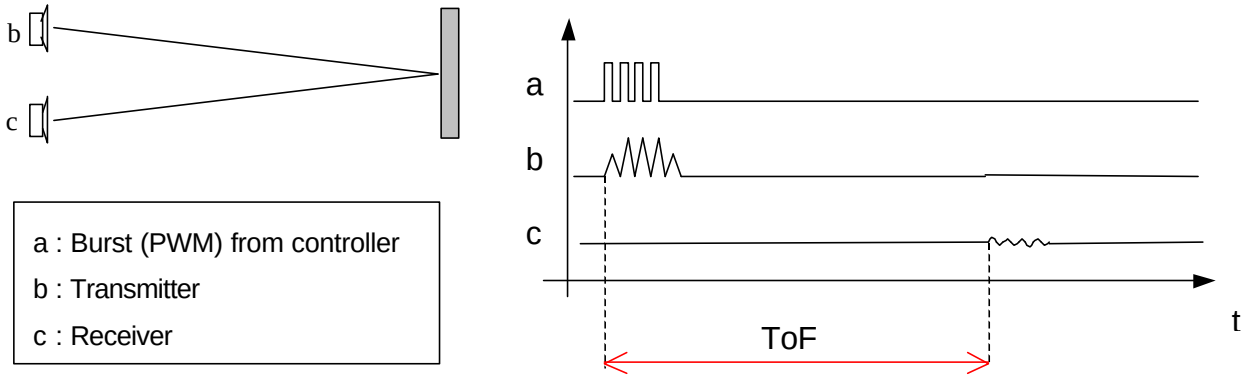

Figure 17 – Functioning of ultrasonic sensors (extracted from appendix 11.1.3)

This device is able to measure at a minimum frequency of 40Hz (in continuous mode) what represents a traveled distance of a bit more than 1cm if the robot's speed is 50cm/s. The frequency of the measures depends on the range mode. Three modes are available:

- Range: 1m **Frequency in continuous mode: 160Hz**
- Range: 2m Frequency in continuous mode: 80Hz
- Range: 4m Frequency in continuous mode: 40Hz

Moreover the sensor presents a **precision of about 1mm** (in 1m mode). It is also able to measure distances as close as 2cm. Of course, it is independent of the ambient light and well adapted to our environment made of straight planks.

The communication protocol is the I²C. It is possible to connect several sensors to the same bus.

The drawback of this range finder is the **limited angle between the device and the reflecting wall.** We have determined these maximum angles for the 1m-mode:

- Vertically: ~20°
- Horizontally: ~35°

Another disadvantage is the possibility of **multi reflections** of the ultrasound in the maze that can come back to the receiver at a bad moment and can be interpreted as a good measures (see § 7.2.1).

In order to implement the **wall following controller** (see § 5.5 for more details), two of these sensors have been mounted above the wheels, on each side. It is also necessary to know the **distance to the next front wall** in order to determine the global position of the robot without odometry. For this reason one more ultrasonic sensor has been placed at the front of Zephyr, just below the extinction system.

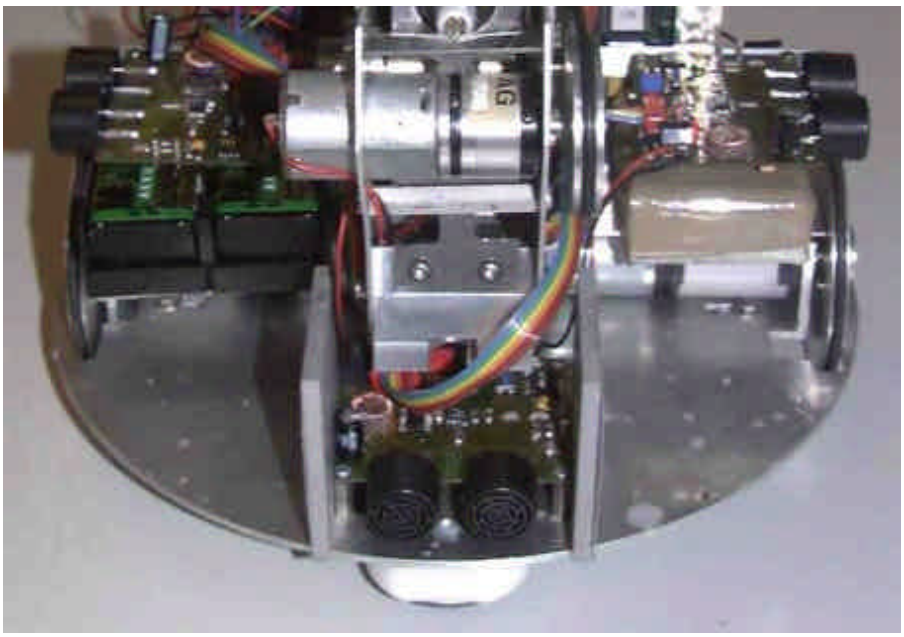

Figure 18 – Ultrasonic sensors arrangement

Now that Zephyr can navigate in the corridors, it needs one more sensor which will allow it to find the candle.

3.3.3 Linear camera

In order to "see" a candle, Zephyr has a TSL1301 linear camera (by Texas Instruments). This chip is made of 102 photosensitive cells, which compose the **102 pixels** of the one and only line. Directly in the chip is an analog shift register, which can transmit the values of each pixel in synchronization with a provided clock. These analog values are then converted into bytes (gray level) by the Kameleon processor by one of its analog converters (see § 5.4.1 for more information about the function that reads this sensor).

Prof. Jean-Daniel Nicoud at the LAMI, EPFL, developed the board that supports this chip with the lens.

The typical image (interpreted by our Supervisor, see § 5.1.2) from such a device is given just below. The robot was placed in front of a candle.

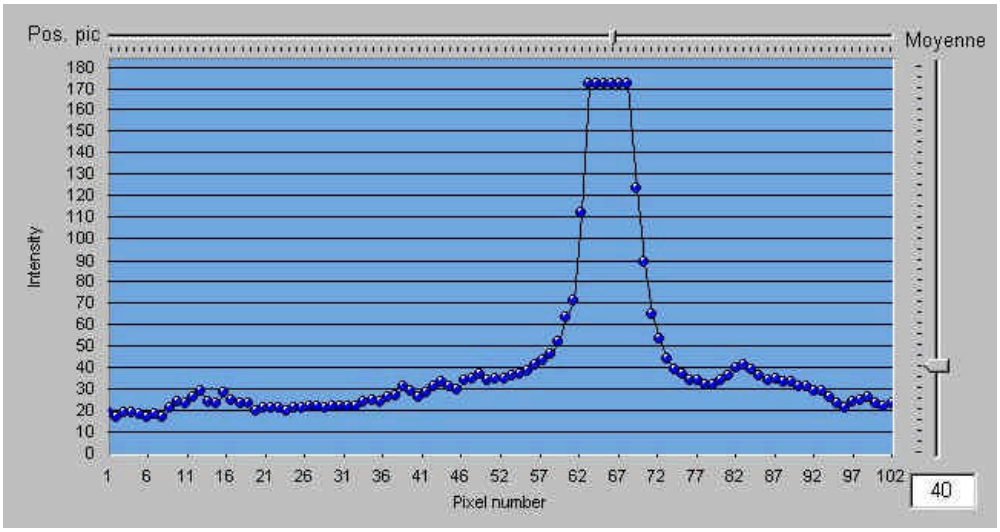

Figure 19 – Image of the linear camera

One of the sensitive parts of this sensor is the lens. André Guignard (LAMI) produces two different types of lenses: cylindrical and spherical (for a comparison between this to types of lenses, see [IV]). Because of the possible height variations of the candle's flame (15 to 20cm), we chose the **cylindrical lens**. The drawback of this version is that all the light that comes from above the maze tends to blind the device. Therefore, we have made a great effort to reduce the integration time.

By calculating the average of the pixel values, Zephyr is able to **determine the distance to the candle**. Below, you can see a graph that represents "average value vs. distance to candle [cm]" during several tries with different ambient light level (spherical lens):

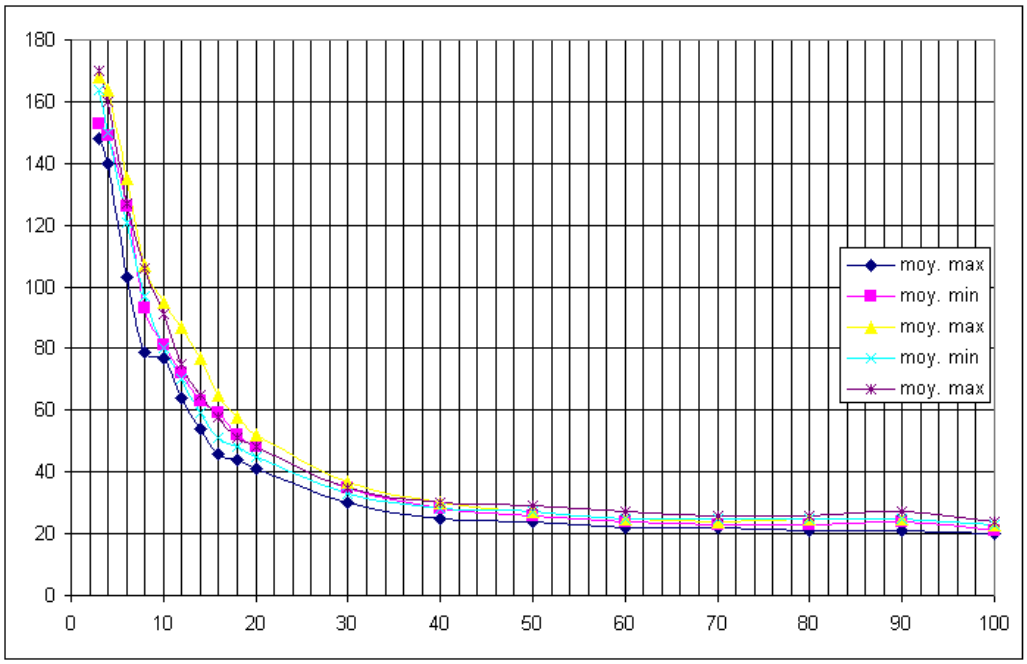

Figure 20 –Pixels average vs. distance to candle [cm]

We can see that if the robot is close to the candle (~10cm), it can measure the distance with a precision of about 3 cm (mean intensity = 100). From 40cm and more, it is practically impossible to calculate a distance.

In conclusion, this device is sufficient to know when the robot must stop to blow out the candle. Yet some problems can appear with the reflection of the candle light on the white walls.

3.3.4 Flame detectors

A good way of sparing time during the competition run is not to enter all the rooms but only the one that contains the candle. Most of the competitors must enter each room and carry out a complete scanning in order to determine the presence of the fire.

To avoid this, we tried to find a sensor capable of **detecting a flame from outside the rooms**. Our search led us to a PbSe photoconductive detector, sensitive in the far infrared, because the candle flame emits a lot in the infrared spectrum.

Figure 21 - PbSe diode

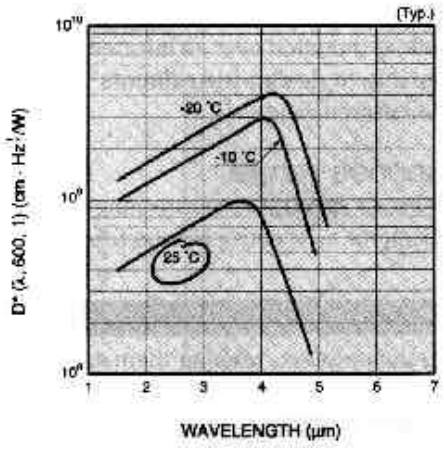

Figure 22 – Sensitive area of PbSe diode

This diode produces a current proportional to the received infrared light intensity. This current has to be converted into a tension and interpreted through several stages of analog electronic devices:

- First stage: current-tension conversion and amplification (adjustable with a potentiometer).
- Second stage: low pass filter in order to eliminate the noise (with an adjustable maximum threshold).
- Third stage: rectifier.

Then the output voltage is simply converted by Kameleon's microcontroller. There is a threshold in order to determine the presence of the candle.

The main problem was to reduce the ambient light effect. Therefore the diodes (one per side) are placed in a sort of sandwich (see Figure 23), which keeps the sensitive area far away of direct floodlighting of ceiling spots and its reflectance on the floor of the maze. This mask tries to let pass the candle rays only.

In order to improve this function, we also put horizontally polarized filters just in front of the diodes. Note that when a light ray is reflected by a smooth surface, it becomes polarized. Thus this filter allows to make a distinction between reflected rays, which come from the candle (horizontal) and those which come from the ceiling lights (vertical).

