An Interactive Maze Scenario with Physical Robots and Other Smart Devices

Marc Jansen, Maria Oelinger, Kay Hoeksema, Ulrich Hoppe
University of Duisburg-Essen
Faculty of Engineering
Institute for Computer Science / Interactive Systems
{jansen,oelinger,hoeksema,hoppe}@collide.info

Abstract

This paper describes an educational application that combines handhelds (PDAs) and programmable Lego bricks in a classroom scenario that deals with the problem of letting a robot escape from a maze. It is specific to our setting the the problem can be solved both in the physical world by steering a Lego robot and in a simulated software environment on a PDA or on a PC. This approach enables the students to generate successful sets of rules in the simulation and to test these sets of rules later in physical mazes, or to create new types of mazes as challenges for known rule sets. In this paper we describe the technical setting for this scenario, different pedagogical scenarios and we will report an evaluation with a group of studens in a school environment.

Keywords: Challenge-based learning, smart devices, motivation

1. Introduction

The work presented in this paper is closely related to the COLDEX (COllaborative Learning and Distributed EXperimentation) project. Within the project the partners try to enhance learning and experimentation scenarios for students with the use of real scientific devices. Furthermore the students should learn the working mode of scientists and work with real scientific data. Additionally this project focuses on orchestrating learning groups both locally and on distance. Therefore the project has not only project partners in Europe but also contacts to Latin America and especially two associated project partners in Chile.

Additionally one goal of the project is to embed smart devices transparently into the learning process and thereby enrich the learning process. This is motivated by former work done in the area of ubiquitous computing by the COLLIDE group. Another project that shows the ideas for ubiquitous computing of the COLLIDE group was the NIMIS (Network Interactive Media In Schools) [1] project that deals with the question how to orchestrate a classroom for early learners.

Within the work presented here we combined different smart devices to enrich the learning environment: we built an ad-hoc classrom based on a wireless network, handheld computers acted as clients within this ad-hoc scenario and programmable Lego bricks, so called RCX (Robotic Command eXplorer), were used to allow physical experiences with robots.

The major goal for the students was to orchestrate sets of rules that allow robots to escape out of given mazes. Therefore the students had two possibilities to create mazes: on the one hand they could create virtual mazes with the help of a little tool running on networked handheld computers and on the other hand they could build physical mazes out of stones.

Later on the students had to orchestrate sets of rules which the robots should follow to find their way out of a given maze. These sets of rules could be exchanged within the ad-hoc network of the classroom. This allows that different groups of students can test the sets of rules from other groups and the groups can try to orchestrate mazes the robots can not escape from with a given set of rules. This leads to a certain competetive scenario that fits perfectly in the sense of challenge-based learning: one learning group can take a set of rules of another learning group, orchestrate a maze that is not solvable with this set of rules, give the so far unsolvable maze back to the learning group that orchestrates the set of rules and this group can now update and change the set of rules so that the robot can again find its way out of the maze.

Additionally this set of rules could be sent to the physical robots using the IR interface of both, the handheld computer and the RCX.

Figure 1 - The programmable Lego brick RCX

This paper will beside the pedagogical background also explain the technical setting for this scenario, point out the problems behind the technical challenge to combine different kind of smart devices and provide solutions to a number of those problems and also evaluate the work done by the students. The last paragraph will deal with future developments about this scenario and ideas how the

still existing technical problems can be solved in the future.

2. Learning Scenario

The following learning scenario was designed to be used in school lessons with children in the age of 13 - 15 in computer science or in exploratory mathematics lessons. It has been tested in an extracurricular "project week" (see below).

2.1 The Problem

Our learning scenario is developed from a well-known problem: defining a maximally general strategy to let a robot escape from a maze. This problem has, e.g., been proposed and worked out by Abelson and diSessa (1982) [2]. It was part of the Logo folklore, though due to its non-trivial nature not among the most frequently used Logo projects. In the treatment by Abelson and diSessa, the robot "understands" standard turtle commands (move *forward* and turn *right/left*) and it "senses" its direct neighbourhood (*free* or *wall* in front, to the right or to the left). The robot can be just a software creature or a physical "floor turtle". A first systematic strategy is wall-following (left or right). To overcome limitations caused by "islands" in the maze, an absolute heading is introduced and exploited in a more sophisticated algorithm.

