 ISSN: 1903-248X

Open Roberta

– a Web Based Approach to Visually Program Real Educational Robots

Markus Ketterl

Dr. rer. nat, Research scientist

Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme IAIS.

Beate Jost

BSc, Research scientist

Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme IAIS.

Thorsten Leimbach

MBA, Research scientist

Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme IAIS.

Reinhard Budde

Dr. rer. nat, Research scientist

Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme IAIS.

Abstrakt

Robotter til brug i undervisningen er et efterspurgt pædagogisk værktøj til hands-on introduktion til moderne informations- og kommunikationsteknologi. "Roberta - Læring med robotter" initiativet fra 2002 har til formål at engagere og motivere piger og drenge til at interessere sig for informationsteknologi og naturvidenskab. Mere end 35.000 børn og unge har deltaget i 600 dokumenterede Roberta forløb. Dermed er Roberta blevet en fast bestanddel i de tyske uddannelselandskab. Men programmering af pædagogiske robotter og vedligeholdelse computer hardware er stadig besværligt for lærere i klasselokalerne – hvilket ofte rapporteres af elever og lærere i Roberta-netværket. Det vigtigste mål det efterfølgende initiativ "Open Roberta" er at overvinde de tekniske udfordringer ved et åbent og fuldt webbaseret programmering miljø for lærere og elever, der kan bruges direkte i webbrowseren hjemme eller i klasseværelset. Den software der præsenteres – "Open Roberta Lab" består af visuelle programmeringsværktøjer til udvikling og tilslutning af reelle pædagogiske robotter uden langtrukne systeminstallationer. Et yderligere teknisk aspekt af papiret er indførelsen af NEPO® meta programmeringssprog som en væsentlig del af Open Roberta Lab.

Abstract

Educational robots have become an often asked educational tool for a hands-on introduction to modern information and communication technology. The ”Roberta - Learning with Robots” initiative aims to engage and motivate girls and boys to take a sustained long-term interest in information technology and natural sciences since the project inception in 2002. With more than 35.000 children and young people in over 600 documented Roberta courses a year – Roberta has become a permanent fixture in the German education landscape and the pedagogical concept, created books, course material and additional tools are being used successfully in other European countries. However, programming educational robots and maintaining complex computer hardware is still a hassle for teachers in the classrooms - as frequently reported from student participants and Roberta network teachers. A main goal of the presented successor initiative Open Roberta is to overcome technical challenges by providing an open, fully web based programming environment for teachers and students alike that can be used directly in the web browser at home or in the classroom. The presented software - the Open Roberta Lab consists of visual programming tools for the development and connection of real educational robots without long-winded system installations, preparation tasks or technology getting in the way. A further technical aspect of the paper is the introduction of the NEPO® meta programming language as an essential part of the Open Roberta Lab.

Introduction and the Roberta approach

Constructing and using robots for programming is an ideal tool to communicate knowledge that is important for understanding technical problems. Researchers and developers have done a competent job since the first inception of classroom turtle programming back in the 1970’s (Feurzeg, 2006), (Papert, 1980). Kelleher and Pausch (Kelleher and Pausch, 2005) have already worked out that Educational robots are highly motivating for boys and girls hence keeping this motivation on a high level for children and for teachers is essential. By designing, constructing, programming and testing mobile robots, children learn the basic concepts of today’s technical systems. But it also pertains to philosophical questions, such as those concerning intelligence and autonomy of artificial systems. By working with robots, kids & co learn to interact with sensors, actuators such as motors, software programs in a very playful way (Kelleher and Pausch, 2005). Educators call for making computer science a cornerstone of the curriculum, even for grade-school kids (Tucker, 2003).

Much earlier than many other coding projects like Girls who code (founded 20121), Ladies learning code (founded 20112) or Women who code (founded 20113), Fraunhofer IAIS started an initiative called Roberta to

engage and motivate especially young girls to take a sustained long-term interest in science, technology, engineering and math (STEM - acronym referring to the academic disciplines of science, technology, engineering and mathematics). Different studies conducted during the project period from 2002 to 2008 reveal that the Roberta concept also promotes active participation of boys in STEM topics (Rethfeld and Schecker, 2006). For more than fourteen years, Roberta especially targets the lack of engineers in general but with a focus on female engineers in Germany and other European countries by raising children interest in technical professions. Heart and soul of this successful project are certified and trained Roberta teachers that spread the word and help newbies as well as advanced users to build and program robotic devices (Bredenfeld and Leimbach, 2010). To participate at a basic Roberta-Teacher-Training no previous knowledge is needed, but mostly the candidates have a didactically or technical background. Two main objectives of the Roberta-Teacher-Training are: Number one is to increase the emphasis of the teacher about gender-sensitive course design. The second one is to provide a hands-on introduction to the robots, the didactic material and the course concept of Roberta (Wiesner-Steiner et al., 2005).