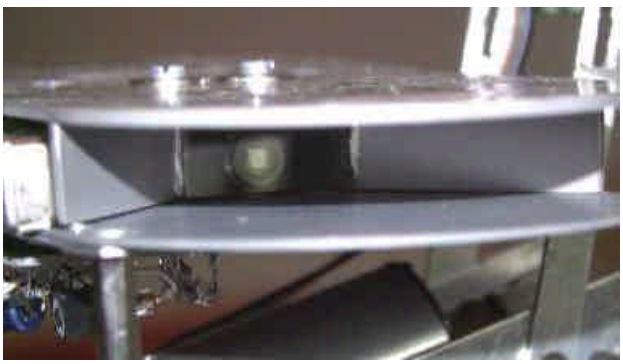

Figure 23 - Flame detector

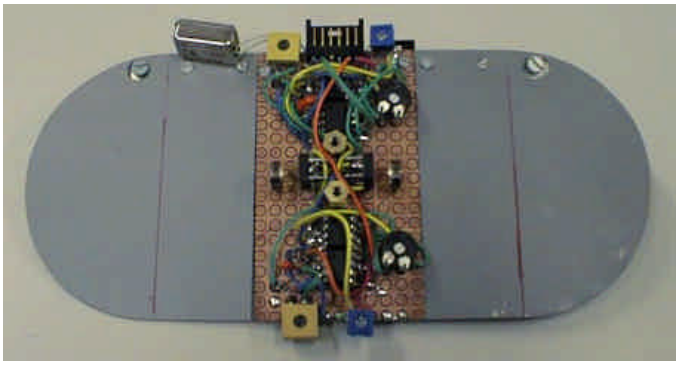

Figure 24 – Internal sight of flame detectors

We have to confess that this device was difficult to adjust. In particular, we had to do some settings in the maze, just before the run, which is a very tricky maneuver. However, it provides a great advantage when it works correctly.

3.3.5 Microphone for sound start

Just below the "sandwich" that contains the flames detectors, we put an electronic board, which is able, with a microphone, to provide an impulse when it receives a 3,5kHz sound:

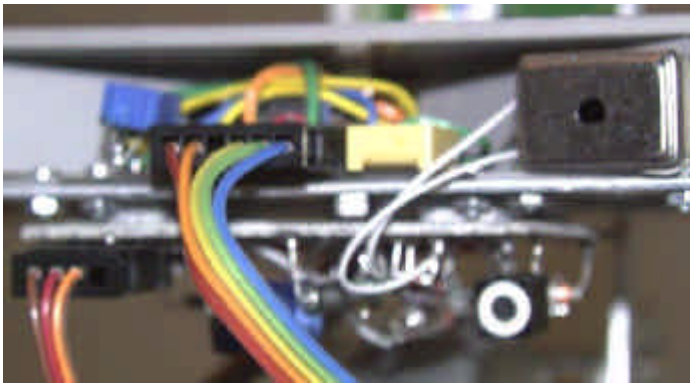

Figure 25 – Microphone for the sound start

Figure 26 – Sound producer

This device allows starting Zephyr with a beep sound without touching the robot. This capability gives us a bonus of 5% on the global score at the contest (see § 2.2).

3.3.6 Infrared sensors*

After the contest, we wanted to implement some obstacle avoidance behaviors. The ultrasonic sensors are not well adapted for this task because of their limited angle. Moreover there is no overlap between the side and front sensors.

As a result **7 infrared proximity sensors** have been mounted **every 30°** from one side to the other. It is very important to maintain an overlap between two nearby sensors.

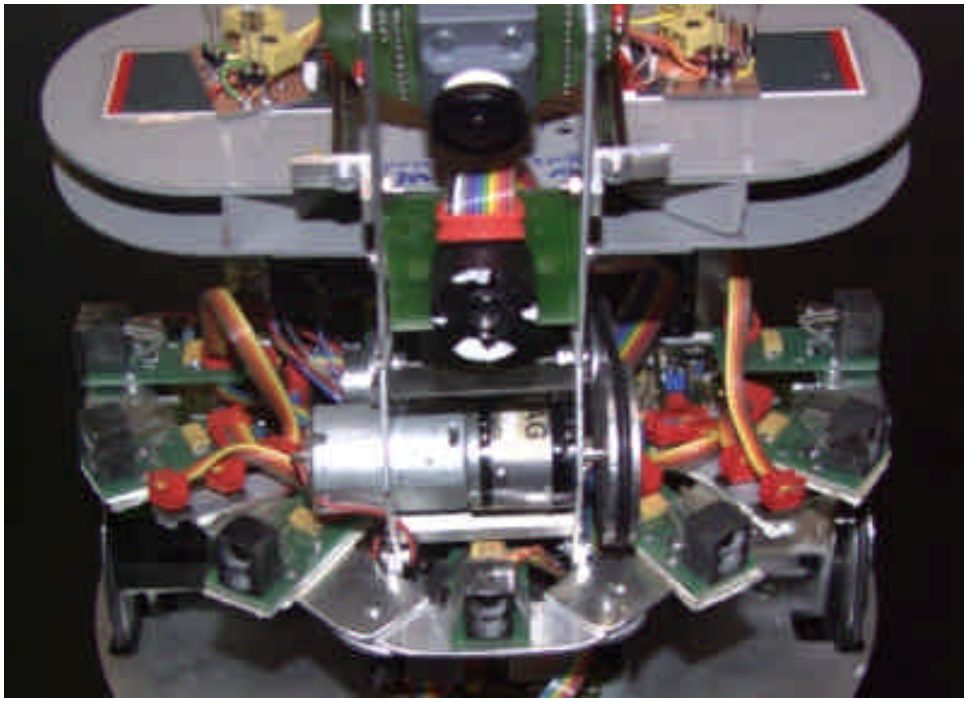

Figure 27 – Infrared sensors and their arrangement on the robot

These devices are simply made of an IR diode and a phototransistor. An impulse is applied to the diode and, after about 300μs of rise time, the phototransistor gives an output tension proportional to the reflected infrared light on the walls.

In order to avoid the ambient light to influence the measure, the phototransistor is measured once before the pulse of the LED and once again after the rise time. Then the difference is taken as an image of the distance to the object. There is **no local processor** for the control of these devices: each sensor is connected to an analog input of the main processor.

We have to notice that the output of this sensor is **highly non-linear and surface dependant** (for example, a white wall reflects more light than a black wall). The **maximal measurable distance is about 25cm** (white smooth surface) but as soon as the object does not have well suited characteristics it is dramatically decreased.

Another drawback of this sensor is the rise time of the phototransistor. If each sensor is read one after another, it takes about 2.3ms (see § 5.4.3 for more information).

*3.3.7 2D camera**

This camera has been added after the contest in order to implement some behavior based modules with the vision as input.

This camera from K-Team (see [VI]) is based on a VV6300 chip (see [VIII]) and a Motorola processor mounted on two Khepera turrets:

- A **CPU-68331** turret with a 16-bit wide 512KB flash memory, an 8-bit wide 128KB static RAM, a serial link (**RS232** of max 57600 bit/s) and a K-Net bus connection (**SPI for the connection with the Kameleon board**, for example). This turret runs a specialized version of KOS (the BIOS from K-Team).
- A camera turret hosts the VV6300, which is a highly integrated CMOS image-sensing device from VISION™. In addition to a **160 x 120 pixel** color image sensor array, this chip includes on-chip circuitry to drive and sense the array. The **frame rate** is variable **between 0.3f/s and 60f/s**.

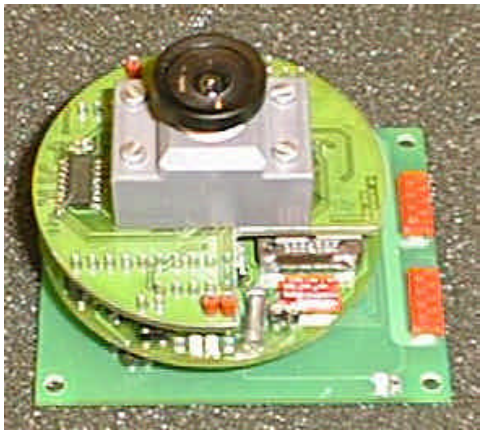

Figure 28 – 2D camera

Figure 29 – Typical image from the camera

The color (RGB) is coded on each four nearby pixels in a so-called Bayer colorization pattern (see Figure 30). The black and white images are obtained by computing the average of each Bayer pattern. Therefore the b & w images are 80 x 60 pixels large.

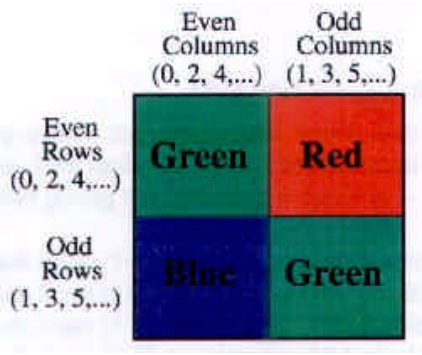

Figure 30 – Bayer pattern

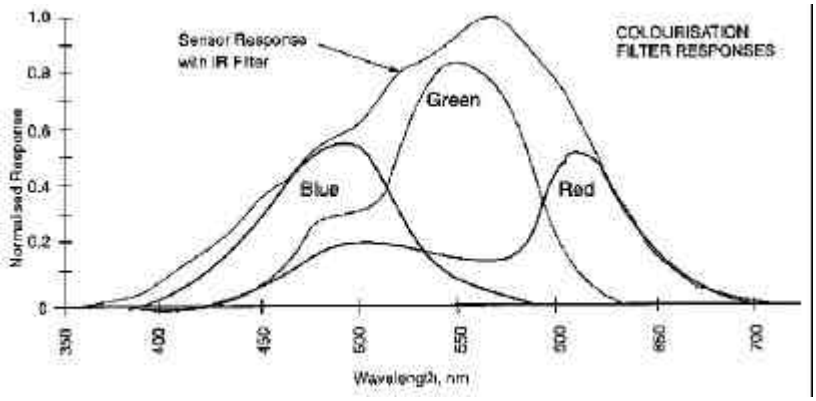

Figure 31 – Spectral response of each pixel

See [VII] for more information about Bayer pattern processing. See § 5.4.4 about the image processing and the reading of the camera.

*3.3.8 Phototransistors**

In order to provide a way to maintain a light gradient (see § 6.4.2) when the robot is very close to the candle, 4 phototransistors have been mounted on the flame detectors platform. The OP550SL from OPTEK have been chosen for their simplicity and their relatively wide opening (about 60°). These NPN transistors are serially connected with a resistance between GND and 5V. Since no more analog channel was available (see appendix 11.1.2), each output (the emitter pin) has been connected to a digital I/O port. Thus these sensing devices have a **Boolean response**. See § 5.4.5 for more details about the reading of these sensors.

The range of detection is about 0 to 10cm. For greater distances, the infrared sensors (see § 3.3.6) are able to provide the light gradient due to the candle.

3.4 Power supply

Right from the start, due to the important variation of the requested motor power, we opted for **separate power supplies**. On one hand, a set of accumulators supplies the electronic part (main board, processor, sensors). On the other hand, another set feeds the inconstant needs of the motors.

For the electronics, we chose an accumulator made of ten 1.2V, 600mAh elements. The consumed power is about 400mA at 12V. This tension source passes through a regulator (on the Kameleon board) in order to provide a well-stabilized 5V for the board. Therefore the tension can drop to 7V before Zephyr's "brain" stops working. With this power supply, the electronics reach **autonomy of about one hour**.

For the motors, two 9V, 110mAh serial accumulators are sufficient since the current peaks are very short. Here the tension variations are not critical and the **18V nominal tension** can sometimes fall to 10V.

4 NAVIGATION

4.1 General considerations

In this chapter we will introduce how Zephyr navigates in the maze and which are the strategies to find the candle. The navigation described here is entirely based on classical approaches.

Basically the navigation is divided into **three consecutive parts**:

- Navigation in corridors
- Navigation in rooms
- Return to starting position (same principle as navigation in corridors)

This same division may be found in the description of the software (see § 5.3).

The navigation in the rooms used in this part of the project is based on odometry and the reading of the linear camera (see § 3.3.3). A completely different approach has been chosen during the second part of the project and will be described later in chapter 6.

4.2 Navigation in corridors with ramps

Since Zephyr should be able to drive over ramps that are placed in the corridors it is not possible to only use the odometry for the navigation. Even if there were no ramps and at very low velocities Zephyr did not manage to finish an entire course based on odometry. The maximal velocity achieved when moving in the corridors is about 60 cm/s, which makes it definitely impossible to only use odometry for navigation. Hence, a new powerful navigation concept had to be found.

Therefore we implemented a tool to follow a wall based on distance measures with a side ultrasonic sensor (see § 3.3.2). This tool, which will be explained in § 5.5, makes it possible to pass over a ramp without losing the right reference since the controller will keep the robot at a constant distance to the wall regardless of the traveled distance of each wheel.

However all turns and several short forward movements are still based on odometry only. But any error due to such an odometry-based movement will automatically be corrected when the robot begins the next follow-wall-phase.

Thus, the follow wall controller allows to move along corridors regardless of the floor's surface shape, and to compensate any error generated by imprecise odometry before hand.

All this leads us to **three basic movements** from which the whole navigation in the corridors is constructed:

- Follow a wall (ultrasonic side sensor)
- Turn (odometry)
- Move forward (odometry)

Note that the robot decides to stop following a wall by either reading the front ultrasonic sensor (detecting an end of the hallway in this direction) or by detecting the end of the wall when the side value jumps to a much higher one.

If one takes a look at the plan of the maze, it becomes obvious that there are basically **two directions to explore the rooms and search the candle**: one clockwise, the other anticlockwise. Both variants have been implemented. However it is the clockwise variant that is advantageous because rooms 1 and 2 are very close and therefore two rooms can be explored in a very short time right at the beginning, which is not the case for the anticlockwise variant.

