3

Teaching practical engineering for freshman students using the RWTH – Mindstorms NXT toolbox for MATLAB

Alexander Behrens, Linus Atorf and Til Aach
Institute of Imaging & Computer Vision, RWTH Aachen University, 52056 Aachen Germany

1. Introduction

As a powerful programming and simulation tool, MATLAB®(The MathWorks, 1994) becomes more and more important in today's curricula of electrical engineering. Its intuitive way to map matrices and vector algebra from mathematical formulas to program algorithms allows an easy and fast introduction to programming basics, especially for beginners. Furthermore, the manifold functionalities of MATLAB enable the user to abstract and solve complex engineering tasks and mathematical problems, which become important when teaching core electrical engineering and computing concepts. Thus, MATLAB is often used as a valuable tool to develop demo applications and address real–world problems in freshman courses (Devens, 1999; Director et al., 1995). Many examples are given in the literature, such as introduction courses to digital signal processing (DSP) (Anderson et al., 1996; McClellan & Rosenthal, 2002; McClellan et al., 1997; 2002; Saint-Nom & Jacoby, 2005; Sturm & Gibson, 2005; Vicente et al., 2007), applied automatic controls (Narayanan, 2005), computer programming (Azemi & Pauley, 2008) as well as to graphical user interface (GUI) design (Lee et al., 2005). Since MATLAB is also widely used in industry for algorithm and simulation development, the acquisition of advanced programming skills in MATLAB becomes important in engineering education.

Besides the establishment of project–based laboratories using interactive software tools, many practical projects showed that robotics can be used in an efficient way to teach and motivate students (Azlan et al., 2007; Christensen et al., 2004; Cliburn, 2006; Dagdilelis et al., 2005; Klassner & Anderson, 2003; Lau et al., 2001; Maher et al., 2005; Michaud, 2007; Mota, 2007; Neilsen, 2006; Patterson-McNeill & Binkerd, 2001; Pomalaza-Raez & Groff, 2003; Sharad, 2007; Vallim et al., 2006; Williams, 2003; Ye et al., 2007). Thus, they overcome the problem of dropping motivation during traditional and more theoretical lectures of core electrical engineering and computing concepts. Studies showed that a pedagogical approach that places students in situations where they "feel like engineers" is likely to enhance student motivation best (Vallim et al., 2006).

Driven by both above teaching aspects, the combination of MATLAB and robots is used for a new first–semester learning module, established in 2007–2008 in the curriculum in Electrical Engineering and Information Technology at RWTH Aachen University, Aachen, Germany (Behrens & Aach, 2008; Behrens et al., 2008; 2010). In this laboratory for freshman students,

termed "MATLAB meets LEGO Mindstorms", digital signal processing is combined with computer programming and problem–oriented engineering. It gives the students their first insights into practical methods and basic engineering concepts and helps them to apply their knowledge to other challenges later on in their studies. After only two months of lectures and seminars, the students participate in this mandatory full–time eight–day project. Working together in teams, the students enhance their first MATLAB programming skills and apply mathematical foundations, which are taught in the affiliated lecture "Mathematical Methods of Electrical Engineering". To avoid an exclusive focus on programming, real–world problems and practical tasks are emulated by using LEGO®Mindstorms®NXT robots (The LEGO Group, 2006c). Besides six mandatory exercises, the students are given time to create their own applications and to define creative robot tasks. The students collaborate in teams of two and four, and are involved in discussions and presentations. For a high student motivation and an increased learning effort during the project, good supervision and a simple and intuitive interface between MATLAB and the Mindstorms robots are essential to ensure fast prototyping and program development. Based on the objective to teach MATLAB fundamentals to beginners and to realize innovative robot applications in a short period of time, the MATLAB ↔ robot interface must also provide high usability and a well structured documentation. Therefore the new "RWTH – Mindstorms NXT Toolbox" for MATLAB has been created and applied in the laboratory. Furthermore it is published as free and open source software (RWTH Aachen University, Germany, 2008), and accessible for third party projects.

1.1 Previous Work

In previous work, G. Gutt (2006) provided a first remote control MATLAB ↔ Mindstorms interface, which uses additional communication software to establish a Bluetooth connection between MATLAB and Mindstorms NXT robots. Since the received Bluetooth packets are always buffered in files, this implementation does not provide a direct and intuitive computer–robot communication suitable for first–semester projects. Another implementation using Simulink®, complex simulation models, and advanced control engineering concepts was developed by T. Chikamasa (2006). This toolbox provides a simulation mode and produces embedded code, which does not allow the program code to be debugged step–wise. Also, it focuses on advanced control theory and requires an initial familiarity with Simulink, which can hardly be expected of freshman students.