Following the Logo approach [3], the learning task here is a programming task (from scratch!). In an adaptation that we have used in our own teaching, we have turned it into a "reactive programming-by-example task" (Hoppe 1997, not published) which makes it somewhat easier to handle: The robot has to react to the current situation description, a triple of what there is in the cell to its *left*, in *front* and to its *right*. These three cells (on a floor of square cells) can either be *free* or contain a piece of *wall*. Each user-defined reaction will be added to the memory as a rule which will be applied under the same circumstances. In a situation to which no existing rule applies, the user/learner will be prompted to enter a new action. Rules can be generalised by replacing concrete elements of situation descriptions by jokers which would match any value. The user will only react by defining actions in concrete situations without having to define global control strategies (local reactive programming).

The following is an example of a rule set for following a wall to the right. It can be developed from only three example situations:

```
front=free & right=free => FORWARD
front=free & right=wall =>
                FORWARD, turn RIGHT
front=wall => turn LEFT
```

Here, FORWARD is measured in unit steps depending on the cell size, and turns are measured in units of 90°.

In our example scenario this problem is mapped onto a physical environment in which the robot is a Lego Mindstorms vehicle, controlled by a PDA. It is intuitive to have a mobile controller next to the mobile physical device. The portability situation enables students to contrast simulation and bodily enactment. Yet, when we implemented the software, we encountered a number of obstacles in fully implementing this using available technology. We will describe our experience and first steps towards overcoming the obstacles.

2.2 Specific Learning Goals

The guiding question in this scenario is: How to construct a set of rules which enables the robot to find a way out of an arbitrary maze from every starting point? If we do not find a general solution, can we find one for restricted classes of mazes (e.g. mazes without islands)? Indeed, to meet the first challenge additional constructs are needed, yet there are quite elegant solutions for restricted classes. If a rule set fails within a maze it should be examined what property of the maze it could not cope with. Can the set of rules be improved to solve the maze, or is it generally not possible to do so? Additional questions might be whether two developed sets of rules are equivalent or not. Later on the students can try to find minimal sets of rules to solve a given class of mazes. The essential cognitive challenge in this kind of reasoning is to predict global behaviour from the locally defined rules, a very interesting challenge from a computer science point of view. To enable this reasoning a global view of the rule set has to be provided.

The guiding questions lead to some requirements for the arrangement of the learning environment. We decided to create a learning environment including both a virtual simulation and a physical maze with the following characteristics:

- Within the scenario it should be possible to revert to an expressive simulation, in particular to
  - create virtual mazes
  - view a maze situation with different perspectives (two or three dimensional)
  - run robots within developed mazes
  - develop and modify sets of rules while the robot is running (to support reactive programming).
- There should be a possibility to build real robots which behave according to developed sets of rules trying to find a way out of a real maze made up of physical bricks. There should be a mode of "runtime programming" similar to the mode within the simulation and a mode of transferring a complete set of rules to the robot.
- All (provisional) results of the students' work should be annotated and stored centrally, accessable for other groups.

- Every group should give a summary of a day's work, which is stored centrally and accessable for other groups and the teacher.
- Real time testing with a robot sometimes is a bit time consuming, all groups have to share one big physical maze. It should be easy to alternate between virtual and real situation using the same data base for developed sets of rules.

2.3 Usage of Wireless and Mobile Technologies

Wireless and mobile technologies are an important part of the implementation of this scenario. Following previous developments [4], we decided to use PDAs connected within a WLAN. Each PDA provides a complete simulation environment and can be used as a programming device for the real robots via an infrared channel. Simulation and robot programming are possible within one working environment. The used data base is stored on a central data server within the WLAN. Each PDA can access and modify this data base.

The programmable Lego brick RCX is another important element of the scenario. It is the heart of the robot in the physical maze, and it allows learning groups to build and program individual robots at reasonable prices. The Lego programming software which comes with the Robolab kit enables a nice start into programming but is of quite limited expressiveness. A LeJOS [5] enabled RCX is Java-programmable, which we used to provide a special "reactive programming interface" for the special task which we thought was adequate for children at the age of 13 - 15.

2.4 Pedagogical Strategy

At the beginnning of the project, guiding questions should be worked out by the students. A possible entrance to the scenario is to start with the maze simulation on the PDA only in three dimensional mode without the possibility to save any rules. This is the most ego-syntonic way to empathize the robots situation. The students stray through the maze and recognize the benefit of thinking about a strategy to get out of it. When the main challenge of the scenario is sufficiently explored by the students, work can start over and the fully featured simulation (different perspectives, rule-saving, global view of rules) can be used. Predefined and self-developed mazes can be saved, commented and stored at the rule server within the WLAN.

