Chapter 2
Myon, a New Humanoid

Manfred Hild[1,2], Torsten Siedel[1], Christian Benckendorff[1], Christian Thiele[1], and Michael Spranger[2,3]

Abstract This chapter introduces the modular humanoid robot Myon, covering its mechatronical design, embedded low-level software, distributed processing architecture, and the complementary experimental environment. The Myon humanoid is the descendant of various robotic hardware platforms which have been built over the years and therefore combines the latest research results on the one hand, and the expertise of how a robot has to be built for experiments on embodiment and language evolution on the other hand. In contrast to many other platforms, the Myon humanoid can be used as a whole or in parts. Both the underlying architecture and the supportive application software allow for ad hoc changes in the experimental setup.

Key words: humanoid robot, modular architecture, power autonomy, antagonistic actuation, distributed neural network, sensorimotor loop, embodiment

2.1 Introduction

The robot Myon, which is shown in Figure 2.1, has been designed for research on cognitive robotics, in particular experiments on artificial language evolution as described in this book (Steels et al, 2012b) and elsewhere (Steels and Spranger, 2009). It incorporates basic properties of the precedent humanoid platform "A-series" which has also been used for language games, as described in other chapters of this book (Kubisch et al, 2012; Höfer et al, 2012). Both humanoid robots need to be able to perceive the world, autonomously wander around, recognize and manipulate different objects, and communicate with other robots. Moreover, the Myon robot is also intended as a research platform for biologically inspired behavior con-

[1]Neurorobotics Research Lab, Humboldt-University Berlin, e-mail: hild@informatik.hu-berlin.de
[2]Sony Computer Science Laboratory, 6 rue Amyot, 75005 Paris, France
[3]Systems Technology Laboratory, Sony Corporation, Minato-ku 108-0075, Tokyo, Japan

L. Steels and M. Hild (eds.), *Language Grounding in Robots*,
DOI 10.1007/978-1-4614-3064-3_2,

trol, using tight sensorimotor loops and antagonistic joint actuation. Based on these objectives and prototypical use cases we can identify the following requirements:

- **Availability.** Normally, a research team around a humanoid robot consists of more than ten researchers. The robot must never be a competitive bottleneck. If the robot is *low-cost*, several robots can be afforded. Since often only part of the robot is needed, a fully *modular* robot further relaxes the bottleneck situation.
- **Flexibility.** Experimental designs need to be realizable immediately and at any place – be it the isolated research using a single joint, or a scenario with several robots. This again demands a *modular* robot, but in addition also a *fully autonomous* one. And this implies that the robot needs to have *distributed energy supplies and processing power*.
- **Simplicity.** Researchers need to be productive quickly. The robot needs to be fully operational in milliseconds. There must be *no booting process* at start-up. The user wants to comfortably design sensorimotor control loops graphically on the computer without the need for programming and download them onto the robot in a second where they are stored in *non-volatile* memory and are operational at any time.
- **Adaptivity.** Researchers may want additional sensors, continuously record the sensorimotor data during the experiment, and synchronize cameras to the robots time frames. All body parts of the robot therefore need to be equipped with a *standard interface* that addresses the aforementioned needs.
- **Transparency.** Researchers always have to know (and visually see) what is going on inside the robot. All results should be reproducible. This demands *guaranteed timing* and *visual status feedback* distributed all over the robot's body. This visual feedback system should also be available to the researcher, as it is often necessary to monitor a hidden neural signal at any place on the robot's body.
- **Affordability.** Eventually, a researcher will somehow break the robot. This is a major problem with expensive platforms, but with a *low-cost* platform the researcher can be more daring. It is important that researchers do not feel inhibited by using an expensive platform where they are afraid to break something.

There are more desirable aspects for a good humanoid research platform, but those are the most important ones and the ones that we have tried to achieve with the Myon design. We have derived them from a long history of building and using robots (Hild, 2007), and most researchers involved in similar experiences agree.

The first half of this chapter outlines the overall system architecture of the Myon robot and focus on its modularity and compliant antagonistic actuation system. We also address the robot's power autonomy and the flange design which allows the robot's body parts to be detached and re-attached at runtime.

Robust hardware is an essential requirement to conduct experiments in embodied robotics, however, a sophisticated robot is only as useful as its accompanying control architecture, which manages everything from low-level sensorimotor loops to visual processing and high level behavioral decision making. Usually, hybrid architectures are used to cope with the different constraints, e.g., simple, but highly reactive, reflex

loops for motion control versus non-time-critical processing of complex and large decision trees and language processing.

In the second half of this chapter, we introduce the distributed architecture DISTAL (Hild et al, 2011c) proposed and implemented for the Myon robot. It is specifically designed to handle large neural networks and supports metamorphoses of the robot while running. We will illustrate the advantages of DISTAL regarding its ease of use, detail its implementation on 32-bit ARM RISC processors, and also introduce the complementary graphical application software BrainDesigner which helps the user in neural coding.

2.2 The Humanoid Robot Myon

Humanoid robots are highly complex systems and as such prone to damage and malfunctioning. This is especially true if not only an upper torso with head and

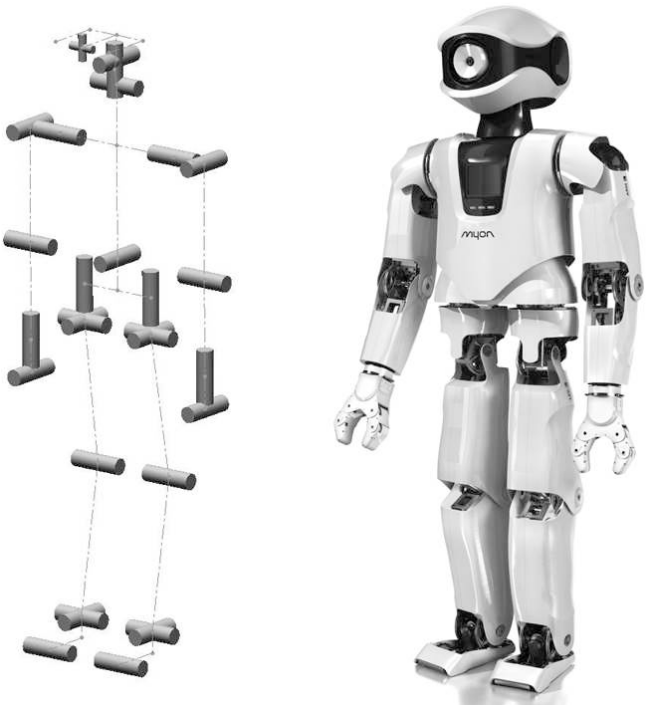

Fig. 2.1 The humanoid robot Myon. Left: Schematic diagram of the joints' positions and orientations. Right: Image of the functional robot including the exoskeleton shells.

arms is used, but a full body, which is necessary to test and analyze sensorimotor behaviors for walking or stable standing. There are several approaches to remedy the situation, depending on the experimental settings that are planned to be addressed with the humanoid platform. If full autonomy is not needed, then energy supply and processing power can be placed outside the robot and the bare skeleton can be optimized for maximal mechanical robustness. Also, the type of the actuators plays a crucial role. Pneumatic-driven humanoid robots can withstand a drop from more than one meter height onto the ground without any problem, although it has to be noted, that pneumatic actuators have much longer response times than electric motors (Hosoda et al, 2010). Hence, if the robot needs to be mobile within the experimental setting then electric actuators and on-board batteries have to be used.

