D. Findrik (0808798)
J. Hampl (0802690)
L. Kamburov (0838290)
Intelligent Autonomous Robotics

FINAL REPORT

Abstract

We designed, built and programmed a robot capable of autonomous exploration and recognition of resource sites in its environment. Resource sites consisted of dark flooring which we used two light sensors to detect, walls and a switch that we used two whiskers and two IR sensors to detect and ligh t of a specific frequency emitted after triggering the switch that we detected with two light sensors. A Hall sensor was used to avoid getting stuck. We used a purely reactive architecture for the robot. We carried out an experiment to test the efficiency of our robot in completing its task and we found that out of eight runs it found an average of 2.6 sites and performed 1.5 dances correctly. The main restrai nt of our robot is the lack of high level planning that could help optimising its site searching strategy.

Introduction

The goal of this project was to build a robot that was supposed to move around in a lab environment searching for "resource sites". Resource sites were represented as black cardboard parallelograms taped to the floor. Two sides of the resource site had walls 50 centimetres high with a slight gap where they would meet. Here a switch was attached to a light diode that emitted a light pulsing at a specified frequency.

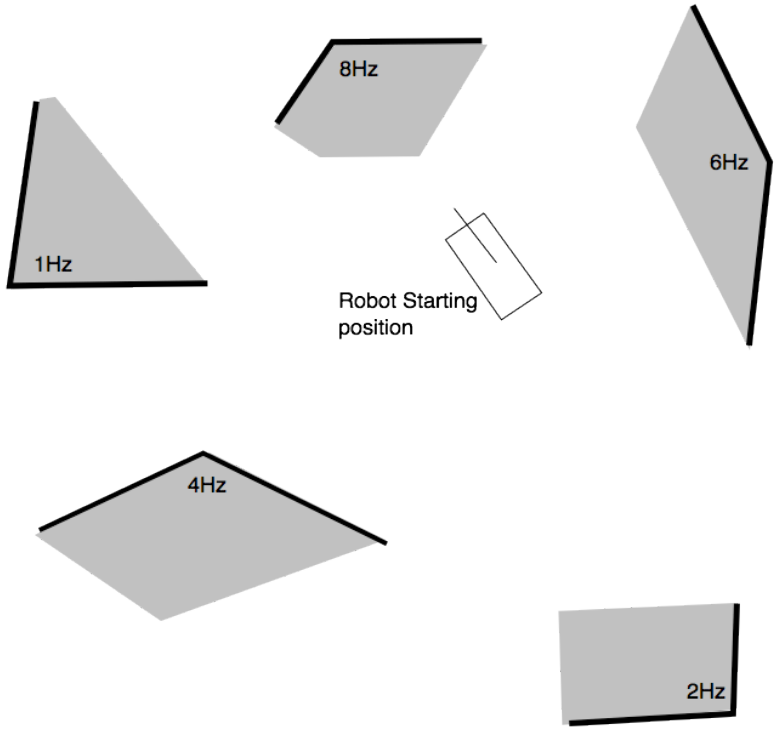

Figure 1: Layout of the arena

The robot had five minutes to find as many as possible of these resource sites. When a resource site was found the robot was expected to trigger the switch and measure the frequency that was emitted by the light switch. Then one of the following victory dances should have been made based on the frequency of the light switch:

Frequency	Movement
0.5 Hz	Backwards, 360° turn left
1 Hz	Backwards, 360° turn right
2 Hz	Backwards, stop, backwards
4 Hz	Backwards, 180° turn left
6 Hz	Backwards, 180° turn right
8 Hz	Backwards, stop, forwards

Table 1: Light frequencies and required movements

Methodology

Hardware specification

The robot has gone through several changes and reached its final state on the last week. It consists of three main distinguishable parts: rear, middle, and front.

The rear part had the two wheels on the sides powered by a motor which is attached via a track. The use of the track makes placing the wheels simpler and thus allows more flexibility. We have also used cross-bracing in the rear to make the structure more robust. To avoid outer interaction with the wheels, we have surrounded them by lego. Also, as the rear is the most robust part of the robot, we have centered most of the weight here by placing the battery and the Fit PC on top of it.

The middle part consist of the power board and the interface board which are part of our structure and integrated into the Lego parts. On the bottom of the middle part, we installed two light sensors and a light bulb. We used cross-bracing in the middle part as well to increase robustness.

The front part holds the servo motor which has a wheel attached to it for steering. It also includes the two whisker sensors, the front-facing and the top IR sensor and the top light sensors. The only weak point of the design is the attachment of the front wheel to the servo, however, it proved to be satisfactory for our purposes during the test runs. For more information about the sensors see below, also see figures 2, 6, 7, 8 for pictures of the structure and Table 2 for basic measurements.