With more than 35.000 participating children and young people in over 600 documented courses each year, Roberta has become a permanent fixture in the German education landscape. But the project has also been successfully exported to other European countries within different funding periods and/or sub projects (see for example ”Roberta goes EU” (Leimbach, 2009)). From crowd sourced generated course and hands-on material, tutorials and edited books up to specific educational robotic kits (based on LEGO) one finds everything needed to not just start to learn coding but also to remain motivated and gain knowledge about ICT, software development and programming, electrical engineering, mechanics, physics and robotics (Wiesner-Steiner et al., 2007). These are the main technical aspects of Roberta. Kids learn that designing, implementing and the construction of technical systems is a creative process that may be rather challenging, but fun and finally also rewarding (Wiesner and Schelhowe, 2004). A further core element of Roberta courses is teamwork. A group of two to three children work together with their robot. This develops and strengthens self confidence by following the System Design Engineering paradigm as mentioned in (Leimbach et al., 2014a). Using robots like LEGO Mindstorms allows children to easily go beyond hands-on in a variety of ways. Resnick names this the constructionist approach. Students can try out things for themselves. M. Resnick: ”They do not simply manipulate physical objects, they construct personally meaningful products” (Resnick, 1998). With the help of educationally and technically adapted robots, even young children can

learn the basics of robot construction and programming in less than two hours. By designing, constructing, programming and testing mobile, autonomous robots, children learn how technical systems are developed and that technology is fun.

During the last years computer platforms and available tools have changed a lot (e.g. ubiquity, price, user expectations). Whereas many of today's educational robotic platforms and programming environments (with hardware connections) have not yet adapted to these new user expectations (e.g. easy to use, no installation hurdles, platform independent, usable on the go or at home) as noted in (Jost et al., 2014). Feedback from Roberta project participants certified a positive overall impression (Leimbach et al., 2014b) but the lack of web based solutions, ready for actual hardware connections have been named frequently as an important factor for the acceptance of the project in the future. Open Roberta, which has been started in 2013, is our open, platform independent technological answer, that bridges the gap between real educational robot hardware and today's web browser. The remainder of the text is organized as follows: The second section gives an overview of related work in the educational robotic domain and sheds light on the advantages of visual coding which is being used in the user interface components of the Open Roberta programming tool called the Open Roberta Lab. The drawbacks reported by the community section gives a resume on lessons learned over the years from our participating network teachers. The main part of the text introduces Open Roberta and the Open Roberta Lab technology, the frequently asked successor of the Roberta project. The technical section of this work is additionally introducing the meta programming language NEPO®, which allows to translate and run code on robotic hardware almost directly from any web browser. The paper concludes with information regarding participation possibilities and future work.

Related work and the benefits of visual coding

Learning to program is difficult for newbies of all ages. In addition to the challenges of learning to form structured solutions to problems and understanding how programs are executed, beginning programmers also have to learn a rigid syntax, command sequences, package and conventions that may have seemingly arbitrary or perhaps confusing names. Today programmers still continue to work with largely textual representations of source code despite the fact that these textual representation cannot easily convey the complex graph-based relationships between pieces of source code across packages, online resources, local folders or foreign libraries and code snippets. Tackling all of these challenges simultaneously is overwhelming and often discouraging not only for beginning programmers.

The Visual Programming Paradigm has been around for several decades - since the early 1960's, researchers have built a number of programming languages and environments with the intention of making programming accessible to a larger number of people. Definitions of visual languages have taken many approaches. Some have taken a formal mathematical approach to defining a visual language (Chang et al., 1986), (Halbert, 1984). One can summarize positive attributes such as a better user experience (Booth and Stumpf, 2013), an easier entrance to programming paradigms (Powers et al., 2006) and they are helpful to prevent typical syntactical errors due to the idea that the syntax of the statements is already implemented into the visual shapes of the statements (Kelleher and Pausch, 2005), (Rosenbaum et al., 2010). Graphical programming and iconic language representations are available since the release of the Sketchpad system from 1963 (Sutherland, 1964). Other early examples can be found in work from (Hirakawa et al., 1988), (Kozen et al., 1987), (Pong and Ng, 1983). In (Boshernitsan and Downes, 2004) the authors further mention the genesis of the Pygmalion system (Canfield-Smith, 1975) as one of the first icon-based programming systems, in which the user created, modified, and linked together small pictorial objects (icons) with defined properties to perform computations. Later examples are PICT (developed by Glinert (Glinert, 1985)) and Prograph from 1983 (Cox and Mulligan, 1985) or Blox in 1986 (visual programming language made up of puzzle-like pieces that fit together)(Salvendy, 1987). We also saw iconic languages in data processing and electronics with LabVIEW since the beginning of the 1980's. In the mid 1990's we had Simulink (Matlab) for data processing. In education, Alice, eToys, Squeak and later Scratch were providing visual block programming in the late 1990's. Also to mention sprite characters for animations were already in use in Atari game consoles or the Commodore C64 in the 1980's and before. By the early 2000's LEGO also shipped the first version of a visual block programming tool with their Mindstorms kits for educational robotics. The similarity of appearance between multiple visual programming interfaces today is the natural outcome of what development, user experience and user interface teams have collectively learned that works well from over 50 years of exploration in that space. In recent months many previously mentioned visual coding elements and concepts are being adapted to online platforms, web tutorials and educational platforms, but they focus mainly on online simulations of interactions and provide no connection to and feedback from actual hardware. We have seen a great number of activities directed at engaging people of all ages into computer programming. For example, The Hour of Code (Wilson, 2015) has organized more than 70 thousand events worldwide, and has been tried by over 95 million people, from elementary students to heads of state. Another prominent example is Code.org (Wilson, 2013). The online platform is a non-profit foundation dedicated to growing computer science education and provides free online tutorials that anyone

can try in a few minutes without deep knowledge. Further projects that are targeting the STEM domain in a playful way and that support self directed learning with online resources are for example: Codeacademy4, Scratch5, Code School6, Programr7, BlueJ8, Codelearn, CodeAvengers, CodeAbbey, Codecakes and many more. But also on the frontier of educational hardware we have seen many new achievements since the inception and success of the LEGO Mindstorm robots. Figure 1 on page § depicts an overview of available educational robotic systems.