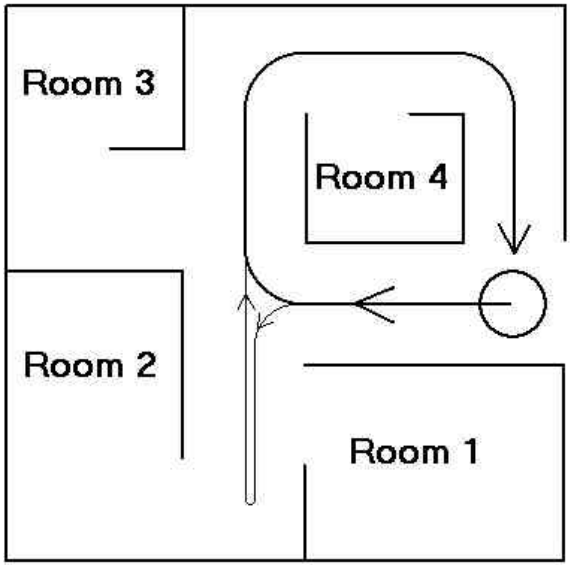

Figure 32 - Clockwise course variant

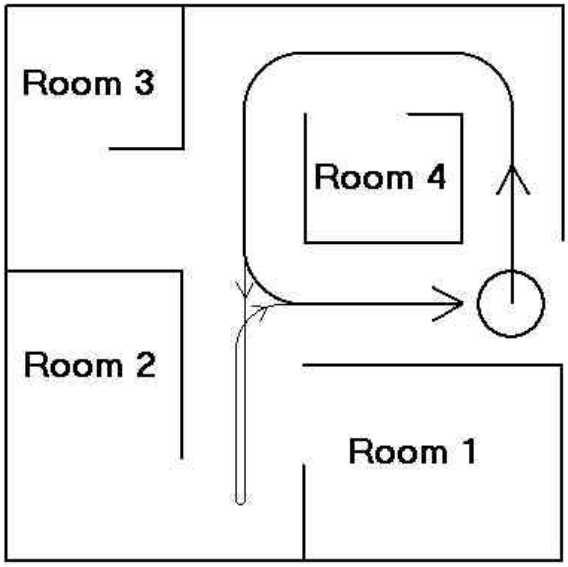

Figure 33 - Anticlockwise course variant

Zephyr has got flame detectors (see § 3.3.4) allowing detecting a candle without entering a room but from the corridors. If those sensors do not work correctly for any reason, it may be possible that the robot does not find a candle after an entire course. It is at that moment that the anticlockwise variant will be used. Zephyr will then explore again each room but by entering them by default without making use of the flame detectors. Note that this is only a security mode. If it should ever be necessary to make use of it, the score will be too bad anyway.

In the following we will describe in detail the procedure for the clockwise course since this is the most used variant.

4.2.1 Clockwise course variant

Zephyr basically has got **two variables to define at every moment its current (strategical) position in the maze**. It will always be moving from one room to the next one. The room series is 1-2-3-4 for the clockwise course and Zephyr will always follow exactly this order. It is the **variable RoomMode** that defines, which room has to be achieved next.

To move from one room to the next a series of movements is defined consisting basically of the movements described above. This series of movements is always the same between the two same rooms as well. The **variable MovementCounter** defines which movement is currently being executed.

The following detailed description of one entire course is therefore divided into the 4 trajectories. The numbers are also reported in the graphic of the course and represent the beginning of the corresponding movement.

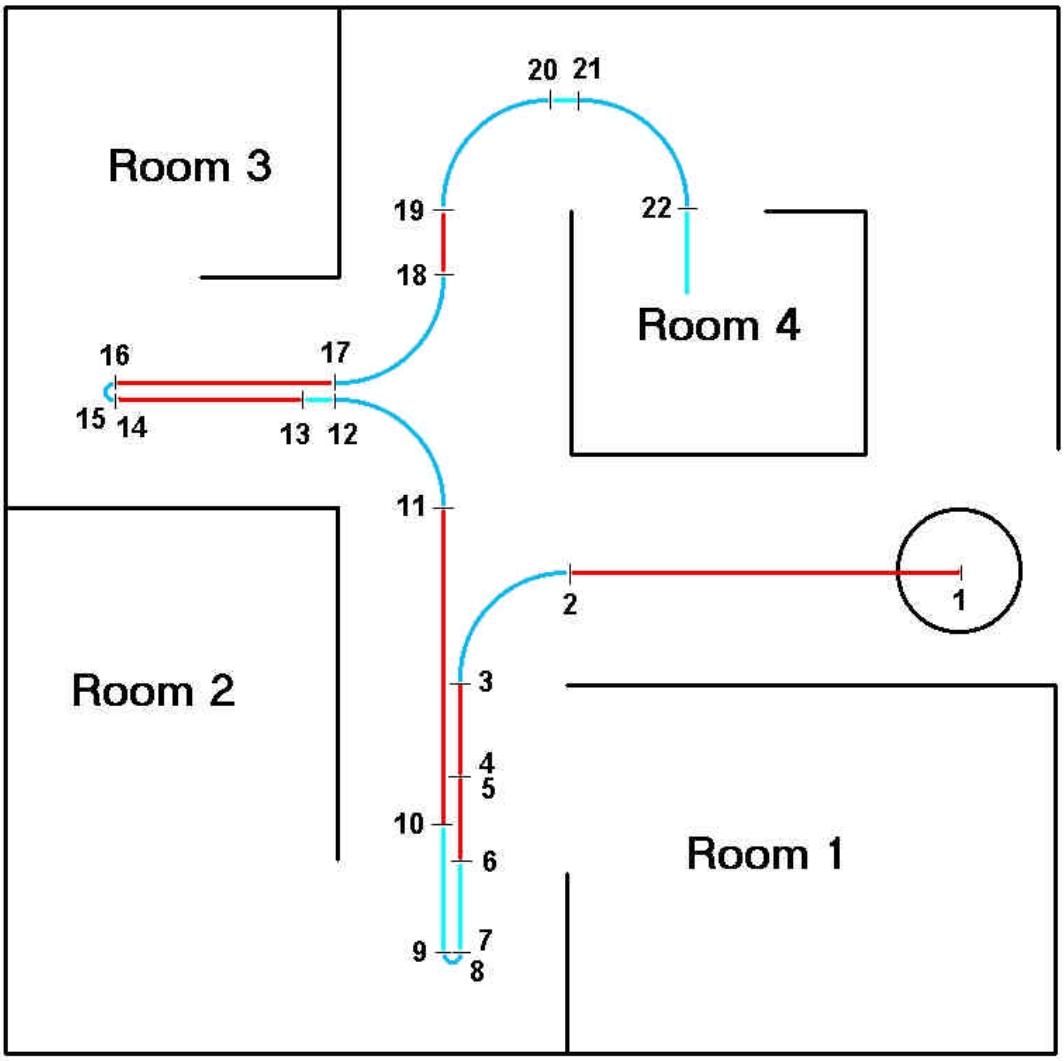

Figure 34 - Movements of clockwise course variant

Move from start to room 1

1. Follow left wall
2. Turn 90°
3. Follow right wall
4. Check left flame detector

Move from room 1 to room 2

5. Follow right wall
6. Move forward for 20 cm
7. Check right flame detector

Move from room 2 to room 3

8. Turn on yourself 180°
9. Move forward for 25 cm

10. Follow left wall
11. Turn 90°
12. Move forward for 4 cm (in order to be certain to detect a wall)
13. Follow left wall
14. Check right flame detector

Move from room 3 to room 4

15. Turn on yourself 180°
16. Follow right wall
17. Turn 90°
18. Follow right wall
19. Turn –90°
20. Move forward for 5 cm
21. Turn –90°
22. Move forward for 20 cm

Comments

When arriving at room 4 the robot enters this room by default. Since it has not found a candle so far, it can only be in the fourth room unless a malfunction of the flame detectors.

The odometry-based movements are terminated when the wheels have traveled the requested distance; the wall following is normally terminated on reading of the front ultrasonic distance sensor.

The blue lines in the graphic represent odometry-based movements (forward and turn), whereas the red lines stands for wall following.

Note that the bends were implemented slightly differently for the contest in Hartford in order to pass over all the ramps.

4.2.2 Entering a room

The described trajectory supposes that the candle will be in room 4 and in none of the other rooms. That is why after each check of the flame detector the robot decides to move to the next room. If a candle is detected in one of the rooms before, the room entering procedure is executed, which goes as this:

1. Turn on yourself 90°/–90° (depending on room)
2. Move forward for 45 cm

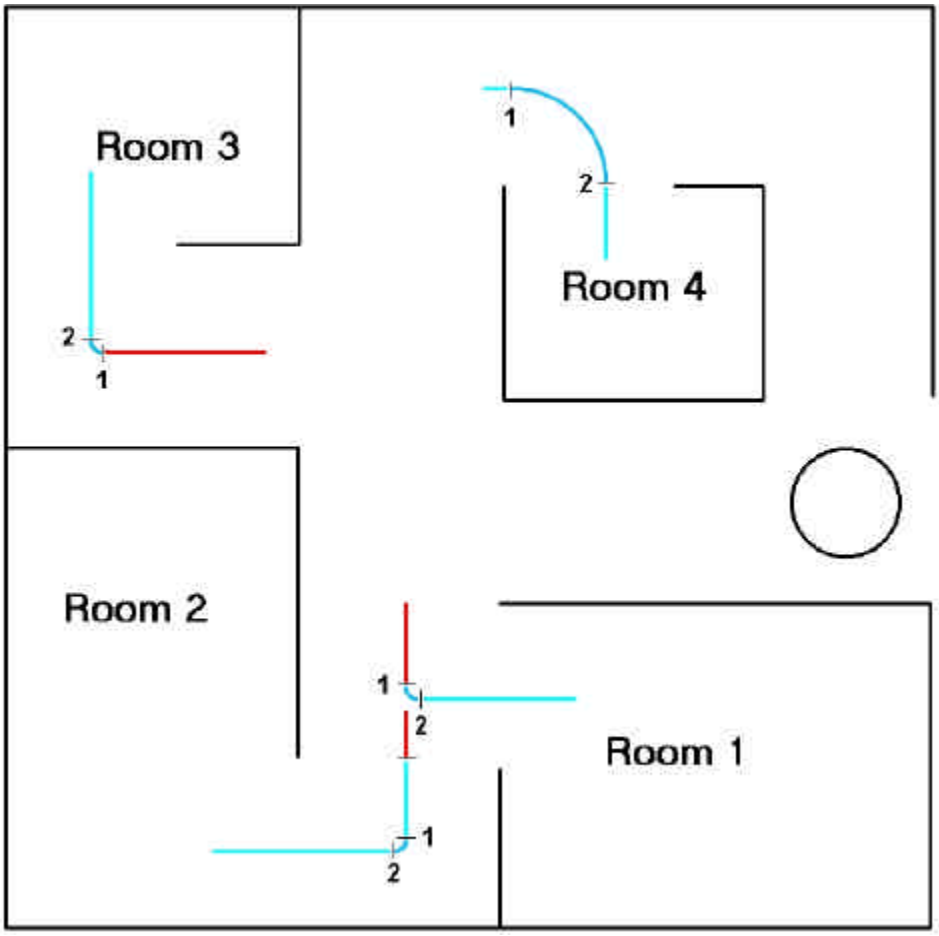

Figure 35 - Entering procedures for all rooms

The distance to move forward is corrected by using the last distance measure to the wall of the ultrasonic side sensor in the follow wall controller.

4.3 Navigation in rooms

4.3.1 Room entrance

At the room entrance the contest rules define that there must not be a candle in a certain zone. This zone is a rectangle that is 33 cm deep into the room. In this area can be no obstacles as well (see § 2.2). The robot will freely move into this zone before continuing with the extinction procedure.

Once the robot has entered a room, it has to find and extinguish the candle. The linear camera is used to «see » the candle. Right at the beginning an image is taken by the camera. If the candle is in the vision field the robot moves directly towards it, if not it starts to turn on itself to scan the entire room for the candle.

The navigation in the rooms is entirely based on odometry. Particularly the procedure to return to the door after extinction is purely based on odometry. Therefore odometry must be switched on during the entire presence in a room, which was not necessary for the navigation in the corridors during the wall following.

If the robot « sees » that the candle is placed very close to a wall, it cannot move straight towards the candle in order to avoid bumping into a wall. It will therefore aim at a point

slightly beside the candle in the first place. Only when it gets very close to the candle it can move straight towards it.

4.3.2 Candle extinction

When the candle is found with the linear camera, it is first centered in the image of the camera. This avoids the candle to leave the vision field just when the robot starts to move forward.

The robot moves towards the candle and gets slower as it approaches the candle. When the robot is close enough to be able to extinguish the candle (maximum 30 cm) the candle has to be centered in front of the robot once again in order to direct the extinction system to the flame.

The extinction motor is then switched on and the candle is extinguished within a split second. The robot verifies the extinction in case of failure.

4.3.3 Return to room door

Now the robot has to return to the room door from where it extinguished the candle. This procedure is purely based on odometry and can be divided into 2 phases.