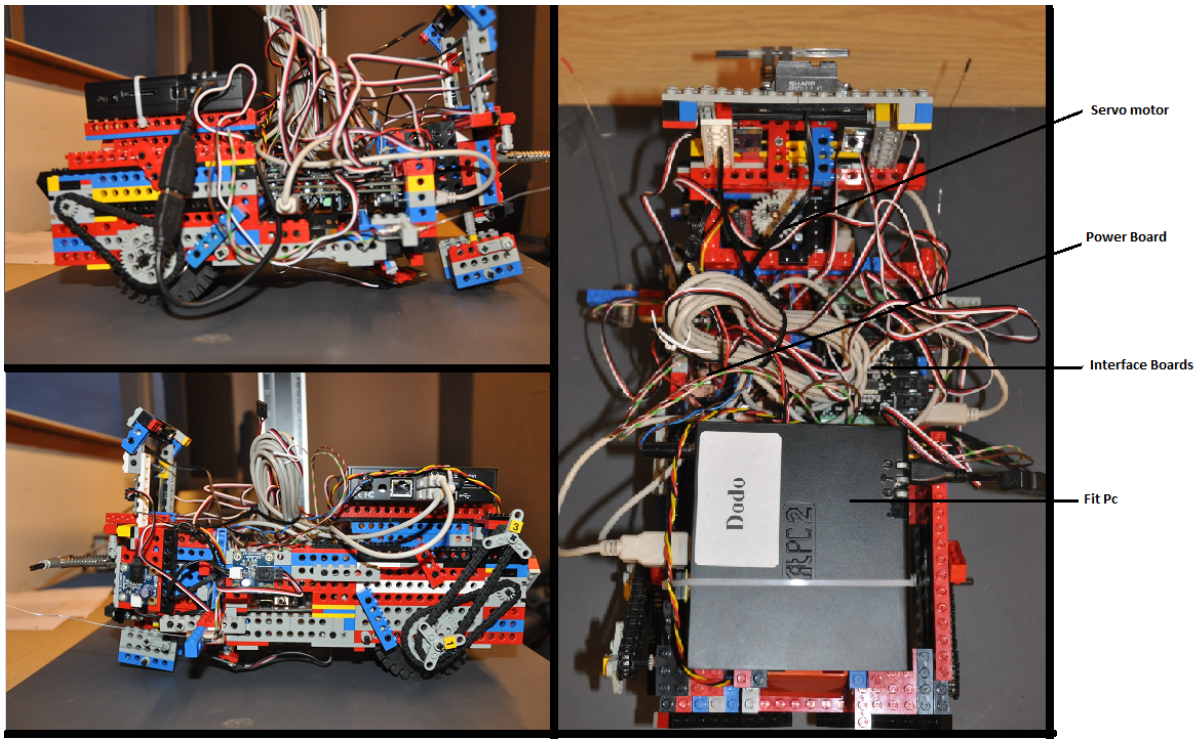

Figure 2: Image of the robot from the two sides and top

Greatest width	23.5 cm
Total length	39 cm
Height	22 cm
Wheelbase	8.9 cm
Rear wheel axes from the back	8 cm

Table 2: Basic measurements of the robot

Sensors

Front-facing IR

This sensor measures the closeness of the objects in front of it and returns an integer. It should give the greatest value when the object is touching the sensor, however, due to the implementation of the sensor, the highest value is obtained when the obstacle in front of the robot is approximately 8 cm away. To have a greater understanding about the values returned by the sensor, we have carried out a test and measured the values received from the sensor between 0 and 17 centimeters with 0.5 cm increments and the result is shown in Figure 3. The robot is programmed to obtain readings from the sensors continuously and retreats when the value from the sensor rises above a certain threshold, which represents a greater distance than the length of the whiskers (6.5 cm from the IR). In this case, the robot drives back and turns to avoid the obstacle.