Figur 1. Overview Educational Robots[9]

Many of them rely on visual programming concepts to attract and motivate newbies of all age and their success proves them right. The listing includes Out of the box systems, Assembly kits, Crowd funded robots, Humanoid, Self-build and microboards or even 3D printed robots.

Drawbacks reported by the community

Most new students have no particular interest in programming. We have about 20 seconds to engage them before they get bored and wander off to play video games. Every barrier to entry (installing Java, or a slow download, or learning English, or a messy UI) represents a significant loss of audience. Nowadays interdisciplinary teaching is the way teachers promote the school children's technical competence. These teachers are not necessarily math, physics or computer science experts. Quite often they are responsible for different school subjects or are primary school teachers. Certainly some of the interested teachers already know how to interact successfully with different (education) robotic platforms, programming languages and corresponding coding environments but there is a significant amount of teachers without basic knowledge. Issues that

arise quite often are technical preparation tasks like system installations for manifold platforms, network/device setup issues, firewall restrictions and/or huge required software downloads (not allowed) before they even can start teaching. Another reality is that technical staff is limited or oftentimes completely missing at schools (responsible administrator). Moreover there is a mixture of old/new computer systems, operation systems and available tools and devices. Closed source systems and exclusive software vendors are also of strategic long term concern. Many schools have started to focus on using tablet devices convinced to reduce the administrative overhead. But this technical changeover does not play well with the current status of available educational robotic environments as noted in (Jost et al., 2014).

Since the inception of Roberta back in 2002 the project and partners were using different programmable LEGO robots and integrated development environments (IDEs) over the years. Integrated development environments are referring to software programs and runtimes that provide comprehensive facilities to computer programmers for software development on a target computer. Table 1 on page § depicts a time overview of used educational robotic systems, IDEs and supported platforms used by Roberta participants.

Until 2013 the Roberta initiative used only the proprietary LEGO Mindstorms NXT/EV3 IDE software that is available on Windows PC or Mac. Roberta teachers had to install the software and the required license, drivers and updates as needed. They also needed to make sure that computers and the software are functioning properly before each class. Oftentimes this was reported as a complete showstopper and a barrier of entry for many teachers since maintaining software and providing PC support is a job they are often not comfortable with.
Several different environments are being offered in the context today. In most cases they belong to or are designed for a specific robotic system. In (Jost et al., 2014) we took a closer look at available (educational) robotic IDEs and involved programming/coupling merits and pitfalls - Systems under evaluation in this study: Enchanting10, EV31 (The LEGO Group, 2014a), GRAPE (Enderle, 2009), miniBloq (Rahul et al., 2014), ROBO pro (Fischertechnik GmbH, 2014), NXT-G (The LEGO Group, 2014b), RobotC2 (Robomatter, 2014), RoboMind11, Ardublock (Vandevelde et al., 2013).

		Platforms					
DE	**Year**	Windows	Linux	Mac OS X	Web	**Programming**	**Robot**
RIS	2002	X	-	-	-	Event based	RCX
NQC	2004	X	X	X	-	Procedural based on C	RCX
NXT-G	2006	X	-	X	-	Dataflow based on Lab-VIEW	NXT
NXC	2007	X	-	-	X	Procedural based on C	NXT
EV3-G	2013	^x	-	X	-	Dataflow based on Lab-VIEW	EV3, NXT
leJOS	2006	X	X	X	-	Object oriented based on Java	RCX, NXT, EV3
NEPO	2014	X	X	X	X	Procedural based on Java	EV3, others planned

Table 1: History of supported educational robotic hardware, IDEs and supported platforms in Roberta.

These commonly used graphical robot programming environments provide IDEs for multiple computer platforms and operation systems and also present graphical programming alternatives in addition to a pure code based representation to users. Certain robot hardware can be connected and user generated programs can be downloaded from the computer desktop IDE to the attached robot device. To choose the right one out of them for a specific school or learning task is not easy. A lot of questions have to be answered before a decision for a system can be made. Like for example: What computer systems do we use and maintain? Is the software available for our operating systems? Who can do the installation for one or more classrooms? What do we have to pay? How long will this system be

supported? How do we select the right target robot hardware (e.g. costs, stability)? How do we interact with the robots, taking administration restrictions at our school into account? Manifold educational robotic system have been invented or enhanced during the last years. But did they really take into consideration what teachers, the main multipliers, really need - especially in the classroom?

Roberta teacher training is being evaluated regularly in order to improve the program and adjust to changing demands. Results can be found in (Leimbach et al., 2014b). Based on these user surveys and additional community feedback the ideal system infrastructure needs to support the following features: Visual graphic programming support with a switch-over to see the actual text based code representation, fully web based with possibilities to run the infrastructure remotely or independently on a local school server or single computer, connections to robots should be easy (e.g. wifi, Bluetooth) and it should not be restricted to a certain vendor. The technical complexity like compilation and program preparation should run in the background (transparent for the technophiles), price sensitive (or free) in order to allow schools to easily join, test and participate as a individual or as a group (school class). Currently this demand can't be fully satisfied by available tools, frameworks and out of the box development kits (Jost et al., 2014). In summary one can say that it is important to lower the installation complexity by providing generic open tools that build upon concepts and ideas people are already familiar with (e.g. web browser) which can be used across platform borders and devices. The pure focus on a single educational robotic platform is also somewhat outmoded taking the current fluctuation and availability of smart computer toys into account. These are the reasons why we have started Open Roberta, the successor of Roberta and the Open Roberta Lab technology platform as explained in the following sections.