Thus, no satisfying software interface between MATLAB and LEGO Mindstorms NXT fulfilling the requirements of a direct and powerful interface was available. Therefore the new RWTH – Mindstorms NXT Toolbox, which is fully integrated into the MATLAB environment and maps the complete functionality of the Mindstorms hardware to the user, was developed. After a period of only four months development time, the first toolbox release and the practical exercises used in the first semester term of the project "MATLAB meets LEGO Mindstorms" were finalized by a core team of eight supervisors in 2007. Since then the toolbox has been consequently improved and extended.

2. LEGO Mindstorms NXT

LEGO Mindstorms NXT is a low–cost and widely used toy robot kit. It is available as a commercial hardware package for private use, as well as an education set (The LEGO Group, 2007). The NXT education set includes a programmable NXT intelligent brick with an integrated USB and Bluetooth communication interface, four different types of sensors (touch, sound, light, and ultrasonic distance sensor), and three servo motors, as illustrated in Fig. 1.

Fig. 1. LEGO Mindstorms NXT hardware of the standard education kit: Five sensors (light, sound, ultrasonic, and two touch sensors), three servo motors, and the programmable NXT intelligent brick.

Furthermore several different plastic LEGO bricks are provided for construction. The NXT brick contains an Atmel®32–bit ARM processor running at 48 MHz, 256 KB flash and 64 KB RAM memory. Its monochrome graphical LCD display has a resolution of 100×64 pixels. In total four sensor input ports supporting both a digital and analog interface, as well as three output ports for motors or lamps are available.
In addition to the sensors included in the standard Mindstorms kit, many other sensors are provided by third party vendors. HiTechnic (2001) offers a wide range of additional analog and digital NXT sensors, like e.g. compass, acceleration and infrared sensors. Supported by LEGO, the sensors are integrated in the common plastic shells and designed like the standard NXT sensors. Furthermore CODATEX (2007) distributes an RFID sensor and individual ID–tag transponders. Mindsensors.com (2005) offers advanced sensor kits and controller interfaces for Mindstorms, which are not encapsulated as HiTechnic sensors. In Table 1 a short overview of the most common NXT sensors is given.
Beyond the variety of commercially available sensors, LEGO provides a hardware developer kit specification (The LEGO Group, 2006b, Hardware Developer Kit) which can be used for individual sensor development. Examples of customized sensors are given by Gasperi et al. (2007).

2.1 NXT Programming Languages

To control LEGO Mindstorms NXT robots, a wide range of programming interfaces is available in the literature. These include compiler–based programming languages (C, C++, Java, .NET), interpreted languages (MATLAB, Python, Perl), as well as graphically oriented tools and simulation software (LabVIEW, RoboLab, Simulink). Despite the high variety of available packages, all programming concepts can mainly be categorized by two properties. The first one is determined by the type of program execution.

Embedded code: In this scenario, programs are usually developed on a computer using a programming development software first, e.g. NXT–G (National Instruments Corporation,

Sensor	Analog/ Digital	Vendor	Toolbox support (v4.03)
Touch	A	LEGO	yes
Light	A	LEGO	yes
Sound	A	LEGO	yes
Ultrasonic	D	LEGO	yes
Color	A	LEGO	no
RFID	D	CODATEX	yes
Compass	D	HiTechnic	yes
Accelerometer	D	HiTechnic	yes
Gyro	A	HiTechnic	yes
Color	D	HiTechnic	yes
Color V2	D	HiTechnic	no
IRSeeker	D	HiTechnic	yes
IRSeeker V2	D	HiTechnic	no
EOPD	D	HiTechnic	no
IRLink	D	HiTechnic	no
IRReceiver	D	HiTechnic	no
NXTCam	D	mindsensors.com	no
Sony PlayStation Controller	D	mindsensors.com	no

Table 1. Overview of most common LEGO Mindstorms NXT sensors.

2006), NXC (Hanson, 2006), ROBOTC (Robotics Academy, 2006), leJOS (Solorzano, 2007), and then translated into NXT bytecode. After downloading the code onto the NXT, no external computer is required anymore. The program code is always executed on the NXT hardware platform.
The NXT's firmware usually provides a virtual machine and executes bytecode while taking care of low–level hardware functionality. In some cases, the embedded program code can also be plain text and is executed by an interpreter, e.g. pbLua (Hempel, 2007). Due to direct hardware access to sensors and motors with minimal latency, real–time applications are possible. On the other hand, the program complexity is restricted by the limited resources of the NXT, such as memory, CPU speed, and display resolution.

Remote control: Programs using a remote control concept typically run on a computer or other host devices, e.g. a mobile phone. Commands specified in the LEGO Mindstorms NXT communication protocol (The LEGO Group, 2006a, Bluetooth Developer Kit) are sent to the NXT via Bluetooth or USB connections. These commands are then interpreted and executed by the firmware. In a similar way sensor and motor data can be retrieved. Since the actual robot control programs do not run on the NXT platform, they can utilize all resources, devices and technologies of their host systems. However, they are limited by the set of available remote commands and by the transfer time delay, which often impedes the realization of true real–time applications.