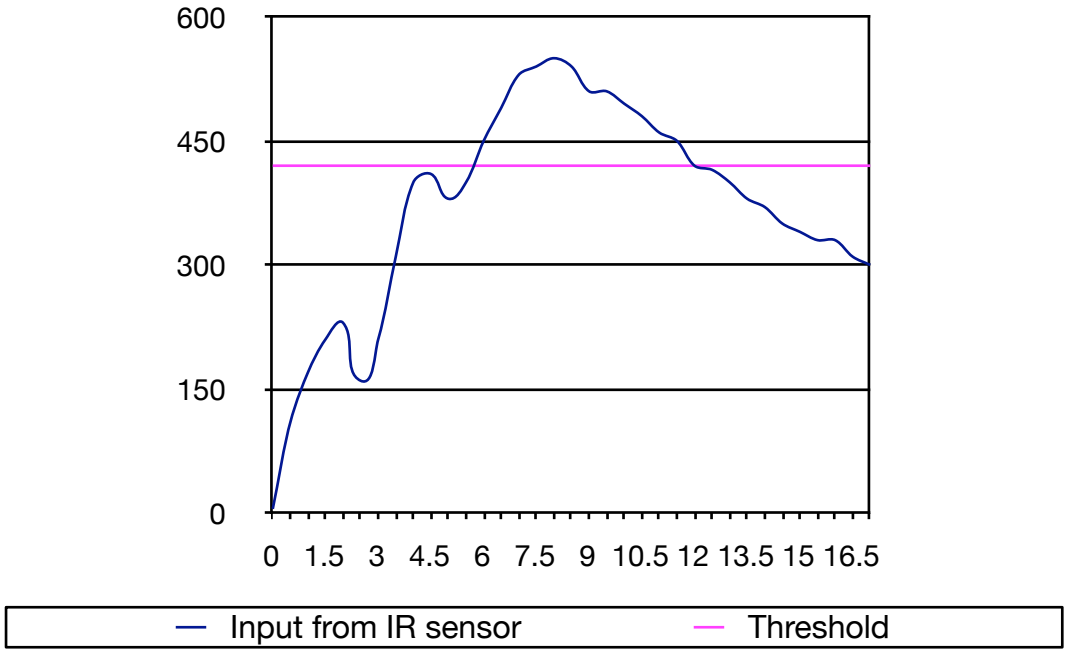

Figure 3: Graph showing the values obtained from the IR sensor with respect to distance. Our threshold is also indicated.

Whiskers

Two whiskers were attached to the robot that are pointing forwards and outwards - one on the left and one on the right. These sensors are triggered when the whisker touches the ring that surrounds it at its base and returns the value 1, otherwise 0 (see Figure 4). The purpose of using the whisker sensors is to avoid obstacles that are invisible to the front-facing IR sensor and making sure that the robot does not attempt to enter a gap that is thinner than itself. This latter use of them was inspired by cats' whiskers which serve the same purpose. The whiskers are 14.5 cm long and sticking out 2 cm to the side. The robot's reactive behavior to the whiskers being triggered is shown in Figure 5. The drawback of the whiskers is, however, that the input we obtain is noisy due to the oscillation of the whiskers. In our implementation we overlook this fact as they usually settle down by the time the reactive behavior is executed.

