Vienna University of Technology
Institute of Computer Engineering

Bachelor's Thesis

LRBJOP: A Lego Robot Controller PCB for the Java Optimized Processor

Alexander Dejaco
alexander.dejaco@student.tuwien.ac.at

Peter Hilber
peter.hilber@student.tuwien.ac.at

September 30, 2007

Abstract

Lego Mindstorms is a robotics invention system of the *Lego Group*. It was intended for children, but turned out to be a great toy for adults, too. It combines electric components like sensors and actuators with Lego bricks and *Lego Technic* parts, such as gears, wheels and axles, to build robots and other automated or interactive systems.

The Lego invention systems have a powerful infrastructure which makes it easy to construct various kinds of mechanical and electric systems, like robots. However, *Lego Mindstorms* originally had very limited program capabilities (e.g. no usage of variables, expressions and function calls in the RCX code). Therefore, a more powerful processor was desirable.

In this project, we use *Lego Mindstorms* sensors and actors, but build our own Printed Circuit Board to use the *Java Optimized Processor* (JOP) [Sch05b] as a central processing unit instead of the Lego RCX. The result is a PCB which provides everything needed to control robots with JOP, and has some additional features to play with, too.

[0]All trademarks and registered trademarks are the property of their respective owners.

Contents

List of Figures

Acknowledgements

We would like to express our gratitude to the people of the Real Time Systems department of the Institute of Computer Engineering, for their support and interest in this project.

We would also like to thank Martin Schöberl for the help provided throughout the project, since a successful completion would not have been possible without his guidance.

1 Introduction

At the Real Time Systems department of the Vienna University of Technology, Martin Schöberl has started a project combining *Lego Mindstorms* and an FPGA running *JOP*, the Java Optimized Processor [Sch05b]. This project interfaces sensors and motors of the *Lego Mindstorms* series with an FPGA substituting the Lego RCX[1].

The RCX is a programmable, microcontroller-based brick for the *Lego Robotics Invention System* [2]. The primary goal of this project is to design and build a Printed Circuit Board which provides access to Lego Mindstorms sensors and actors, and contains JOP, running on an FPGA, as central processing unit.

Figure 1: The JOP FPGA board.

[1]Robotic Command eXplorer

[2]In 2006, the *Lego Mindstorms NXT Robotics Toolset* was introduced (including a new *NXT* brick), which is to replace the *Lego Robotics Invention System* product line, and provides new sensor interfaces not covered by our project and the resulting board. See `http://mindstorms.lego.com`.

1.1 Lego Mindstorms

Lego, or the Lego Group, is a well-known producer of building toys. Lego toys are typically based on interlocking plastic bricks. *Lego Technic* is a product line of Lego which allows to build more complex models with movable parts. In addition to bricks, pieces such as gears, axles, beams, and pneumatic parts are also included.

Lego is very popular and known around the world. Its story begins in the Thirties, when Ole Kirk Christiansen started a small business in Denmark, which among other things, created wooden toys, from which the today's LEGO products derive. The name Lego derives from the Danish words "LEg GOdt", meaning "play well".
The *Lego Technic* product line, featuring advanced models with movable parts, was introduced in 1977.

Lego Mindstorms is a robotics build framework. The Lego Mindstorms *RCX* product line was introduced in 1998. In 2006, it was replaced by the *NXT* product line, which features various improvements[2]. Lego Mindstorms combines Lego bricks, and Lego Technic parts, and electric components. The system is built up around a microcontroller brick, called *RCX* brick, respective *NXT* brick. Electric components include motors and various sensors. These can be connected to the RCX/NXT using cables with interlocking bricks at their end.

Lego Mindstorms is, interestingly, also popular among adults.

1.2 Lego Mindstorms Limitations

While Lego Mindstorms allows easy construction of robots, its hardware and software capabilities are somewhat limited.

The RCX contains an Hitachi H8 microcontroller with 32K external RAM [Hit], providing three actuator output-, and three sensor input-ports.

Programs for the RCX could initially only be developed in a graphical programming environment. No use of variables, complex expressions, and function calls were possible therein.

However, successful reverse engineering allowed third parties to replace the firmware. Many programming languages have been made available for the Lego Mindstorms microcontroller. An example is *leJOS*[3], a Java Virtual Machine for the RCX, which is used to introduce students into programming.

The goal of our project was not to replace the firmware, but to replace the RCX microcontroller by an FPGA running JOP. The RCX brick does, however, not only contain the microcontroller, but e.g. H-bridges, buttons, and an LCD display too. The RCX should therefore be replaced by a Printed Circuit Board.

1.3 Goals

The first goal of our project was to create a PCB layout, with interfaces to Lego sensors, Lego motors and possibly other devices, such as a microphone, a camera, etc.[4]

The power consumption of the board should be low, and its size not too big for integrating the board into a robot. Simple and robust solutions to attach the board to a robot consisting of Lego bricks and to connect multiple Lego sensors and actors to the board, needed to be found. On the FPGA side, a suitable API for the JOP processor was desired. Real-time enabled Java, as provided by JOP, could then be used to program Lego robots.[5]

The resulting PCB should be suitable for use in university courses about embedded real-time systems and the Java Virtual Machine in hardware.

The second goal was to build a sample Lego robot using the board and the JOP processor, to proof the concept. This robot should also be suitable for demonstration purposes of JOP.

[3]See `http://lejos.sourceforge.net/`

[4]The Lego Mindstorms sensor interface had already been analyzed by Schöberl [Sch05a], and others.

[5]It remains of course possible to use other processor designs, or only a hardware description language, with the board.

1.4 JOP - The Java Optimized Processor

JOP is an implementation of the Java Virtual Machine in hardware [Sch05b]. It is being developed by Martin Schöberl at the Vienna University of Technology.[6]

JOP is a simple and small processor, which can be implemented in a low cost FPGA. It features predictable execution time for embedded real-time systems. JOP is open-source and free for education, research and personal use.

[6] See `http://www.jopdesign.com/`

2 Related Work

2.1 Systronix JCX

The Systronix *Java Control System* JCX[7] is a very similar project by a commercial vendor. Multiple boards replace the Lego Mindstorms RCX and may be combined according to the users needs. Interestingly, the CPU used is the aJile Java processor [Har06], which is also a JVM in hardware. The JCX targets schools and universities, among other interest groups.

2.2 Robots and robot circuit boards

There is a very large community of robot builders, and therefore a lot of robot circuit boards can be found. For instance, very popular boards used to build all kinds of robots are available at robotikhardware.de[8]. A nice source for robot kits, components, motors, and even solar-powered robots is solarbotics.com[9]. As for comparable, non-Lego-based robots, there is a lot of scientific work available. For example, there is a very nice robot built by David P. Anderson at Southern Methodist University Dallas, Texas. It is named nBot, and is a two-wheeled balancing robot [10]. Similar work has been done by Rich Chi Ooi at the University of Western Australia, where he examined the suitability of different controller schemes to guarantee stability of a balancing robot. Originally, the demonstration robot was aimed to be another two-wheeled balancing robot (see section 7).

Another project we took interest in, is [PBN03]. It is an effective educational robot with indoor/outdoor capability, back-EMF speed control and a Java programming interface. For the idea of the Inverted Pendulum Robot, we related to [Mig99], a diploma thesis, which shows real-time feedback control programming for balancing an inverted pendulum.

[7] See `http://jcx.systronix.com/`
[8] See `http://www.robotikhardware.de/`
[9] See `http://www.solarbotics.com/`
[10] See `http://www.geology.smu.edu/~dpa-www/robo/nbot/`

2.3 RCX related projects

Regarding the Lego-robot community, there also exist a lot of projects of reverse engineering or substituting the RCX, and a lot of robots of course. An example for scientific work regarding reverse engineering the RCX, is [Pro98]. A lecture for a seminar at Stanford, where the capabilities and limitations, including details about RCX's internal architecture, are discussed.

An interesting replacement firmware for the RCX is LejOS. It is a Java based alternative (and more powerful) operating system which can be used in the RCX to control Lego robots [11]. A similar project exists for a C and C++ programming environment, it is an open source operating system for the Lego RCX, named BrickOS [12].

2.4 Lego part evaluations

An important source of information regarding the Lego components, especially about the Lego motors internals, are the evaluations and comparisons done by Philippe E. Hurbain[13].

2.5 Miscellaneous

In the process of research for the requirements of constructing an inverted pendulum- or a two wheeled balancing robot, we came across an analysis of the control behavior of a two-stage inverted pendulum by Weijing Zhang [Zha96], using a fuzzy controller. Other notable related sites/projects are roboternetz.de[14], which is a german robotics and electronics community-site with a lot of interesting discussions and projects, the popular german electronics forum elektronik-projekt.de[15], and the Toy Robots Initiative [16], which operates at the Robotics Institute at Carnegie Mellon University Pitts-

[11] See `http://lejos.sourceforge.net/`
[12] See `http://brickos.sourceforge.net/`
[13] See `http://www.philohome.com/motors/motorcomp.htm`
[14] See `http://www.roboternetz.de`
[15] See `http://www.elektronik-projekt.de/`
[16] See `http://www.cs.cmu.edu/~illah/EDUTOY/`

burgh, which aims to commercialization of robot technologies used in education, entertainment and art, are also useful resources.

