SATAKUNTA UNIVERSITY OF APPLIED SCIENCES

Tero Mäkelä

USING BRICKOS WITH LEGO MINDSTORMS RCX BRICK AND ESTABLISHING INFRARED COMMUNICATION

Information Technology training programme

Degree Programme in Software engineering

2007

BRICKOS:IN KÄYTTÖ LEGO MINDSTORMS RCX PALIKAN KANSSA JA INFRAPUNAYHTEYDEN LUOMINEN

Mäkelä, Tero Kristian
Satakunnan ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto
Tammikuu 2007
Niemi Juha, DI
UDK: 004.41, 004.42, 004.45
Sivumäärä: 52

Avainsanat: Tietotekniikka, Ranska, Robotin Ohjelmointi, Cygwin, Firmware

---

Opinnäytetyössä tutkittiin vaihtoehtoisen käyttöjärjestelmän brickOS:in asentamista LEGO Mindstormsin RCX palikkaan sen ohjelmointia C ja C++ kielillä sekä RCX:n reaaliaikaista kontrollointia käyttäen PC:tä ja infrapunatornia.

Projektin aihe annettiin Valenciennesin yliopistolta Ranskasta, jossa tekijä oli vaihto-opiskelijana työharjoittelua suorittamassa kolmen muun suomalaisopiskelijan kanssa.

Ilman aiempaa kokemusta kyseisestä aiheesta täytyi ensin selvittää miten kaikki toimii ja miten asiat saadaan toimimaan. brickOS:in ja RCX:n tutkiminen oli asia johon paneuduttiin ennen kuin alettiin tekemään mitään muuta. Ensimmäinen selvitettäväksi annettu tehtävä oli se, että pyyhkiytyykö alkuperäinen firmware pysyvästi pois, kun brickOS asennetaan ja kävi ilmi, että näin käy joka kerta kun patterit poistettiin joten se ei ollut enään jatkamisen esteenä.

Seuraavaksi selvitettiin oikea tapa asentaa tarvittavat ohjelmat brickOS:in asennusta varten ja miten niitä käytetään. Monien kokeilujen jälkeen löydettiin oikea tapa asentaa tarvittavat ohjelmat ja myös miten niitä käytetään.

Ohjelmien tekeminen tiettyjen funktioiden aikaansaamiseksi todettiin olevan suhteellisen helppoa, mutta infrapunayhteyden mahdollistaminen niin, että RCX palikkaa pystyi ohjaamaan PC:ltä oli todellinen haaste joka lopulta pystyttiin toteuttamaan.

Näitä tietoja voidaan käyttää keskenjääneiden tavoitteiden toteuttamiseksi tai uusien projektien perustana.

USING BRICKOS WITH LEGO MINDSTORMS RCX BRICK AND ESTABLISHING INFRARED COMMUNICATION

Mäkelä, Tero Kristian
Satakunta University of Applied Sciences
Information Technology training programme
Degree Programme in Software engineering
January 2007
Niemi Juha, Msc
UDK: 004.41, 004.42, 004.45
Pages: 52

Key-words: Information Technology, France, Programming Robot, Cygwin, Firmware

---

The aim of this thesis was to examine the installation of an alternative operating system brickOS into LEGO Mindstorms RCX brick, programming it with C and C++ languages and the real time control of the RCX using a PC with an infrared tower.

Subject for the project was given by Valenciennes University in France where the author was an exchange student carrying out his practical training with three other Finnish student.

Without earlier experience about the subject first it had to be found out how everything works and how to make things work. Investigating brickOS and RCX was a matter that had to be done before doing anything else. The first thing that was given to work out was if the original firmware would be definitely destroyed when brickOS would be installed and it turned out that it happened every time batteries were removed from the RCX so that wasn't an obstacle to continue anymore.

The next thing to be found out was the right way to install all the necessary programs to install brickOS and how to use them. After many trials the right way to install necessary programs was found out and also how to use them.

Making programs for effectuation of specific functions was discovered to be quite easy, but to enable infrared communication so that the RCX brick could be controlled from PC was a real challenge that was ultimately accomplished.

This information can be used to accomplish unfinished goals or as the base of new projects.

TABLE OF CONTENTS

1 PEOPLE WHO I WOULD LIKE TO THANK

Jean Paul Becar, for having many interesting ideas and guiding us through our work.

Jean Charles Canonne, for his support, help and giving me the best laugh during my entire placement in Valenciennes University. Gotta love those Belgian ‘beaches’ ^_^

Laurent Vermeiren, for being there and supporting us.

Mika Impola, for helping me in various things throughout the project.

Sami Aziz, for being himself and making every day a little funnier.

Mikko Kirkanen, for assisting me with some of the work with the project.

Lionel Boidin and Guillaume Lagache, for helping out on this project and for being great guys.

And God, for having faith in me ^_^

Thanks guys, I couldn’t have done it without you.

2 INTRODUCTION

2.1 Background information

Me and three other lucky persons Mikko, Sami and Mika got to go to Valenciennes University in France (Picture 1) to carry out three months of practical training.

Picture 1. Valenciennes /21/.

After arriving there we all got to choose our own subjects which we would be doing. Even though everyone ‘decided’ what they would do already in Finland, after we arrived the teachers instead gave us the possibility to choose out of different projects which one we could do, and instead of the database I decided to take on the robot programming project. It was completely new and different to me and I was very excited to get a chance to do something like this.

Programming robots was something I had wanted to do for a long time, but I never had the chance or necessary resources to do so. I feared a little that I couldn’t accomplish the goals given to me since I lacked the experience, but I didn’t let it get me down.

We were told by the teachers that one of the most important goals was to teach us information about something that we didn’t know before. To learn something new.

2.2 Starting situation and ideas for the project

The basic idea was to install alternative operating system brickOS to the LEGO Mindstorms RCX brick and to make programs to it with C and C++ languages. I studied the brickOS operating system, found out how things work and tried to achieve the goals given to me.

At the beginning I had no knowledge of how the communication works and neither did I know anything about the RCX brick or brickOS. A lot of time went to finding information about these things and to dig up example programs for research in order to understand and learn how to make programs that could execute wanted functions. These are some of the ideas that were given to me to work on.

- To establish communication between PC and the RCX brick which is using brickOS and to control the brick with PC using LEGO USB tower.

- To send and retrieve information from the brick.

- To be able to pick up an egg without smashing it using robotic arm constructed from LEGO bricks while being controlled through PC with infrared connection.

- To control the RCX brick with a force feedback joystick connected to the PC and to get force feedback effect.

3 HARDWARE BACKGROUND INFORMATION

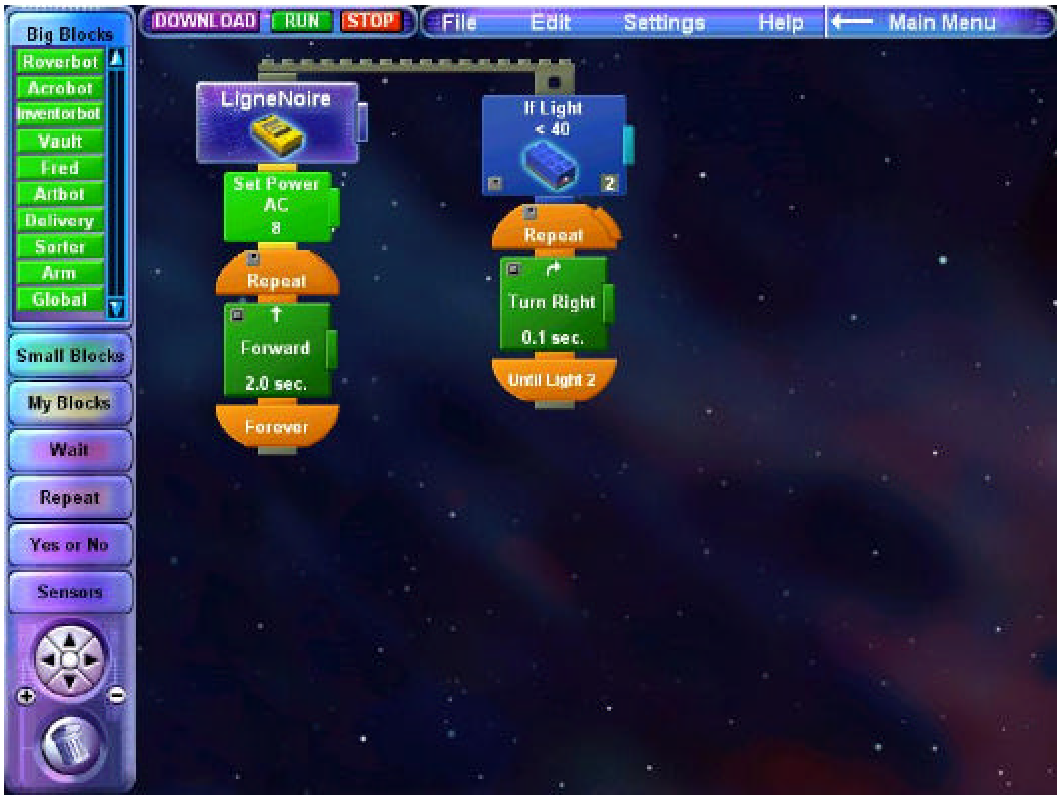
3.1 Key components

The RCX is the programmable LEGO brick that transforms models into robots and controls their actions.

You can control the RCX either by remote control, meaning a direct control from a computer using infrared tower or LEGO Mindstorms Remote Control or by autonomous control.