The second way to categorize Mindstorms interfaces is specified by the required NXT firmware. While some implementations are adapted to the original LEGO NXT firmware, other pro-

gramming languages need a specific or customized firmware on the NXT for program execution.
NXT firmware: The standard configuration of the NXT includes the LEGO Mindstorms NXT firmware, maintained as open source code by LEGO. Its main purpose is to execute bytecode generated by LEGO's standard programming language, NXT–G. This firmware also supports the NXT communication protocol to execute so–called direct commands, remotely sent by other devices or computers.
Besides the official LEGO release, some firmware modifications are available, keeping full compatibility to compiled NXT–G binaries and to direct commands. The most prominent example is John Hansen's enhanced firmware, which fixes known bugs and adds advanced functionality. It comes with the Bricx Command Center (Hanson, 2002) development environment for the programming language NXC (Hanson, 2006).

Custom firmware: In the literature a variety of custom firmware versions is available. Some are based on the original release by LEGO, whereas others provide alternative firmware implementations. The custom firmware usually provides a virtual machine that can execute bytecode or plain text for a certain programming language. Prominent examples are leJOS (Solorzano, 2007) for Java programs, the Lua interpreter pbLua (Hempel, 2007), NXTalk (HPI Software Architecture Group, 2006) as a Smalltalk implementation, and ECRobot (Embedded Coder Robot) for Simulink (Chikamasa, 2006).
Another purpose of custom firmware is the execution of machine code directly on the ARM CPU, or the integration of specialized programs straight into the firmware. One implementation providing such capabilities is given by nxtOSEK (Chikamasa, 2007). Other efforts provide toolchains or compilers for custom firmware development, such as NXTGCC (Pedersen, 2006) or the IAR Embedded Workbench (IAR SYSTEMS, 2009) for LEGO Mindstorms NXT.

The most common interfaces are listed in Table 2. Note that the list is not exhaustive at all.

Name	Language Type	Standard Firmware	Embedded/ Remote
leJOS NXJ	Java	no	embedded
iCommand	Java	no	remote
NXC	C–like	yes	embedded
ROBOTC	C–like	no	embedded
NXT++	C++	yes	remote
Mindsqualls	.NET	yes	remote
MS Robotics Studio	.NET	yes	remote
NXT_Python	Python	yes	remote
LEGO::NXT	Perl	yes	remote
NXT-G	LabVIEW–like	yes	embedded
RoboLab	LabVIEW–like	no	embedded
ECRobot	Simulink	no	embedded
RWTH – Mindstorms NXT Toolbox	MATLAB	yes	remote

Table 2. Most common programming languages for LEGO Mindstorms NXT.

3. RWTH – Mindstorms NXT Toolbox for MATLAB

Since the target audience of the RWTH – Mindstorms NXT Toolbox for MATLAB are freshman students without any or only basic programming skills, the main objective of the toolbox is a direct and intuitive usability of the control interface. Beginners must be enabled to start with simple high–level commands to obtain results rapidly, while intermediate users can use more advanced functions. Using MATLAB as development tool, essential key features such as easy visual debugging by step–by–step execution, 2D and 3D plotting capabilities, a GUI designer, and additional toolboxes are directly provided. Furthermore advanced algorithms and technologies, as well as external hardware such as webcams can easily be integrated into individual robotic projects. However, an intuitive and consistent development environment will only be preserved, if the algorithms are entirely developed in MATLAB code. Thus, the usage of additional third–party software is avoided. As an exception, external USB and Bluetooth hardware drivers are used.
In addition to good usability, a well–written documentation is essential, especially for beginners. Apart from a list of functions and appropriate descriptions, genuine algorithmic examples are provided. Tutorials and step–by–step guides integrated in the toolbox help students to get started and extend their knowledge. Since software which is compatible to different operation systems can easily be distributed in bigger education projects, the framework is designed in MATLAB to run on Windows, Mac OS, and on Linux platforms. Furthermore low–level implementation details for hardware interaction (such as certain drivers or external libraries) are masked by a universal abstraction layer. Thus, the users are able to utilize both Bluetooth and USB connections to the NXT promptly without making any modifications to their program code.
Using the original LEGO NXT firmware the toolbox functionality is mainly limited to the MATLAB $\leftrightarrow$ NXT communication specified by the Mindstorms communication protocol. However, the usage of the original firmware allows a lower toolbox development effort, and a less complex initialization procedure, since the NXT does not have to be flashed again with a custom firmware.