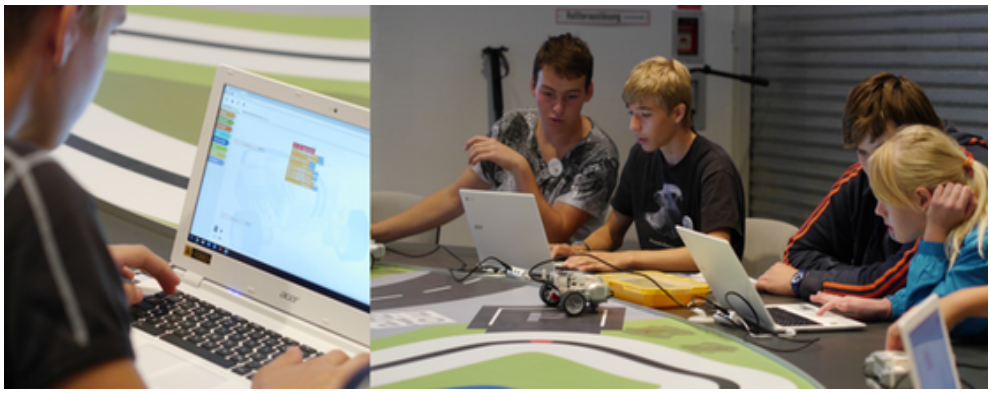
Open Roberta

The Open Roberta12 community project is open to all interested institutions and individuals including commercial providers. It was initiated together with Google13 in 2013. It is centered around the frequently asked technology enhancement of the Roberta - Learning with Robots initiative which targets all issues reported from the Roberta community and aims at bringing educational robotic programming to the next level. The authors of this work are actively involved in the system design, architecture and software development but also steer and maintain teacher qualification, project planing and community engagement. The free to use and open source based Open Roberta Lab is the name of the technical build of the Open Roberta community. The created software is

free available and can be downloaded at GitHub14. This includes technical descriptions and HOWTOs. Updates have been releases frequently since the beginning of the project and we have seen contributions from external individuals and groups.

Open Roberta Lab

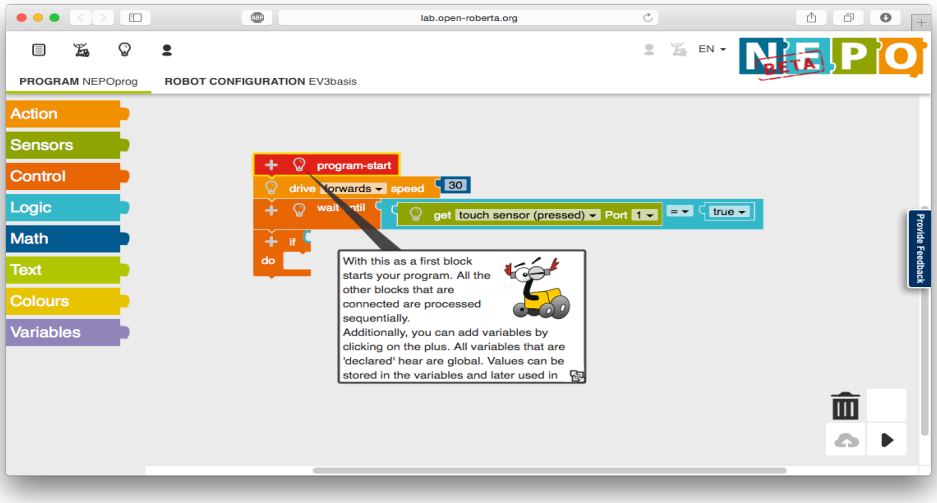
The Open Roberta Lab is the connection to the user. The online programming environment Open Roberta Lab enables children and adolescents to visually program real robot hardware directly from the web browser or by using the build in online robot simulator. As a cloud-based application, the platform can be used without prior installation of specific software but runs directly in any modern browser, independently of operating system and device. This enables beginners to seamlessly start coding without deeper technical knowledge. The program compilation and machine code preparation is handled completely on the server side (see section 6 on page § for further explanation). It is important to mention that the server can be installed locally or in a closed classroom environment without any dependency to outside internet network connection if needed.



Figur 2. The Open Roberta Lab used by kids in a classroom

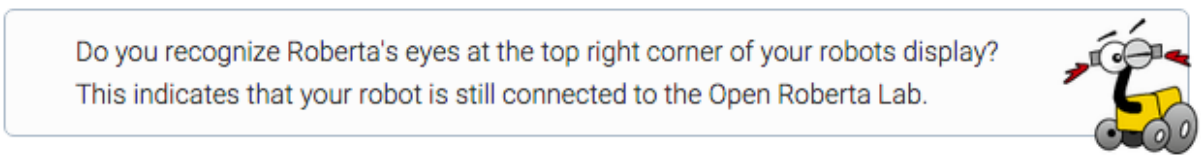
The user facing applications running in the browser are implementing lessons learned from our Roberta network and follow design recommendations from various groups and researchers (e.g. (Teague, 2002), (Kelleher and Pausch, 2005), (Praßl, 2006), (Zimmermann and Sprung, 2008), (Sprung et al., 2010)) regarding technology (HTML and JavaScript), look and feel (Web 2.0), wording (easy language with additional help texts and pictures) as well as features (e.g. user awareness, program sharing, social coding) tailored to our intended target group. The user interface incorporates programming technology based on the frequently used Blockly project (Marron et al., 2012). We have chosen Blockly to not have to do everything ourself and benefit from new features