Within synchronous phases, the groups explain the results of their work to the class and put the developed mazes and sets of rules to the other groups disposal. All the time sets of rules are tested on new mazes to check their quality. Sometimes the sets of rules can be improved, but on some mazes the sets of rules will fail in general.

After this first simulation period (or maybe by division of labour) each group builds a robot, the physical existing model is added to the virtual simulation. The robots can be set into the physical maze and can be directed by transferring previously developed sets of rules. The PDA is the tool to get a connection from the rule server within the WLAN to a single robot (via infrared). The robot can also be set into the maze with an empty set of rules and can be programmed during runtime using the PDA similar to the runtime programming environment within the simulation.

3. Technical Description of the Scenario -

This paragraph will describe the technical setting of the scenario. Overall there are three different things that have to be explained: the used devices, the ad-hoc wireless network and the communication between the different kind of devices.

3.1 The Handheld Devices

As handheld devices we have choosen the Compaq iPaq 3850. It shows good a performance based on a reasonable amount of memory (64MB) and good calculation power. Furthermore this device has got a wide variety of possibilities to upgrade it. Compaq builds so called expansion packs which allow to easily extend the handheld with different products like multimedia extension, memory upgrades or PCMCIA cards. In our case we decided to use an expansion pack with both, the possibility of PCMCIA cards and an additional battery that increases the time that the handheld could be run without being recharged. As a WLAN card we decided to choose the WL110 card from Compaq since it is 802.11b compliant. Additionally to the WLAN connection the iPaq has an infrared port that could also be used as an interface to exchange data with the handheld. Actually, this infrared port was used to transfer data from the handhelds to the Lego robots.

One problem we had to solve while working with the handheld device was the question about how to programme the device. There are different Java Virtual Machines (JVM) for the iPaq. The range of Java support starts with support for the Java 2 Micro Edition (J2ME) and goes to full support of Java 1.4.1 (J2SE 1.4.1). Since the Java support heavily depends on the operating system running on the handheld we had a look at three different operating systems for the iPaq: Pocket PC2002, SavaJE 2 and Linux.

The Java support while using Pocket PC2002 is relatively poor. There are only J2ME implementations available and for example the J2ME RMI profile is not compatible to the RMI version running under J2SE 1.4.1. Also seeing that the CDC profile of J2ME that is available for the iPaq is from a J2ME point of view a quite rich profile it is relatively poor if it is compared to the J2SE 1.4.1 standard im-

plementation. To use the infrared port of the iPaq additional drivers have to be installed when using PocketPC 2002.
The next operating system we had a look at was SavaJE 2 which is a fully Java based operating system for smart devices like handhelds or mobile phones. It supports the full J2SE 1.4.1 implementation but lacks the support for the infrared port of the iPaq, at least within the beta version that we tested. Another problem with using SavaJE 2 is that the developers of SavaJE decided not to work any longer on the iPaq version of their operating system but to concentrate their work fully on smart devices like cell phones which means that only the beta version will be available for the iPaq but no final one.
The last operating system we had a look at was Linux. There exist some distributions that supports the iPaq and we decided to choose a distribution called Familiar because this distribution seemed to have the best support. After the installation of this distribution, which was not at all as comfortable as for the other operating systems, we installed a special J2SE implementation for the iPaq under Linux. This implementation is fully J2SE 1.3.1 compliant but with performance problems when the GUI is built using Swing.
As the result of our tests with different operating systems for the iPaq we decided to stick to Linux as the operating system, also if it does not support the most up to date Java implementation. Additionally Linux has some advantages for the usage of the infrared port which was very important for us since the Lego robots only have an infrared port.

3.2 The Programmable Lego Brick (RCX)

For building the physical devices we decided to take the Lego Mindstorms technology which is based on a programmable Lego brick called RCX as shown in Figure 1. With the help of Lego it is very easy for students to build physical robots. The RCX allows the students to control three sensors and three motors. In our scenario we put two different kind of sensors on the RCX, on the one hand we had a bump sensor that allows later on to react when the robot runs in front of a wall and on the other hand we had two rotation sensors to control the movement of the motors. We only needed two motors in our scenario since the robots only had two motorised wheels that allowed them to move through the maze.