3 Lego infrastructure and parts

3.1 Lego

The Lego infrastructure is well thought through, and very easy to use. Nevertheless, very much can be achieved with the *Lego Technic* parts in combination with *Lego Mindstorms*. To interface the *Lego* systems with another controller than the standard RCX, it was necessary to evaluate the sensors and actuators at the electronical level.

3.1.1 How Lego sensors work

The Lego sensors need a supply-voltage, which is switched off periodically, with *period T*, for the *off-time t*. T is 3ms in the RCX and t is approximately 0.2ms. So, the power is switched off for about 200 us. During this time, the resulting current sink provides the output signal.

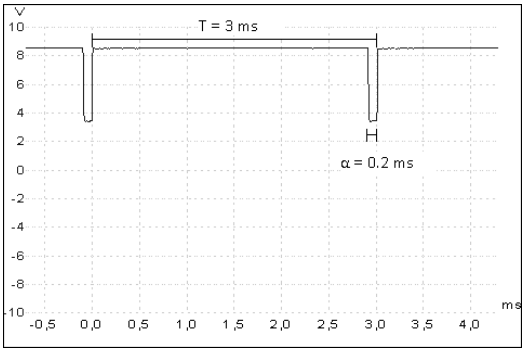

Figure 2: RCX sensor evaluation.

3.1.2 Lego motors

The Lego motors are simple 9V brushed direct current motors. They are suitable for voltages up to 12V and contain a protection diode, which serves two functions. Firstly, to prevent that a motor can be driven with a voltage higher than 15V, and secondly, to ensure, that no, by the motor generated current flow, can be applied to a possibly attached circuit. Since Lego motors

are normal DC-motors, they can act as generators, which can be used for back-EMF reading.

3.1.3 Lego cables

The Lego cables are a robust way to connect all electronic Lego components in a simple and foolproof way. They are constructed in such a way, that regardless how a component or another cable is connected, there can never occur a short circuit just by connecting in the wrong orientation.

3.2 Preliminary work

The work done in this Bachelor's thesis consisted mainly in the design of the PCB, and the related software. The Lego Mindstorms sensor interface had already been analyzed by Martin Schöberl [Sch05a], among others. He also realized a sigma-Delta analog/digital converter for Lego Mindstorms sensors. This and other circuits diagrams were taken from previous projects[17].

Using a prototyping board, a simple Line-Follower robot had also already been constructed[18].

[17] The circuit we used for the microphone input was designed by Jens Kristian Rasmussen and Mikael Lundsgaard under supervision of Martin Schöberl.

[18] See `http://www.jopdesign.com/board.jsp#simpexp`.

4 Technical documentation

4.1 Overview

On the one hand, LRBJOP was designed to use low power and to interface with Lego hardware to be used as a robot controller. On the other hand, LRBJOP has a lot of features and a big prototyping area, so that it can also be used for evaluations and prototyping.

The complete schematics can be found in section D.

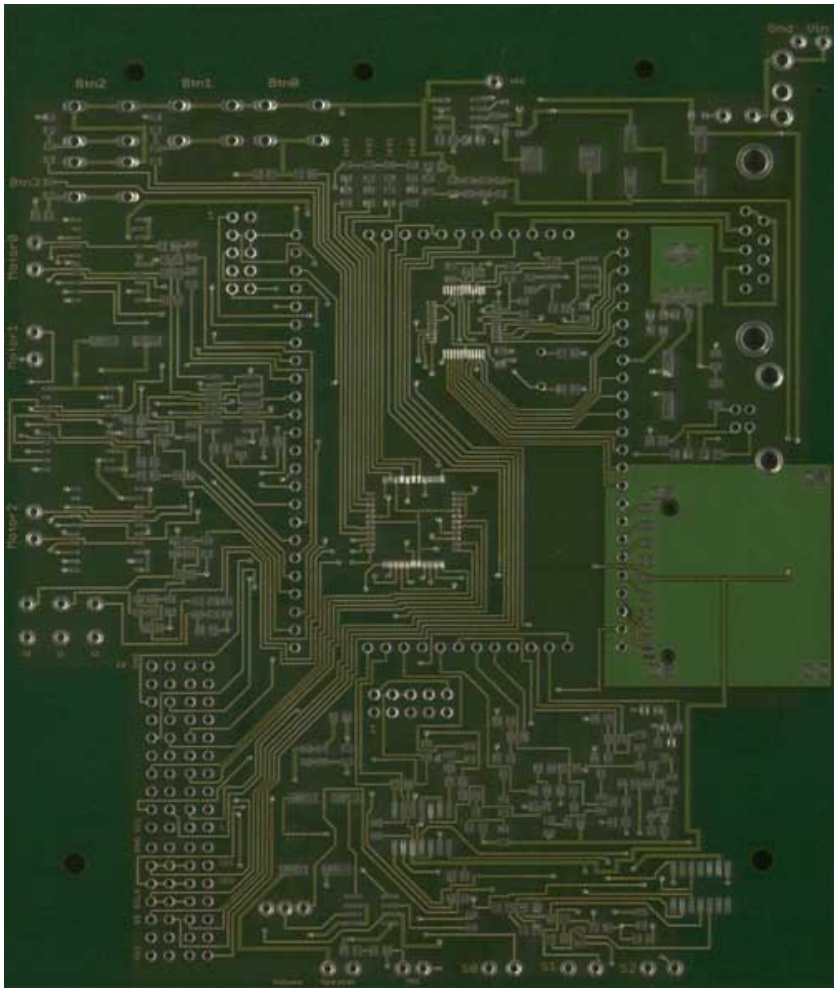

Figure 3: The Printed Circuit Board.

4.1.1 Board details

- Dimensions of the board: 140 mm x 116 mm.
- Layers: 2, top and bottom layer.

4.1.2 Using LRBJOP

There are a few important things to know before using our board.

Using our pre-built board:

We suggest using USB to upload the FPGA design and the JAVA program. The board supply must be connected prior to attaching the USB connector.

Constructing your own board:

The board design for the LRBJOP is open-source and can be found at our site[19], as well as a complete part list with ordering information. The part list can also be found in the attachment (See F).
When the production of the circuit board is complete, and you have soldered all the components, the FPGA board should not be connected, yet. It is recommended to program the EEPROM of the FTDI USB Chip and the PLD, before.

The PLD is used to provide additional pins, it should be programmed before the FPGA board is inserted to prevent that output pins might be connected to output pins. To program it, the Quartus project *pld.qpf* should be used to compile and upload the design.

To program the EEPROM, it has to be flashed via USB using a programming-tool. The tools, the needed drivers, as well as a detailed description how to install them, are available at our site[19]. The installation manuals can also be found in section E.

[19]See `http://stud3.tuwien.ac.at/~e0327019/lego/`

Finally, the FPGA board can be connected, and the board is ready for usage.

4.2 Circuits and Components

A complete part list is available at *http://stud3.tuwien.ac.at/~e0327019/lego/* and in section F.

4.2.1 Power Supply

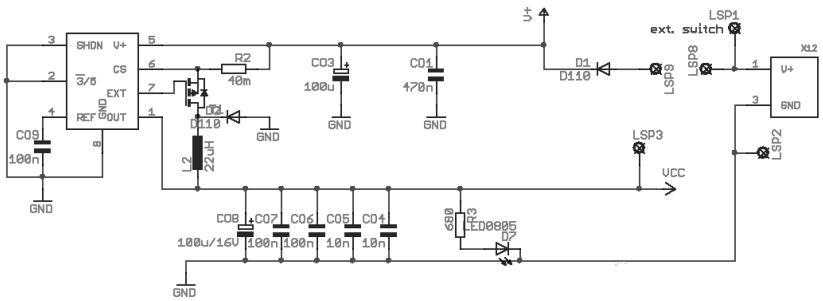

Figure 4: Power supply schematic.

The voltage regulator which is used is a MAXIM 1626 High-Efficiency, Step-Down DC-DC controller. The output voltage of the MAX1626 is preset at 5V or 3.3V. In our case VCC is set to 3.3V. The MAX1626 switching controller provides efficiency greater than 90% between 3 to 2000mA.

The input voltage can be up to 16.5V, although this counts only for the DC-controller. There are some components which are connected directly to the unregulated input voltage. This means, the connected voltage should not exceed 12V.

The DC-controller is able to generate VCC (3.3V) until the input voltage drops below 3,7V. The high-voltage components however, need at least 5V to function correctly. This mainly concerns the amplification component for the speaker output.

In addition, some Lego sensors require even higher voltage. For instance, the Lego acceleration and tilt sensor needs 9V supply, and works well only with an external supply, or a fully charged battery pack.

Circuit protection is supplied by a Schottky barrier diode, providing reverse voltage protection up to 30V.