on the user interface side. Our development team is in ongoing discussion for the best possibility to return parts of our extensions (as described in sections 6, 7) back to the Blocky members. First available Open Roberta Lab software releases enables its users to program LEGO Mindstorms EV3 robots in the browser. A variety of different graphical programming blocks extend the feature set of Blockly by providing mechanisms to interact with motors, sensors and the core of LEGO robots, the EV3-Brick. Upcoming software releases are aiming at a broader online programming support for additional educational hardware (like robots, toys, etc.). Figure 3 on page § depicts the web part of the software running in the browser.



Figur 3. Open Roberta Lab in the web browser

The Lab provides platform features like user login, program saving/sharing and easy hardware pairing over wifi or if no network is available with a USB/Bluetooth connection. If no hardware is around there is also the option of a basic robot hardware simulation that includes a sensor/obstacle emulation. Before one can start programming and running programs on a robot, he or she has to connect the hardware to the Open Roberta Lab. Pairing is done with a session code shown on the robot display (see figure 4 on page §) that needs to be entered into a dialog in the Open Roberta Lab. As previously mentioned our first fully supported robot device is the EV3 (part of popular LEGO Mindstorm set).



Figur 4. EV3 Menu with 'Hello Roberta' and active server connection

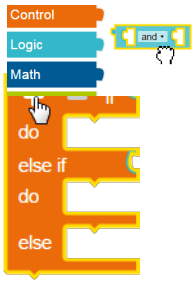
Introduction to programming with NEPO

NEPO is how we call our graphical programming language along with its hardware connection layer. It is an open source meta programming language that can be used in the Open Roberta Lab. NEPO translates to 'New Easy Online Programming'. The visual appearance and usability ideas follow well known concepts implemented by tools like Scratch (Resnick et al., 2009) or Blockly (e.g. (Marron et al., 2012)). NEPO eliminates the syntax and logic battle and lets kids focus on the logic behind conditionals, loops, variables and other core concepts without worrying about unbalanced parenthesis or missing semicolons. A major extension is the creation of visual blocks that map to features that robotic hardware really is capable of. The Open Roberta language supports event based procedural and object-oriented programming by assembling graphical objects. These object shapes only allow a certain connections of elements and prevent typical syntax errors. Different levels of experience hide/show features for newbies or advanced users are available. This includes also user centered keywords and an easy language close to natural speaking.
The most prominent extension and difference to other visual programming solutions that target educational robotics, coding initiatives or browser programming environments is that NEPO does not stop in the browser. The NEPO code will be compiled to a code that can be directly run on the target robot. A NEPO block always represents and encapsulates a certain robot functionality. Features can easily be recognized through the associated block category, for example ≫sensors≪. Blocks are being interconnected and will be executed by the robot according to their order. Only blocks that are successfully connected are executed in the program. Depending on the mode of a block, the number of the connectors and possibilities may vary. Some attributes of NEPO blocks exemplified below:

- Start Programming – Each program has a predefined starting point. This is the red "program-start" block, which is always available and indelible.

- Type Safety – Generally all parameters and variables belong to a specific data type. Therefore the user has to declare the type before the variable is used. Further the user can identify the type by the color on the input and output connections.

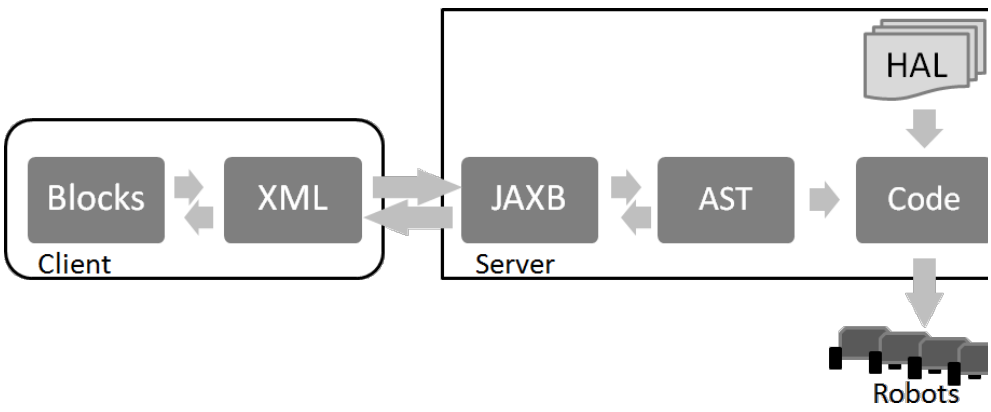
- Color matching toolbox – recognize semantics by colors.



- One Click Expandable – Easily expand blocks if needed, e.g. extend the if statement by applying several else if statements.

Behind NEPO - the complete workflow from block to code towards hardware

The beginning of the whole workflow is the graphical program defined by the block sequence. The graphical blocks are JavaScript objects represented in the browser DOM. Initially thDr.ey are based on a predefined XML representation schema. Figure 5 depicts the pathway from a client block over XML to the hardware abstraction part that translates the program sequence to code suitable for the target robot.