3.1 Software Design

The RWTH – Mindstorms NXT Toolbox is a framework to control NXT robots remotely. Since MATLAB is an interpreted language, the use of embedded code is omitted. This is obvious, because the development of a full MATLAB runtime and a virtual machine or interpreter for the NXT platform with only 256 KB of available flash memory and 64 KB RAM is unfeasible. Thus, the user program is executed by the host computer, which highly outperforms the NXT's computational resources, especially regarding CPU speed. However, the characteristics of the established communication channel between NXT and computer, i.e. limited bandwidth and time delay, impede real–time control loops for wireless robots. Also, the complete functionality of the NXT is not immediately available via the specified remote commands. But aside from this technical point of view, the remote concept still combines a powerful programming environment with an adequate way for beginners to control robots, analyze data, and get results very quickly.
Based on this concept 117 MATLAB functions are provided by the toolbox (version 4.03), organized in a multi–layer software architecture. A global overview of these command layers and the hardware interaction is shown in Fig. 2.
Using individual motor and sensor settings, high–level functions and control loops are available within the third and fourth command layer. Relying on low–level functions, direct com-

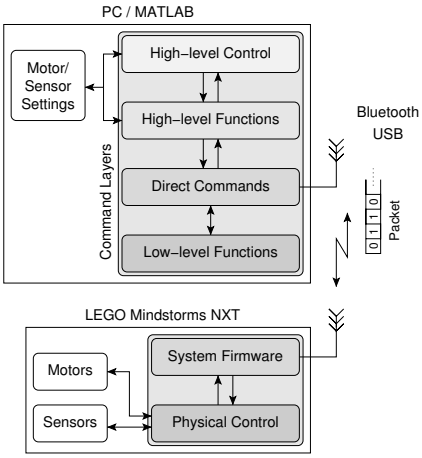

Fig. 2. Overview of the communication between MATLAB and NXT hardware using a multi–layer architecture.

Length, LSB	Length, MSB	Command Type	Command	Byte 5	Byte 6	...

Fig. 3. Structure of a valid Bluetooth packet, defined by LEGO's NXT Bluetooth communication protocol. For a USB communication the first two bytes describing the length of the data packet are omitted.

mands are transmitted via the USB and the wireless Bluetooth communication channel. Each of these commands is specified in the packet–based NXT communication protocol and consists of exactly one data packet. Optional reply packets can be requested for each command. The packet structure is illustrated in Fig. 3.
In the case of transmission via Bluetooth the first two bytes determine the total length of the packet. The command type specifies which category the command is from and whether the NXT should send a reply packet or not. The next byte defines the individual command. What follows is payload and depends on the command. When a command packet is received by the NXT brick, the firmware interprets the content and acts accordingly, e.g. by controlling motors or sensors.
From the technical point of view, the interface of the PC Bluetooth hardware (e.g. a Bluetooth USB stick) is based on the serial port profile (SPP), which uses the radio frequency communication (RFCOMM) protocol and emulates serial ports. Hence, the whole Bluetooth communication is carried out via virtual serial ports. Those are called COM ports in Windows, or can found in the device folders /dev/rfcomm on Linux and /dev/tty on Mac OS, respectively. For data exchange via USB, no additional computer hardware is required, except a USB cable and a free USB port. When the NXT is connected to a Windows or Mac OS machine, the direct commands exchange data with the NXT USB driver "Fantom" (DLL–library and system driver). Since LEGO does not offer any specific NXT USB driver for Linux, the open source library "libusb" (Erdfelt, 2008) is then loaded by the toolbox to handle the USB communication. Via USB connections, direct commands are typically executed within 3 ms (depending on host system specifications), including the time to receive a reply–package if requested. Using Blue-

tooth, a larger latency of about 30 ms is experienced every time the NXT has to switch from transmission to receive–mode and vice versa. Although a lag in the order of some seconds can be observed infrequently (depending on Bluetooth link quality and surrounding interference), the overall communication protocol can be considered reliable.

3.1.1 Command Layers

Table 3 shows a complete overview of the toolbox functions, categorized in different command layers.

Low–level Functions: The lowest layer consists mostly of private functions, which are not directly accessible by the user (i.e. most of them reside in the "private" directory of the toolbox directory structure). These functions include debug procedures, named constants, look–up operations, so–called MATLAB "prototype files" handling external libraries and drivers, as well as functions for binary packet management. Since many low–level functions are called frequently, optimization techniques like look–up tables and mex–files are used for maximal performance.

Direct NXT Commands: This layer provides the first usable front–end of the toolbox. According to the NXT communication protocol, packet–based commands are defined, which can be sent to the NXT via a USB or Bluetooth connection. The interface of these direct commands is implemented as close as possible to the protocol specifications to make it easy for other developers to extend or adapt the open source code. Abstract functions to handle the communication between NXT and computer — independent from the connection type and operating system — are integrated. In relation to the OSI reference model (Day & Zimmermann, 1983), these functions represent the presentation and application layers of the protocol stack.

High–Level Functions: To provide a more user–friendly interface than the direct NXT commands, high–level functions are established. Going far beyond the implementation of the communication protocol, certain feature and parameter combinations are hidden from the user to focus more on essential and robot–related problems. For example, instead of requiring knowledge about specific sensor settings, operation modes, and timeout periods in order to operate the NXT sensors, straightforward instructions are provided for simple actions such as "open sensor, retrieve data". Also complex I^2C command sequences, which are used with digital sensors, are combined into single functions. Possible exceptions are caught wherever possible, and if program execution cannot resume, meaningful error messages are generated. Furthermore main functions to establish and control the NXT $\leftrightarrow$ PC connection via Bluetooth or USB are provided. Some advanced functions are given to read or write complete sets of firmware registers (so-called I/O maps) at once.

