A modular approach for controlling mobile robots

K. Regenstein, T. Kerscher, C. Birkenhofer T. Asfour, J.M. Zöllner, R. Dillmann

Interactive Diagnosis- and Servicesystems (IDS), Research Center for Information Technologies (FZI),
Karlsruhe, 76131, Germany
**E-mail: regenstein@fzi.de, kerscher@fzi.de, birkenhofer@fzi.de, asfour@fzi.de, zoellner@fzi.de, dillmann@fzi.de*
www.fzi.de/ids

A variety of different robots was built at our institute. As these robots differ as well in size, shape and in actuation principle it would be very time consuming and inefficient to tailor a computer and hardware architecture especially to the specific robot. In this paper it will be described how common aspects in robot control can be identified and how modular hardware components can be derived from a modular software framework and a respective computer architecture. A decentralized computer architecture based on embedded PC systems connected to local controller modules via CAN Bus was developed. The requirements and restrictions that led to the development of these controller modules and their associated power amplifier boards will be described

Keywords: Computer Architecture; Modular Control Concept; Hardware/Software Co-Design

1. Introduction

In a large number of robotic systems a decentralized architecture is used.[1–4] At the Research Center for Information Technologies (FZI) different kind of robots - like humanoid robots, four- or six-legged walking machines, mobile platforms and snakelike sewer inspection robots - are developed. For these robots we designed a computer architecture based on embedded PCs and distributed controller modules connected with each other via one or more CAN-Busses.[5] Though the requirements concerning the distributed components are quite different we wanted to implement a persistent design that could be used in all robots with only small amount of adaptation. The main issues for the controller modules used in our robots are space requirement, power consumption, several inputs for sensor value acquisition and communication interfaces (i.e. CAN-Bus). As none of the available of-

the-shelf products suited all these needs we decided to build a controller module and an associated power amplifier ourselves.

2. Computer Architecture

The mechatronical construction of a robot can roughly be divided into mechanical aspects and into aspects of setting up the electronic and computer system. In this section we will describe how the electronic system of our robots is set up and how a computer architecture suiting the needs in these robotic systems was designed.

We started by identifying the concepts of how a robot should accomplish given tasks and thus proposing a control architecture that then was the basis for developing the computer architecture. We chose a hierarchically organized control system for the robots with the three following levels:[6]

- The task planning level specifies the subtasks for the multiple subsystems of the robot. Those could be derived from the task description autonomously or interactively by a human operator
- The task coordination level generates in sequence/parallel primitive actions for the execution level in order to achieve the given task goal. The subtasks are established by the task planning level. The execution of the subtasks in an appropriate schedule can be modified/reorganized by an operator using an interactive user interface
- The task execution level is characterized by control theory to execute specified sensory-motor control commands. This level uses task specific local models of the environment and objects, which represent the active scene

According to the control architecture the computer architecture is structured into three levels as well. Choosing suitable devices for these three levels yielded that the requirements of the task planning and task coordination level could be met with industrial PCs and PC/104 systems. As cabling in the robot is a major issue it is desirable to reduce cabling efforts as much as possible. Because of this we decided to use a decentralized system for sensory-motor control. By placing controller modules close to the motors and sensors cabling can be reduced to one common power supply and a bus connection. Wires for supplying the motors and connecting the sensors to the controller can be kept short and have not to be passed through moving joints. To fulfil the requirements of the task execution level we designed the so called Universal Controller Module (UCoM) that in combination with our

Fig. 1. The Universal Controller Module (UCoM) (upper left) and the 3way-brushdriver (lower left); Schematic overview of data flow on the UCoM and to the piggyback board

motor controller 3way-brushdriver (Fig. 1) is responsible for the sensory-motor control of the robot. The features of the combination of UCoM and 3way-brushdriver will be described in section 3 in more detail.

As software framework we use the Modular Controller Architecture (MCA2)[7] that was developed at our institute and is available under GPL online here.[8] The idea behind MCA2 is to structure the software into reusable modules with simple interfaces. Each module has the three data channels: control data, sensor data and parameters. Via these data channels information is exchanged between the modules. A number of modules can be combined into a group which has the same interface as one module.