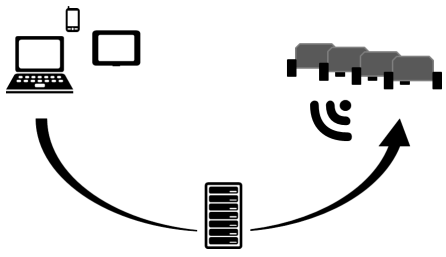
Figur 5. NEPO workflow: From client blocks to machine code

For further processing the created program is stored back into XML on client side and send to the server if needed (e.g. to check against the robot's hardware description or to start the program on the robot). Once the XML is available on server side a tree of Java objects is created by unmarshalling the XML based on JAXB (Java Architecture for XML Binding). Therefore a well defined XML schema for the representation has been developed. The unmarshalling procedure can be reverted by marshalling Java objects back

to XML. A second tree, the Abstract Syntax Tree (AST) is created to add additional information for the final code generation. As an example, class visitors are added to each node of the AST to enable transformations to JAXB by gathering the required class attributes or by checking semantics (following the visitor pattern). The last step is the code generation itself. This transformation can either be source code or compiled machine byte code. For the EV3 system Java source code is generated out of the AST. Depending on the type of robot it is also possible to apply a special Hardware Abstraction Layer (HAL). With the HAL useful available libraries can be bound to access the robots hardware features. In order to generate executable EV3 programs the leJOS libraries15 are accessed from the HAL. At the end of the workflow the generated code is compiled on server side if necessary. The code is now ready to be executed on the robot. To receive and execute code, the operating system of the robot has to provide the capability to connect to the Open Roberta server to receive the program or the binary. Different possibilities are mentioned in the next subsection.

leJOS - one operating system ready to connect to Open Roberta

The operating system of the robot is generally the bottleneck of the workflow. Ideally the robot is capable to connect via wifi to the internet, thus to the Open Roberta server. This way of connection makes it possible to easily exchange information, programs and commands between the robot and the server. The stock EV3 firmware does not easily allow modifications without flashing/overwriting the system. The alternative open source leJOS16 operation system provides the required combination of libraries and firmware to fit to our needs. Some of the educational robot systems may not have access to the internet. For those devices it is still possible to establish a connection to the Open Roberta server via Bluetooth, USB, near field communication (NFC) or even audio. Figure 6 on page § depicts the current used connection between the Open Roberta server and an EV3 robot. Via an unique token, generated on the EV3 system, each robot can be identified and allocated to the right client (user) session.



Figur 6. Open Roberta: Indirect connection from client to robot via the server

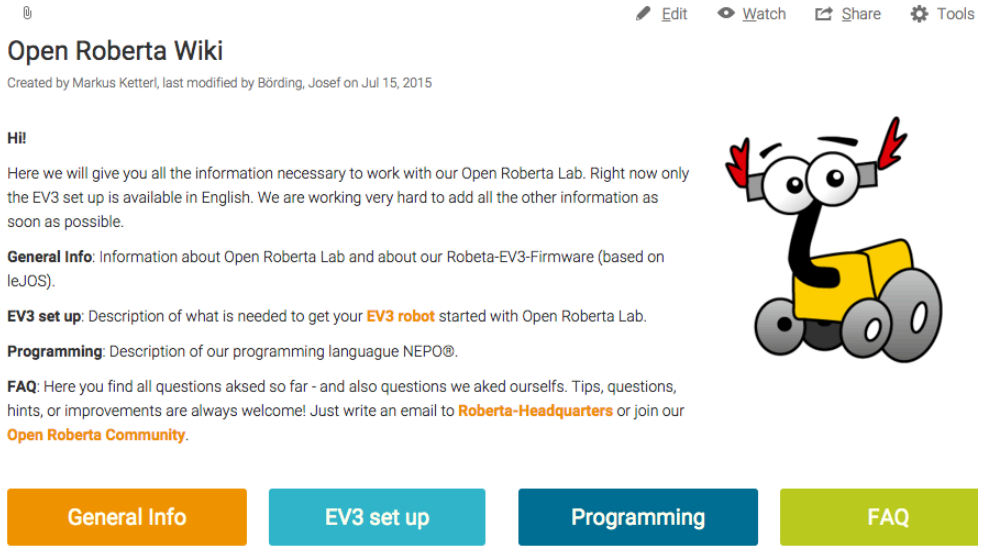
For the EV3 robot a slightly adapted leJOS operating system was created. Once the user has established a connection, programs and other commands e.g. firmware updates can than be exchanged between the Open Roberta server and the EV3 robot until the connection is stopped again by the user. Figure 4 on page § depicts the Open Roberta icon on the right hand side. By selecting this icon the robot tries to connect to the Open Roberta server via HTTP. At the same time it generates a token which can easily be confirmed in the web browser.

7.2 One concept useful for other robot systems and hardware

The previous section tried to explain that the underlying concept of the Open Roberta architecture is open for code generation for many different robot systems. Where common features can be reused, e.g. graphical blocks for similar robot kinematics, the whole workflow can be adopted to generate the specialized executable code for any other system. As an exemplifying further idea from a foreign domain we've created an extension of the building blocks and visual workflow capabilities of NEPO that can be used by an automatic camera tracking systems to define special target following behavior rules as described in (Wulff et al., 2015).