2.2.1 Mechanics for Run-Time Modularity

The Myon robot, as shown in Figure 2.1, all in all is 1.25 m tall, weighs 15 kg, and consists of six body parts (head, torso, arms, and legs) which are fully autonomous in terms of energy supply and processing power. An overview of the robot's main parameters are given in Table 2.1. The robot exhibits 32 degrees of freedom and 48 actuators. Joints which need a large amount of torque, e.g., the knee, are driven by several actuators in parallel, using series elasticities. Besides the camera, there are over 200 sensor values of the following types: joint angle, motor angle, motor current, motor temperature, acceleration force, contact force, battery voltage. Following the outline in (Hild et al, 2011b), we will now detail selected parts of the mechanical construction.

Module	Mass (kg)	Joint	DOFs (number)	Actuators (number)	(type)
Head	1.4	Eye	4	4	Micro servo
		Neck	3	3	RX-28
Arm (2x)	1.1	Shoulder	1	1	RX-28
		Elbow	1	1	RX-28
		Wrist	1	1	RX-28
Gripper (2x)	0.2	Fingers	1	1	RX-28
Torso	2.5	Shoulder (2x)	1	2	RX-28
		Waist	1	1	RX-28
		Hip (2x)	1	1	RX-28
Leg (2x)	3.0	Hip	2	5	RX-28
		Knee	1	3	RX-28
		Ankle	2	5	RX-28
		Foot	1	–	passive
Shells (total)	2.5		–	–	
Total	15.0		32	48	

Table 2.1 Overview of the robot's mass, degrees of freedom (DOFs), and number of actuators. Except for the eye, there is only one type of actuator used. Joints which need a large amount of torque, e.g., knee, are driven by several actuators in parallel.

The Myon robot is highly modular, not only regarding the use of different internal components, but also as a functional robot platform itself. The robot can be disassembled and reassembled during runtime, since all body parts are fully autonomous in a threefold sense: they all possess their own energy supply, processing power, and a neural network topology which allows for stand-alone operation of single limbs.

An overview of the different body parts is shown in Figure 2.2. The robot has especially been designed for robustness and easy maintenance. It exhibits a combination of an endoskeleton with an exoskeleton, the latter of which can manually be detached without the need for technical equipment. One of the essential parts is a novel flange which firmly connects the body parts mechanically, whilst at the same time relaying the power supply lines and sensorimotor signals. The mechanical details are outlined in Figure 2.3.

As recently stated by Migliore et al (2010), the vast majority of walking robots still loses energy by ignoring the potential benefit of using passive elastic components at their joints. On the Myon robot, elastic components could be incorporated in a very compact way along with the antagonistic actuation system (Siedel et al,

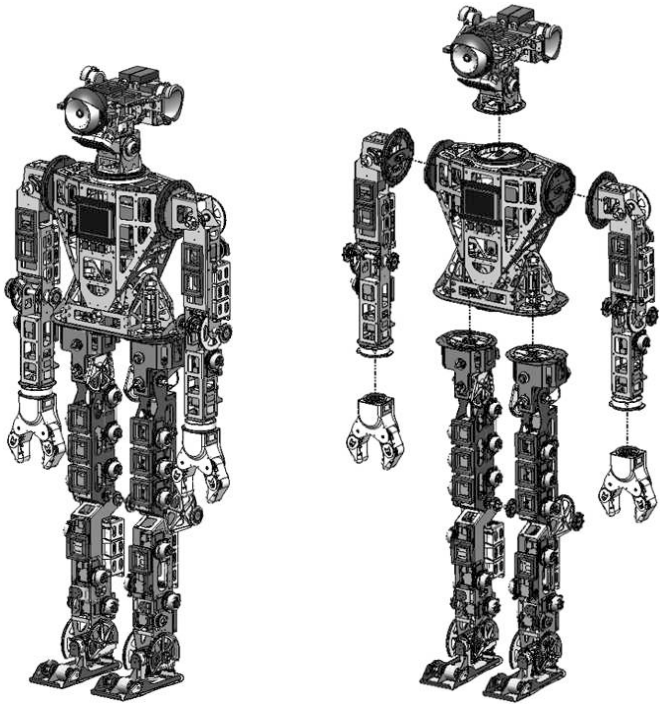

Fig. 2.2 Overview of the body parts of the Myon robot which can be detached and reattached during runtime.

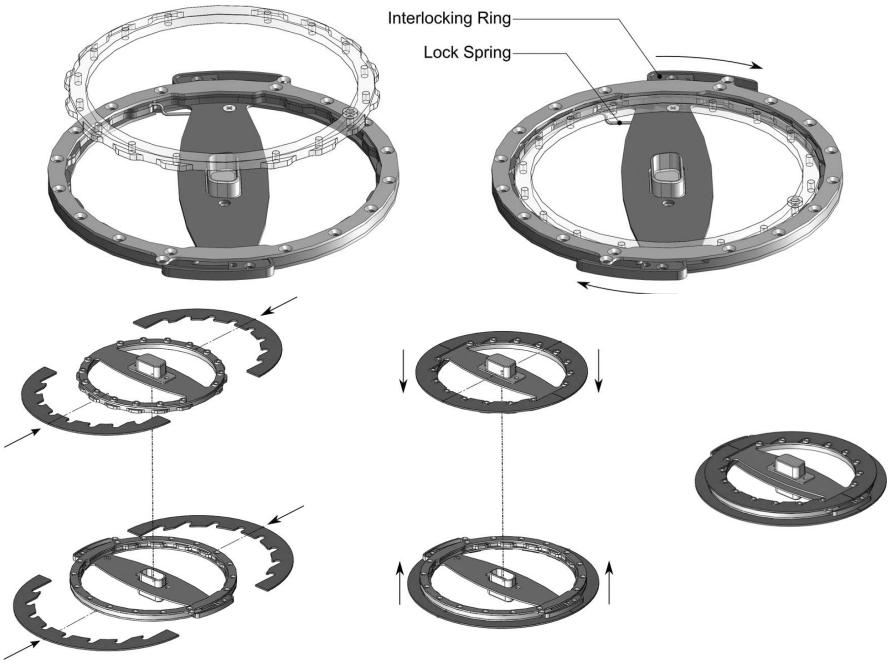

Fig. 2.3 Mechanical details of the flange assembly. Top: An interlocking ring and a lock spring allow for easy snap-in of body parts. Bottom: The exoskeleton is mechanically coupled to the flanges, so that an optimal force distribution between body parts can be guaranteed.