We suggest to use either a regulated power supply providing 9V, or a powerful rechargeable battery pack, since the circuits are designed for 9V supply, the current drain is supposed to be minimal at this voltage.

4.2.2 Sensors

Sensor inputs with ADC:

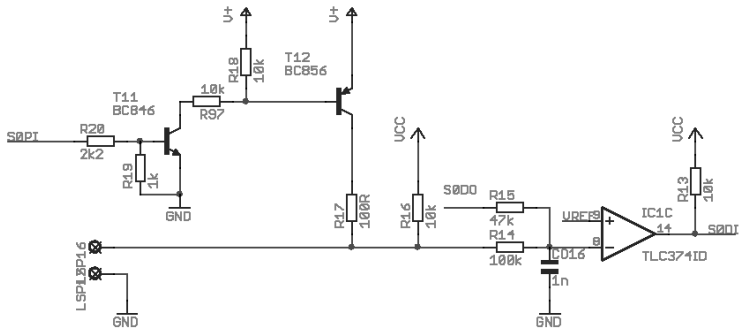

Figure 5: Sensor input with ADC schematic.

On basis of the original schematics from the prototype, we implemented three sensor inputs with ADC, which can be used to either connect Lego sensors, or, for instance, a potentiometer. Each circuit consists of two parts, one contains two transistors (a PNP and a NPN transistor) which are switched by the FPGA to power the connected sensor with the unregulated voltage V+, and some resistors. The second part is the integrator, formed by a comparator, two resistors and a capacitor.[20]

Simple sensor inputs:

Also based on the original schematics, we implemented three simple inputs which for instance could be used to connect switches or buttons, like

[20]See [Sch05a] for the original evaluations.

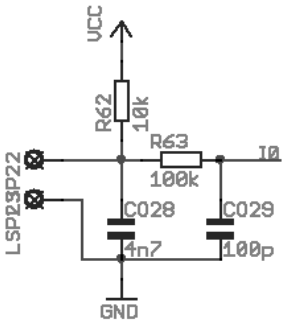

Figure 6: Simple sensor input schematic.

the Lego push buttons. Each input is composed of two solder pads, one connected to ground, the other one is driven at VCC through a 10k resistor, two capacitors to ground and one 100k resistor which leads to an input pin on the PLD. Therefore the simple inputs are low-active, meaning that if there is no connection between the two pads, the digital input signal recognized by the PLD will be high, and otherwise low. The passive low-pass filter consisting of the 100k resistor and a 100pF capacitor reduces response to high frequency inputs.

4.2.3 Microphone and Speaker Circuits

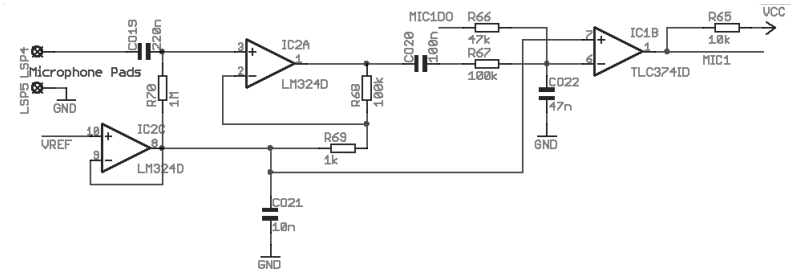

Figure 7: Microphone input schematic.

The microphone circuit is based on the work done by Jens Kristian Rasmussen and Mikael Lundsgaard under supervision of Marin Schöberl. It amplifies the microphone input and then passes the signal to an integrator,

which works exactly like the other sensor inputs.

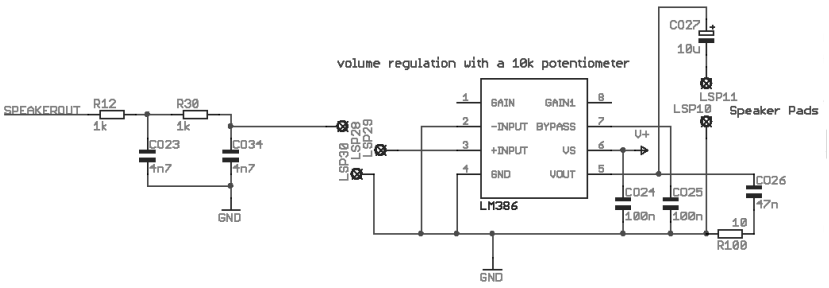

Figure 8: Speaker output schematic.

For the speaker output we used the LM386, a low voltage Audio Power Amplifier. Its an amplifier with low quiescent current drain, designed for battery operation. At the input we connected a 10k potentiometer to be able to regulate the volume. The amplification factor is 20x. It could be increased to 200x by adding a 10uF capacitor between pins one and eight of the amplifier, but this should not be necessary if the supply voltage is high enough. The LM386 can operate at a supply voltage between 4 and 12V. In our case it is powered by the unregulated input voltage V+.

4.2.4 LEDs and Buttons

We added a power LED, which shows if the board is powered. If this LED is not shining although the board is connected, you might want to check the polarity of the input voltage.
There are also four free programmable LEDs, which are switched by the FPGA through the PLD.
For manual inputs, there are four free programmable buttons, which are also connected to the FPGA through the PLD.

4.2.5 Motor driver

There are three HIP 4020 MOS power output H-drivers (Full-Bridge) on our board [Int]. These components show a low standby current drain, have

over-current limit and over-temperature shutdown protection. They feature direction, braking and PWM control. They are operational at a supply range of 3V to 12V. The circuit on the board is designed such that each motor connected to a H-bridge can be steered with three signals: enable, direction and break.

4.2.6 Back-EMF speed measurement

The principle behind back-EMF reading is the Lorentz-force. Whenever a conductor is being moved in a magnetic field, the Lorentz-force inducts a current flow in it, by applying a directed force to the electrons. This is what happens in DC-motors. When such a motor is spinning, the Lorentz-force inducts a current-flow in the connected conductor, when the connections are left floating by the circuit.

That effect is used to measure the rotation speed of DC-motors, by disconnected them from the driving circuit periodically, leaving the pins floating, and in that time, reading the voltage which is generated because of the current-flow in the circuit. The analogue voltage level is converted to a digital value by a sigma-delta ADC. The measured voltage level is proportional to the rotating speed.

In our case, there are back-EMF reading circuits for two of the three possible motor drivers. There are two sigma-delta ADCs for each motor, since we wanted it to be possible to identify the direction in which the motors are rotating. Each one of the converters samples a specific direction.

4.2.7 PLD Device

The Programmable Logic Device is a EPM3064 which connects the buttons, LEDs, SPI signals and unused wires for future use to the JOP processor. The PLD extends the range of available input pins so that all the features can be implemented.

What the PLD does, is basically shifting the bits received by the FPGA over the data line, out to the appropriate pins (e.g. LEDs), and the same

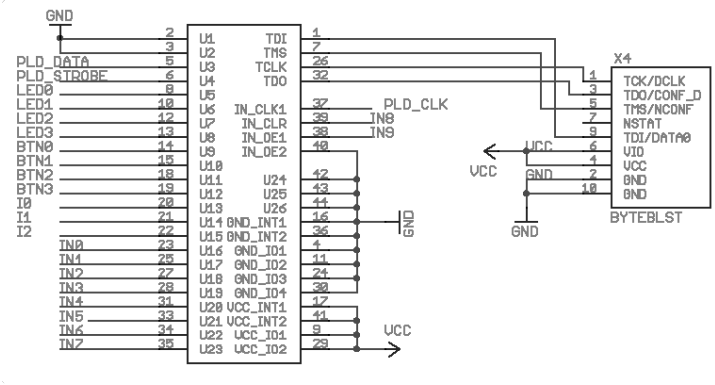

Figure 9: PLD schematic.

in the other direction: it also uses the single data line to communicate the state of the inputs (e.g. buttons) to the FPGA, by driving the appropriate serial bit sequences.

4.2.8 Connectors

There are various connectors fit on the board. First of all, the main connector, which is recommended to upload the FPGA design and the Java program, is the USB interface. It consists of the USB plug, which is connected to the processor through a FT2232C FTDI chip. Secondly, there is a 9 pol serial connector, which is wired also to the FPGA. It is the default choice for uploading the Java program, if using USB is not possible or not desired. In that case, the FPGA design has to be uploaded through a ByteBlaster cable at the FPGA board, directly. The connector used for that matter is X7. Furthermore, there is a SD-Card slot, a grid of solder pads intended for any future usage. Some pads are wired to ground, some to VCC, and some to the PLD which routes the signals to the processor. The detailed assignment can be picked out from the Eagle design files.

Finally, there are also connectors for the ByteBlaster, used to program the PLD and the FPGA. X4 is the one wired to the PLD, X7 is wired to the USB chip, and is connected though a external cable to the FPGA board,

when using USB.

4.2.9 Solder pad grid

The grid of solder pads can be used to add new components in the future. There are unused signals from the PLD connected to some pads for communication. In addition, there are also wires for up to three SPI devices.

4.3 Extensions to the FPGA design

This section deals with how the board components are connected to the JOP softcore processor running on the FPGA.