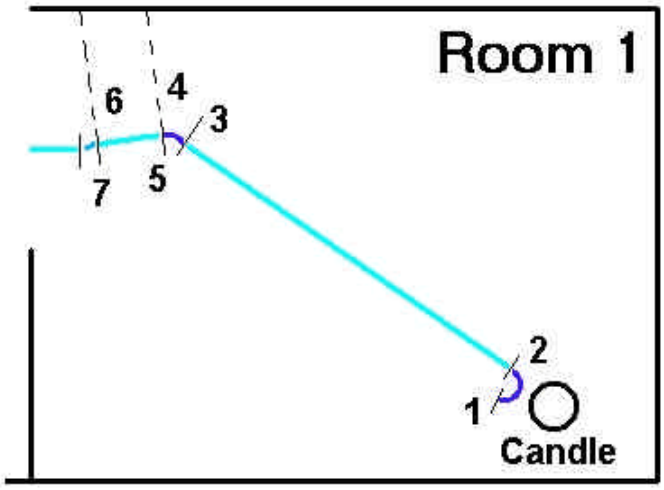

Figure 36 - Return to door procedure

Return to entrance zone

1. Turn to absolute angle (in order to direct the robot to the entrance zone)
2. Move forward (to the entrance zone)
3. Turn to absolute angle (to align to the wall in order to be able to leave the room straight forward)

Correct angle

Since odometry is not very precise, a last correction has to be made.

4. Measure distance to side wall
5. Move forward for 10 cm
6. Measure distance to side wall and compute angle
7. Turn on yourself (computed angle in order to correct alignment)

4.4 Return to starting position

The return to the starting position works exactly the same way as described in “Navigation in corridors” (see § 4.2). Like for the trajectories between two consecutive rooms, a series of movements is defined from each room to the starting point.

5 THE SOFTWARE

5.1 Programming environment

5.1.1 KTProject

The Motorola MC68376 microprocessor can be programmed in C and assembler. For Zephyr we chose to program it entirely in C. This also allowed us to use the preprogrammed BIOS functions.

As development environment we used KTProject by K-Team. This tool uses Cygwin Source Navigator. The C code is downloaded from the PC to the processor via RS232. For development the program is put into the RAM. After development the program can also be loaded into flash memory to be reused at any time.

5.1.2 SupervisorII

The concept of "Supervisor" has already been developed in 1999 and improved until now (see [V] for further information and a downloadable version). The purpose of this tool is to allow the espionage of the robot in a test mode (or during its runs). It is very useful to **test and adjust the sensors** before a run.

Developed with Borland C++ Builder 4, the latest version allows seeing the 102 pixels of the linear camera with its interesting values (peak position, peak value, average of the pixels) and the sensor values (7 infrared sensors, 2 flame detectors, 3 ultrasonic sensors, etc.).

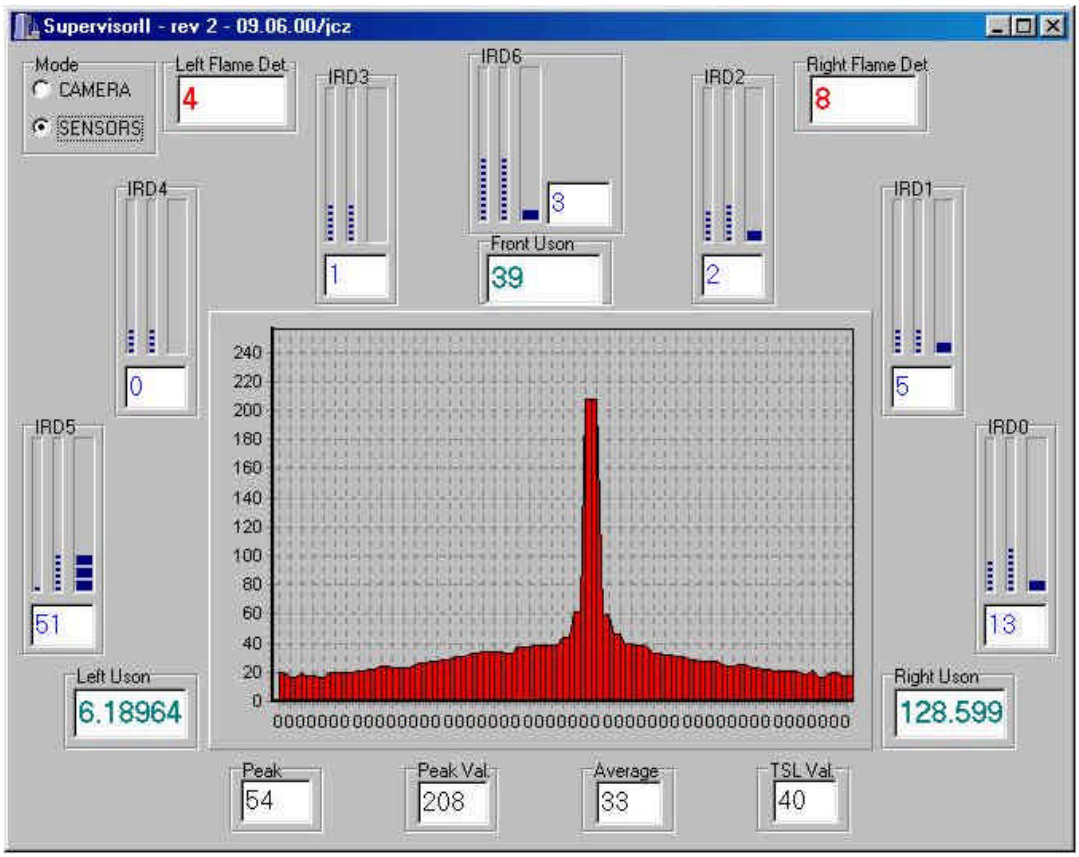

Figure 37 – SupervisorII

The functioning of the system is very simple. It uses an RS232 channel. The serial transfer is initialized to 115200 Bauds. The PC requests the values by sending a specified byte (one for the camera values, another for the sensor values) to the Kameleon. When the Kameleon receives such a byte, it sends back the corresponding values. As you can see, the PC decides for the grabbing frequency but does not send any parameters to the robot.

*5.1.3 Matlab™**

Matlab is a very powerful computation tool for high quantities of data and for visualization. We used Matlab to develop the cylinder extraction from the image acquired with the 2D camera (see § 5.4.4 for details).

5.2 Software architecture

The main program runs on Kameleon's MC68376 processor. This main processor manages all external modules. All decisions are taken on Kameleon.

To assure an optimal functioning, we have opted for the implementation of **two tasks** or processes:

1. **Odometry**: this task executes all odometry computations at regular intervals and updates the current position of the robot in global memory.
2. **Main task**: in this task the whole **strategy** of the robot is implemented. Sensor readings as well as all strategy and navigation decisions are carried out here.

The main process can stop the odometry process when it is not necessary to know the robot's position. This leaves more time for other computations. The robot's strategy was determined such that it is not necessary to know the global position in the maze at any time. Odometry is used only locally at some moments. This has the advantage that odometry errors are not accumulated and that odometry can be stopped when there is no use for it.

The strategy in the main process is divided into **different modes**. Each mode executes a particular and unique movement. This allows reading only the useful sensors in one particular situation and doing the corresponding computations. This way the processor's resources are optimally used. The principle of the different modes also allows keeping a very flexible and clear program structure. Currently more than 15 modes have been implemented.

Each process (odometry and strategy) runs in an infinite loop. In the main process one single mode is executed at every passing in the loop.

The main process is continuously executed. When the odometry process is placed in the execution list, the main process leaves enough time for odometry computations and takes charge again as soon as the robot's situation is determined. Odometry normally is executed in one go without being interrupted by the main process. This is mainly due to the fact that there are two critical sections in the odometry process. Certain modes or movements need to be executed at higher intervals. Therefore the main process can be suspended at any time.

To observe the robot's internal perceptions we use a supervisor on a PC (see § 5.1.2). At the beginning of every execution of the main process, the program controls whether the

PC requests data from the robot. The request is valid if the processor receives any of the possible SCI modes from the PC.

5.3 Modes

5.3.1 Structure

The modes are divided into two main categories:

1. Modes leading the robot to a **requested position**:

   MOVE_CORRIDOR
   TO_DOOR
   GO_HOME

2. **Simple operation** modes. Two types of modes belong to this category:

   a) **Simple movements**

      FORWARD
      TURN
      FOLLOW_WALL
      FORWARD_SEARCH_CANDLE
      ADJUST_ANGLE

   b) Modes to **search and extinguish the candle**

      SEARCH_CANDLE
      CENTER_CANDLE
      TO_CANDLE_FAST
      TO_CANDLE_WALL
      TO_CANDLE_SLOW
      ADJUST_CANDLE
      EXT_CANDLE

The functioning is as follows: the initial mode is always a mode of the first category; that is MOVE_CORRIDOR at the beginning of the exercise. **All modes of the first category use the modes of the second category to reach a certain position.** In those modes a series of movements are defined. These series depend of course on the room the robot wants to reach (MOVE_CORRIDOR) or from which it wants to return to the starting point (GO_HOME). The variable RoomMode specifies the current room. The TO_DOOR mode contains the same series of movements for all rooms. For each movement a mode of the category 2a is called, and after termination of the movement, it returns to the mode of the category 1 by which it has been called.

The modes of the category 2b are always executed in the same order as presented above. They do not need to be called by a mode of the category 1, and therefore they are intercalated between the end of MOVE_CORRIDOR and the beginning of TO_DOOR of the first category.

The majority of the modes have their own **initialization function** by which they are executed.

In spite of the hierarchical structure, all the modes have to be called in the same way and on the same level in the main loop. At every execution of the main process, the current

mode is executed without care for the structure presented above. In fact, this structure is completely virtual for the processor, but very useful for the management and organization of the strategy.

An **additional test mode** is implemented for debugging. The Kameleon decides to execute either this test mode or the contest procedure by testing a button. For instance it is very useful to launch Supervisor during the test mode to observe and adjust the different sensors before starting a course in the maze.

In the following, the different modes are presented more in detail.

5.3.2 Position modes

Navigation in corridors: MOVE_CORRIDOR

This mode manages the navigation in the corridors by using the movement modes. The robot searches the candle in all rooms until it has found it. At that moment the well-defined series of modes to search and extinguish a candle is executed.

Return to room entrance: TO_DOOR

After the extinction of the candle the robot has to return to the room's entrance in order to reorient itself in the corridors. This is done based on odometry: first the robot turns towards the entrance, then it moves straight forward to the entrance and finally aligns itself again with an angle zero relative to the wall.

Because of odometry errors a supplementary realignment is necessary. Moving straight forward for a short distance and taking a side distance measure before and after achieve this. Thus the angle can be computed and corrected.

Return to start position: GO_HOME

Once the room's entrance is found the robot can return to the starting position. The procedure is exactly the same as in the mode MOVE_CORRIDOR.

5.3.3 Simple movement modes

Move straight forward for a certain distance: FORWARD

In this mode the robot moves straight forward for a certain distance based on encoder reading. The velocity and the distance can be specified.

Turn for a certain angle: TURN

With this mode the robot can turn for a certain angle with a certain radius. As for moving forward this movement is based on encoder reading.

Move along a wall: FOLLOW_WALL

This mode executes a wall following controller. It is terminated on a front sensor reading. This distance as well as the distance to the wall, the side and the velocity may be specified (see § 5.5 for more information about this important module).

Move forward and search candle: FORWARD_SEARCH_CANDLE

This mode allows avoiding stopping at a room's entrance to search the candle if it has already been detected when entering the room. In this case the robot can directly pass to the moving-fast-towards-candle-mode without any other maneuvers.

Turn to an absolute angle: ADJUST_ANGLE

In this mode an angle controller has been implemented. The angle is given to the absolute reference. This mode is solely used to return to the room's entrance after extinction of the candle. At a room's entrance the coordinates are stored and the robot moves in the room with odometry functioning.

5.3.4 Candle modes

Search a candle: SEARCH_CANDLE

When searching a candle the robot simply turns around while reading the linear camera. As soon as the difference between the peak value and the average of all pixels (*TSLScan1301 function in § 5.4.1)* exceeds a certain value, the robot continues with the next mode. The fact to take a relative value (difference between peak and average) for this decision makes the robot more independent from ambient light conditions.

Center candle in front of the robot: CENTER_CANDLE

Before moving towards the candle, the robot has to orient itself straight before the candle in order not to lose it just at the start of moving fast forward. The candle is going to be detected at the limit of the field of vision of the camera. If the robot moved forward at that moment, the candle would leave the field of vision immediately.

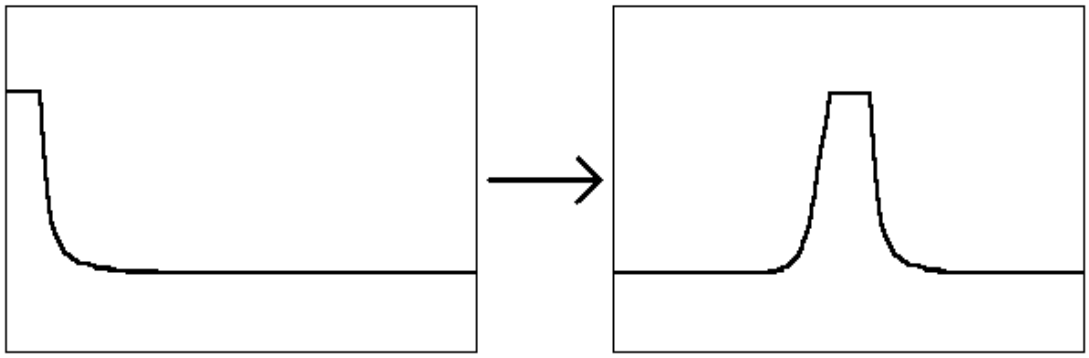

Figure 38 - Center candle in front of robot, grayscale image of linear camera

Move fast towards candle: TO_CANDLE_FAST

The robot moves fast towards the candle while correcting the direction relative to where the peak is located on the linear camera image. As soon as the average exceeds a defined threshold the robot begins to move slowly.