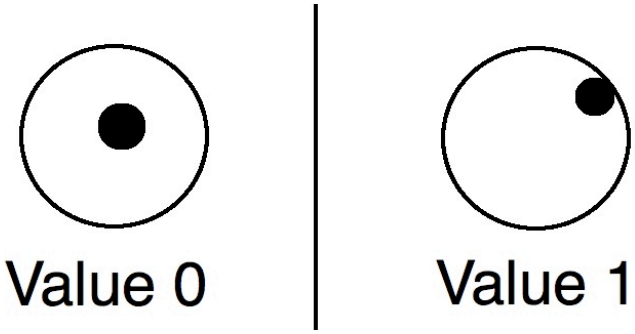

Figure 4: Input obtained from the whiskers

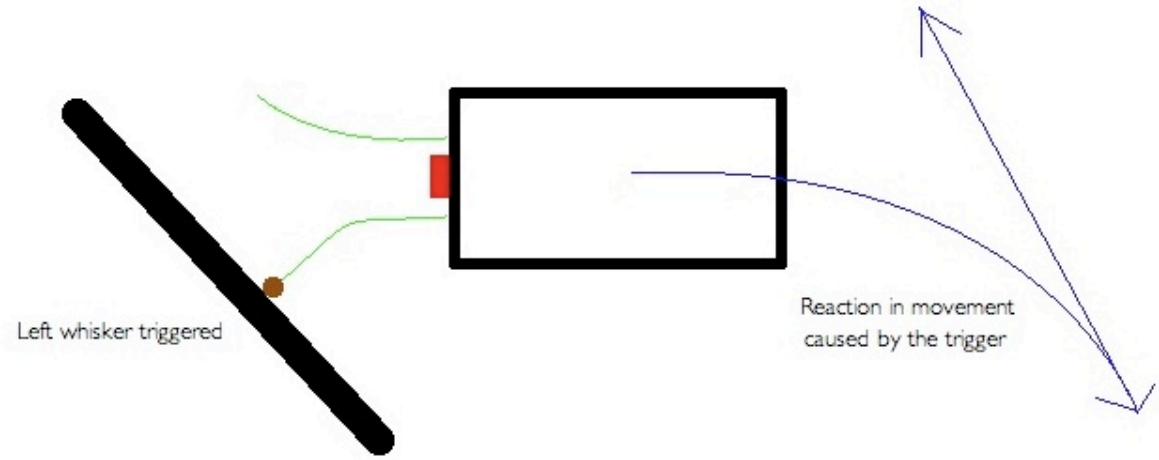

Figure 5: Reactive behavior performed by our robot to triggering the whisker

Bottom light sensors

For the task in Week 4, our robot had to distinguish dark floor from light floor, hence we needed sensors that give information about the floor. We installed two light sensors 8.4 cm apart to the bottom of the robot which measure the brightness of the object. To minimise the effect of changes in light conditions, we have attached a lightbulb to the bottom of the robot between the two sensors (see Figure 6). This provides a consistent source of light and minimises the variance between brightness values obtained from dark and light floor. We use thresholding based on the average of the first values obtained from the left and right sensors after starting our program, assuming our robot starts on light floor. We also use the position of the sensors to navigate into resource sites.

Top light sensors

For Week 5 our robot had to recognise the frequencies with which the lights above the switches were flashing. In order to be able to do that, we have installed two light sensors to the front of the robot at the same height as the flashing LEDs are (15.4 cm). These sensors are at the two sides of the robot as our robot might not be perfectly in front of the light when it triggers the switch, however, this way we can ensure that either of the sensors will reach the threshold required to determine the frequency.

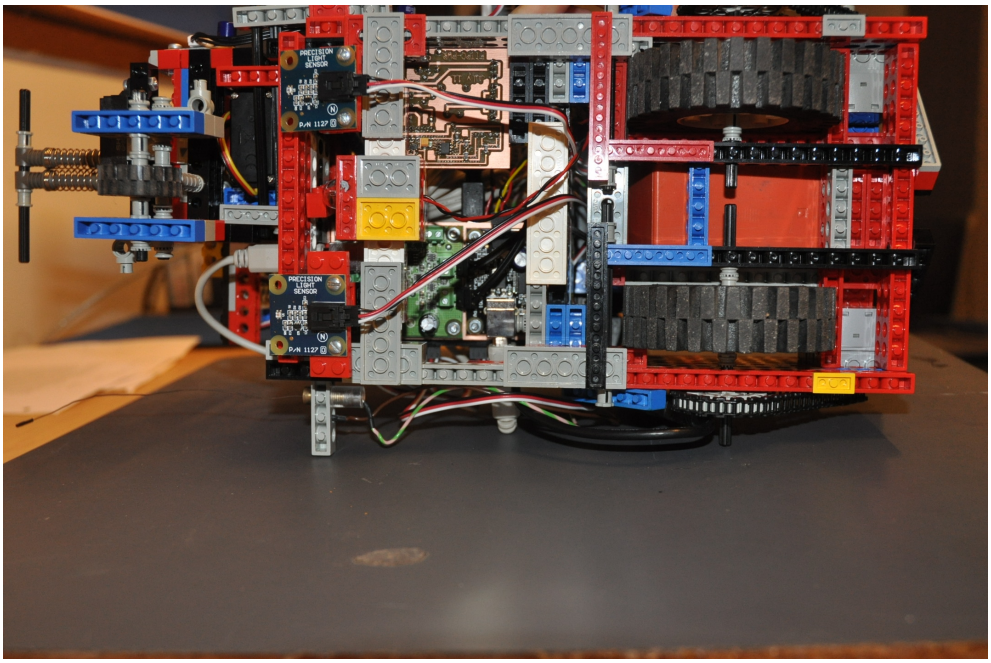

Figure 6: Image showing the bottom of the robot with the light sensors and the light bulb in between them