You can build models and robots using the RCX as the brain. You can write programs and download them to the RCX via infrared tower. After being programmed, the robots are fully autonomous, acting on their own with no support from the computer. Robots take action, interacting with their environment and making decisions based on input from their surroundings, via sensors. Two RCX bricks can even communicate with one another using their infrared eyes /1/.

The original operating system of the RCX can be programmed with Robotics Invention System 2.0 which we used during our task. This is a graphical user interface for programming the RCX. You simply select the blocks you want to use and make a program you need (Picture 2). Simple and effective.



Picture 2. Robotics Invention System 2.0 Software /19/.

3.2 The RCX

Picture 3. LEGO Mindstorms RCX brick /1/.

The base system for using the RCX consists of the RCX itself (Picture 3), an infrared transceiver, and a PC.

Additional components, such as motors, sensors, and other building elements, combine with the base system to allow the creation of functional autonomous robotic devices.

At the core of the RCX is a Hitachi H8 microcontroller with 32K of external RAM. The microcontroller is used to control three motors, three sensors, and an infrared serial communications port.

An on-chip, 16K ROM contains a driver that is run when the RCX is first powered up. The on-chip driver is extended by downloading 16K of firmware to the RCX.

Both the driver and firmware accept and execute commands from the PC through the infrared communications port.

Additionally, user programs are downloaded to the RCX as byte code and are stored in a 6K region of memory. When instructed to do so, the firmware interprets and executes the byte code of these programs /4/.

The RCX has a piezoelectric speaker, which produces 6 distinct tones and can even 'carry a tune'.

Using infrared communication, the RCX can:

- communicate with a computer, sending messages back and forth
- communicate with other RCX bricks: messages can be passed from RCX to RCX
- be controlled via the LEGO Mindstorms Remote Control

/1/.

3.3 Infrared transmitter

Picture 4. Serial and USB infrared transmitters /1/.

Download programs from your computer to the RCX via the infrared transmitter (Picture 4). Transmitters work on both Mac and Windows platforms.

It is available in two cable versions:

- USB cable (requires ROBOLAB 2.5)

- Serial Cable (works with all ROBOLAB software)

Using one transmitter per computer is recommended /1/.

3.4 Remote control

Picture 5. LEGO Mindstorms Remote Control /1/.

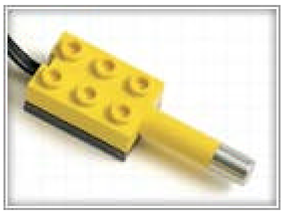
The LEGO Mindstorms Remote Control (Picture 5) enables you to activate the RCX brick at a distance. It communicates with the RCX brick using infrared light, just like a television remote control.

It enables the operator to control one, two, three, or all output ports directly or to run any of the five RCX programs at a distance /1/.

3.5 Input devices

The RCX programmable LEGO brick has three input ports to which can be connected a variety of sensors /2/.

Temperature sensor



Picture 6. LEGO Temperature sensor /2/.

The LEGO Temperature sensor (Picture 6) is calibrated from -20 degrees Celsius to +50 degrees Celsius. The RCX can read and display the temperature in either Fahrenheit or Celsius.

Example: Graph the change in temperature when the ice cube is placed in a glass of tap water. Turn on a lamp or an alarm when the temperature reaches a preset level.

Touch sensor

Picture 7. LEGO Touch sensor /2/.

The LEGO Touch sensor (Picture 7) is a 9-Volt digital sensor calibrated as a "true/false" switch.

Example: Determine when a LEGO Mindstorms robot has run into something - then make the robot turn or back up to get around the object.

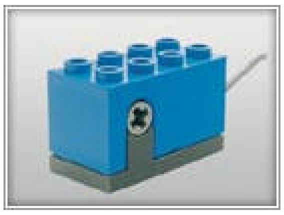
Light sensor

Picture 8. LEGO Light sensor /2/.

The LEGO Light sensor (Picture 8) reads light from 0.6 Lux through 760 Lux. The RCX scales this to a 0-100 percent measure.

Example: Set up a speed trap for a LEGO vehicle to determine its speed as it breaks two light beams.

Rotation sensor



Picture 9. LEGO Rotation sensor /2/.

The LEGO Rotation sensor (Picture 9) reads 16 positions per rotation. Resolution is 500 RPM Max. The RCX will read in either angle degrees or 16th's of a rotation.

Example: Control how far the door on a bird feeder would have to open to provide different amounts of bird feed at different times of a day.

LEGO Camera & Software

Picture 10. LEGO Camera /2/.

The LEGO Camera (Picture 10) and editing software is an ideal extension tool for introducing and enhancing project management. The LEGO Camera can also be used as a sophisticated control sensor.

Extra sensors

To extend your work you can use the LEGO-DCP Adapter, which allows you to combine the DCP Microsense LogIT sensors with the RCX micro computer and

ROBOLAB 2.5 for monitoring sound, temperature, pH, humidity, air pressure, voltage, rotation, current probe, LUX light level, designer analog, barometric/low air pressure and accelerometer /2/.

http://www.dcpmicro.com/

3.6 Output devices

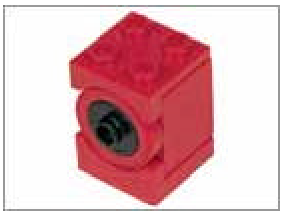
The RCX programmable LEGO brick has three output ports to which can be connected a variety of output devices /3/.

LEGO 9 Volt Gear Motor

Picture 11. LEGO 9 Volt Gear Motor /3/.

The LEGO Gear Motor (Picture 11) is special because of the internal gearing of the motor. Gears are located inside the motor instead of externally. This allows for applications demanding high torque at lower speeds and with little loss to friction.

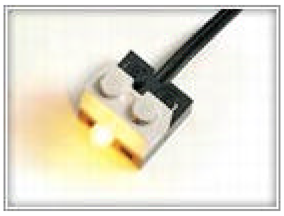
Micromotor



Picture 12. LEGO Mircomotor /3/.

Smaller motor for light duty motion (Picture 12).

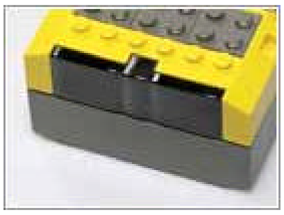
LEGO Lamp brick



Picture 13. LEGO Lamp brick /3/.

The LEGO lamp (Picture 13) is built into a small two-stud brick and integrates nicely into various inventions.

RCX Infrared Eye



Picture 14. RCX Infrared eye /3/.

Using infrared communication (Picture 14) the RCX can communicate with other RCX bricks: messages can be passed from RCX to RCX and communicate with a computer, sending messages back and forth.

4 SOFTWARE BACKGROUND INFORMATION

4.1 Alternative operating systems

These are alternative OS's for LEGO Mindstorms RCX brick, but besides brickOS we will look into them only briefly.

brickOS is an alternative operating system for the LEGO Mindstorms Robotic Invention System. The intent is to allow developers to write C and/or C++ code for the RIS platform /5/.

leJOS is replacement firmware for the LEGO Mindstorms RCX brick - a JVM that fits within the 32kb on the RCX /6/. Yes, you can program a LEGO robot with Java!

NQC (Not Quite C) is a programming language for several LEGO Mindstorms products including the RCX, CyberMaster, and Scout. NQC's syntax is very similar to the C programming language, so experienced C programmers (and Java programmers) should find it very easy to get started with. Even if you aren't an experienced programmer, NQC is relatively easy to learn. NQC uses the same firmware as LEGO's standard tools (RCX Code and Robolab) /7/.

Alternative firmware pbForth is incarnation of Forth. It is designed to work with the LEGO Mindstorms RCX brick /8/.

4.2 brickOS

brickOS was originally known as legOS. This is the reason why some of the references have legOS name in them.

brickOS is an open source embedded operating system and provides a C and C++ programming environment for the LEGO Mindstorms Robotics Kits, allowing the owner of such a kit to program in good old C and also C++ instead of the standard LEGO Programming Language. It was originally developed on Linux, but should run on most Unices as well as Windows.

brickOS consists of an alternative operating system for the Mindstorms RCX and demonstration programs written in C and C++. Also provided are utilities which allow you to download the OS to the RCX and download your compiled programs to the RCX.

The brickOS distribution provides the sources for the operating system, the demo programs and the utilities. You will also need a Hitachi H8 cross compiler assembler and linker (basic tool chain).

Instructions that will help you to obtain and configure these tools can be found at brickOS homepage and also on this document.

After they are installed you will be able to build the OS, the utilities and finally, programs of your own for controlling your creations /9/.

4.3 What Cygwin is and isn't

Cygwin is a Linux-like environment for Windows. It consists of two parts: A DLL (cygwin1.dll) which acts as a Linux API emulation layer providing substantial Linux API functionality. A collection of tools, which provide Linux look and feel. The Cygwin DLL works with all non-beta, non "release candidate", ix86 32 bit versions of Windows since Windows 95, with the exception of Windows CE.

Cygwin is not a way to run native Linux apps on Windows. You have to rebuild your application from source if you want it to run on Windows. Cygwin is not a way to magically make native Windows apps aware of UNIX ® functionality, like signals, ptys, etc. Again, you need to build your apps from source if you want to take advantage of Cygwin functionality /10/.

4.4 Bricx Command Center

Bricx Command Center (BricxCC) is a Windows (95, 98, ME, NT, W2K, XP) program commonly known as an integrated development environment (IDE) for programming the RCX (all versions), Scout, Cybermaster, and Spybot programmable bricks using Dave Baum's Not Quite C (NQC) language. And it supports programming the Scout, RCX2, and Spybot using The LEGO Company's MindScript™ and LASM™ languages via the Mindstorms 2.5 SDK.