High–Level Control and Utilities: Layer four mainly features an object–oriented interface to the NXT actors. The many variable motor options and complex parameter combinations are mapped to properties of the motor class in MATLAB. Functions with integrated control capability handle conditional tasks while interacting with the motors, e.g. pausing further program execution until the servo motor has rotated to a certain position, or helping a motor to reach a specific encoder target. To overcome limitations of the direct commands provided by the NXT firmware, optionally a customized and advanced motor control program with a higher precision control can be used, which runs embedded on the NXT.

Layer	*Description*	*Output/Motors*	*Input/Sensors*	*General*	*Bluetooth/USB*
4	**High–Level Control and Utilities**	**NXTMotor** .ReadFromNXT .SendToNXT .Stop .WaitFor .ResetPosition **NXC_MotorControl**		**OptimizeToolboxPerformance** **GUI_WatchMotorState** **GUI_WatchSensor** **ToolboxTest** **ToolboxBenchmark** **ToolboxUpdate**	**COM_MakeBTConfigFile**
3	**High–Level Functions**	**DirectMotor Command** **StopMotor** **SwitchLamp** NXC_ResetErrorCorrection	**OpenLight** **GetLight** **OpenSound** **GetSound** **OpenSwitch** **GetSwitch** **OpenUltrasonic** **GetUltrasonic** **USMakeSnapshot** **USGetSnapshotResults** **OpenAccelerator** **GetAccelerator** **OpenColor** **CalibrateColor** **GetColor** **OpenCompass** **CalibrateCompass** **GetCompass** **OpenGyro** **CalibrateGyro** **GetGyro** **OpenInfrared** **GetInfrared** **OpenRFID** **GetRFID** **CloseSensor**	**readFromIniFile** MAP_GetCommModule MAP_GetInputModule MAP_GetOutputModule MAP_GetSoundModule MAP_GetUIModule MAP_SetOutputModule NXC_GetSensorMotorData	**COM_OpenNXT** **COM_OpenNXTEx** **COM_CloseNXT** COM_ReadI2C **COM_SetDefaultNXT** **COM_GetDefaultNXT**
2	**Direct NXT Commands**	**NXT_SetOutputState** **NXT_GetOutputState** **NXT_ResetMotorPosition**	**NXT_SetInputMode** **NXT_GetInputValues** NXT_ResetInputScaledValues NXT_LSRead NXT_LSWrite NXT_LSGetStatus	**NXT_PlayTone** NXT_PlaySoundFile NXT_StopSoundPlayback NXT_StartProgram NXT_GetCurrentProgramName NXT_StopProgram NXT_SendKeepAlive NXT_GetBatteryLevel NXT_GetFirmwareVersion NXT_SetBrickName NXT_ReadIOMap NXT_WriteIOMap NXT_MessageWrite NXT_MessageRead	COM_CreatePacket COM_SendPacket COM_CollectPacket
1	**Low–Level Functions**	**MOTOR_A** **MOTOR_B** **MOTOR_C** *byte2outputmode* *byte2regmode* *byte2runstate* *outputmode2byte* *regmode2byte* *runstate2byte*	**SENSOR_1** **SENSOR_2** **SENSOR_3** **SENSOR_4** *byte2sensortype* *byte2sensormode* *sensortype2byte* *sensormode2byte* *waitUntilI2CReady*	**DebugMode** *isdebug* textOut *dec2wordbytes* *name2commandbyte* *commandbyte2name* *wordbytes2dec*	checkStatusByte *createHandleStruct* *checkHandleStruct* *getLibusbErrorString* *getVISAErrorString* *getReplyLengthFromCmdByte* *fantom_proto* *libusb_proto*

Table 3. Overview of the toolbox functions categorized in different command layers.
(NXT_* = NXT direct commands, COM_* = Functions related to the NXT communication, MAP_* Functions related to the NXT I/O maps, NXC_* = Functions which interact with the NXC program "MotorControl", **bold** = Main functions, *italic* = private functions)

In addition to the comfortable motor interface, several tools are offered in this layer: Utilities to monitor the current motor and sensor state, an assistant to create a Bluetooth configuration file, an update notifier, as well as various tools for benchmarking and integrity testing the toolbox.