4.3.1 The SimpCon protocol

The customized interface to the board is written, like JOP itself, in VHDL. It is connected to the JOP processor, in the same style as other components, as a single SimpCon slave (with no pipelining). SimpCon is a simple standard for system-on-chip interconnect [Sch07] and used to connect other components such as a UART or a FPU to JOP.

4.3.2 Analog sensors interface

Similar sigma-delta analog/digital converters are used to read Lego Mindstorms sensors (see 3.1.1), a microphone, and to do back-EMF measurement of the motor rotation (see 4.2.6).

The major part of a sigma-delta ADC is realized in hardware (see 4.2.2), but a part of the feedback, and the integrating part is calculated inside the FPGA. The software part is rather simple. The digital input is filtered through a majority vote of the last 3 observed values. The inverted value is then fed back to the hardware part. Integration is done by a counter. The Lego Mindstorms ADC software part has an additional power switch, since these sensors are supplied with current most of the time and are only read every few ms.

4.3.3 Motor interface

The VHDL component *lego_motor* is used to steer an H-bridge with a hardware PWM signal, and measure the rotation of a connected motor through its back-EMF motion feedback. Two sigma-delta ADCs are used to measure the back-EMF (see 4.2.6). An introduction to back-EMF measurement can be found at [Inc06].

4.3.4 Audio playback interface

A speaker (see 4.2.3) can be connected to LRBJOP. A corresponding (simple) VHDL module does sound synthesis of PCM audio data using an 8 bit PWM channel. Every sample, the duty cycle is simply set to the audio amplitude from a Java program.

JOP running on the FPGA is fast enough to support a sampling rate of 44.1 kHz[21] by changing the duty cycle at appropriate times through a Java program[22].

4.3.5 Interface to pin extension PLD

The Cycore FPGA board for Altera's Cyclone EP1C12Q240 has 49 pins (power supply pins not counted). This pin count was not enough for the features we wanted to include on the board. A small PLD, the Altera EPM3064A, was therefore added to provide 23 more pins. Three pins - clock line, strobe line (for synchronization) and a bidirectional data line - suffice for a simple serial interface between the FPGA and the PLD providing the additional pins.
Depending on whether the pins are configured as input or output, one pin state is transmitted in the corresponding direction, each cycle. The signals routed through the PLD are digital signals where a variable delay of less than 25 cycles can be tolerated, such as those for LEDs and buttons.

[21] A sampling rate of 44.1 kHz is widely used for uncompressed audio data.

[22] When real-time scheduling is used, the scheduling overhead only allows a slightly slower sampling rate. A more advanced VHDL sound synthesis module could support real-time scheduling by reading more than one PCM value at a time from the processor.

4.4 A Java library for convenient access

The VHDL code that makes the board available to the processor is written as a *SimpCon* slave (see 4.3.1). A SimpCon slave can be accessed in Java by calling the `Native.wr()` and `Native.rd()` methods, located in the `com.jopdesign.sys` package, with special address parameters[23]. Access to the raw interface of the *sc_lego* SimpCon slave requires the user to do some bit arithmetic[24], and other calculations. Direct communication through SimpCon access methods is therefore tedious and error-prone.
The Java package `lego.lib` enables easy and nearly foolproof access from a Java application, hiding the implementation details. All standard LRBJOP features, such as buttons and motors (connected to the board's H-bridges), are represented by specialized classes - `lego.lib.Buttons` and `lego.lib.Motor`, in this case.
These classes contain many convenience methods, which should allow intuitive programming of a robot. E.g. the `Motor` class has a method for setting the PWM duty cycle of the H-bridge as a percentage value.
There is extensive Javadoc documentation on the library[25]. Along with the `lego.lib` package, some example programs making use of the package are included in the JOP source[26].

4.4.1 An example Java program

The example program is a simple line-follower robot, making use of JOP's real time threads, and `lego.lib`. The value of an infrared sensor is read, and compared to a hard-coded value to determine if the robot is currently on the line or not. Based on the outcome of the comparison, the motors are driven.

[23]These method calls are ultimately translated to special bytecodes.
[24]One of the reasons for this is that the address space for a single SimpCon slave is only 4 bits. Therefore, multiple values are made accessible together through a single address.
[25]Javadoc documentation is available online at `http://stud3.tuwien.ac.at/~e0327019/downloads/documentation/`.
[26]The JOP source is available at `http://www.jopdesign.com/`.

An example program using `lego.lib`:

```
import lego.lib.*;
import joprt.RtThread;

public class LineFollower {
        static final int IR_SENSOR = 2; // IR sensor index
        static Motor left, right;

        public static void cycle() {
                int val = Sensors.readSensor(IR_SENSOR);
                boolean black = val > 285; // black or white?
                if (black) { // steer to the left
                        right.setState(Motor.STATE_FORWARD);
                        left.setState(Motor.STATE_BRAKE);
                } else { // steer to the right
                        left.setState(Motor.STATE_FORWARD);
                        right.setState(Motor.STATE_BRAKE);
                }
        }

        public static void main(String[] args) {
                left = new Motor(0);
                right = new Motor(1);
                left.setDutyCyclePercentage(75);
                right.setDutyCyclePercentage(75);

                // invokes cycle() every 20 ms
                new RtThread(10, 20*1000) {
                        public void run() {
                                for (;;) {
                                        cycle();
                                        waitForNextPeriod();
                                }
```

```
                    }
                };

                RtThread.startMission(); // starts the thread
            }
        }
```

5 Implementation

At first, we had to define the capabilities the new board should have. After that, the circuits needed to be designed in Eagle. This was done based on the original schematics of the prototype board and other earlier works provided by Martin Schöberl. Subsequent to that, there was to do a lot of testing of the components and circuits, either on the Simpexp expansion board, or on a prototyping solderless breadboard. This took a lot of time, as our experience with electronic components and circuits was limited. When we finally had decided on which components to use, and all circuits had been tested, we came to the next step: the installation of additional sensors and actuators into Lego bricks, and testing them. For that we had to search our attics for childhood toys. The results were a very nice led array, a microphone and a speaker fitted into Lego bricks (see Figure 10), for easy attachment onto Lego robots.

In parallel, the existing rudimentary sc_lego VHDL interface to the motors and sensors was extended (e.g. hardware PWM for motors, communication with a PLD which provides some more pins), and tested on the Simpexp prototyping board. As a next step, we created the *lego.lib* package, which provides convenient access to the sensors and actors available through the extended VHDL interface.

With time we felt it necessary to make adjustments to the circuits and add other components, such as the LM386 amplifier for the speaker output, or the EPM3064 programmable logic device. Some of the components used on the original board were replace by more efficient and adequate ones (e.g. the H-bridges and the voltage regulator), and some new components introduced. When the schematics were completed, we started the placing of the components, routing of the wires, and finally the production of the first version of the PCB.

In parallel, the VHDL design, which originally ran on the Simpexp prototyping board, was adjusted to run on the PCB. E.g., support for download by USB was added.

When the first prototype board was ready, we had to solder the compo-

nents and connectors to the board, and mount the board onto Lego bricks. Then we did extensive testing of the PCB-parts, correction of wiring and soldering-errors, and replacement of inappropriate components. Optimizations of the placement and routing, which resulted in a much smaller board, were also necessary. After that, we gave the second version of the PCB into production. When it was ready, we soldered all the components and connectors on the new board, retested it, and optimized the VHDL Design.

Finally, we created a Lego robot and some example Java programs for JOP to control the robot, such as a crash avoider, a fall avoider, and a audio playback program. The last steps were to hold a presentation of the project and the robot at the Real Time Systems Department, and write the documentation and this paper.

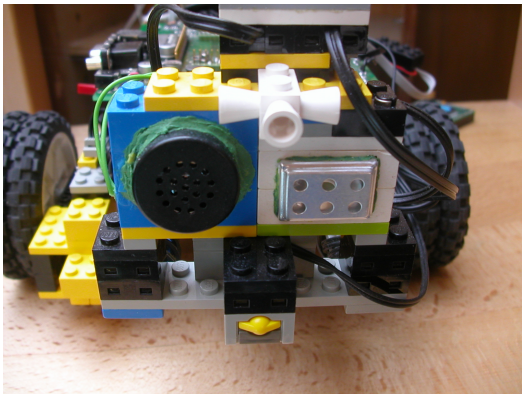

Figure 10: Our self-made speaker and microphone peripherals.

6 Discussion

6.1 Difficulties encountered

Using a $\leq$ comparison for PWM output generation in VHDL caused crashes on Java program startup under some configurations. Using $\equiv$ comparisons instead, which result in similar semantics, made these problems go away.

We found out (thanks to Martin Schoeberl at this point, for his assistance in this matter) that the problem occurred due to wrong ram timing. The code was changed to satisfy the timing constraints of the memory pins, and as a result the problem was gone.

Minor hardware problems were encountered due to wrong wirings and soldering mistakes. Finally, we were able to correct all detected problems. This resulted in a fully functional board with a working interface to the Java Optimized Processor.