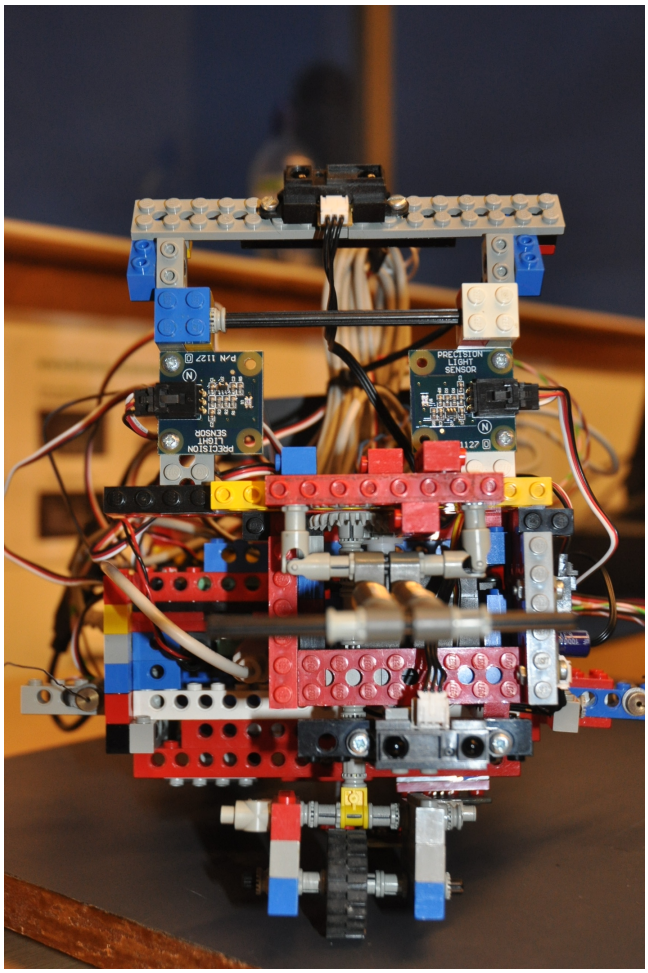

Figure 7: Image showing the front of the robot with the front-facing IR sensor, the top light sensors and the top IR sensor.

Hall sensor

The Hall sensor can detect whether the wheels are moving by changing the output when the bar inside it makes a full turn. We use this sensor to identify the case when our robot is stuck even though none of the other sensors indicate an obstacle. It can also be used to detect objects behind the robot by believing that the robot got stuck whilst driving backwards. The Hall sensor is attached to the top of the rear left wheel and is connected via a track to the axis of the wheel (see Figure 8). The reasoning behind placing it there is that whenever the robot is stuck even powered wheels stop moving and it was easier to install them then to place them on the front wheel.