How to get involved

The created technology and its concepts are free to use for anyone and are available as open source contribution17 released under the Apache License, version 2. In a first step, the development team at the Fraunhofer Institute for Intelligent Analysis and Information Systems (IAIS) reached out to teachers, IT and education experts within the partnering Roberta network as well as to universities and their students to involve them in the development work. In the ongoing second step, the open source community has been opened to all interested parties and programmers. By doing that Roberta still follows its main mission namely the encouragement of female newbies in order to help them becoming role models for the next generation of programming experts.



Figur 7. Open Roberta Wiki

The programming platform Open Roberta Lab18 is online available for tests and for usage feedback reports. Additional information, tutorials, tips and developer instructions are available in the user wiki19 as shown in figure 7.

Conclusion and Future Directions

The article presented Open Roberta, the successor project which arose from the Roberta initiative. Essential parts of Open Roberta are the meta-programming language NEPO as well as the cloud based Open Roberta Lab. The OR Lab is intended to be an open source online programming platform for pairing with different educational robotic systems and additional hardware.

Our current work focuses on an extension of the abstraction layer to further support additional intelligent (educational) toys and gadgets but also to open the system for other use cases. Actually NEPO is biased towards a conventional imperative language. From that ”compatible” Java, Python or C code is generated. But there are other great paradigms that fit well to robot programming. One example from the field of real-time languages are synchronous programming languages (like Esterel, Lustre). They support parallel programming without threading, priority based scheduling among other. Code generation is reminiscent of creating hardware circuits. Adding constructs of these languages as blocks to NEPO is an demanding exercise in compiler construction. If you are coding a program it is very helpful to have dubbing functionality.

Within Open Roberta we are looking for a functionality which shows the

developers which (NEPO) programming block is currently activated and running on the robot. This functionality needs an established and stable connection between the robot and the Open Roberta Lab. Another currently open field and option is social coding, user awareness and user collaboration support in the platform (features that can also be switched off by a teacher). We guess that this extensions might address sociological barriers (including people not seeing the relevance of programming or perceiving computer science as being a socially isolating career path) that are harder to identify than technical difficulties.

Acknowledgment

Besides the members of the Open Roberta community and the Roberta teacher network we would like to thank Google for their generous project support.

[1]https://en.wikipedia.org/wiki/Girls_Who_Code

[2]http://ladieslearningcode.com/about/

[3]https://www.linkedin.com/company/women-who-code

[4]https://www.codecademy.com

[5]https://scratch.mit.edu

[6]https://www.codeschool.com

[7]http://www.programmr.com

[8]http://www.bluej.org

[9]https://educational-robots.zeef.com/roberta.roboter0

[10]http://enchanting.robotclub.ab.ca/tiki-index.php

[11]http://www.robomind.net/en/

[12]http://www.open-roberta.org

[13]http://9to5google.com/2014/11/04/googles-open-roberta-project-is-teaching-germanys-youth-how-to-program-robots/

[14]http://code.open-roberta.org

[15]http://www.lejos.org

[16]http://sourceforge.net/p/lejos/wiki/Home/

[17]http://code.open-roberta.org

[18]http://lab.open-roberta.org

[19]https://wiki.open-roberta.org

References

Booth, T. and Stumpf, S. (2013). End-user experiences of visual and textual programming environments for arduino. In Dittrich, Y., Burnett, M., Mørch, A., and Redmiles, D., editors, End-User Development, volume 7897 of Lecture Notes in Computer Science, pages 25–39. Springer Berlin Heidelberg.

Boshernitsan, M. and Downes, M. S. (2004). Visual programming languages: a survey. Technical Report UCB/CSD-04-1368, EECS Department, University of California, Berkeley.

Bredenfeld, A. and Leimbach, T. (2010). The Roberta Initiative. In in Workshop Proceedings of Intl. Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR 2010, pages 558–567.

Canfield-Smith, D. (1975). Pygmalion: A Creative Programming Environment. PhD thesis, Stanford, CA, USA. AAI7525608.

Chang, S., Ichikawa, T., and Ligomenides, P. (1986). Visual languages. Management and information systems. Plenum Press.

Cox, P. T. and Mulligan, I. J. (1985). Compiling the graphical functional language prograph. In Proceedings of the 1985 ACM SIGSMALL Symposium on Small Systems, SIGSMALL '85, pages 34–41, New York, NY, USA. ACM.

Enderle, S. (2009). The qfix Robot Kits. In Gottscheber, Achim and Enderle, Stefan and Obdrzalek, D., editor, Research and Education in Robotics — EUROBOT 2008, pages 84–95. Springer Berlin Heidelberg.

Feurzeg, W. (2006). Educational technology at bbn. IEEE Annals of the History of Computing, 28(1):18–31.

Fischertechnik GmbH (2014). ROBO pro.

Glinert, E. P. (1985). Pict: Experiments in the Design of Interactive, Graphical Programming Environments (Iconic, Programming Languages). PhD thesis. AAI8508052.

Halbert, D. C. (1984). Programming by Example. PhD thesis. AAI8512843.

Hirakawa, M., Iwata, S., Tahara, Y., Tanaka, M., and Ichikawa, T. (1988). A framework for construction of icon systems. In Visual Languages, 1988., IEEE Workshop on, pages 70–77.

Jost, B., Ketterl, M., Budde, R., and Leimbach, T. (2014). Graphical programming environments for educational robots: Open roberta - yet another one? In Multimedia (ISM), 2014 IEEE International Symposium on Multimedia, pages 381–386.

Kelleher, C. and Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. ACM Comput. Surv., 37(2):83–137.

Kozen, D., Field, J., Chen, W., and Teitelbaum, T. (1987). ALEX : an alexical programming language. Technical Report TR-87-0835, Cornell University (Ithaca, NY US).