6.2 Known issues

There are a few minor issues when using the LRBJOP PCB, detailed in this section.

We also had some trouble with the PLD-communication. The most issues were solved, but there might still be some bit errors now and then. The reason is not fully determined, yet.

6.2.1 Hardware

When downloading the design using USB, the current supply must be connected before the USB cable. Otherwise, JOP will crash after the download. Re-downloading after the crash *may* also fix the problem.

6.2.2 VHDL design

The motor PWM does not scale to the frequency the design is compiled to run on. The PWM frequency is suitable for 80 MHz, and similar frequencies.

6.2.3 Not yet tested

The full range of functions of LRBJOP, and of its VHDL and Java software, was not tested during this project. For the functionality of LRBJOP which was not used, no corresponding VHDL and Java software is provided, yet.

In particular, the SD-Card interface, and the SPI interface were not tested. Both consist only of wirings, however.

6.3 Possible improvements

Auto-negotiation of the direction for pins on the pin extension PLD would be feasible. As of now, the direction is set through the lego_pld_pack VHDL package shared by the PLD and the JOP design.

A better protection of the fronts side of the PCB (when part of a Lego robot) is also desirable. This could be attained by attaching a superstructure.

7 Demo robot

While the primary goal of the project was to design the PCB and the related software, a secondary goal was to build an example robot as a proof of concept and for demonstration purposes.

In early stages of the project, the intention was to build a two-wheel balancing robot. This seemed an impossible mission, given the relatively feeble Lego motors. Therefore the goal was altered to build a similar, but less challenging inverted pendulum balancing robot. By moving forth and back, the robot should be able to balance a long pole. The bottom of the pole is attached through an axle on the robot.

Many variations of both a PID controller and a Fuzzy logic control system, and of the robot itself, were tried. Only Lego and Lego Mindstorms parts where used. The deflection of the pendulum was measured using a tilt sensor and a potentiometer coupled to a moving axle to which the pole was attached.

Ultimately, no control system was able to balance a deflected pendulum for prolonged periods of time. We believe it is rather difficult to build such a pendulum balancing robot combining Lego parts, the feeble Lego motors and the, for this purpose, perhaps overly big LRBJOP board.

As a makeshift, a simpler robot was devised which uses Lego sensors to navigate through unknown surroundings (See Figure 11. Using a pressure sensor mounted to a ski on the front side, it can detect a sheer. Using an infrared sensor, it can detect obstacles. Back-EMF measurement allows the robot to detect when it is stalled and making no progress, so detecting that it should turn back.

As a proof of concept, the sample robot shows that LRBJOP allows robot steering through a simple Java application.

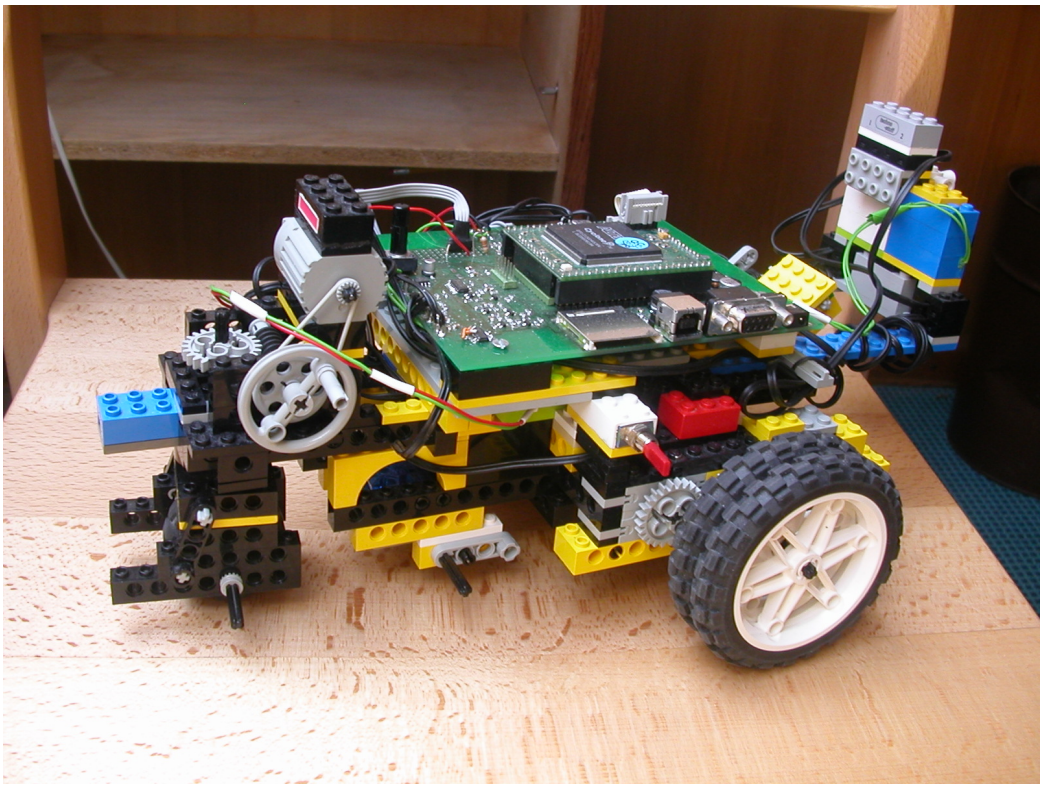

Figure 11: Our demonstration robot.

8 Conclusion

We have shown that the Java Optimized Processor is suitable as a robot controller, and that LRBJOP, the Printed Circuit Board designed during this project, is a working interface between JOP - or other softcores - and miscellaneous sensors and actuators, usable to build all kinds of Lego - or other - robots. The board has many features, and worked fine during extensive testing. A disadvantage resulting from its many features is that it is too big for constructing very small robots.

The project hardware and software built, has been used with success on multiple occasions. For instance, LRBJOP has been used in a lecture at the Vienna University of Technology about the Java Virtual Machine in hardware, to demonstrate the capabilities of JOP. It has also been used at the *KinderuniWien* 2007[27], Vienna Children's University.

8.1 Personal resume

Working on this project has been a great exercise for us. On the one hand we had to design, implement, and debug electrical circuits, and on the other hand we had to write and test the VHDL and Java interface, which happened to be more difficult than expected. We had the opportunity to do a project which encompassed many different levels of abstraction, and their interaction: from designing and building the hardware, to interfacing it with the FPGA running VHDL code, to writing high-level Java applications running on the hardware. This was a great experience.

[27] See `http://www.kinderuni.at`

A Technical details

A.1 Printed Circuit Board

Dimensions [mm]	Height	Width	Depth
	140	116	2
Electrical Specifications	Min	Typ	Max
Operating Voltage [V]	4	9	12
Idle Supply Current, with FPGA [mA]	70	75	120
Idle Supply Current, test program running [mA]	100	150	200
Supply Current, running motors [mA]	150	250	500
Reverse Current Protection [V]			30

Note: These values were taken under test conditions, with a Cyclone EP1C12Q240C8N and additional peripherals attached (i.e. microphone, speaker, IR-sensor, tilt-sensor, IR-remote-module, LED-array and switches).

A.2 VHDL Design

The design was tested on a *Altera Cyclone FPGA Board* for JOP [Sch06]. The processor was a Altera Cyclone EP1C12Q240C8N FPGA. The design was tested at speeds up to 80 MHz. Although Quartus 6.0 reported some timing violations, and suggested a maximum speed of ˜67 MHz, we observed no timing problems.

B File list

File/folder description	Path	Location
FPGA Quartus project	quartus/lego/jop.qpf	JOP source
PLD Quartus project	quartus/lego/pld.qpf	JOP source
Example programs	java/target/src/app/lego	JOP source
SimpCon configuration	vhdl/scio/scio_lego.vhd	JOP source
SimpCon slave providing interface to board	vhdl/scio/sc_lego.vhd	JOP source
Components for accessing actors and sensors	vhdl/scio/lego/*	JOP source
Java library to access board features	java/target/src/app/lego/lib	JOP source
Javadoc documentation	-	Project homepage
Board schematics	-	Project homepage

The **JOP source** is available at `http://www.jopdesign.com/`.

The **LRBJOP project** homepage is at `http://stud3.tuwien.ac.at/~e0327019/lego/`.