Move towards candle close to wall: TO_CANDLE_WALL

To extinguish a candle, which is placed close to a wall, the robot must not move straight forward. To avoid crashing into walls, a bended approaching curve is implemented in this mode.

Move slowly towards candle: TO_CANDLE_SLOW

The robot moves slowly towards the candle while correcting the direction relative to where the peak is located on the camera image. As soon as the average of all pixel values exceeds a defined threshold the robot stops in front of the candle with a precision of about 15 cm. This is absolutely sufficient to guarantee the efficiency of the extinction system.

Center candle in front of the robot: ADJUST_CANDLE

Before starting the extinction, the robot has to place itself straight before the candle because the horizontal dispersion of the extinction system is very tight.

Extinction of candle: EXT_CANDLE

Once stopped in front of the candle, the extinction motor is started. It is later stopped by interruption just when the extinction system is armed again. This allows putting out the candles more efficiently. By observing the pixel average, the program controls whether the candle has been extinguished. If so, the robot may return to its starting position. If not, it will leave the extinction system switched on until the candle is not visible on the linear camera anymore.

5.4 Sensor integration

5.4.1 Reading of linear camera

This function reads the 102 pixels of the linear camera by A/D conversion on one of the converters available on the processor. The function takes about **3ms** for execution. We could diminish this duration by directly getting access to the address of the output port used to control the camera instead of using the BIOS function.

To **adjust the integration time**, 102 clocks are sent without doing the pixel reading. Afterwards comes a controllable delay time before the reading.

This camera reading must deliver the position of the peak (maximum light intensity = candle position), the peak value and the average of all pixels. **The pixel values are not stored in memory, but all computation is done directly when reading each pixel.**

When the robot is close to the candle, some pixels normally saturate. The peak is then situated in the middle of this saturation part. When the candle is close to a wall the **light reflections on the white walls may produce up to 3 saturating peaks including the candle**. All of them have to be evaluated and the right one finally has to be assigned to the candle.

5.4.2 Reading of ultrasonic sensors

An I^2C bus does the ultrasonic sensor reading (see § 3.3.2). This function directly returns the measured distance of the requested sensor. "uso_read" gives the distance in an integer variable with a resolution of 5mm or 1cm depending on the mode (1m or 2m) whereas "uso_read_best_res" returns the distance in a variable of type double with a precision of about 1mm.

5.4.3 Reading of infrared sensors*

As described in § 3.3.6, the infrared sensors are connected in a star configuration. Thus the main processor has to compute two analog-to-digital conversions for each phototransistor, one before switching on the LED and another after. As the rise time is about 300μs, it takes 2.3ms for the reading of the 7 sensors, including the 14 A/D conversions.

In order to reduce this time, 4 sensors (90° right, 30° right, 30° left, 90° left) are read simultaneously. After that the 3 remaining sensors (60° right, front, 60° left) are read. In this configuration, there is no overlap between the first and the second group of LEDs. Hence the acquisition time has been reduced to **0.9ms**.

This function (distance values) is used by the "obstacle avoidance module" (OAM). It is also used (ambient light measurement) as the sensorial input for the light gradient module (LGM, see § 6.4.2).

5.4.4 Reading of 2D camera*

The VV6300 camera is used to **detect the cylinders in the rooms** from "far" away, since the infrared sensors can only detect it very late.

The camera has its **own MC68331 processor** to acquire the image and to do some processing on the image data. Communication between Kameleon and the camera is established with some flags or 8-bit variables on the camera. The Kameleon can access those flags via the K-Net bus (SPI).

The image is 160 x 120 pixels where always two green, one red and one blue pixel contain the color information (see § 3.3.7). Since we only need a black-white image to detect the cylinder, one grayscale pixel is obtained by taking the average of the four-color pixels. This gives us an **80 x 60 pixel grayscale image**.

We implemented a program on the camera's processor to **extract the position and the width** of the cylinder in the current image (see Figure 39). This has to be done in a very simple way in order to keep the computation duration as low as possible. Therefore **only some of the 60 lines are used and the cylinder extraction is accomplished in one single passing.**

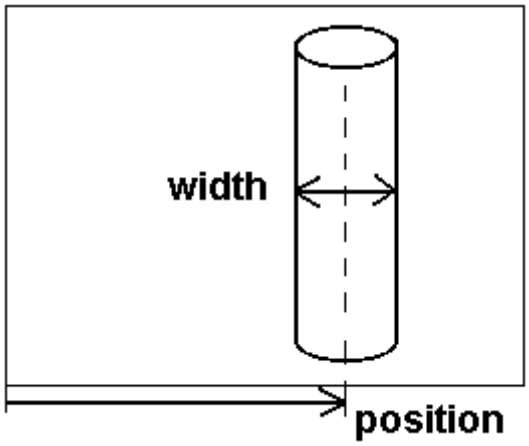

Figure 39 - Cylinder Extraction

You can see two typical images of the cylinder taken with the camera in the following figures. These are images where cylinder extraction is particularly easy to handle.

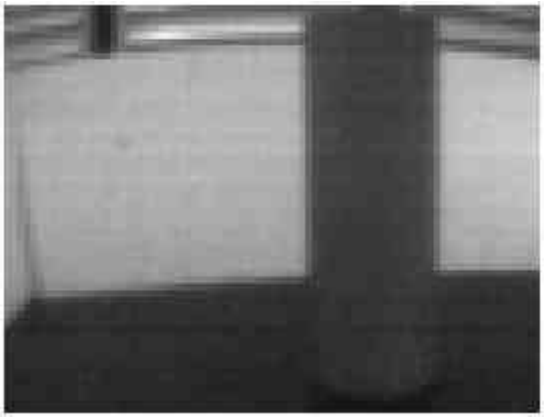

Figure 40 - Cylinder in a room

Figure 41 - A candle behind the cylinder

The cylinder extraction is based on one single line consisting of 80 pixels in 8-bit grayscale (256 levels). In the following two figures you can see line number 20 (out of 60 lines) of above camera images (Figure 40 & Figure 41).

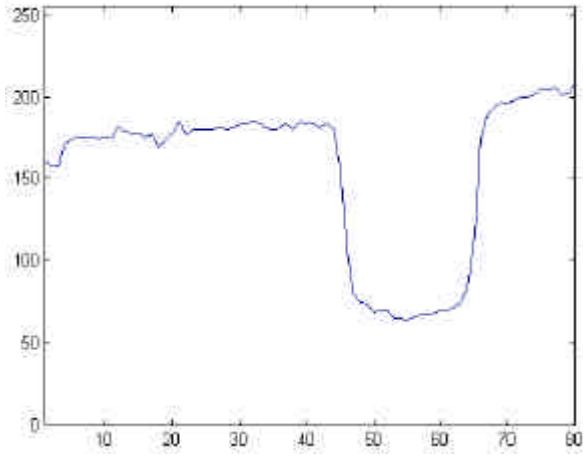

Figure 42 – Cylinder (single line)

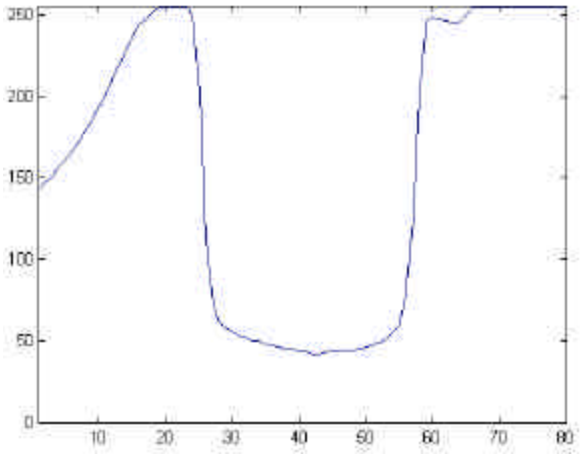

Figure 43 – Candle and cylinder (single line)

Detecting falling and rising edges, where a cylinder is characterized by a falling edge followed by a rising edge, does the cylinder extraction. For a candle it would be exactly the contrary, that is a rising edge followed by a falling edge. Since a candle may also

appear in the images at the same time as the cylinder, this has to be taken into account. The next figures show such a case.

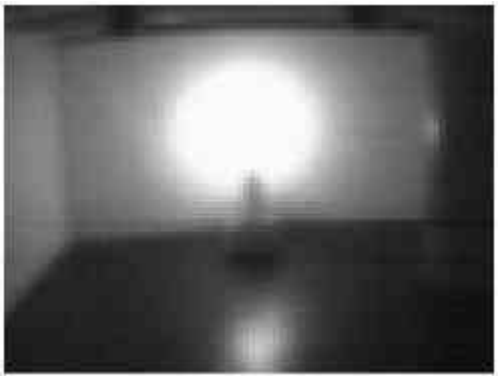

Figure 44 - Cylinder and candle

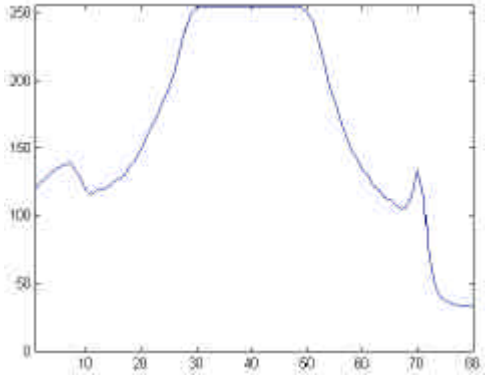

Figure 45 - Cylinder and candle (single line)

It becomes clear that **edges of a cylinder are always lower** (in gray-scale terms) than edges of a candle. In this context one also has to consider situations where the cylinder is at the image border (as it also is the case in above figures), which takes away one of the two cylinder edges. There are in fact **several special** cases with only half the candle or half the cylinder in the image, and not to forget that the candle might also be alone, in which case none of the two candle edges should be considered as a cylinder edge. This whole processing is always done in one single passing in order to keep computation short.

Edges need to have a certain minimal ascent to be detected. But not only the ascent is important, because then very short rises would be detected as well. This is why a minimal edge height has been defined together with an edge width (which is neither minimal nor maximal) as you can see in the figure below.

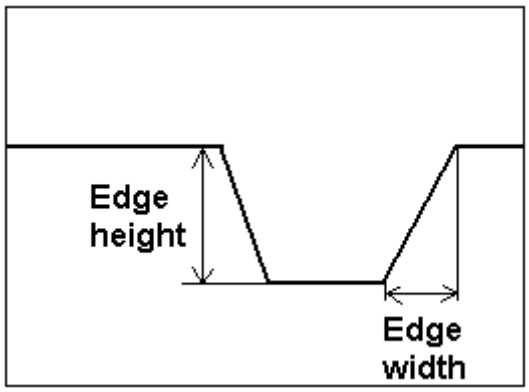

Figure 46 - Edge detection

The reading of this camera is not a very easy task at all. We encountered the following problems:

- The **average acquisition** time of the camera is around 200 ms, which is already quite much.
- This acquisition time is further subject to **high variations**, which can be up to 2s.
- Kameleon cannot access the flags on the camera during acquisition. But the camera is occupied with acquisition most of the time. Therefore **the camera has to be synchronized with Kameleon** in order to allow Kameleon to read the flags when the camera is not acquiring an image.

- From time to time **problems occur in the transmission of the image from the sensing chip to the camera processor.** The image is then shifted left with a different shift for the different image lines. This makes appear some noise at the right image border. This noise can be detected in some cases and the image can be shifted back to the correct position.
- The cylinder detection in a very short time with noisy images is not always possible.

The following figures show such a shifted image with two different noise levels appearing on the right.

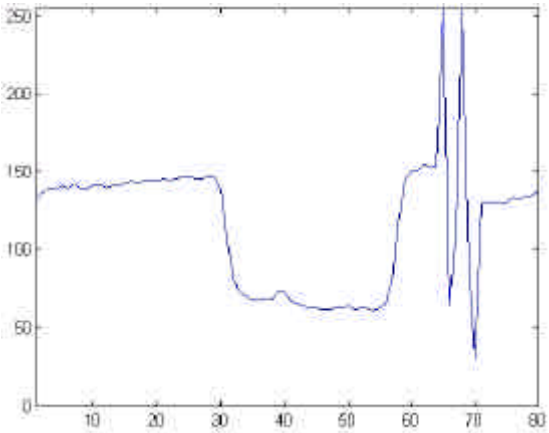

Figure 47 - Shifted image and noise

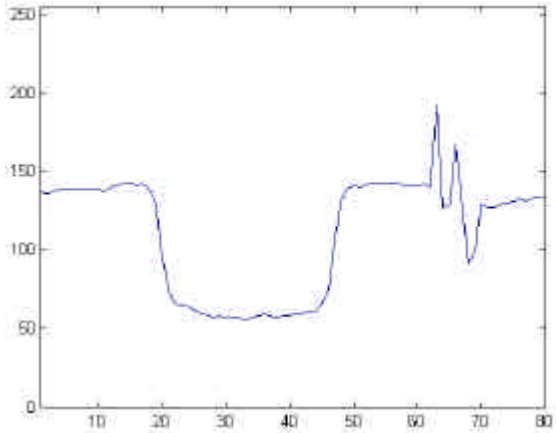

Figure 48 - Shifted image and noise

The noise on the left image is very easy to detect, whereas the noise on the right is already harder to detect if one wants to save computation power. Note that the transmission problem might be resolved by putting a bigger FIFO register between the chip and the processor.