2011a). This not only opens up the research field of energy-efficient walking behaviors, but in the first instance protects the gears against high external impact forces, so that researchers can rely on a robust system while conducting their experiments.

Figure 2.4 shows the construction details of the actuation system. Each actuator (see top row, drawing 1) is equipped with a special carrier (drawing 2).
This carrier has been built by means of rapid prototyping using ABS plastic material. Its bottom part is connected to the joint via a wire rope. On top of the carrier sits a torsion spring with high stiffness (see top row, drawings 3 and 4). Several of this building blocks are then attached to the same joint. Each ankle joint, e.g., is driven by four actuators in parallel. This guarantees high driving torques, if all four actuators are driven in the same direction. Using antagonistic techniques – inspired by biological mechanisms – the non-linearities like friction and backlash can be overcome. Due to the modular approach, also alternative types of clutches can be installed, e.g., novel overrunning clutches (Hild et al, 2011b; Siedel et al, 2011b).

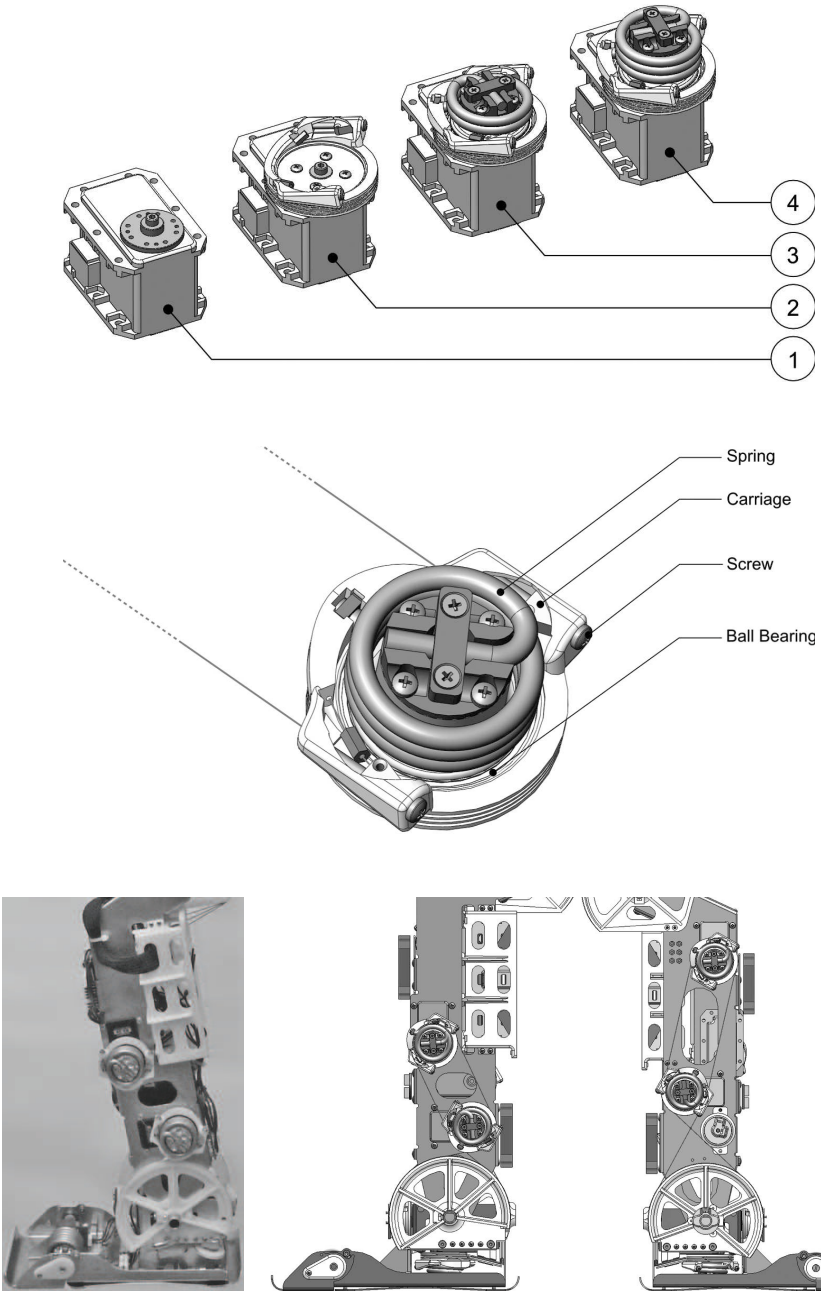

Fig. 2.4 Robot Myon exhibits a compliant actuation system. Top: A robust spring coupling consisting of four components protects each actuator and introduces series elasticity. Middle: The forces are transmitted via wire ropes. Bottom: Several actuators with spring couplings are coupled together onto one joint (here: ankle joint). The series elasticities compensate for tolerances between active actuators.

2.2.2 Hot-Pluggable Energy Supply

Special attention has been paid to the energy distribution within the robot. Long cables would induce power losses, so it is a good design practice to place the batteries nearby the actuators within each body part. Energy is recruited from the local batteries whenever possible. However, the robot can also run with a single battery pack which is inserted at any single body part, even though best performance is achieved when all six packs are in place. Regarding highly dynamic body motions, like full-body stabilization or lifting of heavy objects, there may be very high currents necessary locally, e.g., in one arm or at both ankle joints. If this is the case, then power from *all* body parts is recruited and sent to the place with the highest demands. This is realized fully via analog circuitry. The details can be found in Figure 2.5. Also, on each local processing board (there are up to 32 per robot) there are six parallel stabilized power supply lines to decouple the different analog and digital subparts of the control circuits.