C Acronyms

EMF	Electromotive force
FPGA	Field-programmable gate array
GC	Garbage collection
JOP	Java Optimized Processor
JVM	Java Virtual Machine
PCB	Printed circuit board
PID controller	Proportional Integral Derivative controller
PLD	Programmable Logic Device
PWM	Pulse-width modulation
RCX	Robotic Command Explorer
SPI	Serial Peripheral Interface Bus
VHDL	Very High Speed Integrated Circuit (VHSIC) Hardware Description Language

D Schematics

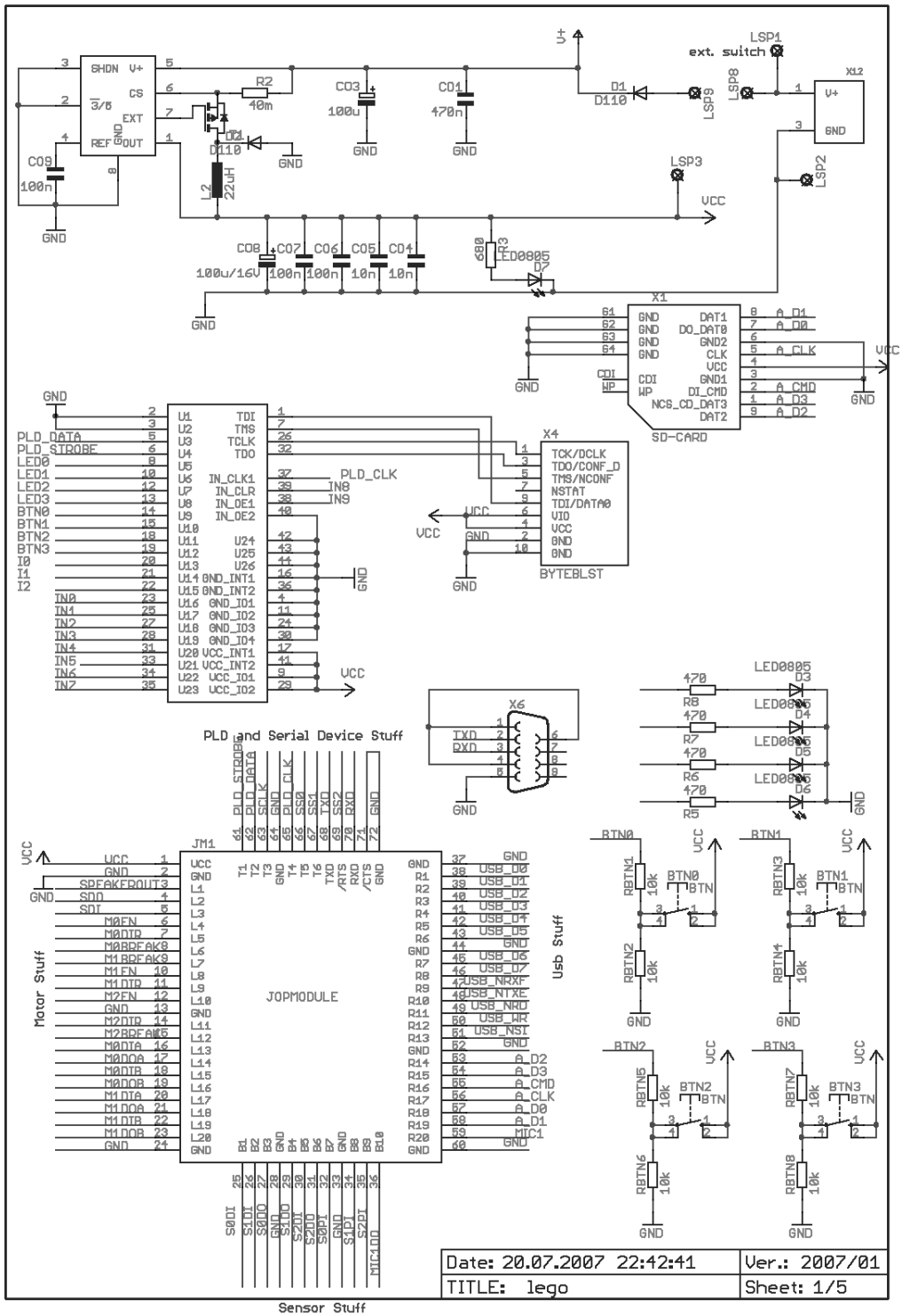

Figure 12: Sheet 1.

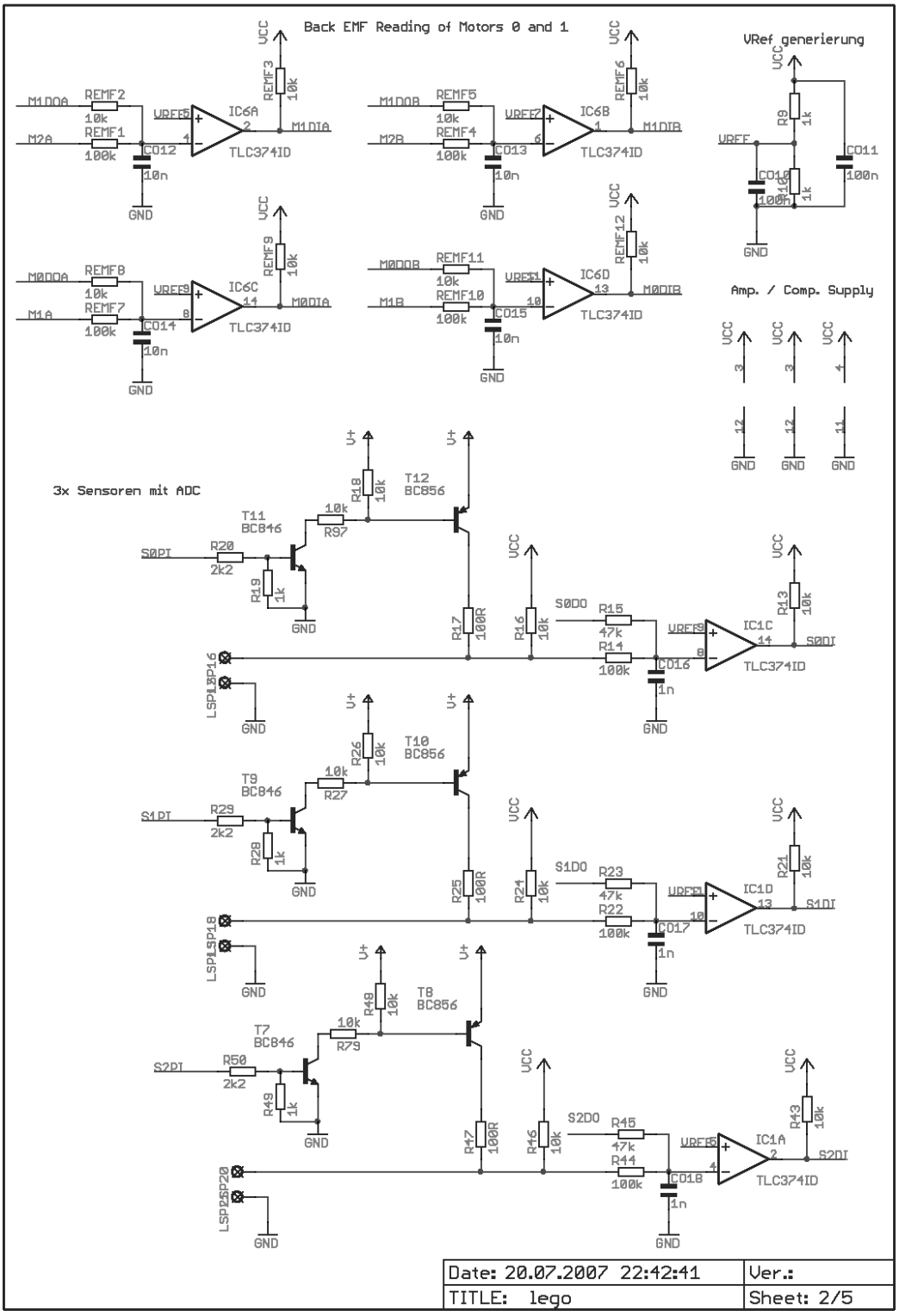

Figure 13: Sheet 2.