Shadows on the walls due to other walls also produce edges like in the image below. You can see the cylinder at the very right border of this image. In these conditions the shadow edges will be considered as cylinder edges. This can be avoided by taking the average of several lines for cylinder extraction, because shadow edges are not vertical. But if we do so, the noise due to shifting cannot be detected anymore since the shift is different for all lines.

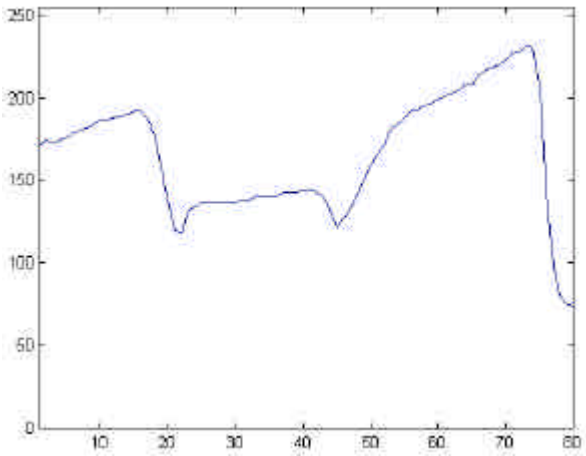

Figure 49 - Wall shadows resembling a cylinder

5.4.5 Reading of the phototransistors*

As described in § 3.3.8, each of the four phototransistors is connected to a digital input of the processor, which works as thresholds. Thus, it is very simple to read these devices. All you have to do is to read the correct digital input port. One global Boolean variable contains the result of this reading for each sensor.

The values provided by this function give **information about the presence of a near candle** and are used by the "light gradient module" (LGM, see § 6.4.2).

5.5 Wall following controller

5.5.1 Necessity of a wall following controller

Since we wanted to perform the contest in the **non-dead reckoning mode** (see § 2.2), we had to implement a robust mode, which would allow Zephyr to determine its position without odometry, in order to drive over ramps and to find its way after imprecise odometry navigation in rooms.

Therefore, we have chosen to implement this wall following controller, **based on the distances given by the ultrasonic sensors**. Once in a corridor, the robot must be able, from any position, to reach its ideal path, which is at **a specified distance and parallel to the wall**. Of course, the initial angle is limited by the side ultrasonic sensors, which have a maximum permissive angle of 35° (see § 3.3.2). Moreover, the initial distance to the wall must also be in a reasonable range compared with the specified path distance.

5.5.2 Implementation

The problem of implementing such a controller is not very simple because of the **non-holonomic** nature of our two-wheeled robot. In fact, we cannot directly enslave the position of the robot to the ideal path because this robot cannot move perpendicular to its wheels. Before reaching the ideal line, it has to turn in its direction.

Therefore we decided to enslave the theta angle (between the wall and the speed vector of the robot) so that it would be proportional to the error between the current distance to the wall (dp) and the requested distance to the wall (D):

$$req\_theta = K \cdot error = K \cdot (dp - D)$$ *where K is a constant*

Then, as we have a current "theta" at each moment and we would like to reach the above-specified "req_theta", we can determine the needed "delta_speed" that will allow the robot to get closer to the requested angle (req_theta):

$$delta\_speed = k \cdot (req\_theta - theta)$$ *where k is another constant*

All the matter, now, is to determine the current theta at each moment and, more difficult, to compute the perpendicular distance to the wall (dp) **when the robot is not exactly parallel to it.**

To compute an approximate angle based on odometry (which provides a delta_theta), then to correct the distance given by the side sensor, solves the problem:

$$dp = d \cdot \cos(theta)$$ *where d is the distance given by the sensor*

Then the approximate current angle based on odometry can be replaced by an angle computed on the basis of the two last perpendicular distances, which is a better approximation of the real theta angle.

Here is a reminder of the used variables (see Figure 50):

```
    n  counter for the controller main loop
    D  requested distance to the wall
    d  current sensor distance to the wall
   dp  perpendicular distance to the wall
theta  current angle relative to the wall
```

req_theta	requested angle based on (dp-D)
delta_theta	angle difference between the last angle and the current angle (given by odometry)
delta_speed	speed difference between left and right wheel

Below, you can see the loop architecture of this controller. For the two first times (n=0 and n=1) we have to initialize some variables. Then (n>1) it is always the same.

- n=0:
  - measure d0
  - delta_speed=0 (straight ahead)
- n=1:
  - measure d1
  - compute theta1 (based on d0-d1)
  - compute dp1 (based on theta1)
  - delta_speed=0 (straight ahead)
- n>1:
  - measure d2
  - compute theta2 (based on delta_theta)
  - compute dp2 (based on theta2)
  - recalculate theta2 (based on dp1-dp2)
  - calculate req_theta (prop. to D-dp2)
  - compute delta_speed (req_theta–theta)

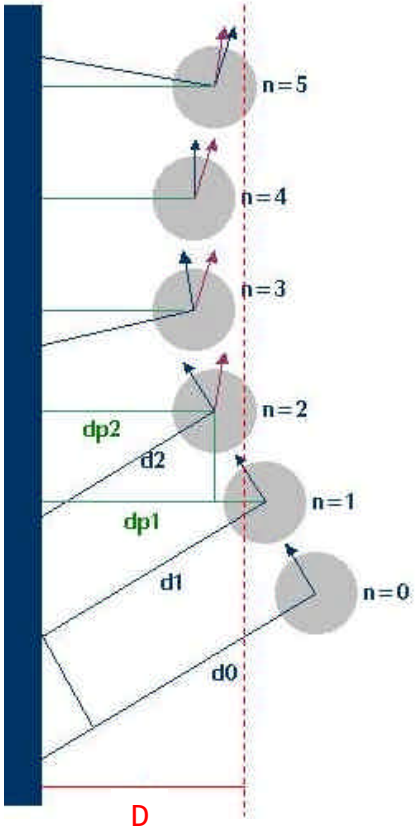

Figure 50 – Wall following

6 BEHAVIOR-BASED NAVIGATION IN ROOMS*

6.1 Why a behavior-based approach?

While the navigation in the corridors can be based on very simple and well-defined features like straight walls, the movement in a room with an obstacle is more difficult to handle with a model-based approach. Let us think a bit about the problem of implementing this path planning with a classical method.

First the robot would have to **locate the cylinder**. There are essentially two ways to handle this problem. The first one consists in taking enough images with a 2D camera in order to cover the entire room. After that, a good and robust image processing merged with odometry information is needed in order to approximate the location of the obstacle. The second way of finding the obstacle is to implement a scanning range finder module, which must be able to build a map of the environment and distinguish between the walls and the cylinder. Both methods are very heavy in terms of processor resources and computation time.

Moreover, when the robot knows the location of the cylinder, it still must **plan its path** by taking care of the candle's position relatively to the obstacle and the walls. Therefore this approach is **too complex** to be valid for a real-time solution, especially if implemented in a 22MHz microcontroller.

This is one of the reasons that let us make a foray into the behavior-based robotics. The second reason is the very short time at our disposal for the development. We also wanted to learn some basic elements of an alternative theory in autonomous robotics and artificial intelligence (AI).

The following paragraph attempts to rapidly summarize the philosophy of the behavior-based approach.

6.2 The behavior-based philosophy

The main part of this paragraph is taken from [2] and [3].

Roodney Brooks developed the subsumption architecture in the mid-1980s at the Massachusetts Institute of Technology (MIT). His approach, **a purely reactive behavior-based method**, flew in the face of traditional AI research at the time. Brooks argued that the *sense-plan-act* paradigm used in some of the first autonomous robots was in fact detrimental to the construction of real working robots. He further argued that **building world models and reasoning using explicit symbolic representational knowledge at best was an impediment to timely robotic response and at worst actually led robotics researchers in the wrong direction**.

The cornerstone of behavior-based robotics is the realization that **the coupling of perception and action gives rise to all the power of intelligence** and that **cognition is only in the eye of an observer.**

6.2.1 Subsumption architecture

Brooks proposed the use of a **layered control system**, embodied by the **subsumption architecture** but layered along a different dimension than what traditional research was

pursuing. Figure 51 shows the distinction between the *sense-plan-act* traditional model (on the left) and the decomposition of the behavior-based or subsumption model (on the right).

The word "subsumption" comes from the verb "to subsume", which means: "to think about an object as taking part of a group". In the context of the behavioral robotics, the name subsumption arises from the coordination process used between the layered behaviors within the architecture. Complex actions subsume simpler behaviors.

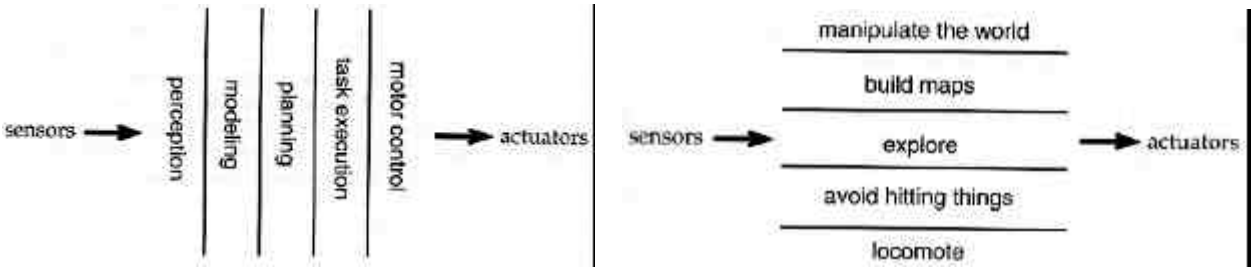

Figure 51 – Traditional and subsumption decomposition (extracted from [3])

Much of the presentation and style of the subsumption approach is dogmatic. Tenets of this viewpoint include:

- Complex behaviors need not necessarily be the product of a complex control system.
- Intelligence is in the eye of the observer.
- The world is its own best model.
- Simplicity is a virtue.
- Robustness in the presence of noisy or failing sensors is a design goal.
- Planning is just a way of avoiding figuring out what to do next.
- Systems should be built incrementally.
- No representation. No calibration. No complex computers. No high-bandwidth communication.

Task achieving behaviors in the **subsumption architecture** are represented as separate layers. Individual layers work on individual goals concurrently and asynchronously. At the lowest level, each behavior is represented using a behavioral module, which encapsulates a particular behavioral transformation function. Stimulus or response signals can be suppressed or inhibited by other active behaviors. A reset input is also used to return the behavior to its start conditions.

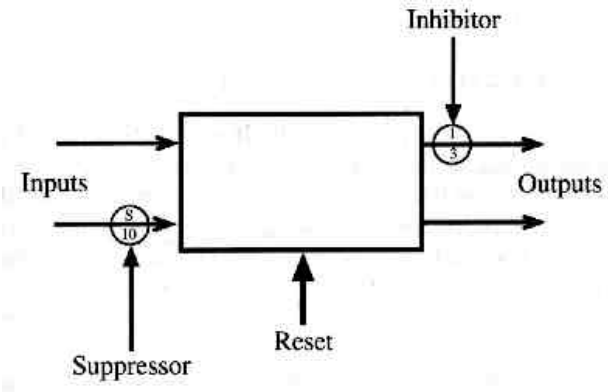

Figure 52 – Behavioral module (extracted from [3])

Each behavioral module performs an action and is responsible for its own perception of the world. There is no global memory, bus or clock. With this design, each behavioral layer can be mapped onto its own processor. There are **no central world models or global sensor representations**. Required sensor inputs are channeled to the consuming behavior.

Figure 53 shows a simple mobile robot with three behavioral layers. The lowest behavior layer, *avoid-objects*, either halts or turns away from an obstacle, depending upon the input from the robot's infrared proximity sensors. The *explore* layer permits the robot to move in the absence of obstacles and cover large areas. The highest layer, *back-out-of-tight-situations*, enables the robot to reverse direction in particularly tight quarters where simpler avoidance and exploration behaviors fail to extricate the robot.

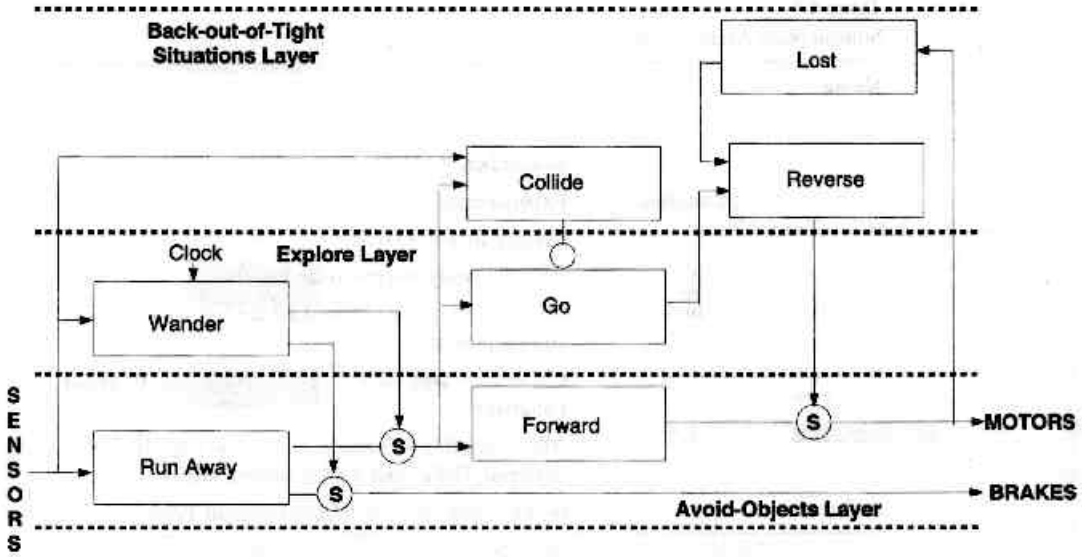

Figure 53 – Simple three-layered robot architecture (Brooks 1987)

This behavior-based system, which was implemented on a radio-controlled toy car (Brooks 1987), is pretty close to what we want to obtain with Zephyr (see § 6.4.1 for a description of the layers and their modules).

