University of Salzburg
Department of Computer Science

Rantanplan

An obstacle avoiding pathfinder

Embedded Softwareengeneering
Winter 2009/10

Martin Aigner
Alexander Baumgartner
Michael Brugger

January 28, 2010

1 Introduction

Rantanplan is a LEGO Mindstorms Robot [1]. Its name origins from the most stupid dog in the wild west as known from Lucky Luke comics. But there are at least two things that Rantanplan is good at: following a track with its sniffing nose and avoiding to crush into a rock while doing so (at least most of the time). Most implementations of pathfinder robots have a fixed "nose". This results in a strange movement behaviour. That means that the robot has to frequently move left and right to check if the track changes its direction.

We tried to get a smoother movement behaviour by building a moveable sensor that keeps track of the path and informs the steering about a change of direction. The advantage is that a deviation from the given track can be detected very quickly and then a corresponding correction can be initiated immediately. Furthermore we realised the functionality of being able to handle barriers that are on the path by moving around them. Then Rantanplan has to find the path again after it got around the barrier.

2 Hardware

LEGO Mindstorms NXT is a programmable robotics kit released by LEGO in late July 2006. The so called NXT Intelligent Brick is the control component. It can take input from up to four sensors and control up to three motors. For easy connecting the brick to the programmers computer it contains a usb port. The processing unit is a 32 bit ARM7 Atmel microprocessor running at 48 MHz. There is also 256 KB of non-volatile flash memory and 64 KB of RAM inside the brick.

3 Programming a NXT Brick

LEGO provides a LabVIEW based drag and drop tool to combine sensor information with control flow. Of course this is not suitable for our purpose where we want full control over the timing behaviour of the signal flow.

There exist quiet a lot of third-party programming languages and tools for the MINDSTORMS NXT version. The most widely used are Next Byte Codes (NBC), an assembly language and the high-level language Not eXactly C (NXC) that is built on top of NBC. Both are written and maintained by John Hansen, member of the MINDSTORMS Developer Program and are available under the Mozilla Public License [2].

3.1 Tasks and concurrency

NXCs concurrency paradigm is called tasks. Tasks are the implementation of threads on the NXT. They have a similar declaration as functions but can not be called via the stack. Instead tasks can be started and stopped from other tasks asynchronously.

Shared data is realized with global variables. NXC provides a small standard library [3] that contains semaphores and semaphore-operations to assure mutual exclusion of critical code sections.

4 About sensors

We use three different types of sensors to perform the activities [1]:

- Colour Sensor, moveable sensor that keeps track of the path:
  The Sensor assists in helping the robot to 'see'. Using the NXT Brick, it enables the robot to distinguish between light and dark, as well as determine the light intensity in a room or the light intensity of different colours.
- Ultrasonic Sensor, moveable sensor for the ability to see barriers:
  The Sensor helps the NXT Brick judge distances and to check where objects are. The Sensor is able to detect an object and measure its proximity.
- Touch Sensor, for the initial calibration of the position of other sensors:
  The Sensor reacts to touch and release. It can detect single or multiple button presses, and reports back to the NXT Intelligent Brick.

5 Concept

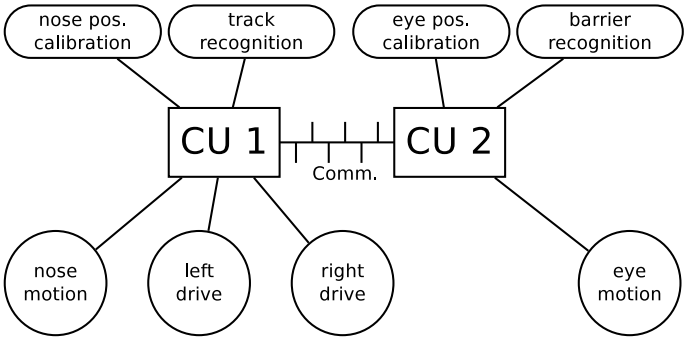

Figure 1: Abstract view on Rantanplan

The main reason for our decision to establish a connection between two NXT bricks is the simple fact that we need control over more motors than there are outputs on one brick.

Figure 1 shows the sensors and actuators that are connected to each control unit (CU). The first unit CU 1 operates as the main steering unit. It controls the drive and the colour sensor motion. The second unit CU 2 controls the direction of the ultrasonic sensor and the check for barriers. CU 1 has the responsibility to tell CU 2 where to position the ultrasonic sensor and to start the evasion procedure when a obstacle is detected. Section 7 describes how the connection is established.