Figure 2 - The iPaq as a proxy

The only way to communicate with the RCX is through its infrared port. Unfortunately the protocol used by the infrared port of the RCX is not IRDA compatible what leads to a not straight forward communication between the iPaq and the RCX since the iPaq usually only supports IRDA. As mentioned before we decided to take Linux as the operating system which allows us to use a certain driver that emulates the communication protocol of the RCX through the IRDA compliant infrared interface of the iPaq.

Another topic to discuss is the operating system running on the RCX. Usually it ships with an operating system especially designed by Lego for the RCX. That operating system has no Java support at all. Furthermore would it not be possible to send data from a PDA to the RCX if the RCX is equipped with the original Lego operating system since there exists no software for the iPaq that is capable of sending in the proprietary Lego protocol. As an alternative we have used the LeJOS operating system which is based on Java. It has a little Java virtual machine that, with respect to the low memory of the device, is quite powerful, e.g. it has full thread support. LeJOS additionally provides an API for communication with other devices that eases the communication with the iPaq so that we decided not to use the original Lego operating System but LeJOS.

3.3 The Ad-Hoc Wireless Network

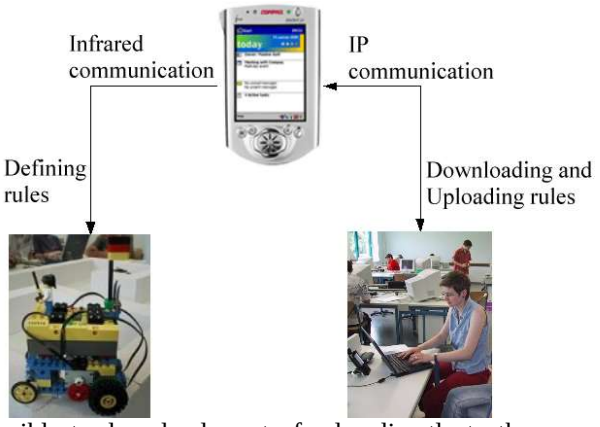
In the classroom itself we set up an ad-hoc wireless network [6]. For this purpose we had a laptop running Linux with a fully configured DHCP server and a Compaq WL410 as the access point.

Within the DHCP server only the network cards that belonged to the iPaq handhelds were registered so that only those cards received an IP address. This is a question about security and ensures that only registered cards are in the network. The wireless network itself was strong enough that the students were not limited to the classroom but it was also possible to work outside the classroom, e.g. in a room with internet access or on the school yard.

This wireless network allowed the students to exchange their sets of rules by uploading them to a central server. Additionally the students could compare their sets of rules to the sets of rules of other groups by downloading them and applying them to a robot that tries to find the way out of their maze.

3.4 Communication between the Different Devices

Since the used devices had very different interfaces for the communication it was not possible to communicate with all of them in the same way. Because the RCX only got an infrared interface for the communication it was not



possible to download a set of rules directly to the server which was only accessible via an IP connection. Therefore we used the iPaq as some kind of proxy since the iPaq was able to communicate on the one hand with the server and on the other hand with the RCX. Usually the students downloaded a set of rules with the help of the iPaq from the central server and then transferred this set of rules to the RCX.

Figure 2 shows how the iPaq worked as a proxy between the RCX and the server.

The infrared communication from the iPaq to the RCX was not only used for downloading set of rules from the server but also for defining a new set of rules in the physical maze. Since the communication back from the RCX to the iPaq was not working properly the students had themselves to determine the situation the robot was in. Afterwards they had to define a rule and send it to the RCX. The problem that the communication back from the RCX to the iPaq does not work is still not solved but for this scenario this was just a minor problem since the students could walk through the physical maze beside the robot and determine the situation themselves.

4. Setting

The maze scenario includes both: the software representation and the physical representation. The first one simulates the maze scenario including the construction of mazes, a virtual robot and the set of rules. The second representation is a maze of physical bricks and the Lego Mindstorms robots which are again guided by rules. The rules are sent from a handheld to the robot.

4.1 Maze Setting and User Interfaces

The software consists of three different parts. The *MazeConfigurator* serves as a construction tool for the mazes which should later on be solved. The *Maze* programm itself contains three different views on the problem: a 3D view of the maze where only the current situation can be seen, a 2D view of the maze as a whole, and the table of the existing rules.