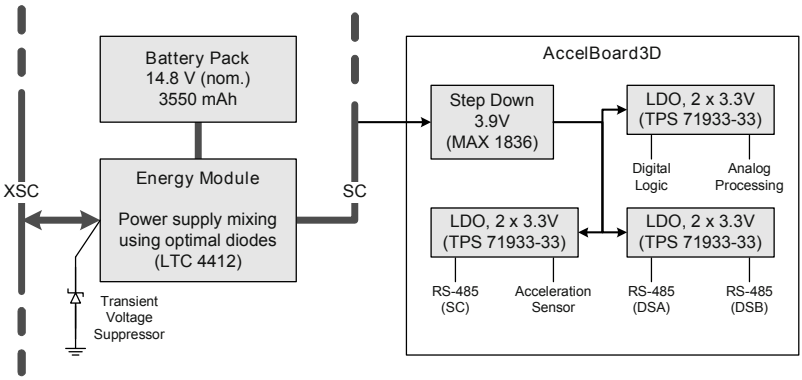

Fig. 2.5 Each body part has its own energy supply. A special energy module takes care about the dynamic balancing between body parts. Within each body part, massive decoupling of all electronics is achieved by the use of multiple parallel linear low-drop regulators. This increases the signal-to-noise ratio of all sensory signals.

2.2.3 Distributed Processing Nodes

The Myon robot exhibits several unique architectural characteristics. Here, we give a summary of the processing nodes and the communication bus between them. An overall diagram of the system architecture is given in Figure 2.6. All processing nodes are connected using the so-called SpinalCord, which is a multi-core bus that

transfers energy, sensorimotor data at a rate of 4.5 MBaud, and a control signal which is used to switch the robot on and off.

Data processing is predominantly done by 25 processing nodes, which are distributed all over the robot's body. They are called AccelBoard3D, since they also possess a 3-axis acceleration sensor, despite the Cortex-M3 ARM RISC processor running at 72 MHz. Up to four actuators are connected to each AccelBoard3D. The actuators are all of the type Robotis RX-28. Whenever several actuators drive the same joint, all of them are connected to the same AccelBoard3D. Also, the corresponding sensory data (angular sensors of the joint and all motors; motor current sensors) is sent to the same processing node, so local processing of antagonistic control paradigms can easily be realized. Those situations are automatically detected by the application software BrainDesigner during the deployment process, as will be described later in this chapter. Each AccelBoard3D also exhibits a mode button and two status LEDs. This is extremely helpful for diagnosis, inspection of internal states which would otherwise be hidden to the user, and switching of operational modes like start, stop and the like.

As the name already indicates, the BrainModule is a special processing node inside the robot's head. When logging the SpinalCord data, the BrainModule is indistinguishable from an AccelBoard3D, but as can be seen in Figure 2.7, the BrainMod-

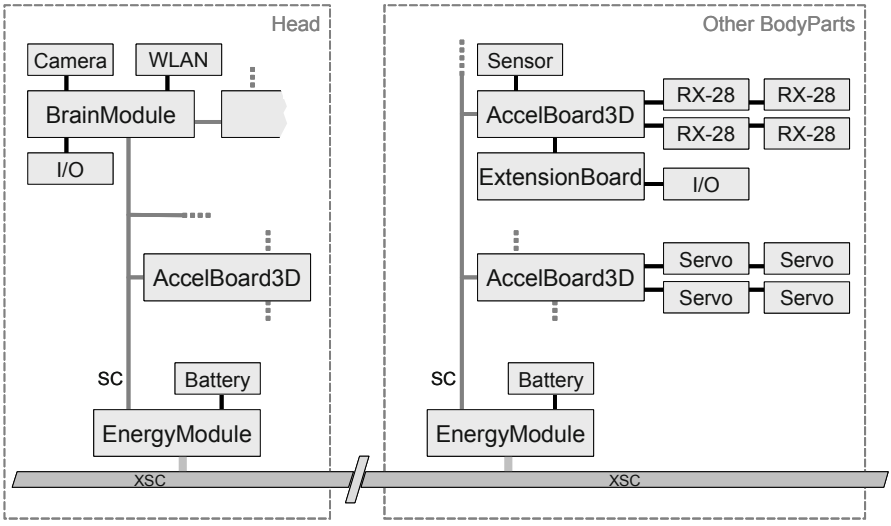

Fig. 2.6 System architecture of the Myon robot. Components within each body part are connected via the so-called SpinalCord (SC), whereas the body parts are connected by the ExtendedSpinalCord (XSC) which includes lines for energy transfer. Each body part exhibits its own local energy supply and processing power.

ule possesses enough processing power to do serious audio-visual processing, e.g., a Hough-Transform.

Along with the digital camera interface, there is a special analog video mixer which allows for video keying and overlaying. This is helpful not only during presentations, but also for standard lab situations, where one wants to see the original camera image with the processed visual data superimposed. A simple overlay, e.g., shows a cross hair which indicates the object that the robot is currently investigating (Kubisch et al, 2012). Since this is all done fully synchronously, the researcher can detect the slightest deviation from the expected behavior. When using the wireless interface to monitor all data on a PC, the resultant quality and reactivity is by far lower, due to the restricted bandwidth.

Configuration of the XILINX Virtex-4 field programmable gate logic (FPGA) is done by an 8-bit microcontroller via a standard MiniSD-Card that contains the necessary FPGA bitfile. Future implementations may also use the MiniSD-Card to

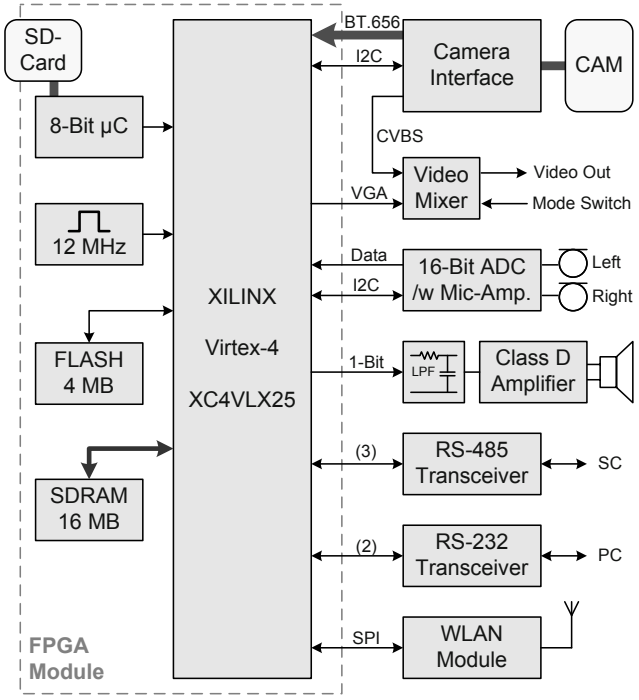

Fig. 2.7 On the one hand the BrainModule is just another processing node of the DISTAL architecture, but on the other hand it possesses considerably more processing power than an AccelBoard3D. This is needed for the audio-visual processing inside the robot's head.

log sensorimotor and visual data during autonomous stand-alone scenarios without the use of a PC.