It supports programming RCX bricks in C, C++, Pascal, Forth, and Java using the brickOS, pbForth, and leJOS alternate firmwares. It also supports programming the new LEGO Mindstorms NXT brick using Not eXactly C (NXC), Next Byte Codes (NBC), and a simple on-brick programming language called NPG.

Version 3.3 of BricxCC which was created by John Hansen is an enhanced revision to Mark Overmars' original program. You can see his web page for an overview of the basic functionality upon which this release was built /11, 12/.

4.5 LNP, WinLNP and LNP Daemon

LNP stands for LegOS Network Protocol. It allows for communication between legOS-powered robots and host computers.

Although LNP can be used for communication between many devices, with any mixture of computers and RCX's, my experience only covers communication between my computer and a single RCX, and only that situation will be discussed. LNP is still very useful in this case, since it takes care of a lot of details and is generally easier to work with than raw infrared communication /13, p. 4/.

Under Windows WinLNP allows the use of various Windows programming languages to communicate with a legOS robot (Appendix 1). Under Linux, the LNP Daemon can be used to communicate with the RCX from the PC, but since all of the work was done with Windows OS's it won't be discussed any more /13, p. 3/.

5 HOW TO MAKE EVERYTHING WORK

There are two ways to compile your files, either with BricxCC installation or with Cygwin installation. My recommendation is BricxCC since GUI makes life a hell of a lot easier.

I installed both of them to my computer since I had some trouble getting some of the files compiled with BricxCC.

If you are going to also do this I recommend that you will do the Cygwin installation first. You should also pay close attention to where you are going to install files as in place Cygwin and brickOS into different directories by the instructions given here.

So just follow these instructions and you should be fine.

Maybe.

5.1 Installing USB tower

Download and install the LEGO USB tower drivers by following instructions of the setup wizard.

http://www.lego.com/eng/service/downloads/patches/Tower164.zip

After you are done you can plug in the USB tower.

5.2 Installing and using BricxCC

Before you install BricxCC, make sure the Windows register is empty of a previous version of BricxCC, brickOS and Cygwin.

If it isn't it can cause some difficulties when compiling programs and transferring them to the brick.

Download these files:

BricxCC 3.3.7.10
http://sourceforge.net/projects/bricxcc/

LEGO Mindstorms SDK 2.5
http://mindstorms.lego.com/sdk2point5/LEGOMindstormsSDK25.zip

BricxCCSetupCygwin
http://prdownloads.sourceforge.net/bricxcc/BricxCCSetupCygwin.exe?download

BricxCCbrickOSleJOS
http://prdownloads.sourceforge.net/bricxcc/BricxCCbrickOSleJOS.exe?download

To install Bricx Command Center, perform the following simple steps:

- Run the BricxCC setup executable. Follow the setup wizard instructions. Choose the default installation.

- Install the LEGO Mindstorms SDK 2.5 to enable MindScript and LASM compile and download support for RCX2, Scout, and Spybot bricks.

- To enable brickOS C/C++/Pascal support and leJOS Java support you can now download an all-in-one setup executable which installs a small subset of Cygwin along with pre-built H8 tools.

  NOTE: If you get an error when you try to run the installer that it is missing BricxCCSetupCygwin.002 that means the download did not complete successfully. Please try to download it again or using a download manager utility of some sort.

- Once you have successfully downloaded and installed the Cygwin configuration and H8 compiler tools then install the minimal brickOS and leJOS for BricxCC and you are ready to go (except for leJOS, which still needs to have a separately installed Java Developers Kit).

/11/.

Using BricxCC is pretty simple. When you start the BricxCC you choose the port where your infrared tower is connected and then you choose the firmware and RCX as the brick you are using.

From here it is easy to choose the programs you want to edit and compile. In case you have difficulties compiling programs you might want to try and put them into the demo directory. It might help.

5.3 Installing Cygwin

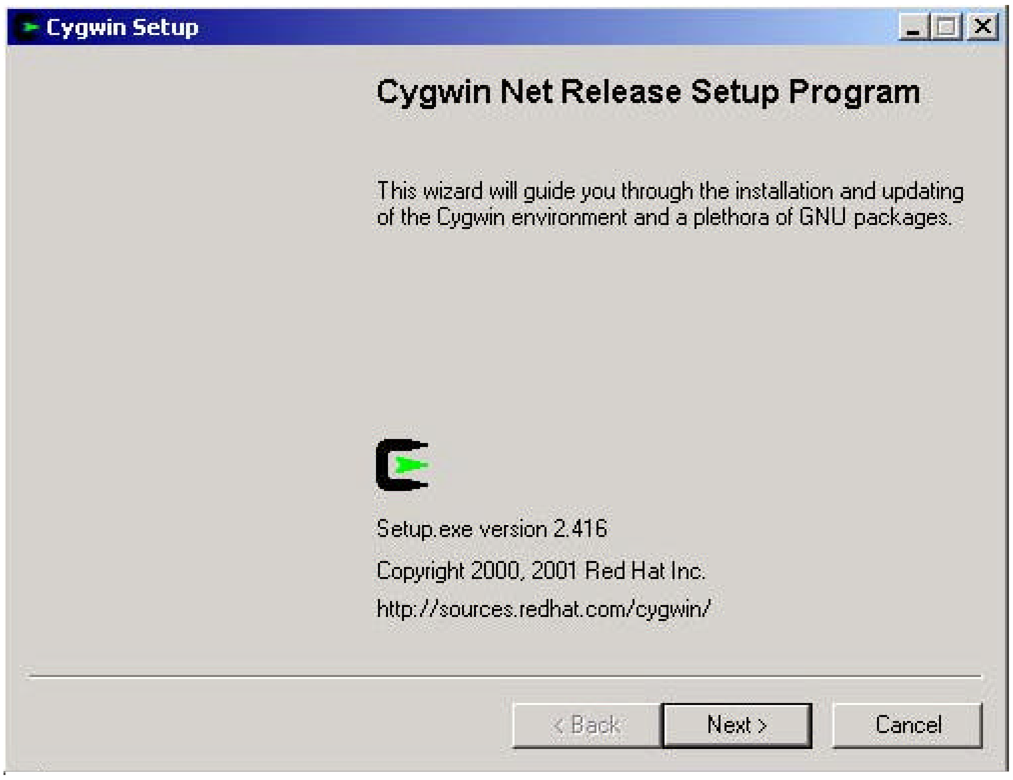
You can use this link to go to a web site to download all necessary files for brickOS installation and to see instructions on how to make everything work:

http://www2.acae.cuhk.edu.hk/~tliu/LegOS_install/legOS_install.htm

First, download and install Cygwin by clicking on this link:

http://www.cygwin.com/setup.exe

Follow instructions of the setup wizard until you get to Select Packages window.


Cygwin Setup

Cygwin Net Release Setup Program

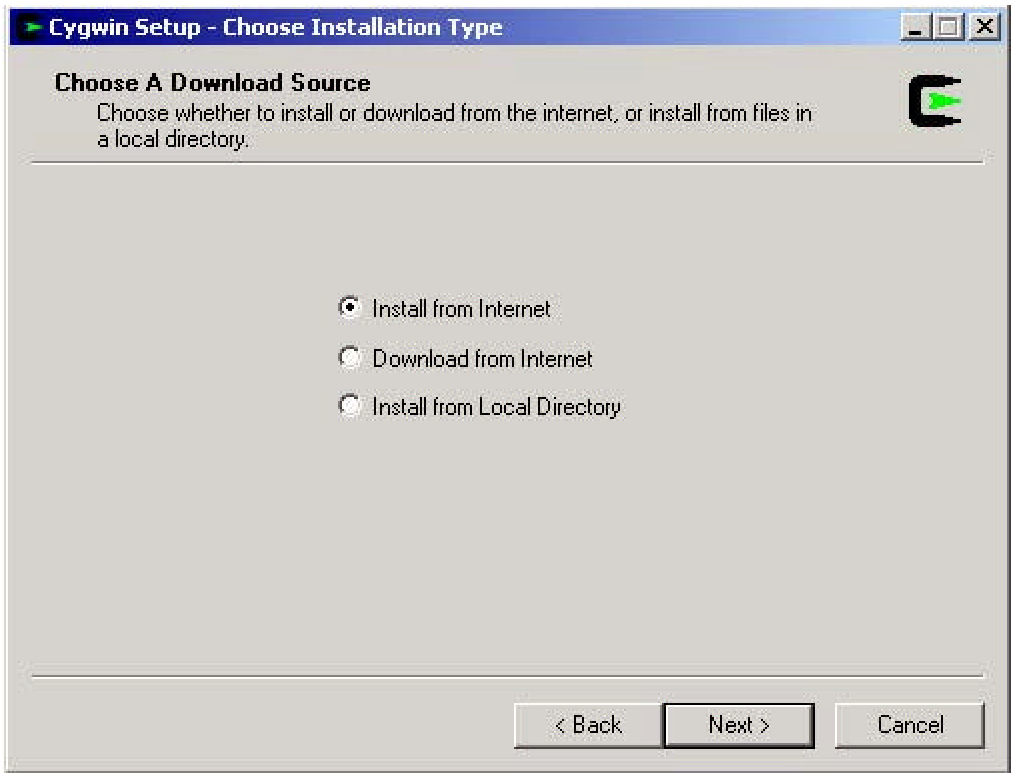
This wizard will guide you through the installation and updating of the Cygwin environment and a plethora of GNU packages.

Setup.exe version 2.416

Copyright 2000, 2001 Red Hat Inc.

http://sources.redhat.com/cygwin/

Picture 15. Installing Cygwin /15/.