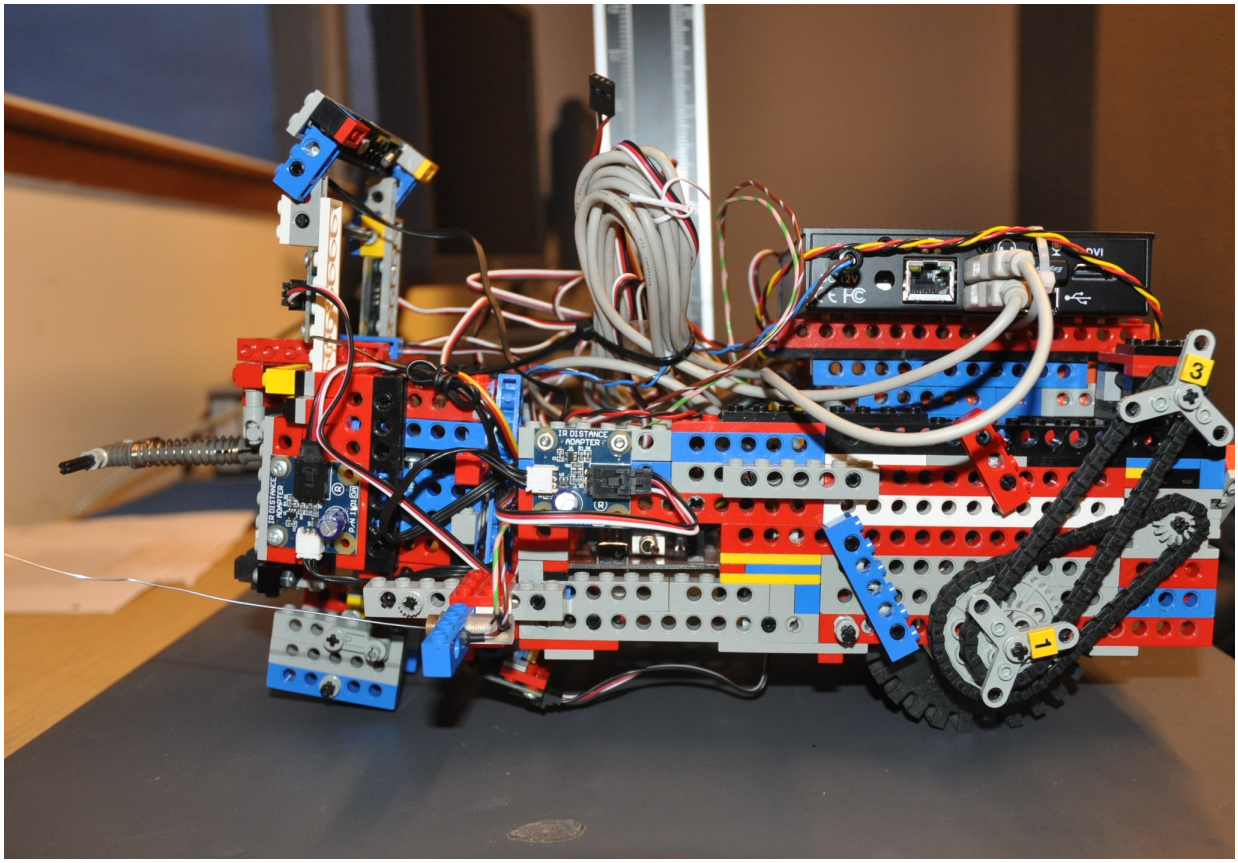

Figure 8: Image of the robot's left side showing the Hall sensor (3) attached to the rear left wheel

Top IR

We have attached an IR sensor, facing approximately 45° upwards, to the front of the robot to detect the gap at the resource site between the walls and help in guidance to the switch (see Figure 9 of gap and IR sensor). This idea was implemented after seeing it working successfully on the robot of group A5.

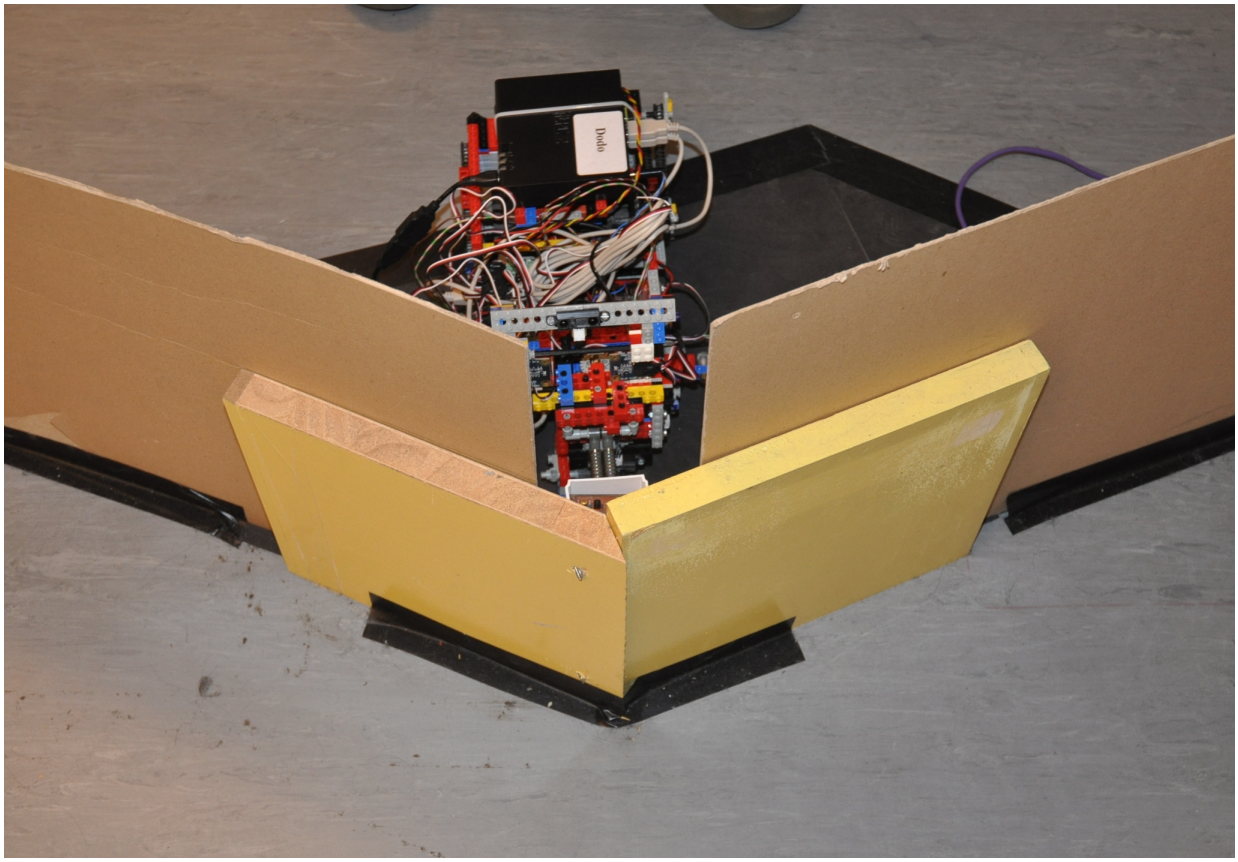

Figure 9: Image of the gap between the walls and the top IR sensor pointing through it

Robotic control

One of our first design decisions (inspired by Brooks, 1990) was to try to create a purely reactive control architecture to solve the task assigned to us. This was partially successful in the sense that we managed to have the robot navigate around in this manner, without the need to use planning or active sensing.