3.1.2 Advanced Motor Control

When trying to control motors via direct commands (i.e. "NXT_SetOutputState"), two problems become apparent. First, the motors cannot be turned to a precise position, since they often overshoot (i.e. turn too far). This characteristic is caused by the motor control of the LEGO firmware. It only turns off the power to the motor when the desired encoder target (property "TachoLimit") is reached, leaving the motor spinning freely in coast mode. An automatic "braking" mode is not available. Instead, the LEGO firmware provides an automatic error correction mechanism to compensate cumulative error displacements. Unfortunately, due to large overshootings, this displacement correction can lead to unexpected results, which causes another difficulty. For example, the next direct motor command will be ignored by the firmware, if the current absolute motor position already exceeds the next target position. Both characteristics clearly impede an intuitive motor control.
Even though the internal error correction of the firmware can be deactivated by overwriting specific bytes in the firmware register using complex input/output map commands, a precise motor control which automatically turns the motor to a specific position is still not available. To overcome this problem, the advanced program "MotorControl" was developed. The program runs directly on the NXT to control the motors in real–time, without being slowed down by Bluetooth or USB latencies. It is programmed in NXC (Pedersen, 2006) and is downloaded on the NXT as a binary 32 KB large RXE file. During execution of MATLAB programs, "MotorControl" keeps running on the NXT as background process, and controls the motor movement in a control loop. The control parameters are specified via the motor class in MATLAB (toolbox function layer four), and then transmitted to the NXC program using a specified message-based communication protocol. Besides a motor position accuracy of ± 1 degree in most cases, smooth braking and acceleration modes, synchronized motor movements, monitored motor information, and a manual emergency stop (by pressing a button on the NXT brick) are supported. Further information about "MotorControl", its features and its communication protocol are given at `http://www.mindstorms.rwth-aachen.de/trac/wiki/MotorControl`. Since it is designed independently from the MATLAB environment, also other Mindstorms NXT remote control interfaces can adapt the concept and utilize "MotorControl" for their own projects.

3.2 Documentation and Toolbox Help

Besides an adequate program interface, a complete and easily accessible documentation of the toolbox functions and their features is very important for a high level of usability. Thus, the documentation of the RWTH – Mindstorms NXT Toolbox is fully integrated into the common MATLAB help browser, just like any other commercial MATLAB toolbox, as shown in Fig. 4. This is achieved by providing specially formatted XML help and content files.
From each m–file a HTML–formatted MATLAB help file is published using the *wg_publish* script (Garn, 2006). Since every major function is located in a separate m–file, the relevant information is extracted from the customized header format. The layout of the HTML pages is designed like the standard MATLAB text layout, using single cascading style sheets (CSS). Besides interface descriptions, the help content includes example code and see–also links.

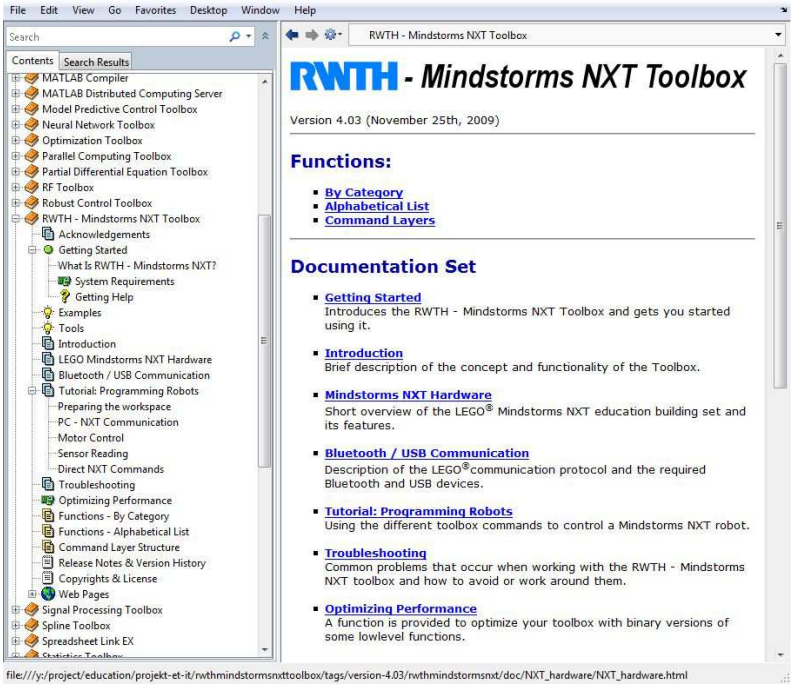

Fig. 4. Documentations and help text of the toolbox integrated in the MATLAB help browser.

Furthermore comprehensive tutorials, first–step demo programs and help pages for beginners are provided. In addition to the help browser support, the common MATLAB help command `help <function>` displays the function's help text in the command window.
Since the toolbox is published as an open source project, the complete source code is well and comprehensively commented so that other developers are able to extend and adapt the toolbox easily. The toolbox (v4.03) consists of more than 14.000 source lines in total. One third are comments, one third are help text, and the remaining third comprises executable code.