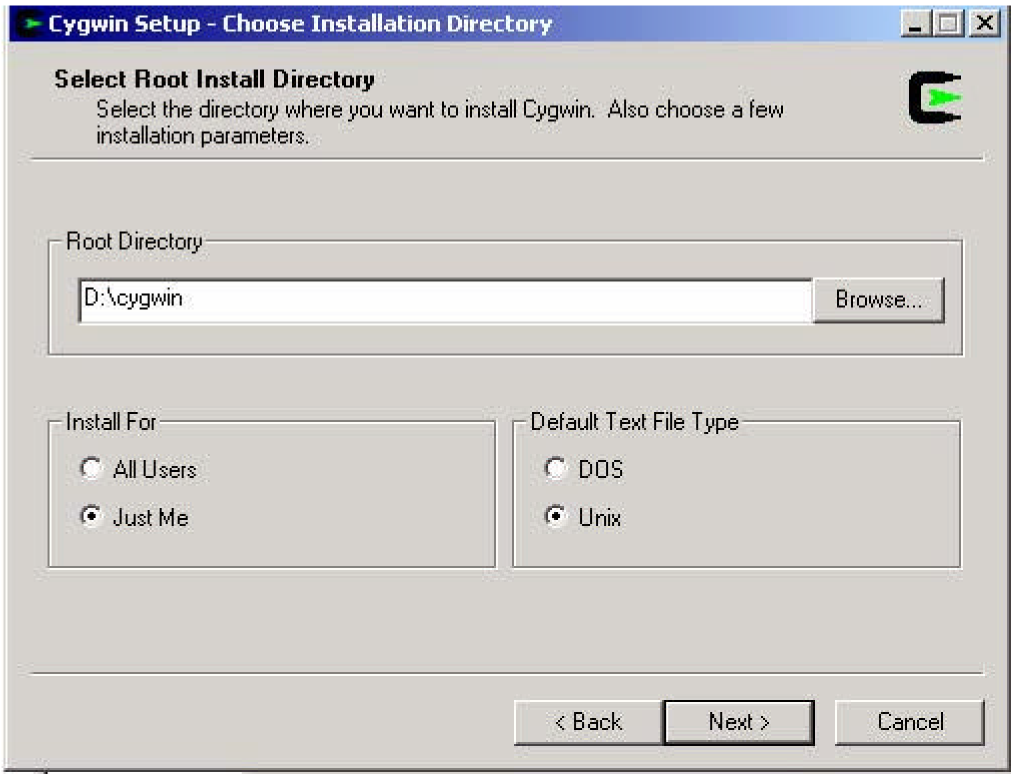

Cygwin Setup - Choose Installation Type

Choose A Download Source

Choose whether to install or download from the internet, or install from files in a local directory.

[✓ Install from Internet]
Download from Internet
Install from Local Directory

Picture 16. Installing Cygwin /15/.



Cygwin Setup - Choose Installation Directory

Select Root Install Directory

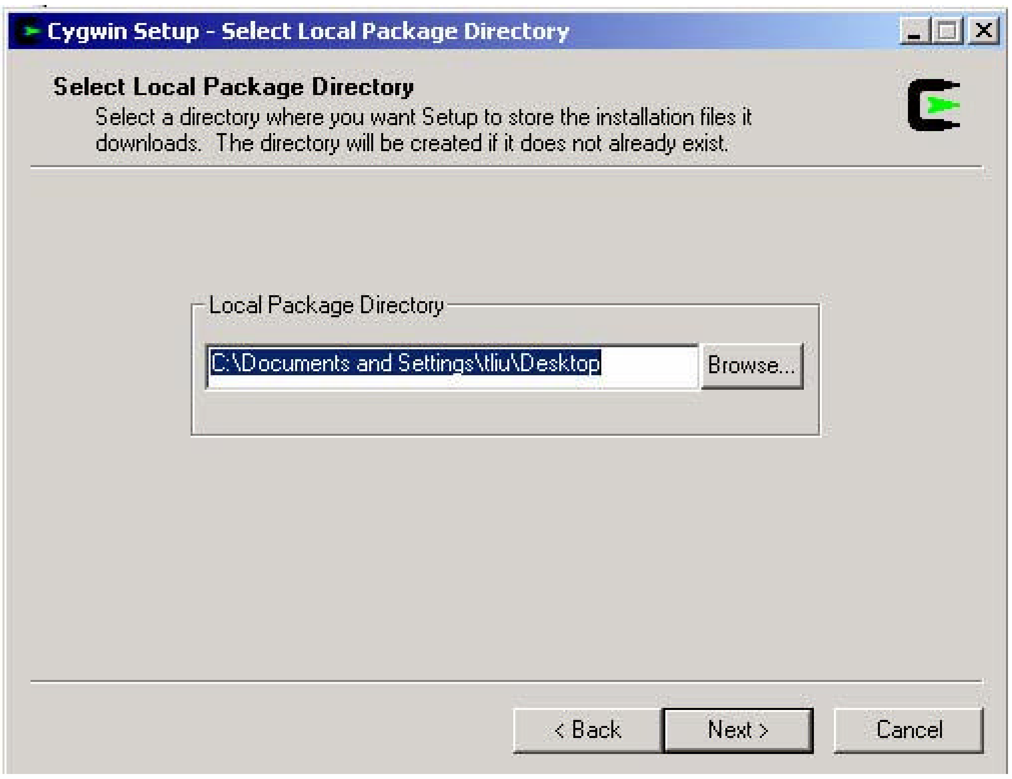
Select the directory where you want to install Cygwin. Also choose a few installation parameters.

Root Directory: D:\cygwin

Install For: ( ) All Users (●) Just Me

Default Text File Type: ( ) DOS (●) Unix

Picture 17. Installing Cygwin /15/.



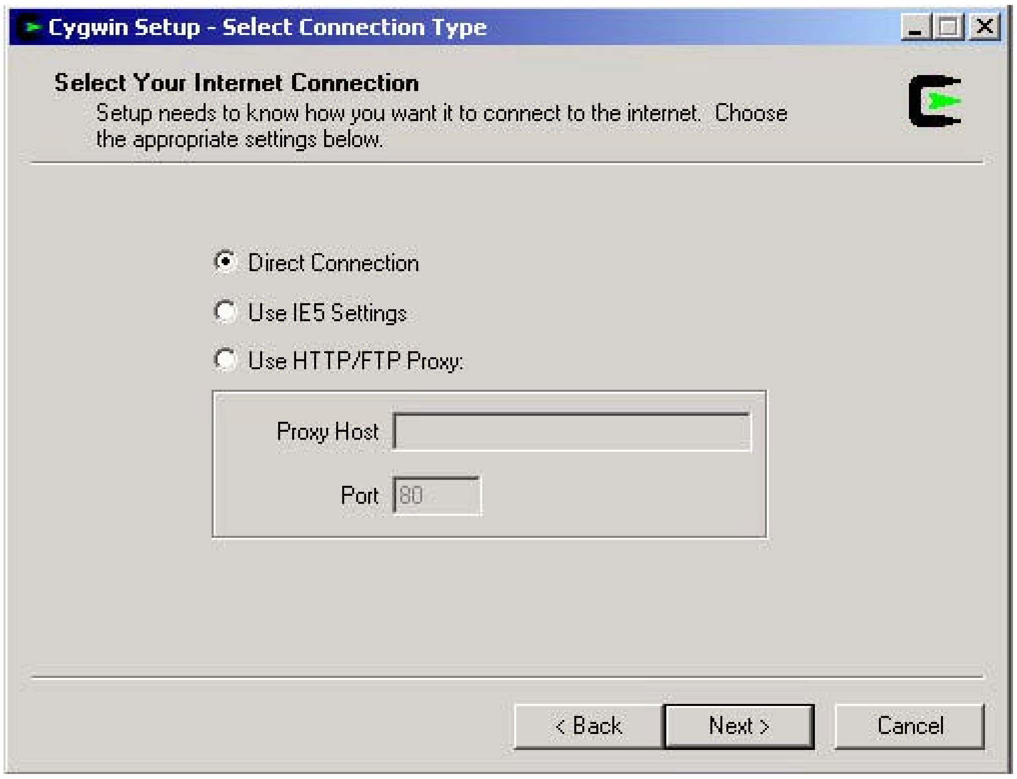
Cygwin Setup - Select Local Package Directory

Select Local Package Directory

Select a directory where you want Setup to store the installation files it downloads. The directory will be created if it does not already exist.

Local Package Directory: C:\Documents and Settings\tliu\Desktop

Picture 18. Installing Cygwin /15/.