Our control algorithm is built in layers, where top layers effect how the bottom layers react to sensory input. Each of these layers are directly dependent on sensory input (there is a bit of state retained in the robot, however there is so little that calling it a model would be exaggerated).

A top level priority is finding out whether the robot is stuck somewhere. If not (or this hasn't been detected yet, see bellow for details), then we use light sensors on the bottom of the robot to detect dark flooring. When on the dark floor all movements tend to be smaller and less pronounced. Also the reaction to the whisker sensors becomes smaller to have a more wall hugging behavior.

The lowest level are reactions to the whisker sensors and the front IR sensors. The remaining sensors are checked for values only when the other sensors mandate a possibility of them being useful.

Strategies

Since creating reactive robots is all about using simple reactions to orchestrate more complex behaviors.

Wall hugging

Inspired by Schank et al. (2006) we have used the two whisker sensors to avoid hitting a wall but intend the robot will back up a little in the direction that the whisker was triggered and then continue driving forwards presumably to hit the wall a couple of centimetres onwards. With this property we have tried to exploit two important properties of the environment. Our robots will have a high probability of ending up in corners (as discussed in Schank et al.; although they generate this behaviour by virtue of the robot's shape only, we considered that approach impractical with LEGO robots) and corners are where all of the light switches are. The second important property was that 6 out of the 10 walls of the resource sites aim at walls of other resource sites. So when the robot follows such a wall when the wall ends it will continue straight and hit the wall in the next resource site and then it will follow that wall all the way to the switch. The 8Hz resource site had the nice property that both of its walls led to another resource site, and this was in fact often exploited by the robot.

Stuck detection

In order to ensure that our robot does not get stuck for an extended period of time, we have installed the Hall sensor to recognise when the wheels are stationary. The input from the sensor changes parity when the magnet on the bar passes the sensor, i.e. makes a full turn. We exploit this behavior along with the fact that our main function, which defines every movement, is called every 50ms to determine the state "being stuck". The simplest approach was to count the number of loops when the input from the sensor has not changed. A single cycle of "being stuck" is defined by seven iterations of the main code called without change in the input. The robot considers itself being stuck after the third such cycle. If at any time the input changes, both the number of cycles and iterations are reset to zero. After the robot has realised that it is stuck, it makes the opposite movement that it was making previously, i.e. either go forward or backward. This is done by storing the intended orientation when the robot starts to move in a given direction.

Frequency recognition

Frequency of the light switch mainly exploits the hardware configuration of the robot which guarantees that if the light is triggered at least one of its two light sensors will be in front of it. Once one of these sensors reach a certain threshold of activation, we stored the time this happened. When both of the sensors went below the threshold and again above it we subtract the two times to get the period and then derived:

$$f = \frac{1}{T} \qquad (1)$$

We have carried out several tests to define the efficiency of the algorithm. In these experiments the robot was aligned facing the switch, the program was started and we let the robot find the switch, trigger it and readings of frequencies were recorded. For all lights that were available in the laboratory, we have measured the frequencies ten times and the result is shown by Figure 9. We can see that the algorithm performs constantly well for lower frequencies with a standard deviation and mean of 0.0024 and 0.9835 for frequency 1 and 0.0097 and 1.9562 for frequency 2. For higher frequencies, the standard deviation has increased and thus we obtained 0.1208 with a mean of 3.9843 for frequency 4, 0.0862 with 5.9253 mean for frequency 6 and 0.1141 with 7.8632 mean for frequency 8. Even though the precision of the obtained values decreased as the frequencies became higher, the 0.3Hz wide area around the frequencies guarantee that the correct frequency is recognised.