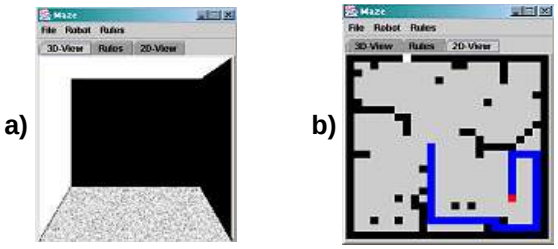


Figure 3 - Maze a) 3D view and b) 2D view

The robots 3D view in the maze reveals, e.g. that the way ahead and to the right is blocked and the way to the left is free. To solve the maze task only by the 3D view is the most difficult task in the scenario, but this also simulates the lifelike challenge best. The 2D view includes the starting point of the robot and its trace.

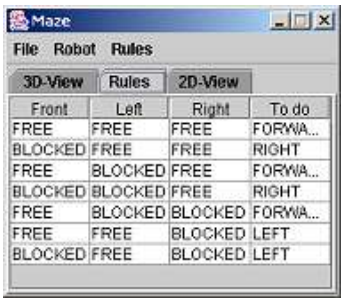


Figure 4 - Table view in the Maze software

The *RulesServer* is necessary to collect the different sets of rules which are uploaded. The PDAs are connected to the RulesServer using WLAN.

Finally there is the *RuleSetter* which actually enables the students to send rules to the robot: the user maps the real maze situation of the robot into the interface, decides where to go in this situation and commits this rule to the robot.

4.2 Project Week

The project took place during a summer project week at the Robert Schuman comprehensive school in Willich, Germany. The event proceeded in two rooms: one classroom and one computer room with internet access. Every day the students had to send a reporting email from each group. Furthermore the students were encouraged to send voluntary personal emails.

The project consisted of 20 students, among these two girls. The heterogeneity of this learning group based not only on the gender aspect, but especially on the different ages; the youngest participants were 13 years old, the oldest students 15 years. Furthermore the pre-knowledge varied, e.g. some already had experiences with their own Lego kits, those differences are visualized in Table 1. The

students at the age of 15 got the extended Lego Mindstorms set. Since they were the oldest ones, they were supposed to construct an escaping robot with the different building materials.

Every day each group of students had access to the maze website and related websites (Lego Mindstorms, University of Duisburg-Essen). The Lego software was available in the computer room.

4.3 Project Week Workplan

First of all we planned to group the students and to present the project website to the groups. This website includes a discussion forum and a chat. The students should first test the discussion forum where they should later on present and discuss the results of their work so that those results could be exchanged during the project and also later on. To ease the task of sending a work report each day we included an eMail form in the internal area of the project website. The teacher should hand out the Lego Mindstorms boxes to the students. A short introduction on Lego Mindstorms should be given to the students and the students should start building their first very basic robots controlling light and bump sensors. Additionally they worked with the Lego Software tutorial to get a first hand on the Lego robots.

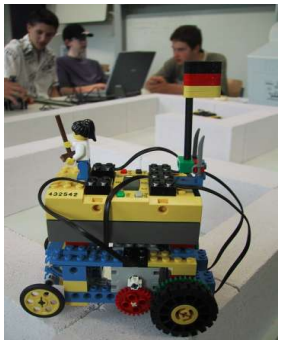


Figure 5 - The robot

Several programming tasks were given to the students to allow them to get a first impression about the challenge to help a robot to escape out of a maze. Subsequently the students were provided with an elementary theory about mazes, how to design them and how to develop strategies to escape from mazes. Within this theory phase the students developed the first feasable sets of rules. Afterwards the students had to develop a special robot which later on should be used in the physical maze.

One of the problems already detected at this early stage was identified as the "dead end" problem: the robot seemed to stuck if it comes to a dead end, i.e. walls in front and to both sides. From there it could not get out, basically because it had no history about its reactions on the way so far. Afterwards the groups started the construction of robots with rotation sensors which are, as a matter of fact, necessary to control the robots precisely.

In the next step the students were handed out the PDAs and an explanation of the software. Then the students could work with the PDAs to test their strategies which they created in the theory phase. Thus they compared the boundaries of the Lego software with the optimized software of the university.

On the last day the students tried to deploy their so far created sets of rules on the physical Lego robots and so to help them to escape out of the physical maze. Finally they presented their results and discussed the developed sets of rules.