3.3 Version History

The first stable toolbox release 1.0 had been completed in December 2007 before the first "MATLAB meets LEGO Mindstorms" lab course started. It featured Bluetooth connections only and provided a basic motor control support via direct commands. Nevertheless the presented robot creations by students were truly impressive (Behrens et al., 2008; 2010).
The toolbox website was created in February 2008, and version 1.0 was made publicly available for download under the GNU General Public Licence (Free Software Foundation, 2007). During summer 2008, USB connections and an improved communication layer were introduced with version 2.0. It enabled the construction of stationary robots with very fast response times. Also the possibility to connect to multiple NXT devices at the same time was another new feature.
Later in 2008, the first embedded NXC program was developed to offer precise motor rotations. Although the control mechanism often led to abrupt motor movements, the position

accuracy was highly improved. The interface to these new motor functions used the object–oriented paradigm for the first time. Additionally, more external sensors were supported. The resulting stable toolbox 2.03 was used during the student project in 2008.
In 2009, the focus was put on higher precision of the embedded motor control program. Smooth braking was achieved by introducing a custom control algorithm. Other improvements include further documentation, stability and performance issues. The version number has finally arrived at 4.03, which is the latest stable version recommended to be used as of May 2010.

3.4 System Requirements

In summary the RWTH – Mindstorms NXT Toolbox for MATLAB can be used on standard PC and NXT hardware. The system requirements of the current release version 4.03 are listed in Table. 4.

- Operating system: Windows, Linux, or Mac OS
- MATLAB Version 7.7 (R2008b) or higher
- LEGO Mindstorms NXT building kit (compatible with NXT 1.0 retail, NXT 2.0 retail, and NXT Education)
- LEGO Mindstorms NXT firmware v1.26 or higher, or compatible
- For Bluetooth connections: Bluetooth 2.0 adapter recommended model by LEGOő (e.g. AVM BlueFRITZ! USB) supporting the serial port profile (SPP)
- For USB connections: Official Mindstorms NXT Driver "Fantom", v1.02 or higher (Windows, Mac OS), “libusb” (Linux), 32–bit version of MATLAB

Table 4. System requirements of the RWTH – Mindstorms NXT Toolbox v4.03.

In the case of using an older MATLAB version such as 7.3 (R2006b), the NXT motors can be alternatively controlled via the classic motor control functions offered until toolbox release 2.04. For more information using individual system configurations, a version guide and changelogs are provided on the toolbox web page.

3.5 Example Code

A basic example program using high–level functions and direct commands is shown in Listing 1. The program first establishes a Bluetooth connection to the NXT, then plays a tone, gets the current battery level, and finally closes the connection again.

```
handle = COM_OpenNXT('bluetooth.ini'); % open a Bluetooth connection using
COM_SetDefaultNXT(handle);             % parameters from configuration file

NXT_PlayTone(800,500);                 % play tone with 800Hz for 500ms

voltage = NXT_GetBatteryLevel;         % get current battery level

COM_CloseNXT(handle);                  % close Bluetooth connection
```

Listing 1. Basic program example.

A comparison between high–level functions and direct commands for sensor reading is given in the next Listings 2 and 3. Both programs request the current raw 10–bit value of the NXT light sensor. In the case of using direct NXT commands (command layer two), data packets have to be requested and verified by the user program code until a valid packet is received from the sensor. This control mechanism is necessary, since the light sensor can still be busy in its initialization step. A control loop, which requests packets every 300 ms in the case of an invalid data is shown in Listing 2.

```
% initialize the sensor
NXT_SetInputMode(SENSOR_1, 'LIGHT_ACTIVE', 'RAWMODE', 'dontreply');

data = NXT_GetInputValues(SENSOR_1); % get light sensor value

if ~data.Valid            % check valid flag, re-request data if necessary
    startTime = clock();  % initialize timeout counter
    timeout   = 0.3;      % set time out to 300 ms

    while (~data.Valid) && (etime(clock, startTime) < timeout)
        data = NXT_GetInputValues(SENSOR_1); % re-request until valid/timeout
    end
end

light = double(data.NormalizedADVal); % use normalized light value (10 bit)
```

Listing 2. Program reads the current value of the light sensor using direct NXT commands.

Using high–level functions from command layer three, the control loop in Listing 2 is hidden from the user to provide a better usability for sensor reading. Thus, the whole program simplifies to only two commands, as shown in Listing 3.

```
OpenLight(SENSOR_1, 'active'); % initialize light sensor

light = GetLight(SENSOR_1);    % get light sensor value
```

Listing 3. Reprogramming of the program code in Listing 2 using high–level functions.