6 Flow-Chart

The flow chart was the first step to define the required functionalities. Figure 2 represents the workflow for the avoidance of obstacles and the path search.

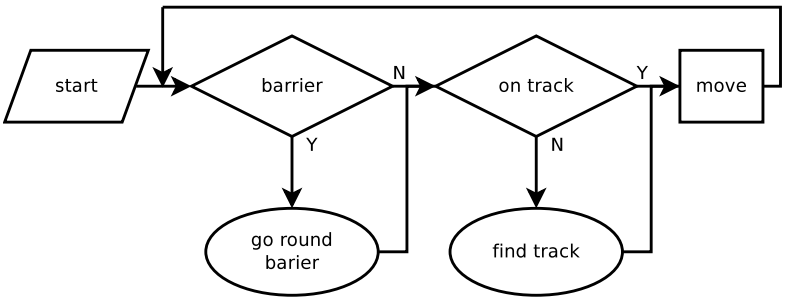

Figure 2: Flow-Chart

Rantanplan starts - after calibrating its sensors - with polling for a barrier. If no barrier is detected Rantanplan keeps track of the path and moves along it. If an obstacle appears, the tracking task is suspended and Rantanplan performs the obstacle avoidance procedure as described in section 9.

7 Communication

There are three possibilities to send messages from the NXT brick to other devices:

- Bluetooth (2,1 Mb/s)
- I^2C (1 Mb/s)
- RS-485 serial high-speed port (10 Mb/s)

We decided to use the serial high-speed port because it is robust and fast. The NXT brick supports this high-speed connection only on the sensor input port number four. Unfortunately this connection is not bidirectional. So we had to implement a protocol that grants mutual exclusive access to the medium.

We did this in a very simple implementation of the token-based protocol [4] with two participants. There is always exactly one receiver and one sender. Immediately after sending one message the sender becomes the receiver and instantly after receiving the message the receiver sends a message.

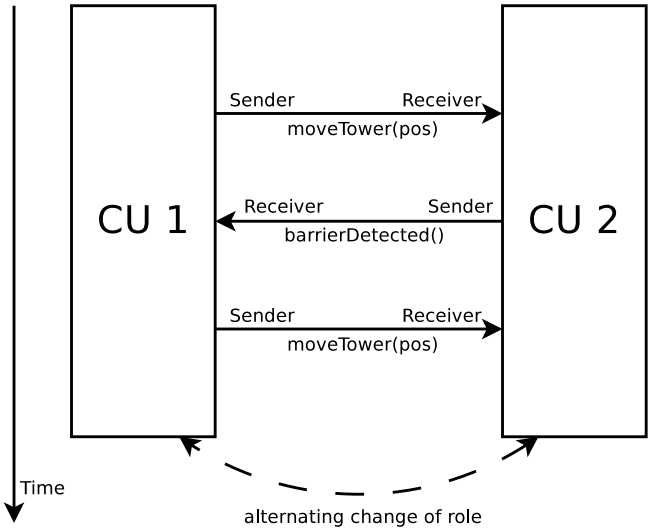

Figure 3: Sender and receiver are permanently changing their roles

As mentioned in section 5 there are two logical units. Each unit owns s single thread which is responsible for the communication. The received message is stored in a global variable which is accessible by other threads that can react on seen barriers or change the tower position if required.

8 Tracking the path

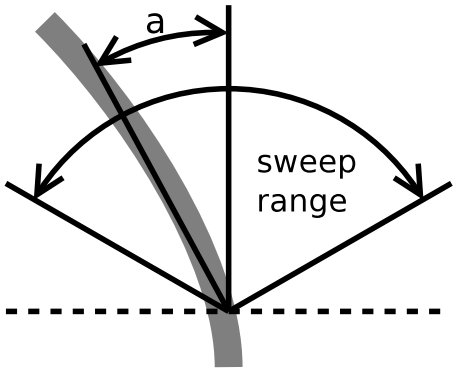

Figure 4: Sweeping range that is covered by the colour sensor

Rantanplan follows a black line on white background. A colour sensor in front of the robot is mounted on a motor that moves the sensor back on the track if the track is left. The rotation count of this motor is processed periodically. Figure 4 illustrates the sweeping range that the sensor can cover. If the correction angle $\boldsymbol{a}$ gets larger than fifteen degrees then an adjusting movement is performed that brings the robot back on the right track.

The advantage of a sweeping colour sensor is that the movement of the robot gets more smooth than with a fixed “nose”.