5. Observation

The group setting is shown in the following table:

group	m	f	class	age	pre-knowledge
1	4	0	K6	13 y	Lego Spybotics
2	2	2	K7	14 y	no
3	3	0	K7	14 y	Lego Mindstorms
4	5	0	K7	14 y	no
5	4	0	K8	15 y	basic programming concepts

Table 1 - Group setting

The participants were asked to form five groups; planned were groups with four members each, the actual teams had three to five members. This was caused in the composition of the participants group and the social context, like friendships and so on, were the students were already integrated in. Finally the students of the K7 formed one group of three, one of four and one of five members.

Group 1, the youngest participants, faced the challenges every day with a high degree of motivation. They arranged duties very efficiently which resulted in a fast way of completing their tasks. Considering their young age, their achievement was above-average.

Group 2 splitted completely in the first half of the week, some members of different groups shared their efforts for the presentation on the last two days of the week. The reason for the group splitting was different expectations about the course and different working attitudes. Nevertheless this group managed to solve the programming task on the starting day first.

Group 3 with three members developed the most complex Lego programmes and always searched for the perfect solution. They identified the complexity of the maze problem most fast but had problems to develop accurate rule sets theoretically. One of the participants in this group discovered the advantage of the *Maze* software: the Lego software is not able to support such a short and easy solution as the university software enables. Among the prob-

lems with the Lego software the one of keeping exactly straight ahead seemed the most difficult one for the programmed robots. This is, however, not relevant for the university robot because of the different strategies to find the way out which bases on a rough knowledge of the direction and the exact knowledge of the situation.

Figure 6 - Strategy discussion

Group 4 with five members was highly engaged, but confused about their strategies and showed various interests. The results of this group, however, were indeed presented by all of the members and elaborated to the best advantage.

Group 5 worked very autonomously. Furthermore they caught up the course although they had to miss one day because of school organisation issues. With their extended set of the Lego kit they were able to construct a freestyle robot which solved the problem as well as the standard robots of the other groups.

A technical problem occurred when the first programmes were sent to the Lego robots: the Lego USB towers disturbed each other. Thus the groups had to agree in a cooperative way before they were able to send the Lego software to their own RCX.

5.1 Learning styles

Mainly there could be observed three different styles of learning. Learners of the first learning style tried to solve the scenario theoretically and did almost not use the possibility of programming by example. Others worked just the opposite way. They did almost solely use programming by example and refused to think about what problems appeared within the programming process. Similar faults appeared several times.

A third type of learners tried to switch between theoretical phases and programming-by-example phases. In fact this group managed to get the best results, because the phases of programming-by-example were supported by the maze software and the theoretical part of the scenario fostered the discussion about the strategy. To support this learning style was the major aim of the scenario.

After a short introduction the maze software was already usable for the students. The software supported them to think in a programming structure although only a few of them had experiences in programming.

The students were aware of the supporting nature of the maze software, because in this environment they could concentrate on the maze strategies themselves instead of the problems of directing the robots straight away very exactly, for example. Furthermore they re-discussed their strategies in their own groups as it is shown in Figure 6. The students also exchanged their knowledge of good strategies with participants of other groups. Finally they were able to create better sets of rules than they did without the support of the maze software.

5.2 About Motivation

As proposed in [7] from the first day the scenario seemed to be motivating to nearly all of the students. The main interest focused on the RCX and the robots to be built, the question how to get out of a maze was discussed heavily too. Looking at the daily reports and the number of sent emails it is obvious that motivation raised during the project, particularly on the fourth day when the PDAs were introduced.

Although the presentation on the last day was not expected to be reported, some students reported their experiences on Friday. Only on Wednesday the motivation was decreasing because of the more theoretical discussions at this day. The project forum was used several times by the students within their sparetime at home. Several students even sent test report emails to the university. Cooperation between different-aged students could be discovered while working with the PDAs.

The possibility of exchanging sets of rules between the groups was not often used, each group seemed to prefer to find its own solutions. The students might need more time to get used working with shared data bases. Many students suffered because of the end of the project, some bought their own Lego mindstorm kits. To afford this, one student even sold his new playstation, so he can do some "real science" by building own robots.

