Acta Cybernetica **19** (2010) 655–660.

Automata Depository Model with Autonomous Robots[*]

Zoltán Szabó[†], Balázs Lájer[†], and Ágnes Werner-Stark[‡]

Abstract

One of the actual topics on robotis research in the recent decades is the robots' autonomy. The methods of self-sufficient problem-solving of the machines brings on several questions in programming, so mobile robots started to extend as tools of education as well.

Our final goal was in this project to create the model of an automata depository that constitutes a closed system from the users' point of view. We model such circumstances that make autonomy important like extreme high or low temperature, closeness of dangerous materials.These circumstances substantiates the need of robots and that they have to solve their problems self-sufficiently, without any direct human interaction.

The model builds up from two main components: the Central Controlling Unit (CCU), and the group of robots. The robots ply in the depository using the line following method. During their activity may turn up some conflict situations, whose autonomous handling is the main topic of our research. Using the right wayfinder algorithm and the representation of the map of the depository, the robots find out after a short information excange, who of them has to give way to the other in order to solve the conflict in optimal time.

The communication between the LEGO MINDSTORMS NXT Robots and the Central Controlling Unit is based on a BlueTooth connection.

The robots' autonomy means that if they loose connection with the CCU, they can finish their commands that they have already received. Nevertheless navigating their physical relocation and sense any incidental new barrier is absolutely their task.

Keywords: autonomous robot, mindstorms nxt, solving conflict situation

1 Introduction

This project has been created for a Scientific Student Conference. Our final goal was in this project to create the model of an automata depository that constitutes a closed system from the users' point of view. We model such circumstances

[*]This work was supported by the LEGO Hungária Ltd.

[†]E-mail: {szabo.zoltan,lajer.balazs}@door2world.hu, werner@virt.uni-pannon.hu

that make autonomy important like extreme high or low temperature, closeness of dangerous materials. These circumstances substantiates the need of robots and that they have to solve their problems self-sufficiently, without any direct human interaction.

The system consists of two parts: the Central Controlling Unit (CCU) and the group of robots. The user shall not do any interaction with the robots, only with the CCU through the user interface. The CCU and the robots bring the user's queries into effect independently, so we can mention the systems autonomy as well.

First of all we had to find a proper hardware for the project. After having examined some robot kits that can be found on the internet, we chose the Lego Mindstorms NXT set, as the robot can be built up easily and people can start work on the software before long. We decided to use the Lejos NXJ platform what is a little JAVA Virtual Machine running on the NXT Bricks, so we could develop our program in object-oriented method.

2 Build-up and software of the robots

Using the sensors that can be found in the Mindtorms NXT set, we could easily build up a robot that we could use for this research. The robots stand on a three-wheel frame that makes them stable enough. We used only two of the available sensors: the light sensor for the line-following and the ultrasonic sensor on the top of the robot in order to sense if it reaches a barrier and to know if the robots fork-lift is fully under the box it is going to lift.

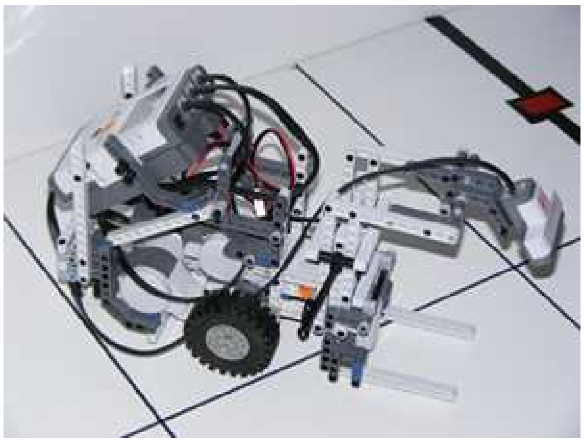

Figure 1: Top view

After having implemented some basic functionalities such as moving forward to a given distance, turning in a given degree, getting to a given point in a virtual frame of reference or lifting up and letting down the fork-lift, we started building their main software. Its tasks are: to process a route computed by the Central