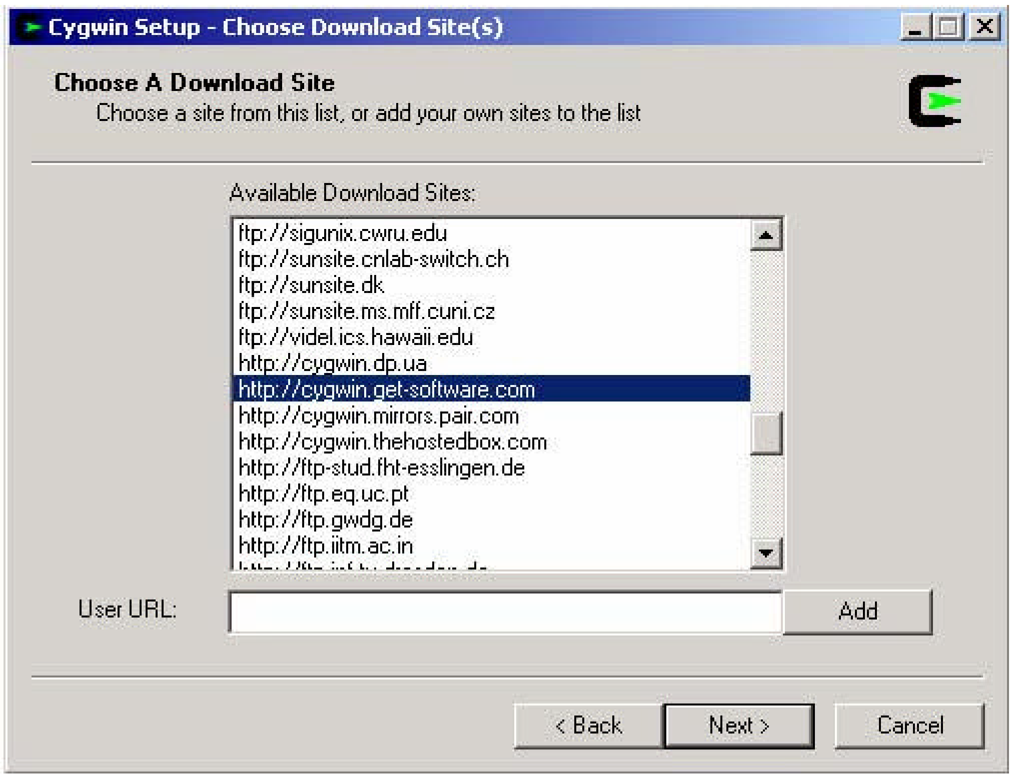
Select Your Internet Connection

Setup needs to know how you want it to connect to the internet. Choose the appropriate settings below.

[✓ Direct Connection]
Use IE5 Settings
Use HTTP/FTP Proxy:

Proxy Host
Port 80

Picture 19. Installing Cygwin /15/.



Choose A Download Site

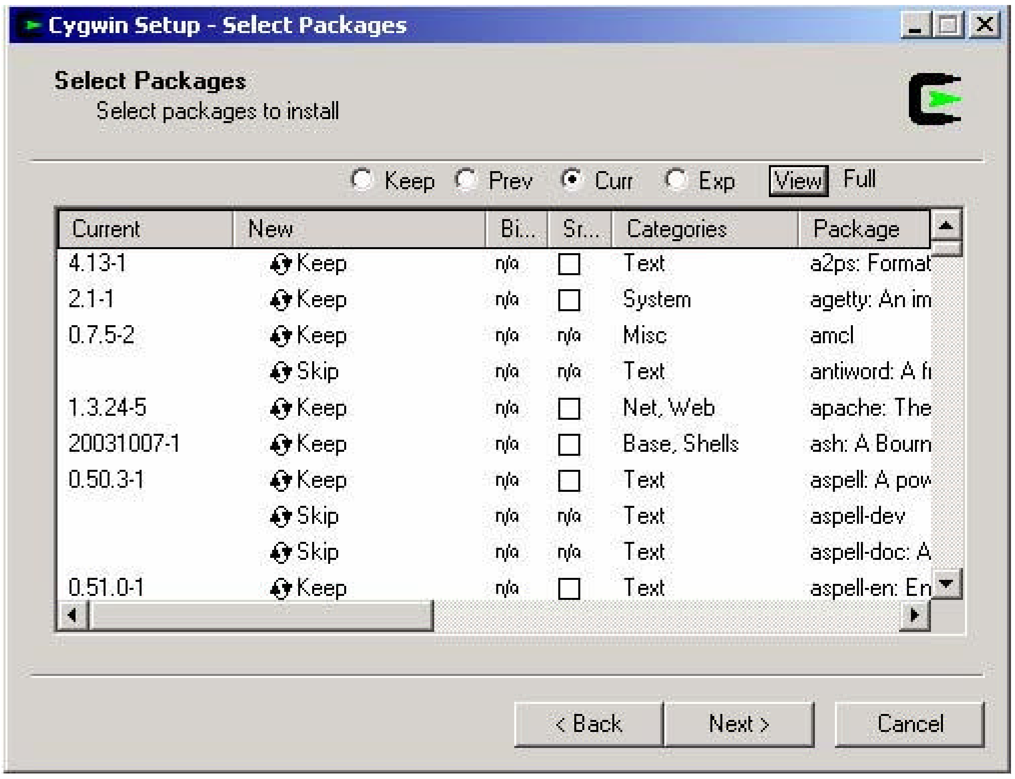
Choose a site from this list, or add your own sites to the list

Available Download Sites:

ftp://sigunix.cwru.edu
ftp://sunsite.cnlab-switch.ch
ftp://sunsite.dk
ftp://sunsite.ms.mff.cuni.cz
ftp://videl.ics.hawaii.edu
http://cygwin.dp.ua
[✓ http://cygwin.get-software.com]
http://cygwin.mirrors.pair.com
http://cygwin.thehostedbox.com
http://ftp-stud.fht-esslingen.de
http://ftp.eq.uc.pt
http://ftp.gwdg.de
http://ftp.iitm.ac.in

User URL:

Picture 20. Installing Cygwin /15/.


Cygwin Setup - Select Packages

Select Packages
Select packages to install

Keep Prev [✓ Curr] Exp View Full

Current	New	Bi...	Sr...	Categories	Package
4.13-1	Keep	n/a	☐	Text	a2ps: Format
2.1-1	Keep	n/a	☐	System	agetty: An im
0.7.5-2	Keep	n/a	n/a	Misc	amcl
	Skip	n/a	n/a	Text	antiword: A fi
1.3.24-5	Keep	n/a	☐	Net, Web	apache: The
20031007-1	Keep	n/a	☐	Base, Shells	ash: A Bourn
0.50.3-1	Keep	n/a	☐	Text	aspell: A pow
	Skip	n/a	n/a	Text	aspell-dev
	Skip	n/a	n/a	Text	aspell-doc: A
0.51.0-1	Keep	n/a	☐	Text	aspell-en: En

< Back Next > Cancel

Picture 21. Installing Cygwin /15/.

This is the most important step. Click the View button until you have this form in the list box. Select four packages from the list:

gcc: C compiler
gcc: C++ compiler
binutils
make

You might also want to check out if these files have been selected, but usually it is not necessary since these are installed by default.

ash
autoconf
autoconf-devel
autoconf-stable
automake
automake-devel

automake-stable

base-files

base-passwd

bash

cpio

cygwin

diff

diffutils

file

fileutils

findutils

flex

gawk

gdbm

grep

gzip

less

libiconv2

libintl1

libintl2

libncurses5

libncurses6

libreadline4

libreadline5

login

m4

mingw-runtime

mktemp

ncurses

patch

pcre

perl

readline

sed

sh-utils
tar
termcap
terminfo
texinfo
textutils
time
w32api
which
zlib
_update-info-dir

Download and unzip Hitachi-H8 cross-compiler folder into Cygwin's root directory either using WinZip (recommended) or Cygwin's own unzip tool. If you get errors while trying to run the compiler later, try to unzip it other way than you did before:

http://www2.acae.cuhk.edu.hk/~tliu/LegOS_install/2000r1_i686-cygwin32-x-h8300-hms.zip

Download and install minimal brickOS and leJOS for BricxCC. Follow the instruction and select your Cygwin folder as its installation directory, you will find a 'brickOS' and a 'leJOS' folder:

http://www2.acae.cuhk.edu.hk/~tliu/LegOS_install/BricxCCbrickOSleJOS.exe

Connect USB tower into the computer.

Launch Cygwin and type the next command lines:

```
cygwin
cd /brickOS
./configure
```

This builds the configuration files and if luck is with you the builded configuration files are correct and everything should work.

brickOS has now been installed to your computer.

/10, 14, 15/.

5.4 Using Cygwin

Now that Cygwin has been installed you can now try to compile example programs. Before trying your program, you should make a new folder under your Cygwin folder like ‘prog’:

```
cd /
cd brickOS
mkdir prog
cd prog
cp /brickOS/demo/makefile ./
```

Now you have made a folder named prog into Cygwin’s root and also copied makefile into this folder. You can now copy a program into the prog folder and compile it:

```
cp /brickOS/demo/helloworld.c ./
make
```

Make command will create helloworld.lx file which can be transferred to the brick with few simple commands, but before you do that you must define which kind of infrared transmitter you are using by this command:

```
export RCXTTY=USB
```

This command isn’t always necessary, but recommended. Next go to util folder where the files that are used to transfer data are located:

```
cd /brickOS/util
```

Then you need to download brickOS into the brick if it isn't already installed:

```
./firmdl3 ../boot/brickOS.srec
```

And now you can send the program into the brick:

```
./dll ../prog/helloworld.lx
```

/15/.

5.5 How to send your own programs into the RCX brick

Make sure the USB tower is connected.

Copy your C program file on the next directory: /brickOS/prog/

Modify the makefile file and change the line:

```
PROGRAMS=helloworld.lx rover.lx linetrack.lx robots.lx c++.lx sound.lx
trailerbot.lx
```

into

```
PROGRAMS=my_prog.lx
```

Launch Cygwin and type next command lines:

```
cd /
cd /brickOS/prog
make
```

```
cd /brickOS/util
export RCXTTY=USB
./dll ../prog/my_prog.lx
```

(This line isn’t always necessary, but it’s recommended.)

Make sure you don’t have a folder with the same name as the file you are compiling or you might receive a message like ‘nothing to be done to file my_prog’.

5.6 How to erase / install firmware?

Erasing the current firmware is a pretty simple to do. All you need to do is to remove the batteries from the brick and then press the red on/off button and that’s it!