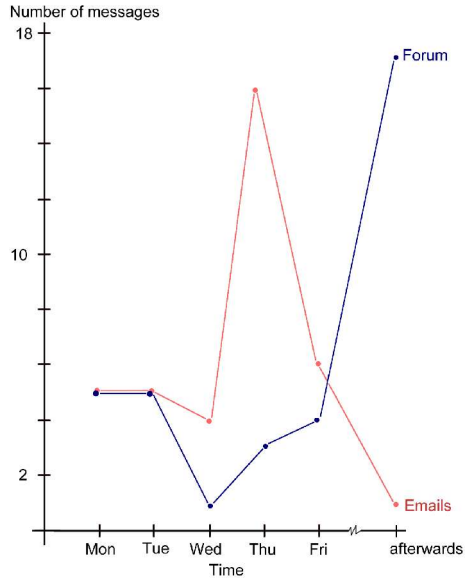


Figure 7 - Statistic of Emails and Forum

The students asked for using the project week forum continuously to have a possibility of exchanging experiences with building and programming new robots at home. Figure 7 shows the usage of the forum after the end of the school project week. Another verification of the students motivation about the project were their wish to design information pinboards in front of the classroom and to write project flyers they handed out on the schoolyard during the summer festival.

Afterall the participants were in high spirits despite of the sophisticated tasks. That can be seen from the final feedback emails and from face-to-face interviews by the teacher in the following week.

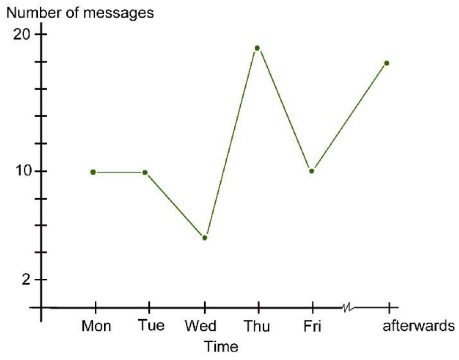


Figure 8 – Statistic of messages

Figure 8 shows the sent messages - both, forum posts and emails integrated. Summarized it can be said that the motivation of the students lasted during the whole project and beyond.

6. Outlook and Future Work

The stability and usability of the system has to be improved, e.g. it might in future releases be possible to send the reporting Emails directly from the software without the usage of a special Email programme, this would also allow to easily add sets of rules to the reports. That might increase the discussion about the sets of rules. Furthermore we plan to create a DEXT (Digital Experimentation Toolkit) from this scenario in the context of the COLDEX project. Those DEXTs should be a set of hardware, in this case some Lego robots and PDAs, a collection of software and explanation of how to use those things in combination. Last but not least we plan to do some more practical weeks in schools in Germany to evaluate more didactical questions already raised in this paper.

Acknowledgements

This research has been supported by grants of the European community with the COLDEX project (IST-2001-32327).

References

[1] Tewissen, F., A. Lingnau, U. Hoppe, G. Mannhaupt., D. Nischk, *Collaborative Writing in a Computer-integrated Classroom for Early Learning*, In Dillenbourg, P., Eurelings, A. & Hakkarainen, K. (editors) Proceedings of the European Conference on Computer-Supported Collaborative Learning (Euro-CSCL 2001). Maastricht, The Netherlands, March 2001, pp. 593-600.

[2] Abelson, H. and A. diSessa, *Turtle Geometry*, MIT Press, Cambridge (USA), 1982.

[3] Papert, S., *Mindstorms: children, computers, and powerful ideas*, Basic Books, New York, 1980, p. 5.

[4] Pinkwart, N., C. Schäfer and U. Hoppe, *Lightweight Extensions of Collaborative Modeling Systems for Synchronous Use on PDAs*, In Milrad, M., U. Hoppe and Kinshuk (eds): Proceedings of IEEE International Workshop on Wireless and Mobile Technologies in Education, WMTE 2002. Los Alamitos (California), USA, 2002, pp. 125-129.

[5] Ferrari, G., *Programming Lego Mindstorms with Java*, Syngress, Rockland (USA), 2002.

[6] Chang, C.-Y., J.-P. Sheu, *Design and Implementation of Ad Hoc Classroom and eSchoolbag Systems for Ubiquitous Learning*, In Milrad, M., U. Hoppe and Kinshuk (eds): Proceedings of IEEE International Workshop on Wireless and Mobile Technologies in Education, WMTE 2002. Los Alamitos (California), USA, pp. 125-129.

[7] Schofield, J. W., *Computers and Classroom Culture*, Cambridge University Press, Cambridge, 1995, p. 192.