9 Obstacle Avoidance

With its eyes (the ultrasonic sensor) Rantanplan is able to see barriers in front of it and of course it reacts on seen barriers.

In the following we describe our algorithm for the evasive action step by step. The size of the barrier has no influence on the algorithm.

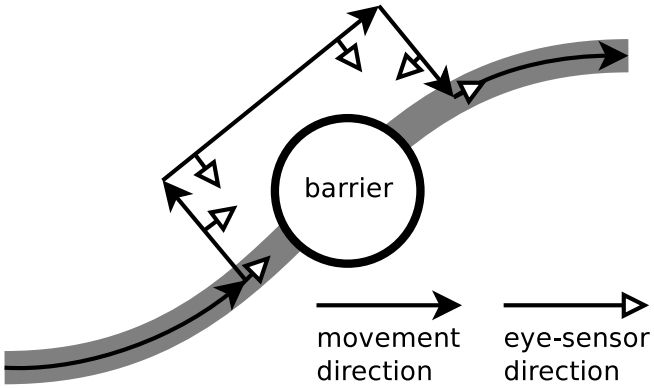

Figure 5: Rantanplan walks around a barrier

1. Suspend the driving and sniffing algorithm and centre the nose (adjust colour sensor position)
2. Turn ultrasonic tower 90° right
3. Turn 45° left out of the track
4. Start searching for a new track [1].
   From now on whenever Rantanplan finds a "new" track the following happens:
   a) Drive just a little bit over the seen track
   b) Turn left until you find the track again
   c) Turn ultrasonic tower 90° left (center the tower)
   d) Release the driving and sniffing algorithm
   e) E N D
5. Start observing the barrier [2]
6. Turn 45° left (finish the 90° left turn)
7. Start driving straight ahead until out of barrier
8. Drive a little bit over the barrier
9. Turn 90° right
10. GOTO 7.

[1] A global variable is set to false. From the tracking thread this variable is set to true if a track is seen

[2] We have three states: Before barrier; Barrier seen; Out of barrier

10 Validation

In this context, validation refers to the process of data validation, ensuring that data inserted into an application satisfies pre-determined formats or complies with stated length and character requirements and other defined input criteria. It may also ensure that only data that is either true or real can be redirected to an algorithm.

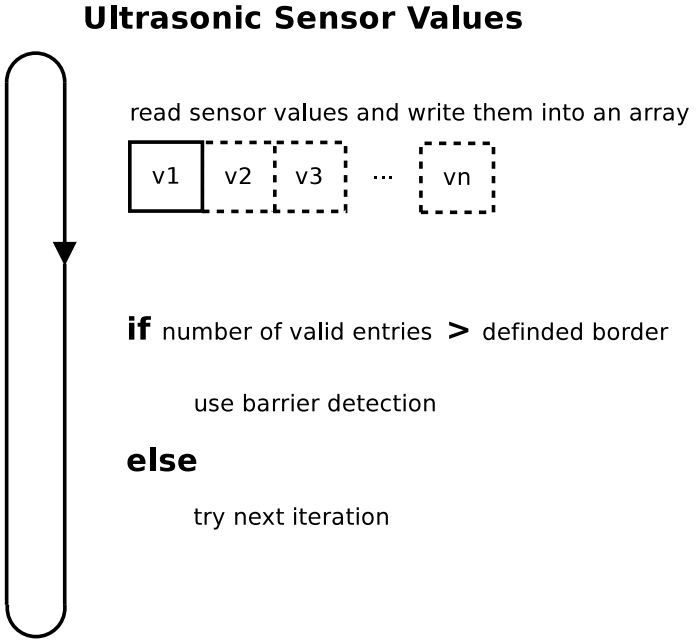

Figure 6: Validation-Process

During the implementation, we have figured out that the ultrasonic sensor partially provides unreliable values. To get rid of this problem, we have used a concept of control engineering: It stores a number of values from the sensor. The values will be redirected to the program only if they pass a plausibility check. For example, five values have been stored and one value is completely different then the others, the validation process will throw away this value and use the others. Figure 6 illustrates the concept.

References

[1] LEGO MINDSTORMS, February 2009. URL `http://mindstorms.lego.com/`.

[2] Next Byte Codes and Not eXactly C, February 2009. URL `http://bricxcc.sourceforge.net/nbc/`.

[3] John C. Hansen. *Lego Mindstorms NXT Power Programming: Robotics in C*. Variant Press, September 2009. ISBN-13: 978-0973864977.

[4] Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems: Principles and Paradigms (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006. ISBN 0132392275.