After you have erased the firmware you can install another one.

With Cygwin:

```
cd /brickOS/util
./firmdl3 ../boot/brickOS.srec
```

With BricxCC:

From the menu choose Tools and Download Firmware. The brickOS firmware should be located in the Cygwin folder right about here:

C:\cygwin\brickOS\boot\brickOS.srec

If you want to download the original firmware to the brick it should be located at this directory:

C:\Program Files\LEGO Software\LEGO Mindstorms SDK\Bin\VPBrick1\RCX2\
firm0328.lgo

6 EXAMPLE PROGRAMS

6.1 Ir3 by Pavel Petrovic

With ir3 you can send messages to the LCD screen of the brick and also to get PC to receive messages from the brick. You have to use Cygwin installation with ir3, not BricxCC. I was not able to compile it under BricxCC, but if you can find a way to do that then I'll tip my hat to you.

Unzip files to cygwin/brickOS/demo folder.

As it says on the readme file replace the makefile of the demo directory with Makefile.Windows, but don't destroy or erase it since you are going to need it later. Follow the instructions of the readme file and you should be able to send and receive messages. Remember to define that you are using USB port.

If you get tired of always having to define the USB port you can edit the file a little like I did. Just find this line:

```
#define DEFAULTTTY   "/dev/ttyS0" /* Linux - COM1 */
```

And change it into:

```
#define DEFAULTTTY   "usb" /* USB */
```

Voilà! If you have trouble to get the communication to work you should try using the WinLNP application running on the background. I had to use it with Windows XP in order to make the ir3 to work.

/16/.

6.2 RCX Repeater 2 by Pavel Perovic

Rcxrepeater2 is an application for using the RCX brick to control other infrared devices like RoboSapien, DogBot, and RoboRaptor etc. You can also control the motors connected to the brick.

With this you'll get far with just taking a look at the readme.txt file. It also uses the Ir3 application to transmit messages, but is a lot more complicated as you can see by taking a look at the code.

By simple commands you can control the motors.

Example : ir3 --tty=usb mot_A_left_7E

This would make the motor connected to A turn backwards.

So go ahead and give it a try ^_^

/17/.

6.3 Sudhanshu Gaur's real-time interface

With this you have to have the only the original firmware lego0328.lgo in the brick no programs are needed.

Use Visual Studio 6++ to compile the files and after that you should go to command prompt and execute it.

Program is pretty simple and with it you can do three things, make the motor turn forward, backward and make them stop. That's it, have a nice day! ☺

/18/.

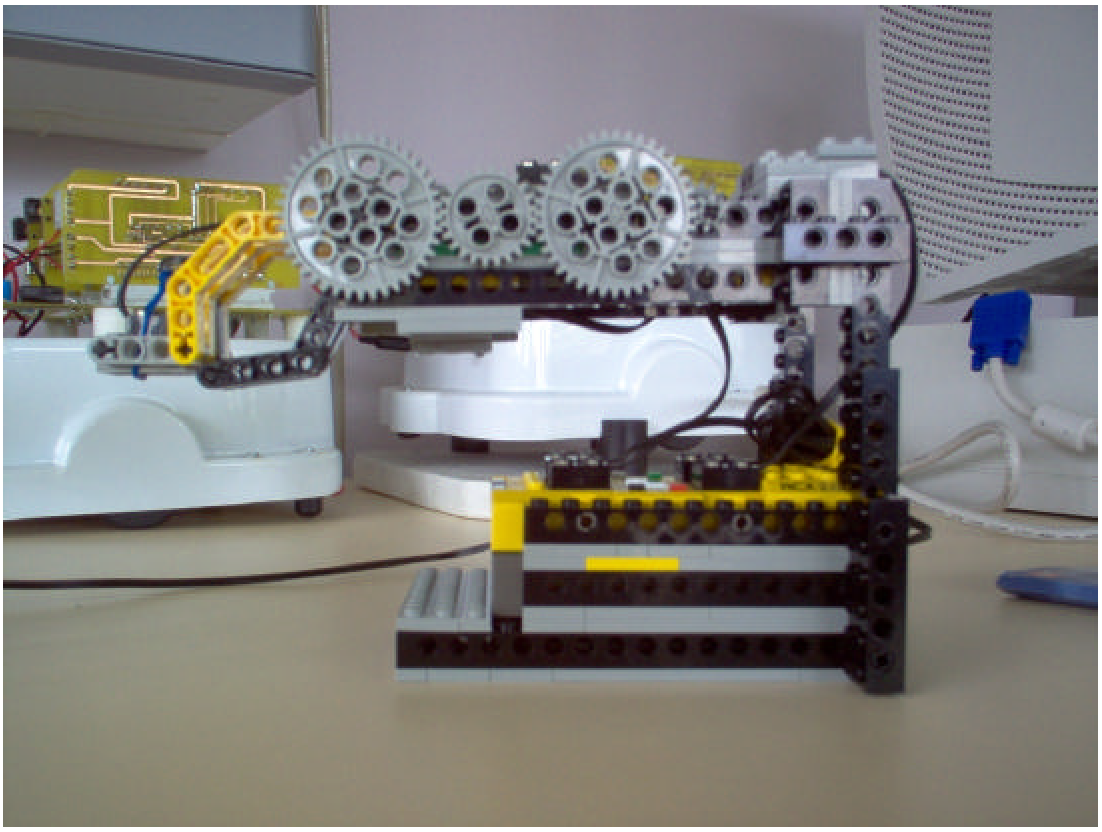
7 WHAT WAS ACCOMPLISH AND WITH WHO?

7.1 Collaboration with others

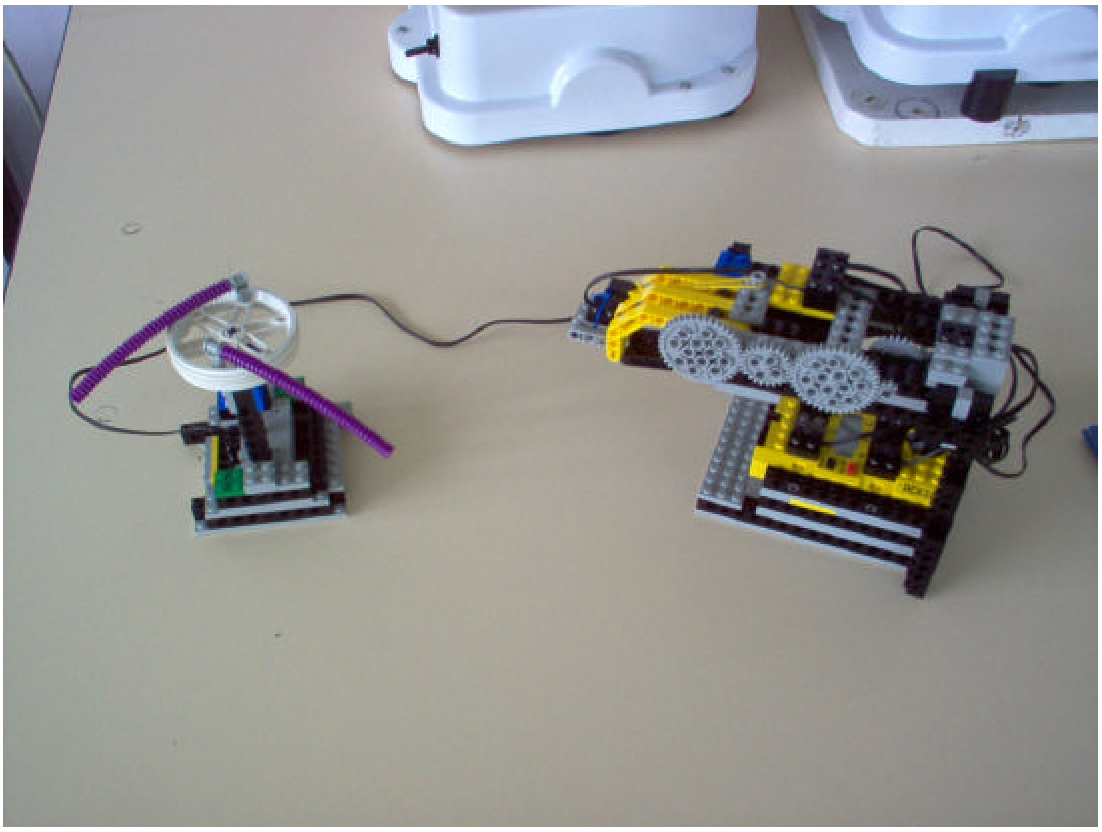
Many people have been helping me with this project. Lionel, Guillaume, Mikko, Mika and Sami have all been very helpful whenever I had trouble with something. Once a week teachers came to see everyone's progress and give us guidance on our work. If something was troubling us, we were always able to ask their help.

Lionel helped me at the beginning when we were studying brickOS and trying to make the Cygwin compiler and BricxCC to work. I got the BricxCC to work on my laptop that had Windows XP and with Guillaum and Lionel we worked together to make the BricxCC to work on Windows 2000. In the end it was thanks to Guillaum's and Lionel's help that we were able to make it work.

They also build autonomous robotic arm able to grasp things without breaking them using a touch sensor (Picture 22). It also had a rotor (Picture 23). The sensor was placed so that when the hand or more specifically fingers closed it touched the sensor thus stopping the motor from rotating and after a while changed the direction of the rotation, started the motor again and opened the hand. The motorhelicop.c program they wrote was made using C language and it has three functions. It plays a tune, reacts to the reading of the touch sensor and rotates motor A and B according to the program (Appendix 3).



Picture 22. Robotic arm with touch sensor



Picture 23. Robotic arm with touch sensor and rotor.