A **priority hierarchy** fixes the topology. The lower levels in the architecture have no awareness of higher levels. **This provides the basis for incremental design.** Higher-level competencies are added on top of an already working control system without any modification of those lower levels.

6.2.2 Coordination in sumbsumption

Coordination in subsumption has two primary mechanisms:

- **Inhibition**: used to prevent a signal being transmitted along a behavioral module's wire from reaching the actuators.
- **Suppression**: prevents the current signal from being transmitted and replaces that signal with the suppressing message.

Subsumption permits **communication between layers** but restricts it heavily. The allowable mechanisms have the following characteristics:

- Low baud rate, no handshaking

- Message passing via machine registers
- Output of lower layer accessible for reading by higher level
- Inhibition prevents transmission
- Suppression replaces message with suppressing message
- Reset signal restores behavior to original state

The world itself serves as the primary medium of communication. Actions taken by one behavior result in changes within the world and the robot's relationship to it. New perceptions of those changes communicate those results to the other behaviors.

The key aspects for design of subsumption-style robots are situatedness and embodiment. **Situatedness** refers to the robot's **ability to sense its current surroundings** and **avoid the use of abstract representations**, and **embodiment** insists that the robots be physical creatures and thus **experience the world directly rather than through simulation**.

Now that the bases are placed, we can proceed to the design of our own subsumption architecture.

6.3 Adaptation of the existing robot

In order to become "behavior-based friendly", we had to adapt Zephyr. Actually the sensors of a behavioral system, a bit like those of human beings, do not have to be very precise, linear and failure free. Yet it is better if there is a certain redundancy and a great number of sensors.

Another adaptation of the contest version of Zephyr is the integration of the behavioral modules in the existing code with its modes.

6.3.1 Perception

For obstacle avoidance, the ultrasonic range finders are not well adapted at all because of their tight permissive angle and the fact that there is absolutely no overlap between the front and side sensors. Therefore, the perception of the real environment is now carried out by the **infrared sensors** (see § 3.3.6), which have been added after the contest. The angle of 30° between each of the 7 sensors allows a good overlap. The same sensors are also able to give information about the light gradient, what ultrasonic range finders are not able to, of course.

When the robot is very close to the candle but does not see it through the linear camera, the infrared sensors are unable to provide a sufficient response as well because they are mounted too low on the robot. Therefore we mounted the **phototransistors** (see § 3.3.8) on top of the flame detector platform.

One of the first ideas was to add the vision system in order to drive the robot in the corridors by a neural-based method. Since the **2D camera** (see § 3.3.7) has a too tight vision angle for this task, we decided to use it to implement a cylinder steering module for the furniture mode (see § 2.2).

6.3.2 Integration in the existing code

All the behavioral modules become active at a very particular moment. When the robot has entered a room and has to search and extinguish the candle the behavior-based approach is used.

The entire navigation before and after stays the same and is still model-based, using the many modes described in the software chapter. This led us to create two behavior-based modes, one to find the candle (mode 22 in appendix 11.2.1 on page 37) and one to find the exit (mode 23 in appendix 11.2.1 on page 37). The first is launched once the robot has entered a room, the second when the candle has been extinguished. When the robot is back at the room door, it jumps back to another mode of the original model-based approach. Therefore Zephyr in its current configuration is a hybrid system.

In principle every behavioral module should have its own independent process. This implies an asynchronous system. Nevertheless we have implemented a synchronous system.

It is important to mention that it would have been possible to create an own process for each behavior-based module instead of having them all in the same process managed by a single mode. This possibility has been evaluated and rejected for the following reasons, which all have the time factor in common:

- To manage 6 or 7 tasks at the same time consumes more computation power of the processor than one task would because of the **switching between the tasks**.
- Several modules use the same sensor values as input. These sensor values would have to be stored in a global memory accessible by all modules. Every time the values are updated or read by a module a **critical section** (piece of code that cannot be interrupted by another process) would have to be defined. This takes time to manage. Or the sensors would have to be read more than once, independently by each module, which obviously takes more time. Moreover two modules cannot read the same sensor at the same time.
- The speed outputs of all modules have to be summed up. Therefore they would also have to be stored in global memory, which also implies critical sections for writing and reading.

This shows that multitasking also has its drawbacks, particularly in real-time applications and with limited computational power.

Above all the organization of the behavior-based modules allows a linear execution of all the modules without changing the behavior of the robot at all. Hence the chosen variant is clearly the better one.

6.4 Subsumption architecture for Zephyr

6.4.1 Organization of the behavioral modules

As described in § 6.2.1, task-achieving behaviors in the **subsumption architecture** are represented as separate layers (individual layers work on individual goals concurrently and asynchronously). Thus we defined **the individual goals of the robot in the rooms**:

- During the extinction phase:
  - **Avoid objects** (walls, cylinder)
  - **Find the candle**
  - **Handle tight situations**
- During the exit phase:
  - **Avoid objects** (walls, cylinder)
  - **Find the room entrance**
  - **Handle tight situations**

These three layers have to be implemented one after another in an **incremental design.** In each layer, each behavior is represented using a behavioral module (see § 6.4.2 for a detailed explanation of each module).

Here below, both diagrams for the extinction and the exit phase are showed. Note that their architecture is very close to Figure 53.

Extinction phase:

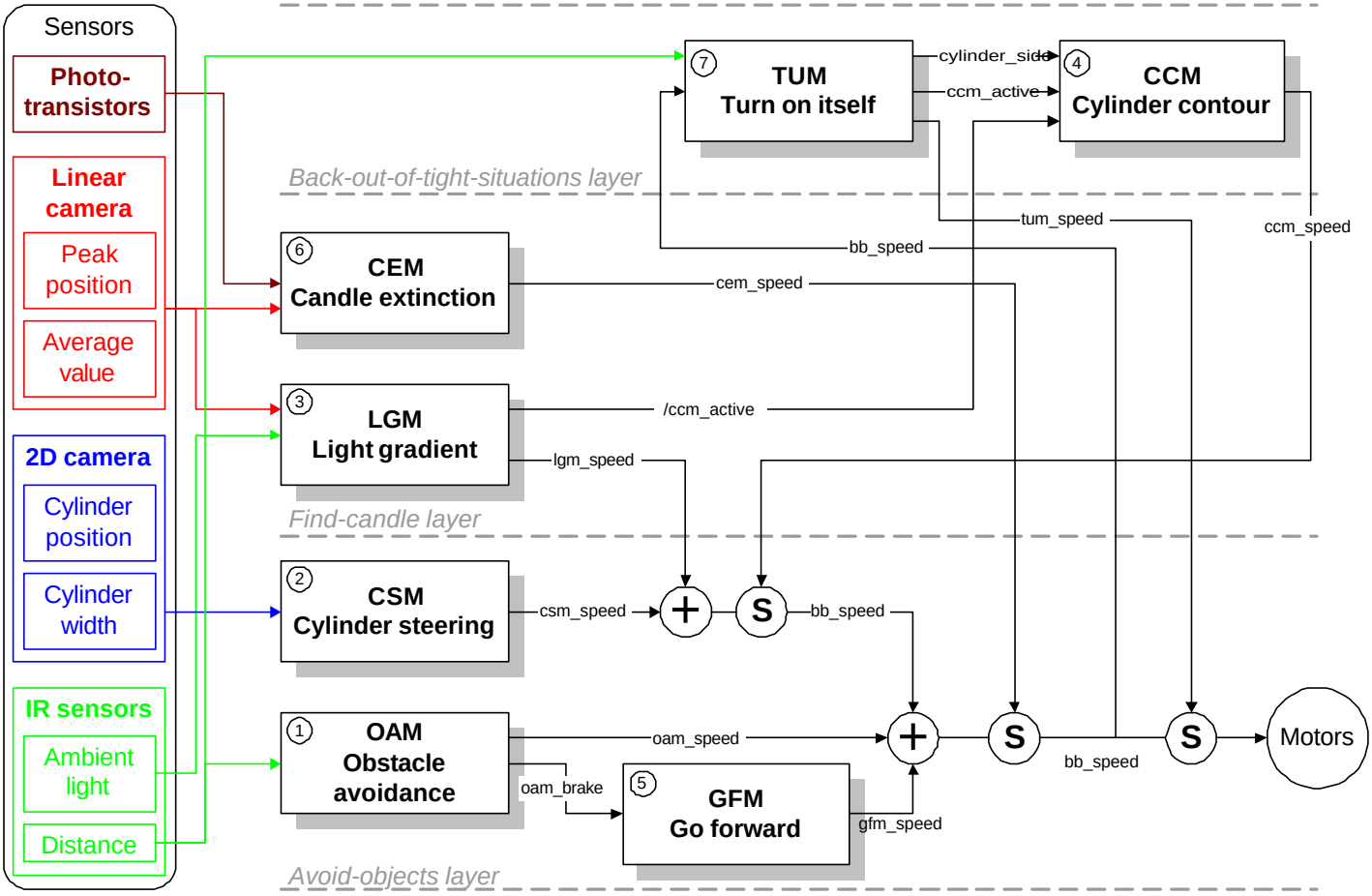

Figure 54 – Organization of the behavioral modules for the extinction phase

Exit phase:

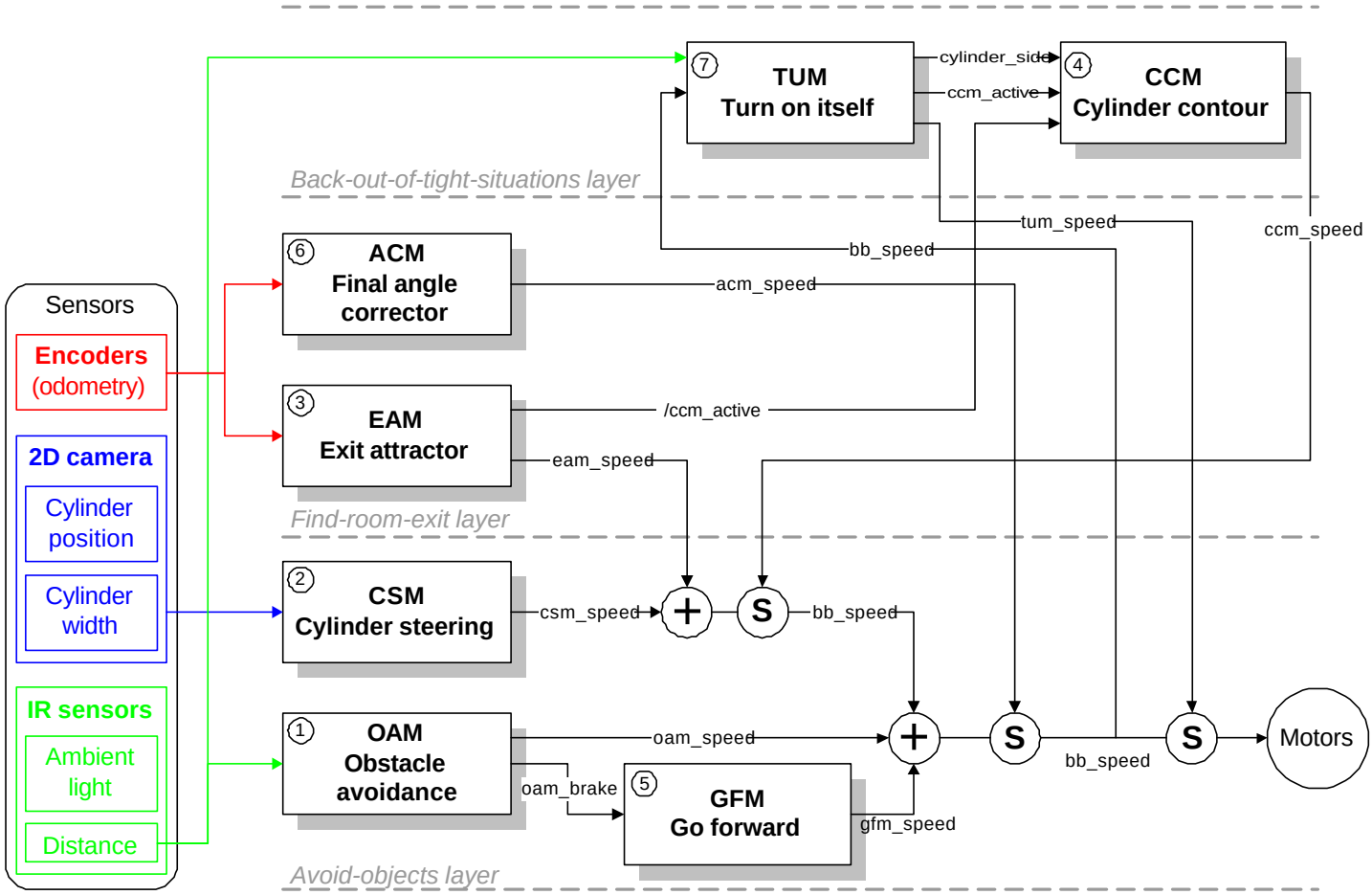

Figure 55 - Organization of the behavioral modules for the exit phase

6.4.2 Implementation of the modules

For the implementation of the modules see page 34 in appendix 11.2.1.