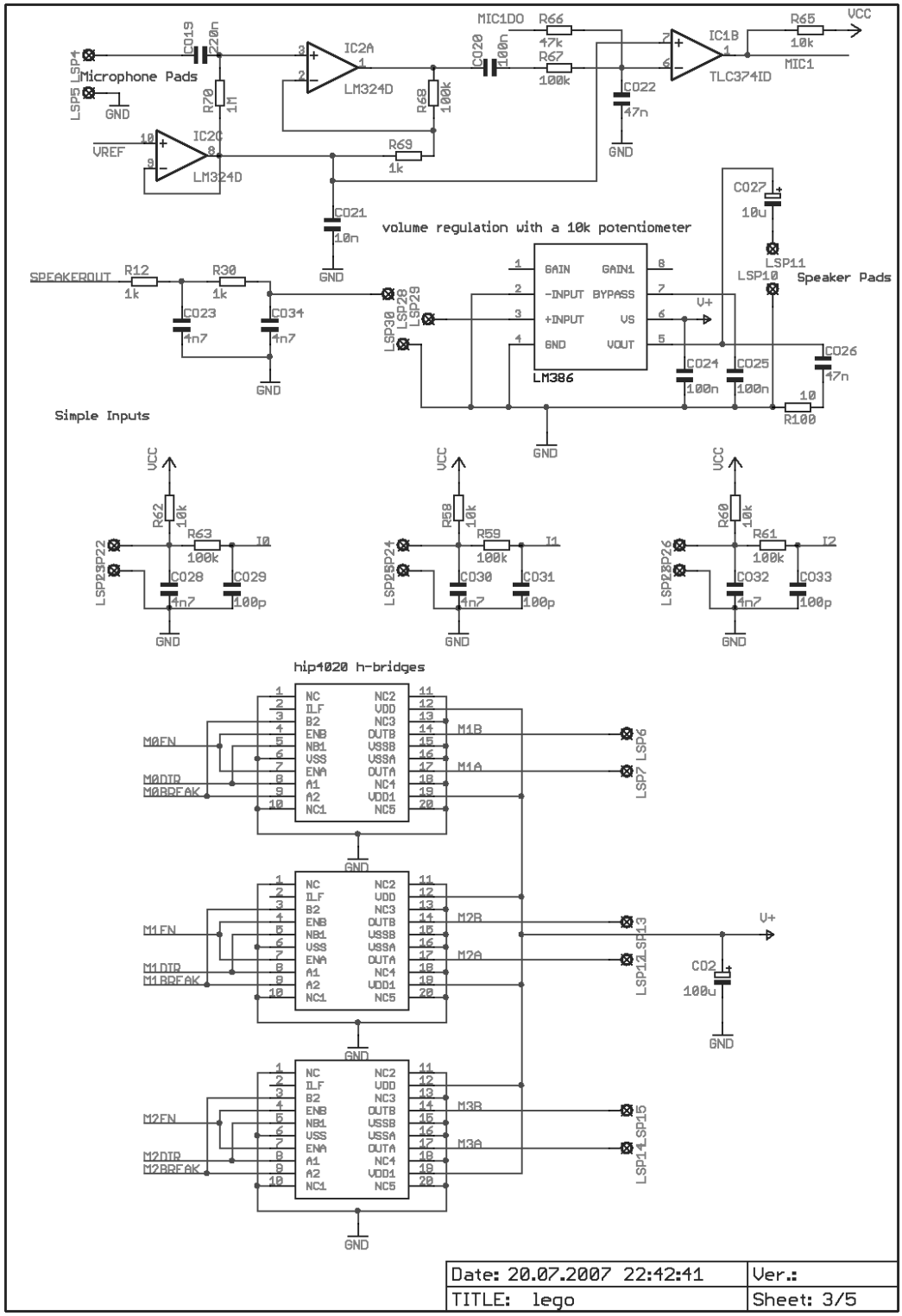

Figure 14: Sheet 3.

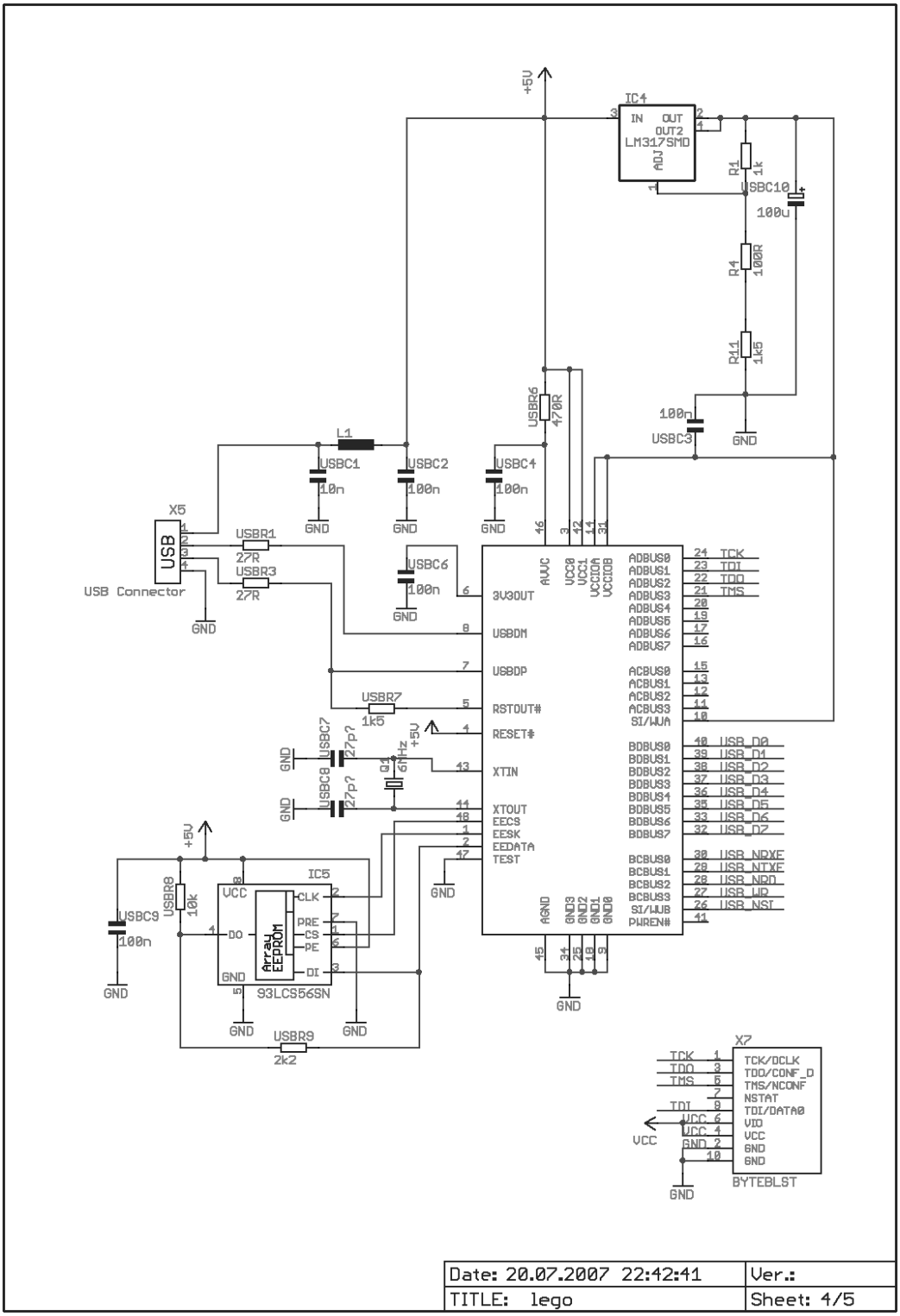

Figure 15: Sheet 4.

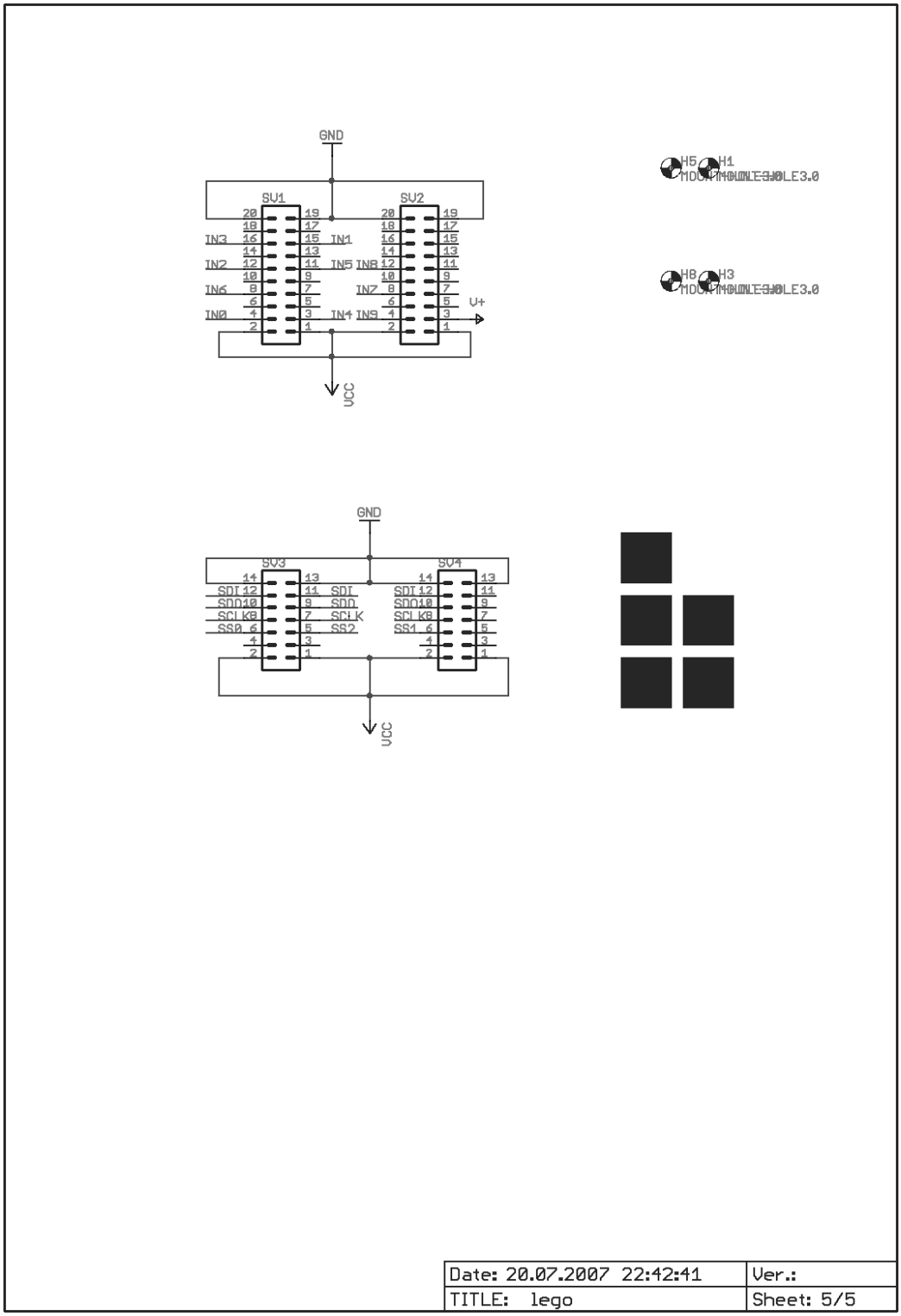

Figure 16: Sheet 5.