Mika was able to make WinLNP to work by modifying and compiling it on his computer. I tried to do it on mine without success. He also helped me to try and understand out how many of the programs I found from the internet actually worked by explaining the code to me. Also when my computer crashed he helped me to get it fixed so that I was able to get all of my work files from the computer.

Every now and then Mikko was helping me with some of the things on the LEGO project. I can’t be specific since I don’t remember well ^_^;

It was the same on Sami, he helped me with some of the little things and I just can’t remember everything.

7.2 My accomplishments

First I tried to find out if installing brickOS would definitely destroy the original firmware. I wasn’t able to find this out, but Guillaume and Lionel found out that the firmware is erased every time when you just take out the batteries.

I read a lot of brickOS / Mindstorms / and all kinds of documentation and tried to contact many of the people that could answer my questions about brickOS, but without success. Only person that ever replied to my questions was Pavel Petrovic and I would have included his e-mails here but it wasn’t necessary since he just confirmed some of the things that I wasn’t certain of. Thanks for the help Pavel!

A lot of data mining was done also in order to get the information needed. How much you ask? You really want to know? I would like to say a dirty word that would describe it perfectly, but since I know it’s not proper to put something like that into a this kind of documentation I’ll just say a lot ^_^

I banged my head to the wall many times while trying to get Cygwin work. It was difficult because the instructions we found were all different from each other! At some point we also realized that we were using old files and needed to get newer ones. In the end we succeeded to make Cygwin to work on both Windows XP and 2000 operating systems.

While we did the installation to Windows 2000 Lionel wrote an instruction manual on how to make everything work which is included in this document. I did the installation and Lionel wrote the instructions.

I tried to make many programs to work that I found from the internet. Many of them didn't work which was frustrating. Eventually I found Pavel Petrovic's programs which didn't work at the beginning but after the discovery of WinLNP and Mika's help in compiling it I got them to work!

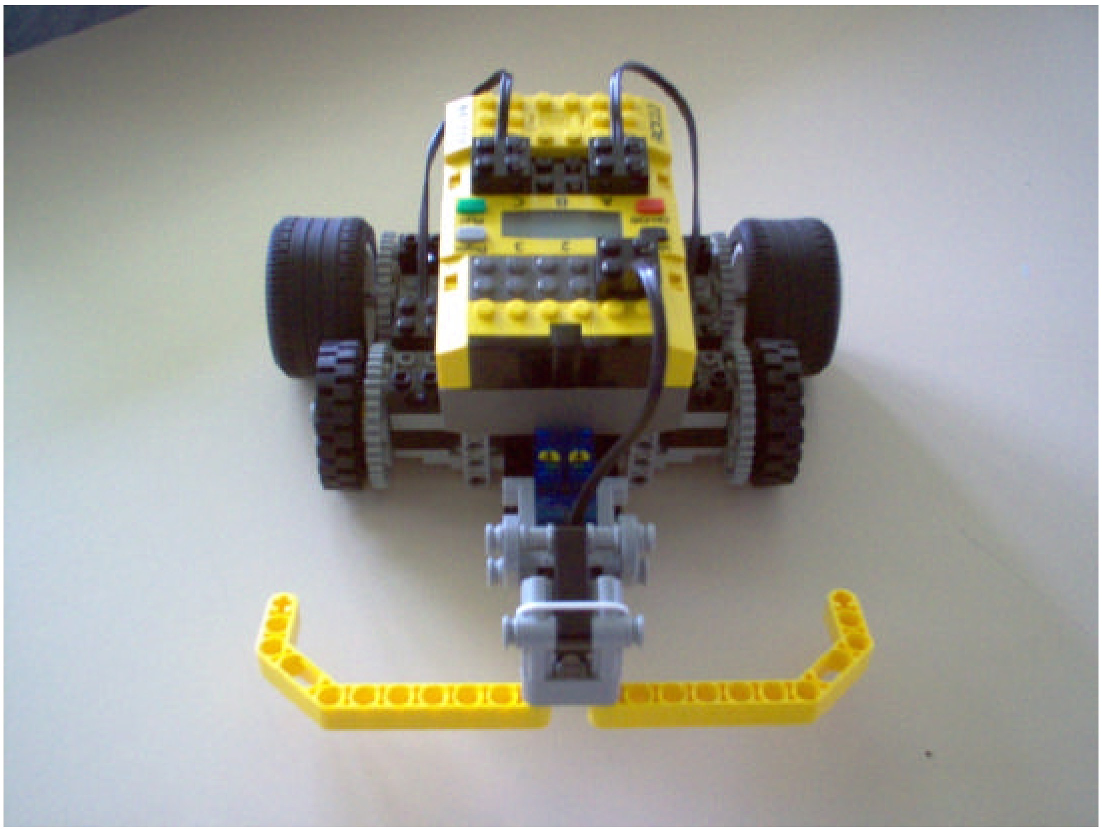
This is what is weird. When trying to run the programs in my XP computer I needed WinLNP to make them work, but when my computers hard drive crashed and I was forced to use Windows 2000, I noticed that I didn't need it at all! I have no idea why.

I did a lot of fighting with my Visual Studio 2005 Express edition also to make it work and in the end I realized I would have been better off with Visual Studio 6. Don't get the Express edition, it sucks.

I continuously complained about the snack machine that we had in the work place. I just don't like ants in my coffee.

I tried to study the codes of many of the programs I had found, but with a little success in understanding them.

I was able to modify the ir3 program so that it wasn't anymore necessary to give command that defines you are using USB infrared tower. Also I modified the talker.lx so that when it receives message of the length of 5 letters the motors start turning so that it goes in circle and if it receives any other message the motors start turning the other direction. I didn't include the code here because it was a simple if else loop. For demonstration I built a LEGO Roverbot (Picture 24) which building instructions could be found from the Robotics Invention System 2.0 Constructopedia /23/.



Picture 24. LEGO Roverbot.

I helped Mikko a little with the translation work of LEGO database from English to Finnish. It was quite difficult because there were a lot of parts that for our knowledge didn't have and never have had Finnish names. So I would like to think us as being pioneers of translating names of LEGO parts to Finnish language ^_^

7.3 Suggestions for future development

What I have done here is laid a good foundation for future development. From here you can go to any direction you want. Here are couple of pointers and suggestions to the people who will continue this project next year.

1. Try to find out if there is a way to compile the ir3 and other applications on BricxCC, I couldn't do this and it bothers me. Having to use two versions of

Cygwin is a pain in the ass and getting the BricxCC to work would make a lot of things easier.

2. Find out if WinLNP is really necessary. When I was using XP I couldn’t make the ir3 program work without it but with Windows 2000 I didn’t need it at all so what’s up with that?

3. If possible I think you should work on making the communication to work without the need to use Cygwin. A good example is the program made by Sudhanshu Gaur.

4. Another thing that would be good is to make an application that gives you a good and proper control of the brick. Adding joystick to control the brick and getting a force feedback effect if it bumps into something would be nice, but I think it’s very difficult to do with Cygwin.

5. Using the original firmware and the program of Sudhanshu Gaur it would be easy to make the joystick control with force feedback to work, so that might be something that you could think of.

6. Establishing connection using Bluetooth would be superb, but I believe it is insanely difficult if you have to do it in C or C++. Then again I don’t know how good coders you are ^_^

   Program and instructions on how to make your own Bluetooth device made by Pavel Petrovic and Richard Balogh can be found at their homepage /20/. However in their page they use leJOS on the brick, not brickOS.

I have also made a list of important or interesting web pages you can take a look at (Appendix 1).

8 FUTURE IS NOW

About a month after my job was finished the next generation of LEGO robotics called NXT was published (Picture 25) /22/. Even though I was painfully aware about this at the beginning of my task and would have loved to get my hands on the NXT, it just wasn't possible.

Picture 25. LEGO NXT /19/.

Drool. Technology behind NXT is far superior to RCX and you can see that quite clearly (Appendix 2). Interesting new things are sound and ultrasonic sensors and also Bluetooth wireless communication which were not available with RCX and should give a bigger kick to the old school of RCX users like perhaps the possibility to control NXT with a Bluetooth equipped cell phone.

What comes into programming this baby, it's pretty much the same as with RCX. So take your pick and stick to it. Since LEGO decided to release open source firmware (which was nice) and also software, hardware and Bluetooth developer kits, the NXT package will probably get all hackers jump out of their pants shouting 'JOYGASM'. Cool kick ass gadgets by superior technology, thanks LEGO!

9 THE CONCLUSION

With the help of others I was able to succeed in establishing infrared communication between PC and the RCX brick and sending information to the brick and receiving information from there. Guillaum’s and Lionel’s robotic arm was also a complete success considering how little parts we had to use.

Building Hideaki Yabuki’s CyberArm IV a robotic arm with feedback sensor was one of the ideas we had at the beginning of our placement but it was soon abandoned because of the lack of parts and purchasing them would have been very expensive. This robot and many more can be found from the book of LEGO Mindstorms Masterpieces /23/.

Other ideas I was not able to do because of lack of time, like using force feedback joystick connected to PC to remotely control the LEGO Mindstorms RCX brick using brickOS and also to be able to have force feedback effect when the touch sensors are activated. I leave those things to the people who will be working with this project in the future. I hope you’ll succeed.

I hope you can find what you need to from this documentation. If not, there is always the Internet. Since I have written everything into this documentation for you guys, there is only one thing that I can say if you’ll run into any problems. Bummer.

Even though originally this was not the project I thought I would be doing I don’t regret choosing it. I learned a lot during the three months that I spent in Valenciennes.