Obstacle avoidance module (OAM)

The obstacle avoidance module is a very classical module. It is totally based on the infrared sensors. The speed values for each wheel are directly connected to the infrared sensor responses through a weight vector.

Cylinder steering module (CSM)

The cylinder steering module tends to push the cylinder out of the 2D-camera image. It always pushes the cylinder to the border to which the cylinder is already closer.

If the whole cylinder is visible, the rotation speed is proportional to the cylinder width, which indicates the distance to the cylinder. In other terms the robot must turn more if it is closer to the cylinder.

If the cylinder is at the border, the rotation speed is simply proportional to the cylinder's position since the width does not give any indication about the distance anymore.

This module is always active except in tight situations and at candle extinction.

Light gradient module (LGM)

The light gradient module's mission is to drive the robot in the direction of the candle. There are two possibilities of achieving that:

- by using the linear camera
- by using the ambient light values provided by the infrared sensors

If the robot does not see the candle directly through the linear camera, it will use the second possibility. Otherwise, the first one is chosen for its higher precision.

This module just has to decide if the candle is found or not. If it is the case, it also will deactivate the cylinder contour module (CCM, see below), which, of course, will only have an effect if it was active before.

It is not its task to stop the robot in front of the candle. Therefore its output can be suppressed by the candle extinction module (CEM), which will achieve this task.

Cylinder contour module (CCM)

The cylinder contour module must turn around the cylinder once the robot has got stuck before. This is achieved with a very simple trick.

This module is actually activated by a message received from the turn module (TUM). The turn module specifies at the same time on which side the cylinder has been detected. Now the cylinder contour module simply sets a rotation speed towards the cylinder to its output. Since the obstacle avoidance module (OAM) tries to avoid the cylinder, this will make the robot turn around it.

Go forward module (GFM)

This module just has to make the robot moving forward. It is responsible for the global speed of the robot in the room. It receives as input an information (oam_brake) from the obstacle avoidance module (OAM), which will tell it if the robot must slow down because of the presence of an obstacle.

Candle extinction module (CEM)

This module is responsible to take the decision of stopping the robot because it is very close to the candle. In order to achieve this goal, it receives as sensor input both values from the **linear camera** and the **phototransistors**.

If the average of the linear camera is sufficiently high, it will stop the behavior-based mode and let the mode ADJUST_CANDLE (see § 5.3.4) doing its job.

If one of the phototransistors is active, the module will suppress all other speed outputs and make the robot turning on itself until the linear camera sees the candle.

Turn on itself module (TUM)

The turn module is able to activate itself. It detects the particular tight situation when the robot gets stuck between the cylinder and a wall. This is the case when the speeds of both wheels are very close to zero and at least one infrared sensor on each side detects an obstacle.

On the wall side more than one sensor is detecting an obstacle, whereas on the cylinder side only one sensor detects the cylinder. This characterizes the side where the cylinder is placed. This information is transmitted to the cylinder contour module by a message.

Once the tight situation detected, this module sets a constant rotation speed to its output. It deactivates itself as soon as the infrared sensor at 60° detects the cylinder again. This sensor also has to be the only one to detect an obstacle; otherwise it could be simply detecting the wall just when the robot begins to turn (see the left image in Figure 56). This is why the sensor at 60° had to be chosen instead of the sensor at 90° because that one could be detecting the wall as the last and only one.

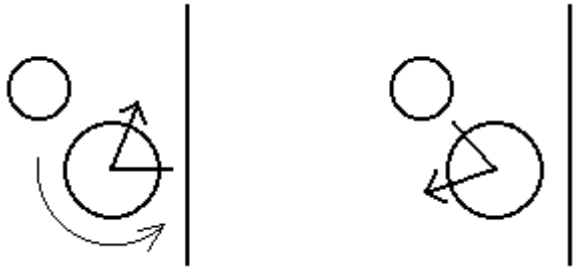

Figure 56 - Turn module

When active, the turn module suppresses the outputs of all other modules.

Angle corrector module (ACM)

The angle corrector module only gives an output rotation speed when the robot has arrived at the door by comparing the door coordinates with the robot's current position. The rotation speed is proportional to the difference between the current angle and the angle necessary to be correctly aligned to the side wall.

Exit attractor (EAM)

The exit attractor module is based on odometry measures and calculations. The door coordinates are known as well as an angle indicating the direction to the door. Knowing the current angle of the robot, the difference to the direction angle is proportional to the rotation speed to put on the motors.

This module can deactivate the cylinder contour module (CCM) as soon as the robot points in the direction of the door.

7 PERFORMANCES

7.1 Positive points

7.1.1 Contest version

Zephyr's good performances are based on **three strong points**:

- its high speed
- its ability to detect candles without entering a room
- its very efficient extinction system

The **high speed** could be achieved with the implementation of the very powerful **wall following controller**. This controller allowed to correct errors due to imprecise odometry and to **pass over ramps** without further problems. Another important point is the good and **robust mechanical construction**, which made it possible to pass over ramps at very high speed.

The extinction system allows Zephyr to put out the candle within a split second, once the candle has been detected.

The software of Zephyr is very flexible and allows adapting Zephyr's behavior to other structures than the used maze. The code is optimized and well adapted to the real-time system that Zephyr is. There are only two processes running on the processor and function calls are avoided if possible.

*7.1.2 Behavior-based version**

A drawback of the model-based approach is its inability to recover from errors, since not any error can be foreseen. The behavior-based navigation on the other hand has this possibility. In addition such a behavior can be obtained with very simple modules, which can be developed very rapidly.

The implementation is very easy, based on the layers proposed by Brooks. The integration in the existing code is very easy, too.

The sensors are generally very simple and fast mounted on the robot. Despite the simple architecture of the system it is very robust. This lies in the heart of the behavior-based approach, which makes it such a fascinating domain.

7.2 Main difficulties

7.2.1 Contest version

Although Zephyr achieved very good performances at **high speeds** and with a **good robustness**, we encountered some difficulties, some of which still persist.

First of all **the flame detectors** are not very reliable because they depend highly on ambient light conditions and they are too sensitive. Nevertheless when they are working correctly, this means a great profit in time.

One of the main issues of the contest were **the ramps**, which could be placed nearly everywhere in the hallways of the maze. In the first place we solved this problem with the

implementation of a wall following controller. But it was not always possible to follow a wall where a ramp could be placed, because either there simply was no wall or the ramp could also be placed in a bend. We then had to choose the appropriate odometry-based movements. The ramps used at the contest in the USA were so smooth in shape and had a surface, which made our front ultrasonic sensor detect them because it was placed too low.

It was very hard to correctly manage **the ultrasonic sensors**. We encountered interferences between the different sensors as well as multiple reflections on the walls finding their way back to the receptor. Both led to distance measures that were to low (they couldn't be too high since then the correct reflection would already have arrived at the receptor before). This was not a particular issue for the wall following controller since the next correct measure would always work out the problem. But to take decisions on the front sensor measures the problem was rather annoying. By installing shields between the sensors and by orienting them correctly the problem could be managed. However those adjustments were sometimes very sensitive. Software filters helped to assure the correct decisions.

The calculation of the **distance to the candle** is subject to high variations. The calculation is simply based on the linear camera average. With reflections on the white walls the average rises, which does not mean that the robot is closer to the candle.

The **odometry** quickly revealed its limitations. It is simply not precise enough to be used alone. Nevertheless there are situations where we were forced to use odometry. Particularly the navigation in the rooms was entirely based on odometry before the implementation of the behavior-based modules. A realignment based on ultrasonic measures became necessary in order to correctly leave a room.

An annoying thing was the **Robotics Extension Board** (REB) breaking down. It actually had to be replaced twice during the project. This was quite alarming, knowing that it is worth around 1000 SFr.

Towards the end of the project we encountered several problems with **memory space**. The processor was sometimes running out of memory and we had to search for the exact nature of this problem.

We used about 6 different sensors on the robot, some of them multiple times. With so many sensors it is obvious that some **time constraints** will become important. Generally it is just the reading of the sensors that takes most of the time. The sensor information has then to be treated and all this in real-time. For some sensors we had to strictly avoid to call any functions in the software, because this takes too much time when used too often.

*7.2.2 Behavior-based version**

The **2D camera** VV6300 is a good example of a sensor taking much time in a real-time application (which actually makes it a rather bad example). Different constraints added up the time for the camera reading and the noise appearing from time to time did not help to make it supply better measure values. Luckily this camera at least has a processor of its own.

Many **additional sensors** had to be mounted on Zephyr to satisfy the needs of the behavior-based modules. The existing sensors were not adapted to them.

Despite the simplicity of the final architecture, the organization of the modules demanded a lot of reflection. Simplicity is not always easy to find.

The method all the same implies some problems. **The weights of the different modules are quite difficult to adjust.** This is mainly based on human perception of the robot's behavior. With so many modules it is very hard to see which module is predominant on the motor control. Sometimes the robot might even get stuck and stop to move due to bad weight adjustment.

Another drop of bitterness is the fact that the robot might leave a room without having found the candle. This happens if it avoids an obstacle such that the light sensors do not receive enough light anymore to be attracted to it. In its current configuration the robot will be lost if it enters a room where there is no candle at all.

8 CONCLUSION

8.1 To conclude and recapitulate

The basic goal of the project was to navigate with a robot in a maze, of which the plan is known before hand, but which may also include some perturbation. We wanted to use two different approaches to execute this task.

In the first place we chose a conventional model based approach. This is generally used in autonomous mobile robotics when the environment is known. A model of the environment can then be worked out, and the robot will navigate in the modeled environment. With this approach a very high speed in the known hallway structure could be achieved.

To navigate in the rooms, where obstacles can be placed anywhere, we chose a behavior-based approach. Some very simple modules such as obstacle avoidance or light attraction, which directly act from the sensors on the motor speeds, were implemented. This gave us a powerful system, which makes it almost impossible to bump into walls and which may always find a way to execute the task differently. But all this is going on at much lower speeds.

8.2 What we have learnt

A mobile autonomous robot requires a complex system made of sensors, actuators, electronics and software. During this project we have learnt to build and manage such a system covering a wide range of fields of knowledge.

We have learnt to apply the theories of managing a non-holonomic robot, to navigate with the robot in a well-known environment and to chose and integrate the needed sensors.

Particularly we have become familiar with the MC68xxx microprocessor, its architecture, ports and multitasking capabilities. It is important to well manage multitasking in real-time applications like a mobile robot. The system has had to be optimized for real-time action. The C program architecture has been conceived for this, and it was not always easy to implement functions for debugging in real-time.

Finally we have seen how a very basic behavior-based navigation approach works, and what are its limitations and advantages.

And not to forget the mental part of the project: we have also been working in a team of five persons. The project management and the time planning represented a big challenge. And managing the stress constraints in contest conditions has been a very useful experience.

8.3 Acknowledgements

We would like to thank Prof. Jean-Daniel Nicoud for his help and assistance during the project.

We are grateful to our assistants Prof. Dario Floreano and Joseba Urzelai for their interesting propositions, their help and their support throughout this project.

Thank to the LAMI staff, especially Marie-José Pellaud for her efficient organization and André Guignard for his availability and technical advices.

We would also like to thank K-Team, which provided the Kameleon board and its Robotics Extension Board, in particular Pierre Arnaud, Olivier Carmona, Christophe Jaquet and Skye Legon.

Thank you to Unai Viscarret for his support with the ultrasonic sensors.

We are indebted to all our sponsors, who made possible the conception, the construction and the correct working of our robot Zephyr.

Patrick Ramer *Jean-Christophe Zufferey*

9 BIBLIOGRAPHY

[1] ROLAND SIEGWART, *Autonomous Mobile Robot*, lecture notes, 1999

[2] RONALD C. ARKIN, *Behavior-Based Robotics*, The MIT Press, 1997

[3] RODNEY A. BROOKS, *Cambrian intelligence*, The MIT Press, 1999

[4] LAURENE FAUSETT, *Fundamentals of neural Networks*, Florida Institute of Technology, Prentice Hall International Inc., 1994

10 RELATED WEB SITES

[I] Official web site of robot Zephyr: dmtwww.epfl.ch/~jzuffere/Zephyr.html

[II] Official web site of the contest in Hartford: www.trincoll.edu/robot

[III] Links on the contest in Lausanne: diwww.epfl.ch/lami

[IV] Lens comparison (for TSL1301): dmtwww.epfl.ch/~jzuffere/CompLentilles.htm

[V] Information about Supervisor: dmtwww.epfl.ch/~jzuffere/Robotique.html

[VI] K-Team, our provider for the Kameleon board: www.k-team.com

[VII] Information about Bayer pattern processing: www.dvcco.com/rgb.htm

[VIII] VV6300: smartvisionproducts.com/Data%20Sheets/VV6300Color.pdf

11 APPENDIX TABLE

11.1 Specifications

11.1.1 Some timings

11.1.2 Pinning

11.1.3 Ultrasonic module (by Unai Viscarret)

11.2 C code

11.2.1 Final program for the Kameleon board

11.2.2 Program for the VV6300 module