Controller Unit, to monitor every important information to the CCU, discover new barriers, to keep in touch with the other robots. Our first idea was to implement a recursive routeplanner algorithm that runs on the NXT Brick, but unfortunately we had to face the fact that the recursion uses all the available memory after the second step. Because of we had no time and close deadlines with this project, we decided to use the CCU to plan the route. This algorithm is a breadth-first search algorithm. The adjacency list stores the path points on the map. By this way we use every time one of the shortest routes. There is a work in progress to develop an other non-recursive algorithm that can run on the NXT Bricks.

The base class of the robots' software is the Controller class. The tasks mentioned above are shared between two threads: one thread is responsible for the navigation and all the other functionalities that help the robot to complete its assignment, the other thread makes the communication.

3 Navigation

The virtual map of the depository is a frame of reference with synchronizing points in the middle of each field. We had some trouble because of the imprecision of the lego parts, that's why we chose the line-following method. Because of the same reason, we had to take care to implement some corrections when turning the robots. If the robot turns 90 degrees in a direction during processing its route, the result was not always correct. The best solution was that if the robot could not sense the line after having turned, it starts iteratively swivel in both directions by increasing angles. This way it will surely find the line. The robots orientate themselves by keeping their last position in their memory. They know that they have reached a synchronizing point by sensing again black line after they had left it before. If they cannot sense black again, then they have left the line because of some reason, so they have to search for it again using the correction method mentioned above.

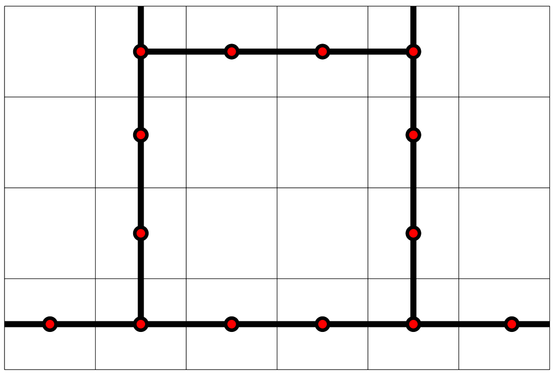

Figure 2: The map

4 Conflict Situations

In our model we determined three types of conflict situations. One is that two robots are on their way, and their routes cross each other. The second one is when only one of them has an actual task (Robot A), the other is idle (Robot B) but staying on the route that Robot A has to process. In the third situation both robots have their task, but one of them could go faster on its route, so it comes up with the other. I this case the conflict handling is very simple, the faster robot only have to wait for the other.

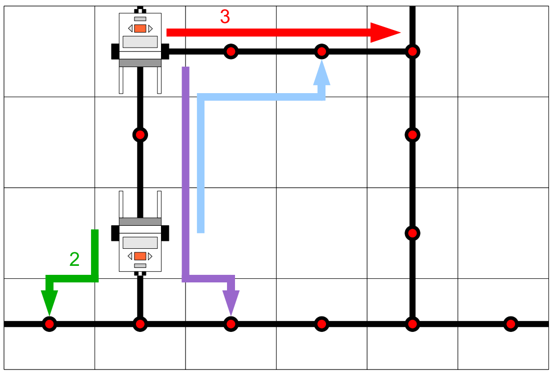

Figure 3: An example conflict situation

In every monitoring message the robots send the actual coordinates on the map. They send these information for each other not just the for the CCU, but because of some issues with the firmware these data go through a pipe in the CCU (it will be explain below). By this way they can calculate the distance from the other. They use the Pythagorean theorem, but we are working on an other more effective method which will use the map for calculating.

When they got the distance they have to determine whether this is a conflict situation and if it is what kind of conflict situation.