2.3 The DISTAL System Architecture

The most widespread robotic infrastructure consists of a fully or partly assembled robot, which is connected to a computer. Highest flexibility is achieved when the computer is within the sensorimotor loop, so structural changes, as well as parameter changes, can be realized on the fly. Since the robot's processing power is not used in this case, the corresponding operating mode is called *transparent mode*. But the application software has then no way to cope with unforeseen robot morphologies, e.g., with the one shown in Figure 2.8.

Fig. 2.8 All body parts of the robot MYON can be detached and reattached during runtime. Here, the head has been replaced by the robot's left arm.

When experimenting with self-explorative algorithms, cables may hinder free movement. Thus, one needs to be able to deploy the running neural network permanently on the robot's processing nodes. This process we call *deployment*. After deployment, it should still be possible to monitor and log sensorimotor data as well as internal behavioral states (called *logging*). Also helpful, especially during presentations, are standard audio-visual signals which are provided by the robot in stand-alone scenarios, i.e., without any additional external computer. Surely, this also has to be

supported by DISTAL. Finally, often program-debug cycles hinder experimenting, so a graphical network editor is important.

There is a long history of architectures for robot control going back to the 1970s. In the beginning robot control was largely understood as a planning problem in a sense-plan-act cycle (SPA) (Nilsson and Fikes, 1970; Nilsson, 1984). Ten years later saw the advent of the behavior-based paradigm (Brooks, 1986) with concrete architectures such as AuRA (Arkin, 1987). Notable architecture examples from the 1990s are SAPHIRA (Konolige and Myers, 1998) which was designed for autonomous mobile robots and BERRA (Lindstrom et al, 2000) which was designed specifically for service robots. An evaluative survey of architectures for mobile robots up to the year 2003 can be found in Orebäck and Christensen (2003).

Along with the continuous increase of processing power, versatile approaches appeared which today can be run on various robot platforms, even though their underlying processor hardware differs considerably. A widely-used open-source framework is URBI (Baillie, 2005), which can be equally well used to control Sony's AIBO, Aldebaran's NAO, or LEGO's Mindstorm NXT robots – just to name a few. Another example is ROS (ROBOT OPERATING SYSTEM) (Quigley et al, 2009). Recent architectures typically attempt to provide a middle-ware layer and support for distributed processing (see also Amoretti and Reggiani, 2010; Heintz et al, 2010; Hawes and Wyatt, 2010; Balkenius et al, 2010; Mitchinson et al, 2010; Martínez-Barberá and Herrero-Pérez, 2010, for more examples). An up-to-date survey is given by Hülse and Hild (2010). However, most of these frameworks do not guarantee real-time performance.

The DISTAL architecture is a realtime framework in the hard sense, i.e., at any time data processing is bound within prescribed time limits. In order to achieve highest performance, we introduced a neural bytecode (NBC) which almost directly translates into compact machine code for the 32-bit ARM RISC processor of the AccelBoard3D. In the following, we address these two main concepts which constitute DISTAL.

2.3.1 Communication Using the SpinalCord (SC)

All processing nodes communicate with each other one hundred times a second using the SpinalCord. Therefore, each participant has a designated time slot, during which it sends its data. For the rest of the communication time, it receives the data from all the other connected participants. The starting time of a slot is relative to the starting time of the participant with the lowest identification number (ID), which has the role of a master and triggers the 10 ms pattern. The whole timing is shown in Figure 2.9 and an example of the SpinalCord data in Figure 2.10.

The communication on the Myon robot lasts 3.36 ms, which leaves 6.64 ms for the calculation of neural networks and the acquisition of sensor values before the next slot starts. Up to 32 participants are intended, whereof six are the energy modules of the six body parts, which have a shorter time slot than the others, because

they only report the charge status of the batteries. The slots of all other participants last 125 μs each, during which they send 27 words (16-bit values). The first word is reserved for a synchronization value (0x5555), and five bits of the second word contain the ID of the participant.

As already mentioned before, the morphology of the robot can change, and therefore new participants can join during runtime. A new participant initially listens some hundred milliseconds and then joins the communication at the correct time. It is even possible that the new ID is lower than the ID of the current master, which leads to a new master. The old one automatically becomes a slave when it receives data from a new master before its own slot. If the master is removed, the second lowest ID will recognize this situation, become the master and the communication continues seamlessly. If the BrainModule is connected to the SpinalCord, it is automatically the master because it has the lowest possible ID, namely zero. It gradually synchronizes the SpinalCord to the 50 Hz signal of the camera, leading to time-consistent sensory data (regarding SpinalCord and camera data). It is possible to shift the communication time by nearly 125 μs per 10 ms slot by starting the communication later, near the end of the slot. Because of a 2.5 μs dead time at the beginning of each slot, moving backwards is possible, too. The 25 words after the synchronization word and the ID contain sensory data and designated fields for motor control voltages, as well as free slots, which can be used by neural networks for the communication between different processing nodes.

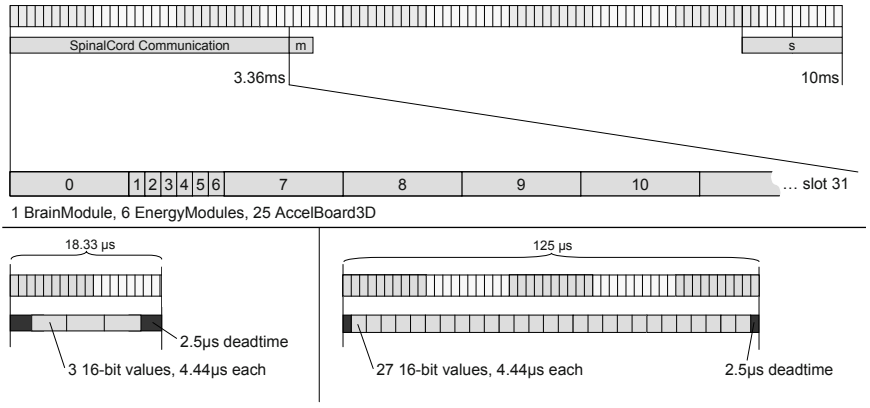

Fig. 2.9 SpinalCord timing of a 10 ms time slot: During the first 3.36 ms all data is communicated between up to 32 processing nodes (*SpinalCord Communication*), than there are almost 6 ms dedicated for motor control and calculation of the neural network. At the end of the time slot, new sensor values are acquired (*s*). The communication takes place between three different types of participants. Every data chunk which is communicated by the BrainModule or an AccelBoard3D is 27 words long and needs 125 μs to be transferred over the SpinalCord, whereas the data chunks of the EnergyModules are only three words long and therefore only take 18.33 μs.