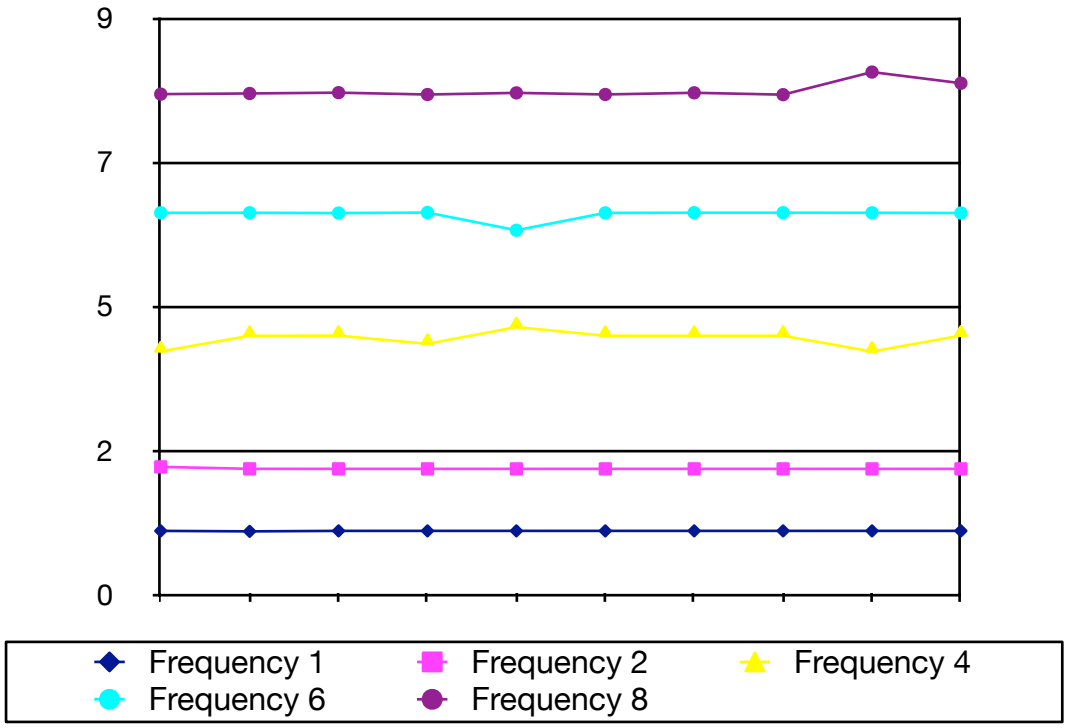

Figure 9: Graph showing the recognised frequencies

Experimental Design

Our main concern was the performance of the robot on its main task. We tested this in eight repeated 5 minute trials. We measured how many resource sites would the robot reach, how many light switches would it manage to trigger and how many victory dances it performed correctly. The robot was always placed in the centre of the arena with a constant starting orientation facing one of the switches. The experimenters did not interfere with the operation of the robot.

We counted as reaching a resource site if the full body of the robot was over the dark area (NB we did not judge whether the robot in fact recognised it was over the dark area since its behavior was very similar to that when it was outside and it was impossible to be sure whilst still keeping the robot autonomous). We counted as triggering the light switch if at least two consecutive flashes were seen and counted as a successful dance if the robot did the prescribed motions within a roughly ±35° range as judged visually by the experimenters.

Results

Main performance results

The robot managed on average to find 52.5% (=2.63 sites, SD=0.92) of the resource sites in the allocated 5 minutes. It's average success rate of triggering the switch after finding the site was 66.7% (SD=0.30). If the light switch was triggered the robot performed the dance correctly 95.8% of the time (SD=0.12).

Run	Black areas reached	Lights triggered	Dances performed
#1	3	3	2
#2	2	1	1
#3	3	1	1
#4	4	2	2
#5	2	2	2
#6	3	2	2
#7	1	1	1
#8	3	1	1
Mean	2.6250	1.6250	1.5000
Std. Dev.	0.9161	0.7440	0.5345

Table 3: Results of the experiment

Discussion

Strengths and weaknesses of current design

Based on the results we can see, there is much to be done in terms of finding the resource sites. The five minute time period seems too short to reach all of the resource sites without a very well planned strategy and some idea of the global layout of the arena. Also performance in this matter was very unreliable, with some runs being very successful and others quite tragic. This seems to have depended on minute physical variance in the environment since the code of the robot is entirely deterministic.

The strength of our robot was that it always acted and almost never got permanently stuck. Another interesting property we observed is that even when some of its sensors malfunctioned or disconnected it managed to continue in its task (albeit with lower efficiency).

Corner oscillation

An interesting side effect of our various behaviors interacting was when the robot got to a corner in the lab it tended to oscillate between driving out in one direction or the other. This is caused by the fact that the robot is basically designed to reach centres of corners (where we presume the light switches are). The robot didn't get stuck in the oscillatory pattern for ever, due to subtle differences in turning on both sides (caused by one wheel having additional friction caused by the Hall sensor) it would eventually manage to get out.

REFERENCES

Brooks, R. (1990). Elephants Don't Play Chess. *Robotics and Autonomous Systems*.

Schank, J., Joshi, S., Tran, J., Taylor, R., May, C. J., & Scott, I.-E. (2006). Rat pups and random robots generate similar self-organized and intentional behavior. *Complexity*.

SOURCE CODE

The source code can be found at http://github.com/gampleman/Robot-IAR.