Leimbach, T. (2009). Roberta goes EU. Technical report, Fraunhofer IAIS.

Leimbach, T., Jost, B., Petersen, U., Börding, J., and Härtig, S. (2014a). Roberta-Grundlagenband EV3. Fraunhofer Verlag, Stuttgart, Germany.

Leimbach, T., Jost, B., Petersen, U., Börding, J., and Härtig, S. (2014b). Roberta-Grundlagenband EV3. Fraunhofer Verlag, Stuttgart, Germany.

Marron, A., Weiss, G., and Wiener, G. (2012). A decentralized approach for programming interactive applications with javascript and blockly. In Proceedings of the 2Nd Edition on Programming Systems, Languages and Applications Based on Actors, Agents, and Decentralized Control Abstractions, AGERE! 2012, pages 59–70, New York, NY, USA. ACM.

Papert, S. (1980). Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, Inc., New York, NY, USA.

Pong, M. C. and Ng, N. (1983). Pigs—a system for programming with interactive graphical support. Software: Practice and Experience, 13(9):847–855.

Powers, K., Gross, P., Cooper, S., McNally, M., Goldman, K. J., Proulx, V., and Carlisle, M. (2006). Tools for teaching introductory programming: What works? SIGCSE Bull., 38(1):560–561.

Praßl, M. (2006). FEMUIS - Frauen und User Interfaces. Diplomathesis, FH JOANNEUM Gesellschaft.

Rahul, R., Whitchurch, A., and Rao, M. (2014). An open source graphical robot programming environment in introductory programming curriculum for undergraduates. In IEEE International Conference on Innovation and Technology in Education (MITE), pages 96–100.

Resnick, M. (1998). Turtles, termites, and traffic jams - explorations in massively parallel microworlds. MIT Press.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y. (2009). Scratch: Programming for all. Commun. ACM, 52(11):60–67.

Rethfeld, J. and Schecker, H. (2006). Evaluationsergebnisse zum projekt roberta - mädchen erobern roboter. In Lehren und Lernen mit neuen Medien., pages 114–116.

Robomatter, I. (2014). ROBOTC for LEGO MINDSTORMS.

Rosenbaum, E., Eastmond, E., and Mellis, D. (2010). Empowering programmability for tangibles. In Proceedings of the Fourth International Conference on Tangible, Embedded, and Embodied Interaction, TEI '10, pages 357–360, New York, NY, USA. ACM.

Salvendy, G., editor (1987). Cognitive Engeineering in the Design of Human-Computer Interaction and Expert Systems, Proceedings of the Second International Conference on Human-Computer Interaction, Honolulu, Hawaii, August 10-14, 1987, Volume 2. Elsevier.

Sprung, G., Zimmermann, L., Strohmaier, R., and Nischelwitzer, A. (2010). Touch::tell:it tendencies and consequences for the usage of educational programming languages. In EDULEARN10 Proceedings, 2nd International Conference on Education and New Learning Technologies, pages 2445–2451. IATED.

Sutherland, I. E. (1964). Sketch pad a man-machine graphical communication system. In Proceedings of the SHARE Design Automation Workshop, DAC '64, pages 6.329–6.346, New York, NY, USA. ACM.

Teague, J. (2002). Women in computing: What brings them to it, what keeps them in it? SIGCSE Bull., 34(2):147–158.

The LEGO Group (2014a). LEGO® MINDSTORMS® Education EV3-Software.

The LEGO Group (2014b). LEGO® MINDSTORMS® Education NXT Software 2.1.6 (inkl. Messwerterfassung).

Tucker, A. (2003). A model curriculum for k–12 computer science: Final report of the acm k–12 task force curriculum committee. Technical report, New York, NY, USA. ACM Order No.: 104043.

Vandevelde, C., Saldien, J., Ciocci, C., and Vanderborght, B. (2013). Overview of technologies for building robots in the classroom. In International Conference on Robotics in Education, Proceedings, pages 122–130.

Wiesner, H. and Schelhowe, H. (2004). Handlungsträgerschaft von robotern: Robotik zur förderung von chancengleichheit im schulischen bildungsbereich. In Fachzeitschrift für Mentoring und Gender Mainstreaming in Technik und Naturwissenschaften. ADA-MENTORING.

Wiesner-Steiner, A., Schelhowe, H., and Wiesner, H. (2007). The didactical potential of robotics for education with digital media. IJICTE, 3(1):36–44.

Wiesner-Steiner, A., Wiesner, H., and Schelhowe, H. (2005). Technik als didaktischer Akteur. Robotik zur Förderung des Technikinteresses. Hochschulinnovation: Gender-Initiativen in der Technik.

Wilson, C. (2013). What's up next for code.org? Computer, 46(8):95–97.

Wilson, C. (2015). Hour of code: Maryland, washington and san francisco move to support computer science. ACM Inroads, 6(3):14–14.

Wulff, B., Wilson, A., Jost, B., and Ketterl, M. (2015). An adopter-centric api and visual programming interface for the definition of strategies for automated camera tracking. In Multimedia (ISM), 2015 IEEE

International Symposium on Multimedia, pages 587–592, Miami, Florida, USA.

Zimmermann, L. and Sprung, G. (2008). Technology is female: How girls can be motivated to learn programming and take up technical studies through adaptations of the curriculum, changes in didactics, and optimized interface design. In iNEER (Ed.), editor, International Conference on Engineering Education (ICEE).