7 Left Leg Lower Bottom	9 Left Leg Lower Top	11 Left Leg Upper Bottom
45 SYNC	99 SYNC	153 SYNC
46 ID/Health/Mode	100 ID/Health/Mode	154 ID/Health/Mode
47 Peephole 0 Uptime Low 1 Uptime High 2 Voltage DS 3 Firmware Version 4 Temp M1 5 6 7	101 Peephole 0 Uptime Low 1 Uptime High 2 Voltage DS 3 Firmware Version 4 Temp M3 5 Temp M5 6 Temp M7 7 Temp M9	155 Peephole 0 Uptime Low 1 Uptime High 2 Voltage DS 3 Firmware Version 4 Temp M13 5 Temp M15 6 Temp M11 7
48 AccelX	102 AccelX	156 AccelX
49 AccelY	103 AccelY	157 AccelY
50 AccelZ	104 AccelZ	158 AccelZ
51 Current DS A M1	105 Current DS A M3+M5	159 Current DS A M13+M15
52	106 Current DS B M7+M9	160 Current DS B M11
53 DS A, Position 1 Position 1	107 DS A, Position 1 Position 3	161 DS A, Position 1 Position 13
54	108 DS A, Position 2 Position 5	162 DS A, Position 2 Position 15
55	109 DS B, Position 1 Position 7	163 DS B, Position 1 Position 11
56	110 DS B, Position 2 Position 9	164
57 Angle LToes	111 Angle LAnklePitch	165 Angle LKnee
58 Angle LAnkleRoll	112	166
59 Force LLeftBack	113	167
60 Force LLeftFront	114	168
61 Force LRightBack	115	169
62 Force LRightFront	116	170
63	117	171
64	118	172
65	119	173
66	120	174
67	121	175
68 DS A, Torque 1 Motor 1	122 DS A, Torque 1 Motor 3	176 DS A, Torque 1 Motor 13
69	123 DS A, Torque 2 Motor 5	177 DS A, Torque 2 Motor 15
70	124 DS B, Torque 1 Motor 7	178 DS B, Torque 1 Motor 11
71	125 DS B, Torque 2 Motor 9	179

Fig. 2.10 Data contained in the SpinalCord, here shown for three different AccelBoard3Ds with IDs 7, 9 and 11 (odd numbers correspond with the left half of the robot's body). The 25 words after the synchronization word and the ID contain sensory data and designated fields for motor control voltages, as well as free slots, which can be used by neural networks for the communication between different processing nodes. The third word within each data slot is a so-called *peephole*. The peephole multiplexes between data which only changes slowly, e.g., the motor temperatures.

2.3.2 Implementing Neural Byte-Code (NBC)

A neural network executed by the AccelBoard3Ds consists of several *calculation units*, like the one shown in Figure 2.11, which are linked together via incoming and outgoing arrows.

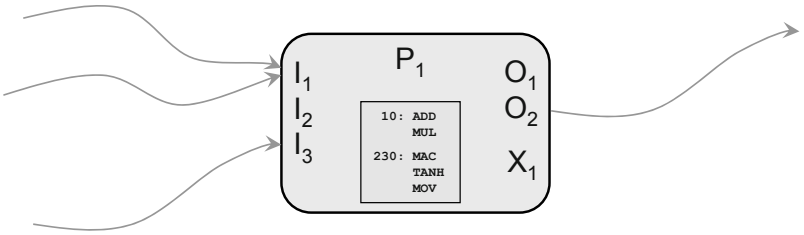

Fig. 2.11 Schematic illustration of a calculation unit. Such a unit can represent an artificial neuron, a delay line, or any other arbitrary signal processing element. It is possible to build recurrent structures by connecting outputs back to the inputs.

After a network of calculation units has been created, two different blocks of machine code are generated for the ARM processor technology used in the processing nodes of the Myon robot. The first block of code maps the network topology, whereas the second block encodes the signal processing taking place within the calculation units.

In what follows, we refer to Figure 2.12. For each unit, a so-called preamble is compiled, which fills the registers of the processor with values according to the network topology. After that, a branch is taken to the compiled code of the unit (a *snippet*). The code for each snippet uses the given values to calculate new output values. In addition to these values, two free registers are available for temporary calculations. The commands of the NBC are similar to those available in the ARM instruction set, e.g., a command for signed saturation exists. A sample code for a weighted synapse reads as follows

```
mul   V0, Input, w
write Output, V0
```

where the first line multiplies the input value with a parameter w and puts the result into the register V0 (which is R8 on the ARM processor), whereas the second line writes this register value to the output.

Each of the calculation units consists of *inputs* and *outputs*, *parameters* (constant for each instance) and *internals* (non-volatile values, which are not accessible from outside). *Outputs* and *internals* together are called *locals* and are represented as a large array in the RAM of the processor (see Figure 2.12). *Parameters* are put directly into the unit's preamble. The calculation order of the units is important in time-discrete neural networks. Therefore, all snippets are executed in a given order which is given by labels attached to the bytecode, e.g., see label '10:' in Figure 2.11.