In addition to the high–level features, the RWTH – Mindstorms NXT Toolbox provides applications for motor and sensor data monitoring. Since e.g. the initialization of parameter settings, sensor tests, or calibration processes are often necessary for the development of individual control algorithms, the users are able to test and measure sensor characteristics, as illustrated in Fig. 5.
An example of using objects of the NXTMotor class (command layer four) to control the NXT servo motors in MATLAB is shown in the next two listings. First, several motor objects for

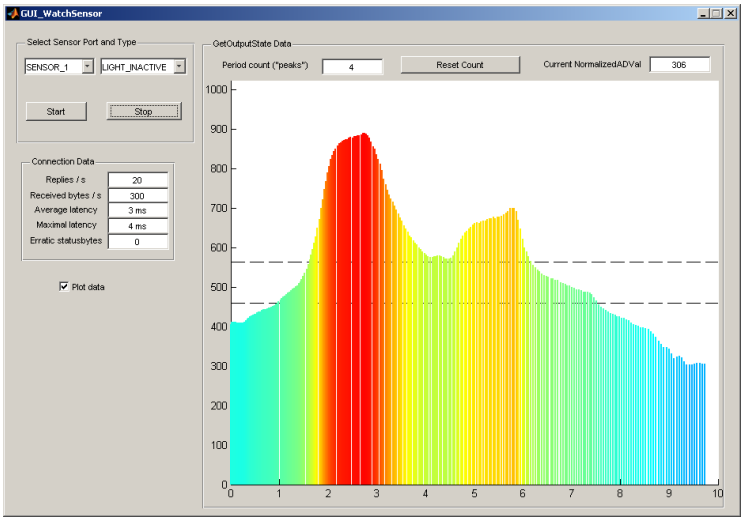

Fig. 5. The "Watch Sensor GUI" provides a comfortabke data monitoring tool for several NXT sensors.

different robot movements are created in Listing 4. Based on these objects, an algorithm of an explorer robot which drives an eight–shaped loop on the floor becomes structured and very simplified, as shown in Listing 5.

```
leftWheel    = MOTOR_B; % set parameters
rightWheel   = MOTOR_C;
bothWheels   = [leftWheel; rightWheel];
drivingPower = 60;      % in percent
turningPower = 40;      % in percent
drivingDist  = 1500;    % in degrees
turningDist  = 220;     % in degrees

% create objects for drive forward:
mForward = NXTMotor(bothWheels, 'Power', drivingPower, 'TachoLimit',
    drivingDist);

% create object for turning the bot left:
mTurnLeft1 = NXTMotor(leftWheel,  'Power', -turningPower, 'TachoLimit',
    turningDist);
mTurnLeft2 = NXTMotor(rightWheel, 'Power',  turningPower, 'TachoLimit',
    turningDist);

% create object for turning the bot right:
mTurnRight1      = mTurnLeft1;           % copy objects
mTurnRight2      = mTurnLeft2;
mTurnRight1.Port = rightWheel;           % swap wheels
mTurnRight2.Port = leftWheel;
```

Listing 4. Initialization of motor objects for different robot movements.

```
for n=1:1:8
  mForward.SendToNXT();                % drive forward
  mForward.WaitFor();

  if (n < 4) || (n == 8)
    mTurnLeft1.SendToNXT();            % make left-turn
    mTurnLeft1.WaitFor();
    mTurnLeft2.SendToNXT();
    mTurnLeft2.WaitFor();
  else
    mTurnRight1.SendToNXT();           % make right-turn
    mTurnRight1.WaitFor();
    mTurnRight2.SendToNXT();
    mTurnRight2.WaitFor();             % resulting route   _
  end                                  % of the robot:    |_|
end                                    %                  |_|
```

Listing 5. Program code of an explorer robot driving an eight–shaped loop.

4. Software Project Management

To maintain the current and previous versions of the RWTH – Mindstorms NXT Toolbox, the revision control system Subversion® (The Apache Software Foundation, 2000) is used. Thus, changes and developments of each single file of the toolbox can be easily controlled. Furthermore merging of new program code contributed by different programmers becomes structured and traceable. In addition to the revision control of source code, the toolbox is administrated using the web–based project management tool Trac (Edgewall Software, 2003). It provides a wiki, an issue tracking system for bug reports, a user administration, and a road map schedule for project management.
Using an individual layout the RWTH – Mindstorms NXT Toolbox is published as an open source software on the web page `http://www.mindstorms.rwth-aachen.de` (see Fig. 6).

5. Educational Projects, Evaluations and Results

5.1 Freshmen Project "MATLAB meets LEGO Mindstorms"

The development of the RWTH – Mindstorms NXT Toolbox for MATLAB was motivated by the establishment of a new laboratory "MATLAB meets LEGO Mindstorms" for freshman students at the RWTH Aachen University, Aachen, Germany. Started in winter term 2007, the project has become an annual mandatory project for each first–semester Bachelor student of electrical engineering. Within this eight–day full–time course three objectives are addressed. First, mathematical foundations are mapped to MATLAB program code. Based on this, more complex tasks and algorithms are then described within the MATLAB environment. Going beyond simulations, real applications are performed by LEGO Mindstorms NXT robots, which are designed and constructed by the students themselves.
While many other robotic education projects are designed for senior students, this project is intentionally established for freshman students. Each winter term almost 400 students participate in the laboratory and are guided by more then 80 supervisors simultaneously. Using about 200 robot kits, students grouped into teams of two are distributed over 23 institutes of the Electrical Engineering Department. The project tasks are separated into three working