First of all they examine the distance to know if it is under the threshold or not. If it is over the threshold they can go on. If it is under or equal with the threshold they have to use one of the avoiding tactics. To determine the avoiding case they get more information. For first the state of the other robot (idle, busy), and the route of the other. They can determine wherher their routes have common point or not. If there is at least one common point then they have to continue the method else they can go on their own route. If they know that their routes cross each other then they search the first free path point and send the length of the avoiding route to the other robot. A free path point is one of the path points on the map along the other's route but not on it. The two robots can decide which has to do avoiding, it depends on who has the shortest route. After the avoiding robot is out of the way, sends a signal to the other to continue its own route. When the robot finished provessing the route, sends a resume signal to the avoiding robot to continue the interrupted task. In this case the avoiding robot go to the coordinate where the task was interrupted, and continues it.

5 System architecture and software

The CCU works as a bridge between the TCP/IP network and the Bluetooth network. It accepts connections from the users via TCP/IP protocol. The users can use a string-based protocol to order actions in the system. We are working on a graphical user interface to ease the users' work.

On the other side the CCU has to build and keep persistent the connection with the robots. Because the CCU initiates the connections, it has to know the name and the address of the Bluetooth devices. The core of the CCU is the ServerCore class. This is the main controller which is responsible for delegating tasks to the robots and serving information to the users. It starts two threads to listen to the user connections and for the robot communication control. These are the channels for the communication between the two networks.

The robot's software has the Controller main class which is responsible for starting the communication thread and for initialising the navigator classes and reset the position of the fork-lift.

The solution of the problem with the firmware Bluetooth implementation is in the communication thread. We had to face the fact that in this version of the firmware we can use only one connection in one time. So we can not communicate with the CCU and the other robots at the same time as we planned before. So we modified the protocol of the CCU-Robot communication and made it able to let through the messages between the robots. We use an interface in these classes to make it possible to modify the program when multiple connections are available.

6 Problems and future plans

Some minor problems turn up because of the sensitivity and imprecision of the sensors. We draw the inference that the light sensor need constant circumstances after its thresholds had been set. It measured false data when the sun shone through the window or when the lights were on. It was very difficult to find the right values for the ultrasonic sensor and the right form for the box in the depository to make them work together.

Our main problem was the imperfection of the BlueTooth connectivity in the NXJ platform. After the connection between a robot and the CCU has been established, we tried to command the robot to build up a connection with the another. Then the robot's software suddenly froze, and we did not know what happened. Having searched for the implementation of the BlueTooth connectivity in the NXJ code, we found that only one listener can be active in a robot. While we were searching for some solution on the internet, we could see that several other projects also missed this feature so we look forward for the next version of the NXJ firmware.

7 Conclusion

We succeed to create a model of an automata depository in which the user only needs to command a computer, there is no interaction needed between human and robot. The CCU and the robots can manage to bring into effect the changes that the user had queried. We found the Lego Mindstorms NXT robots very useful in teaching programming, especially robotics and embedded systems.

References

[1] Druin, A., Hendler, J. Robots for kids: Exploring new technologies for learning. San Diego, CA: Academic Press, 2000.

[2] Wie hauche ich dem kleinen Roboter "Asuro" das Leben ein? `http://www.dlr.de/schoollab/desktopdefault.aspx/tabid-2980/4537_read-6668/`

[3] LEGO, LEGO MINDSTORM set for Schools # 9790. Billund, Denmark: The LEGO Group, 1999.

[4] `http://claraty.jpl.nasa.gov/man/overview/index.php`

[5] Lui, M., Hsiao, Y. Middle school students as multimedia designers: A project-based learning approach. Journal of Interactive Learning Research, 13(4), 311-337, 2002.

[6] Java for LEGO Mindstorms. `http://lejos.sourceforge.net/p_technologies/nxt/nxj/api/index.html`

[7] Cormen, Rivest, Leiserson. Algoritmusok. 2003.

[8] `http://mindstorms.lego.com/eng/Overview/Bluetooth.aspx`

[9] `http://www.daimi.au.dk/~dam/logs/doku.php?id=projekt2#thursday_d._13_12-07`

Received 3rd January 2009