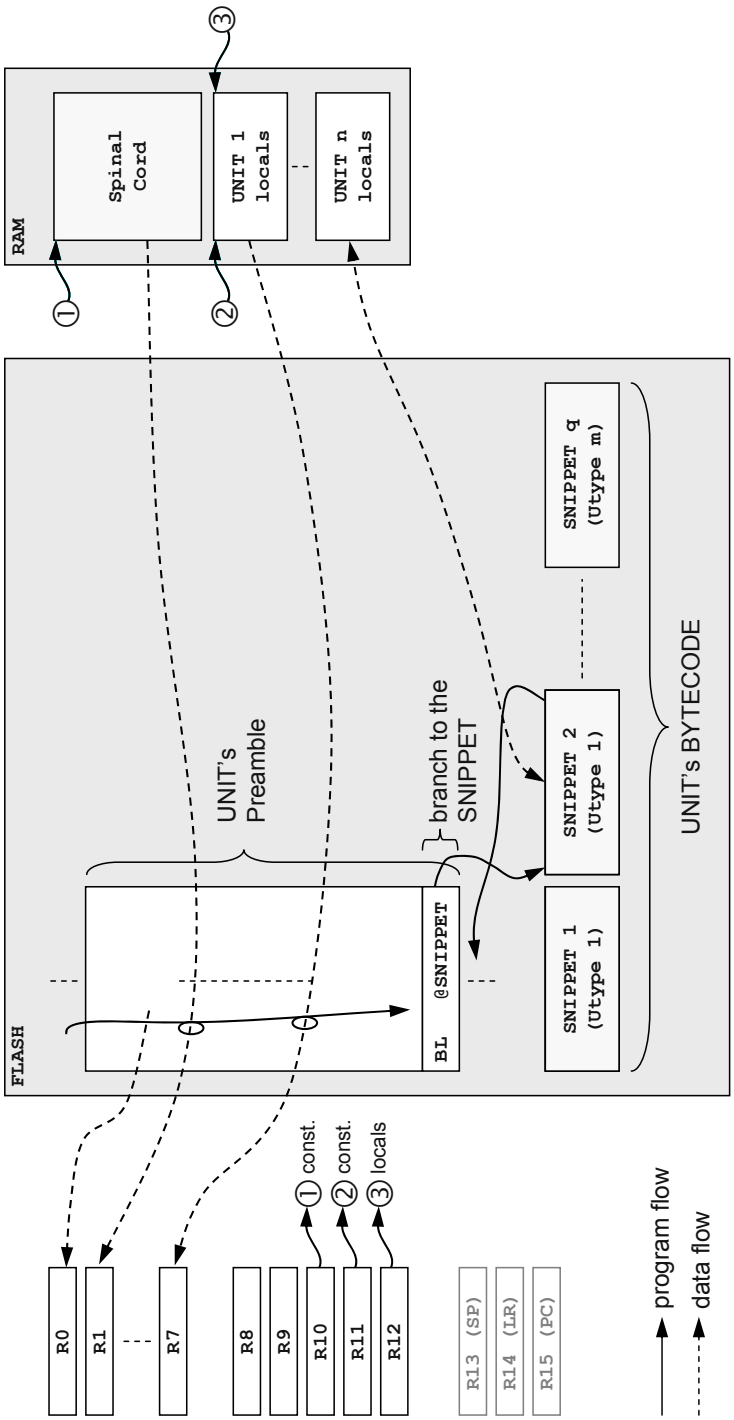

Fig. 2.12 Concept of the execution of the neural bytecode on an ARM processor. On the left side the registers are shown, in the middle the flash memory, and on the right side the RAM. Every instance of a calculation unit has its own preamble code (in flash), after which a branch to the corresponding snippet is taken.

2.3.3 Application Software

Every control architecture has to stand the test in real-world scenarios. DISTAL was used extensively on the Myon robot. The software BrainDesigner (see Figure 2.13) was developed to create artificial neural networks for DISTAL using a graphical interface. Various behaviors, like walking, gripping, and hand-eye-coordination were successfully implemented using the BrainDesigner and DISTAL. Some examples are described in more detail in one of the following chapters (Kubisch et al, 2012), and elsewhere (Kubisch et al, 2011a,b). The application software BrainDesigner offers a graphical interface for assembling neural networks on a personal computer, using the mouse for adding signal processing units (neurons) and connecting them.

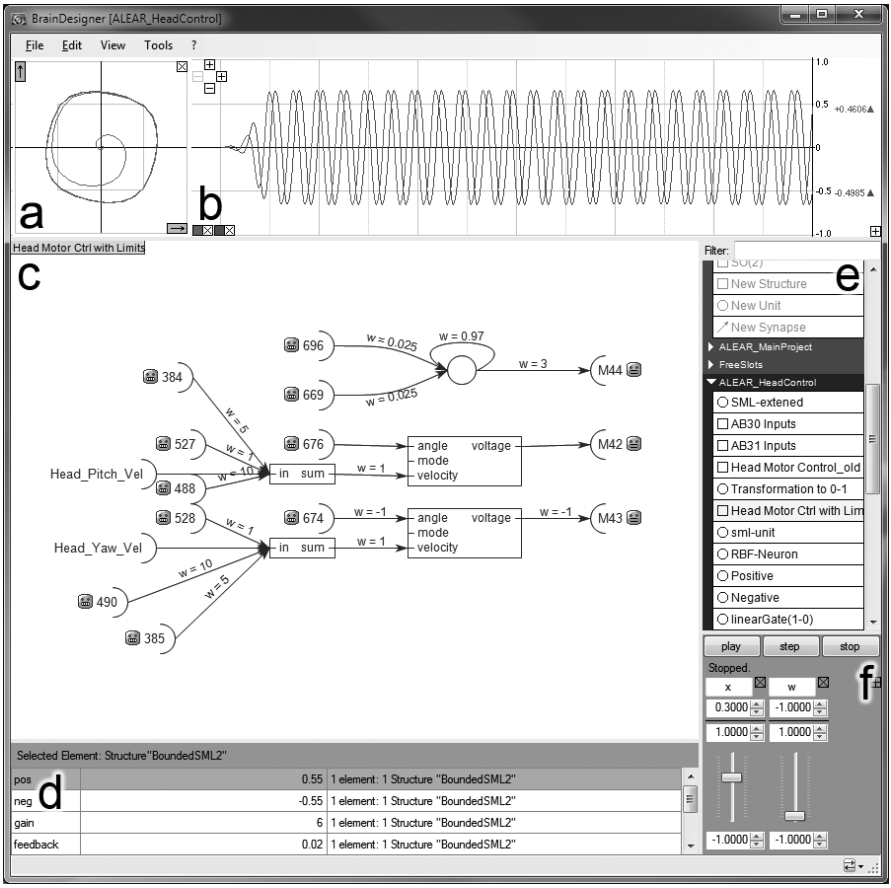

Fig. 2.13 The software BrainDesigner, with a simple neural network loaded (c). Output values can be shown over time (b) or against each other (a). At the bottom (d), parameter changes of included units and structures are possible. (e) Library of units and structures. (f) Parameters can be changed during runtime in transparent mode, using graphical sliders.

Several types of nodes (*neurons*) and directed edges (*synapses*) are available to assemble a network. New types of neurons and synapses can be created, which contain executable code (*Neural ByteCode*) that allows for the implementation of any kind of neural calculation or local learning process. Synapses are independent from neurons – they are treated like any other *unit*. Assembled networks can be encapsulated and included as a building block into other networks, enabling the user to create cascaded network hierarchies.

By using special input and output nodes within the software BrainDesigner, it is possible to read and write values to and from fields in the SpinalCord. Since all sensory data is available in the SpinalCord, and actuators are directly driven from specific SpinalCord values, no additional mechanisms for peripheral connections are needed. The user can choose from a wide range of plug-ins for different robots which are using the DISTAL architecture. For the Myon robot, both operating modes *transparent mode* and *deployment mode*, as described earlier, are available.

2.4 Conclusion

We presented the modular humanoid robot Myon, along with the distributed control architecture DISTAL, which seamlessly supports the Myon robot. Having addressed important use cases of different experimental settings, we detailed the mechatronical design of the Myon robot and the mechanisms of DISTAL which allow for the specific characteristics of those settings. Most important, and at the same time unique amongst humanoid robot platforms, are the ability of stand-alone operation of single limbs and the enabling of runtime-metamorphosis.