This was a once in a lifetime opportunity and I’m glad that I got to experience it.

BIBLIOGRAPHY

1: LEGO. Key Components of Mindstorms RCX brick [online]. [Reference 15.1.2007]. Available: http://www.lego.com/eng/education/mindstorms/home.asp?pagename=rcx

2: LEGO. Input devices for Mindstorms RCX brick [online]. [Reference 15.1.2007]. Available: http://www.lego.com/eng/education/mindstorms/home.asp?pagename=input

3: LEGO. Output devices for Mindstorms RCX brick [online]. [Reference 15.1.2007]. Available: http://www.lego.com/eng/education/mindstorms/home.asp?pagename=output

4: Kekoa Proudfoot. RCX Internals [online]. [Reference: 15.1.2007]. Available: http://graphics.stanford.edu/~kekoa/rcx/

5: SourceForge. brickOS Alternative LEGO Mindstorms OS [online]. [Reference 15.1.2007]. Available: http://sourceforge.net/projects/brickos

6: leJOS. Java for LEGO Mindstorms [online]. [Reference 15.1.2007]. Available: http://lejos.sourceforge.net/faq.html

7: NQC. Not Quite C [online]. [Reference 15.1.2007]. Available: http://bricxcc.sourceforge.net/nqc/

8: pbForth. Alternative firmware for RCX [online]. [Reference 15.1.2007]. Available: http://www.hempeldesigngroup.com/lego/pbForth/homePage.html

9: Markus L. Noga. About brickOS™ [online]. [Reference 15.1.2006]. Available: http://brickos.sourceforge.net/about.htm

10: Cygwin™. Linux-like environment for Windows [online]. [Reference 15.1.2007] Available: http://www.cygwin.com/

11: John Hansen. Bricx Command Center 3.3 (BricxCC) [online]. [Reference 15.1.2007]. Available: http://bricxcc.sourceforge.net/

12: Mark Overmars. RcxCC LEGO robot pages [online]. [Reference 15.1.2007]. Available: http://www.cs.uu.nl/people/markov/lego/

13: Mike Ash. 2000. LegOS HOWTO [online]. [Reference 15.1.2007]. Available: http://legos.sourceforge.net/HOWTO/

14: Paolo Masetti. 2003. brickOS 0.2.6.10 Instructions for installing on Win9x/NT/2000/XP [online]. [Reference 15.1.2007]. Available: http://brickos.sourceforge.net/docs/INSTALL-cygwin.html

15: brickOS Installation Guide - For Windows 98, 2000, NT, XP [online]. [Reference 16.1.2007]. Available: http://www2.acae.cuhk.edu.hk/~tliu/LegOS_install/legOS_install.htm

16: Pavel Petrovic. 2006. Ir3 program [online]. [Reference 15.1.2007]. Available: http://www.idi.ntnu.no/~petrovic/ir/

17: Pavel Petrovic. 2006. RCX Repeater 2 [online]. [Reference 15.1.2007]. Available: http://www.robotika.sk/projects/robsapien/petnrcx.php

18: Sudhanshu Gaur. 2003. Real Time Interface for Lego Mindstroms in VC++ [online]. [Reference 15.1.2007]. Available: http://filebox.vt.edu/users/sgaur/lego.htm

19: Google. Free search engine [online]. [Reference 15.1.2007]. Available: http://www.google.fi/

20: Robotica.sk. 2006. Wireless radio communication [online]. [Reference 18.1.2007]. Available: http://www.robotika.sk/projects/rcxbt/

21: Travel Post. France Valenciennes [online]. [Reference 18.1.2007]. Available: http://www.travelpost.com/EU/France/Nord-Pas-de-Calais/Valenciennes/map/3069076

22: NXT. 2007. LEGO Mindstorms homepage [online]. [Reference 18.1.2007]. Available: http://mindstorms.lego.com/

23: Mindstorms books. Roverbot and CyberArm IV [online]. [Reference 18.1.2007]. Available: http://autonom.ict.pwr.wroc.pl/mindstorms/books/

APPENDIX 1

Web pages important to the project:

Daniel Berger's website which contains information about infrared communication between the brick and PC and also some example programs.

Couple things you should know:

1. I wasn't able to get the example programs to work
2. In the example programs the brick uses original firmware

http://www.kyb.tuebingen.mpg.de/bu/people/berger/mindstorms.html

Focused discussion group for brickOS/legOS. Keywords to find useful information are lnp, WinLNP and anything else you can think of ^_^

http://news.lugnet.com/robotics/rcx/legos/

Homepage of University of Haifa. Bulletin board containing information about many brickOS projects and programs prepared by students. Important files you can download from here are WinLNP USB and WinLNPUSB.reg.

http://math.haifa.ac.il/robotics/student_projects.html

Many interesting things are on the web pages of robotika.sk, go ahead and take a look.

http://www.robotika.sk/maine.php

Other interesting web pages:

Remote Control X by Kevin Gott

This program will allow you to control your LEGO Mindstorms invention with a game controller using the infrared transmitter that comes with the Robotics Invention System set.

What you should know is that I couldn't get this program to work.

http://www.gottcreations.com/software/rmtctrlx/index.php

Here you can find an emulator for LEGO Mindstorms RCX bricks made by Jochen Hoenicke

http://csd.informatik.uni-oldenburg.de/~hoenicke/rcx/

Just a nifty little application that allows you to control multiple bricks with a single infrared tower.

http://www.battlebricks.com/rc/index.jsp

Page containing some information about brick and brickOS

http://www.it.uu.se/edu/course/homepage/realtid/H05/ass2

Pages containing courses on how to program the RCX brick. You can find lessons behind the Course Schedule link.

http://legolab.daimi.au.dk/DigitalControl.dir/

Brick Remote for Linux

http://sourceforge.net/projects/brickrc

APPENDIX 2

NXT technical specifications:

- 32-bit ARM7 microcontroller
- 256 Kbytes FLASH, 64 Kbytes RAM
- 8-bit AVR microcontroller
- 4 Kbytes FLASH, 512 Byte RAM
- Bluetooth wireless communication (Bluetooth Class II V2.0 compliant)
- USB full speed port (12 Mbit/s)
- 4 input ports, 6-wire cable digital platform (One port includes a IEC 61158 Type 4/EN 50 170 compliant expansion port for future use)
- 3 output ports, 6-wire cable digital platform
- 100 x 64 pixel LCD graphical display
- Loudspeaker - 8 kHz sound quality. Sound channel with 8-bit resolution and 2-16 KHz sample rate.
- Power source: 6 AA batteries

APPENDIX 3

```
#include <config.h>
#include <dmotor.h>
#include <conio.h>
#include <unistd.h>
#include <rom/system.h>
#include <dsensor.h>
#include <unistd.h>
#include <dsound.h>
#include <tm.h>

int i;
int j;



static const note_t robots[] = {
  { PITCH_D4,  2 } , { PITCH_C4,  1 } , { PITCH_D4,  1 },
  { PITCH_F4,  1 } , { PITCH_D4,  1 } , { PITCH_D4,  2 },
  { PITCH_F4,  2 } , { PITCH_G4,  1 } , { PITCH_C5,  1 },
  { PITCH_A4,  2 } , { PITCH_D4,  2 } , { PITCH_END, 0 }
};

int son(int argc,char *argv[]) {
  while (!shutdown_requested()) {
    if (wait_event(dsound_finished,0) != 0)
            dsound_play(robots);
  }
  return 0;
}



int helicop()
{
 motor_c_speed(0);
 motor_c_dir(1);
 while(!shutdown_requested())
 {
  motor_c_dir(1);
  for(j=0; j<255; j++)
  {
   motor_c_speed(j);
   msleep(2500/(j+1));
  }
  for(j=255; j>0; j--)
  {
   motor_c_speed(j);
   msleep(2500/(j+1));
  }
  motor_c_dir(2);
  for(j=0; j<255; j++)
  {
   motor_c_speed(j);
   msleep(2500/(j+1));
  }
  for(j=255; j>0; j--)
  {
   motor_c_speed(j);
   msleep(2500/(j+1));
  }


 }
 return 0;
}

int pince()
{
 cputs("GO");          // write a message on tle LCD display
 sleep(1);             // wait 1 second
 cls();                // clear the LCD display
 motor_a_speed(100);   // Define speed of motor a between 0 to 255
 ds_passive(TOUCH_1);  // Define sensor one mode (passive or active mode)
 ds_passive(TOUCH_3);
 i=1;                  // direction of motor
 motor_a_dir(i);       // motor turns in i direction (0=free wheel, 1= fwd, 2=rwd, 3= brake)

 while(!shutdown_requested()) //sub function "shutdown_requested()" defined in brickOS
                              {
                               if (TOUCH_1==1)
                                 {
                                  motor_a_dir(3);     // set brake
                                  sleep(3);           // wait 3 second in last motor's state
                                  if(i==1)            // reverse direction of motor
```

```
                                        {
                                         i=2;
                                        }
                              else i=1;
                              motor_a_dir(i);
                              msleep(300);         // wait 300 milliseconds => to lost the sensor information
                             }
                           if (TOUCH_3==1)
                             {
                              motor_a_dir(3);
                              sleep(6);
                              if(i==1)
                                         {
                                          i=2;
                                         }
                              else i=1;
                              motor_a_dir(i);
                              msleep(300);
                             }
                           }
 return 0;
}

void main()
{
 execi(&helicop, 0, NULL, 1, DEFAULT_STACK_SIZE);
 execi(&pince, 0, NULL, 1, DEFAULT_STACK_SIZE);
 execi(&son, 0, NULL, 1, DEFAULT_STACK_SIZE);
}
```