E Installation manuals

E.1 FTDI driver installation

a) How to remove all the previously installed FTDI drivers (for example if you had to flash the EEPROM previously).
b) How to install the appropriate drivers to use the FTDI Chip to communicate with LRBJOP.

If you are using a pre-built board, where the EEPROM had already been programmed, than you can skip part a).
But if you have previously flashed the EEPROM by yourself, you need to remove all the installed FTDI drivers prior to the driver installation.

a)
If you have already installed some FTDI drivers, you need to remove them before proceeding with part b).
To do that please follow these steps:

1. Download ftdi.zip from the downloads section[28]
2. Disconnect the PC from the internet
3. Run ftdi/FTClean/FTClean.exe
4. Click on "'Clean System"'
5. Proceed how you are told by the driver removal tool

b)
To install the drivers required for communication with LRBJOP, please follow these steps:

1. Download ftdi.zip from the downloads section[28]

[28] `http://stud3.tuwien.ac.at/~e0327019/lego/`

2. Disconnect the PC form the internet
3. Remove all previously installed drivers with the FTClean tool (if you haven't done that yet)
4. Plug in the board
5. When asked for drivers, select driver installation from a specific path
6. For the first device use path ftdi/d2xx
7. For the second device use path ftdi/vcom
8. For the third device use path ftdi/vcom
9. If it worked, you should now have two new USB devices and a new Serial Port installed
10. Go to your Windows hardware manager and look which COM port has been assigned to your Serial Device
11. The appropriate COM port needs to be stated in the Makefile to be able to upload the FPGA design and the Java program

E.2 Flashing the EEPROM of the FTDI USB chip

a) How to install the appropriate drivers to flash the EEPROM.
b) How to flash the EEPROM.

a)
To be able to flash the EEPROM with the appropriate information needed for the communication between a Windows machine and the LBRJOP, its required to install D2XX functionality for both FTDI channels on the chip. To do that please follow these steps:

1. Download ftdi.zip from the downloads section[29]

[29] `http://stud3.tuwien.ac.at/~e0327019/lego/`

2. Disconnect the PC from the internet
3. Install the MProg EEPROM programming tool from ftdi/MProg2.8_Setup.exe
4. Plug in the board
5. When asked for drivers, select driver installation from a specific path
6. For the first device use path ftdi/d2xx
7. For the second device use path ftdi/d2xx
8. If it worked, you should find the two devices of the FTDI chip installed as "'FT2232C Channel A"' and "'FT2232C Channel B"'

b)
Now that the appropriate drivers are installed, you are ready to flash the EEPROM.
To do that please follow these steps:

1. Run MProg
2. Now push Ctrl+C to scan the device. If it works you should get a message that a blank device has been found
3. Now push Ctrl+O and select ftdi/eeprom.ept which contains the device information needed to flash the EEPROM to our wishes
4. There is no need to change anything of the settings which are now displayed. Just push Ctrl+P to program the chip, and that's it

F Part list

A part list with ordering information is also available at our site[30].

[30] http://stud3.tuwien.ac.at/ e0327019/lego

#	Partnr	Value/Desc.	DeviceNr
	lego_brd		
1	AUDIO	LM386M-1	LM386M-1
4	BTN1,BTN2,BTN3,BTN4	BTN	BTN
1	CO1	470n	C0805
4	CO2,CO3,CO8	100u/16V	CPOL-EUE
8	CO4,CO5,CO12,CO13,CO14,CO15,CO21,USBC1	10n	C0805
13	CO6,CO7,CO9,CO10,CO11,CO20 ,CO24,CO25,USBC2,USBC4,USBC6,USBC9	100n	C0805
1		AUDIO Pot.	10k
3	CO16,CO17,CO18	1n	C0805
1	CO19	220n	C0805
5	CO22,CO26,CO28,CO30,CO32	47n	C0805
2	CO23,CO27	10u/16V	CPOL-EUE
3	CO29,CO31,CO33	100p	C0805
2	D1,D2	BAT254	D110
5	D3,D4,D5,D6,D7	LED0805ROT/Orange	LED0805
1	IC4	LM1117MP	
1	IC2	LM324D	4AMP_P3+
3	IC1,IC6	TLC374ID	4AMP_P3+
1	IC3	FT2232L	FT2232C
1	IC5	M93S46-WMN6	93LCS56S
1	JM1	JOPMODULE	JOPMODUL
1	L1		L1206
1	L2	22uH	B82464A4223K
3	MOTOR 1,MOTOR2,MOTOR3		HIP4020
1	PLD1	EPM3064ATC4410	LEGOPLD
1	Q1	6MHz	HC49U-V
1	R2	50m	R0805
1	R3	680	R0805
5	R5,R6,R7,R8,USBR6	470	R0805
7	R9,R10,R19,R28,R49,R69	1k	R0805
33	R13,R16,R18,R21,R24,R26,R27,R43,R46,R48 R58EMF8,REMF9,REMF11,REMF12,USBR8	10k	R0805
12	R14,R22,R44,R59,R61,R63,R67,R68,REMF1 REMF4,REMF7,REMF10	100k	R0805
4	R15,R23,R45,R66	47k	R0805
4	R17,R25,R47	100R	R0805
4	R20,R29,R50,USBR9	2k2	R0805
1	R70	1M	R0805
1	R100	10	R0805
2	SV1,SV2		MA10-2
2	SV3,SV4		MA07-2
1	T1	SI2315BDS	PMOSSOT2
3	T7,T9,T11	BC846 npn	BC846
3	T8,T10,T12	BC856 pnp	BC856BSM
2	USBC7,USBC8	22p	C0805
2	USBR1,USBR3	27R	R0805
2	USBR7	1k5	R0805
1	VREG	MAX1626	MAX1626
1	X1	SD-Card	SD-CARD
2	X4,X7	BYTEBLST	5x2 pinhead
1	X5	USB Connector	USB-7877
1	X6	9pol sub-d Buchse	F09HP
1	X12		DCCON

Figure 17: Part list.

References

[BDW95] Billur Barshan and Hugh F. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transaction on Robotics and Automation VOL. 11*, 1995.

[Har06] David S. Hardin. ajile systems: Low-power direct-execution java microprocessors for real-time and networked embedded applications, 05 2006.

[Hit] Hitachi. Hitachi single-chip microcomputer h8/3297 series. Hardware Manual.

[Inc06] Acroname Inc. Back-emf motion feedback, 2006.

[Int] Intersil. Hip4020 half amp full bridge power driver for small 3v, 5v and 12v dc motors. Hardware Manual.

[Mig99] Tobias Migge. Diplomarbeit inverses pendel. FH Wedel University of Applied Sciences Hamburg, 1999.

[Ooi03] Rich Chi Ooi. Balancing a two-wheeled autonomous robot. University of Western Australia, 2003.

[PBN03] T. Hsiu S. Richards A. Bhave A. Perez-Bergquist and I. Nourbakhsh. Designing a low-cost, expressive educational robot. Carnegie Mellon University Pittsburgh, 2003.

[Pro98] Kekoa Proudfoot. Reverse engineering the lego rcx. Standford University, 1998.

[Sch05a] Martin Schoeberl. An fpga for lego mindstorms roboter. Vienna University of Technology, 2005. http://www.jopdesign.com/lego/.

[Sch05b] Martin Schoeberl. *JOP: A Java Optimized Processor for Embedded Real-Time Systems*. PhD thesis, Vienna University of Technology, 2005. http://www.jopdesign.com/thesis/thesis.pdf.

[Sch06] Martin Schoeberl. Cyclone fpga board. Vienna University of Technology, 2006. http://jopdesign.com/cyclone/.

[Sch07] Martin Schoeberl. SimpCon - a simple and efficient SoC interconnect. In *Proceedings of the 15th Austrian Workhop on Microelectronics, Austrochip 2007*, Graz, Austria, October 2007.

[Zha96] Weijing Zhang. Two stage inverted pendulum. Aptronix Incorporated Santa Clara, 1996.