Using the appealing computational simplicity of time-discrete neural networks (the complexity of which being only bound by the number of processing nodes), we could illustrate that the proposed neural byte-code (NBC) is suitable for graphical editing of neural networks, and at the same time also almost directly translates into compact machine code for the 32-bit ARM RISC processors.

Not only did we present the fully functional robot platform Myon, but also a theoretical framework and a corresponding computational infrastructure, the accompanying application software BrainDesigner, and references to whole-systems example of the robot which is able to autonomously locate, grip and relocate objects by purely neural control paradigms which have been realized with DISTAL. Further research will focus on adaptive neurons and synapses, learning rules, and networks for self-explorative behavior.

Acknowledgements

This research has been carried out at the Neurorobotics Research Laboratory, Humboldt-Universität zu Berlin, with support from the EU FP7 project ALEAR.

References

Amoretti M, Reggiani M (2010) Architectural paradigms for robotics applications. Advanced Engineering Informatics 24(1):4 – 13

Arkin RC (1987) Towards cosmopolitan robots: intelligent navigation in extended man-made environments. PhD thesis

Baillie J (2005) Urbi: Towards a universal robotic low-level programming language. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005.(IROS 2005), pp 820–825

Balkenius C, Morén J, Johansson B, Johnsson M (2010) Ikaros: Building cognitive models for robots. Advanced Engineering Informatics 24(1):40 – 48

Brooks R (1986) A robust layered control system for a mobile robot. Robotics and Automation, IEEE Journal of 2(1):14–23

Hawes N, Wyatt J (2010) Engineering intelligent information-processing systems with cast. Advanced Engineering Informatics 24(1):27 – 39

Heintz F, Kvarnström J, Doherty P (2010) Bridging the sense-reasoning gap: DyKnow - Stream-based middleware for knowledge processing. Advanced Engineering Informatics 24(1):14 – 26

Hild M (2007) Neurodynamische Module zur Bewegungssteuerung autonomer mobiler Roboter. PhD thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II

Hild M, Siedel T, Benckendorff C, Kubisch M, Thiele C (2011a) Myon: Concepts and Design of a Modular Humanoid Robot Which Can Be Reassembled During Runtime. In: Proceedings of the 14th International Conference on Climbing and Walking Robots, Paris, France

Hild M, Siedel T, Geppert T (2011b) Design of a Passive, Bidirectional Overrunning Clutch for Rotary Joints of Autonomous Robots. In: International Conference on Intelligent Robotics and Applications (ICIRA 2011)

Hild M, Thiele C, Benckendorff C (2011c) The Distributed Architecture for Large Neural Networks (DISTAL) of the Humanoid Robot MYON. In: International Conference on Neural Computation Theory and Applications (NCTA 2011)

Höfer S, Spranger M, Hild M (2012) Posture Recognition Based on Slow Feature Analysis. In: Steels L, Hild M (eds) Language Grounding in Robots, Springer, New York

Hosoda K, Sakaguchi Y, Takayama H, Takuma T (2010) Pneumatic-driven jumping robot with anthropomorphic muscular skeleton structure. In: Autonomous Robots

Hülse M, Hild M (2010) Informatics for cognitive robots. Advanced Engineering Informatics 24(1):2 – 3

Konolige K, Myers K (1998) The Saphira architecture for autonomous mobile robots. Artificial Intelligence and Mobile Robots: case studies of successful robot systems pp 211–242

Kubisch M, Benckendorff C, Hild M (2011a) Balance Recovery of a Humanoid Robot Using Cognitive Sensorimotor Loops (CSLs). In: Proceedings of the 14th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines

Kubisch M, Werner B, Hild M (2011b) Using Co-Existing Attractors of a Sensorimotor Loop for the Motion Control of a Humanoid Robot. In: International Conference on Neural Computation Theory and Applications (NCTA 2011)

Kubisch M, Benckendorff C, Werner B, Bethge C, Hild M (2012) Neural Implementation of Behavior Control. In: Steels L, Hild M (eds) Language Grounding in Robots, Springer, New York

Lindstrom M, Oreback A, Christensen H (2000) Berra: A research architecture for service robots. In: IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA'00, vol 4

Martínez-Barberá H, Herrero-Pérez D (2010) Programming multirobot applications using the thinkingcap-ii java framework. Advanced Engineering Informatics 24(1):62 – 75

Migliore SA, Ting LH, DeWeerth SP (2010) Passive joint stiffness in the hip and knee increases the energy efficiency of leg swinging. In: Autonomous Robots

Mitchinson B, Chan TS, Chambers J, Pearson M, Humphries M, Fox C, Gurney K, Prescott TJ (2010) Brahms: Novel middleware for integrated systems computation. Advanced Engineering Informatics 24(1):49 – 61

Nilsson N (1984) Shakey the robot. Technical Note 323, Stanford Research Institute (SRI), Menlo Park, CA

Nilsson N, Fikes R (1970) Strips: a new approach to the application of theorem proving to problem solving. Technical Note 43, Stanford Research Institute (SRI), Menlo Park, CA

Orebäck A, Christensen H (2003) Evaluation of architectures for mobile robotics. Autonomous Robots 14(1):33–49

Quigley M, Gerkey B, Conley K, Faust J, Foote T, Leibs J, Berger E, Wheeler R, Ng A (2009) ROS: an open-source Robot Operating System. In: Proceedings of the Open-Source Software workshop at the International Conference on Robotics and Automation (ICRA)

Siedel T, Hild M, Weidner M (2011a) Concept and Design of the Modular Actuator System for the Humanoid Robot MYON. In: International Conference on Intelligent Robotics and Applications (ICIRA 2011)

Siedel T, Lukac D, Geppert T, Benckendorff C, Hild M (2011b) Operating Characteristics of a Passive, Bidirectional Overrunning Clutch for Rotary Joints of Robots. In: International Symposium on Information, Communication and Automation Technologies (ICAT 2011)

Steels L, Spranger M (2009) How experience of the body shapes language about space. In: Proc. of the 21st Internat. Joint Conf. on Artifical Intelligence (IJCAI'09)

Steels L, Spranger M, van Trijp R, Höfer S, Hild M (2012) Emergent Action Language on Real Robots. In: Steels L, Hild M (eds) Language Grounding in Robots, Springer, New York

http://www.springer.com/978-1-4614-3063-6

Language Grounding in Robots
Steels, L.; Hild, M. (Eds.)
2012, XII, 276 p., Hardcover
ISBN: 978-1-4614-3063-6