3. Controller modules on sensory-motor level

Following the modular strategy that is realised in the software framework MCA2 we wanted to reach this modularity in the computer architecture as well. To achieve this goal not only for one of our robots but spanning all different robots we have to choose a common controller unit. This is important to reduce programming efforts as well. For example you can implement a PID-controller only once and as you use the same hardware that can run the same software in different robots you only have to adapt the PID parameters for the chosen joint. As already mentioned the main issues for the controller modules are space requirement, power consumption, several inputs for sensor value acquisition and communication interface. Especially for motor control the mandatory requirements for the controller module were

- Suitable to control three brushed DC motors at 24 V at up to 5 A
- Achieve cycle times as low as 1 ms
- Able to decode six quadrature coded signals
- Small outline, positioning close to the actuator possible
- Low power consumption
- Interface to access CAN-Bus

In some applications the 5 A might not be sufficient but this was the maximum current that could be realized without exceeding the space limitation. Besides electrical motors we use other actuation principles in our robots. For example in one of our walking machines - Airbug[9] - fluidic muscles

were used and there is still ongoing research evaluating fluidic muscles as actuation. For this kind of actuation a valve driver is needed. Furthermore in some robots like LAURON[10] we need extended sensor input like posture information from gyroscopes and acceleration sensors. To avoid building a special controller module for each of these applications we decided to split the controller module into one part that actually contains the controller and one part that contains the power amplifier, the valve driver or sensor acquisition electronics. As mentioned above we named the part with the actual controller Universal Controller Module (UCoM) as it will be universally used in our robots together with the respective piggyback board.

3.1. *Universal Controller Module - UCoM*

The choice for a suitable microcontroller/DSP for the UCoM could be narrowed down rather quickly as nearly no available controller had all the required features. The Freescale "DSP56F803 16-bit Hybrid Controller"[11] came closest to our needs. This controller is a DSP featuring a set of peripherals usually only known from microcontrollers.

Though this hybrid controller nearly matches the requirements it still misses some essential features. On the one hand side it does not have enough general purpose IOs to control three motors on the other hand it only has two quadrature timers capable of decoding quadrature coded signals. To extend the DSP's flexibility we decided to put an FPGA next to it. As suitable FPGA we chose the Altera EPF10k30A. With this FPGA we can equip the UCoM with a high number of general purpose IOs. We gain a high flexibility concerning routing and assignment of pins to the piggyback board. Through the FPGA we can reassign most of the signals so that it suits the used piggyback board. The FPGA is also used to preprocess data that is exchanged between the UCoM and the piggyback board.

By this approach we can disburden the DSP from tasks that are done in hardware more efficiently. For example we implemented six decoders for quadrature encoded signals. The communication between DSP and FPGA takes place via the external memory interface. As FPGA and the external RAM that we integrated on the UCoM share the external address range we implemented an address decoder into the FPGA. This address decoder deasserts the chipselect for the lowest 64 addresses of the external address range and receives the sent data. For all other addresses the data is routed to the external RAM so that nearly the whole external memory range is available to the DSP and only the lowest 64 addresses are used as FPGA-registers. The DSP always initiates communication with the FPGA

by writing to or reading from a FPGA-register. These registers are used to exchange data between the two devices. So to get the value of a quadrature coded signal all the DSP has to do is access the respective external RAM address.

The UCoM combines the Freescale "DSP56F803 16-bit Hybrid Controller", an external RAM and an Altera EPF10k30A on one board. It is interfaced to the desired piggyback board via two 60-pin 0.8 mm pitch board-to-board connectors. Via this connector the UCoM is supplied with a 5 V power supply. From this 5 V we generate the 3.3 V that are needed on the UCoM. We do not directly feed 3.3 V to the UCoM to avoid problems with voltage drop or disturbances that must be expected due to the wiring close to the motor power wires. The 3.3 V generated on the UCoM are then fed back to the board-to-board connector to be available on the piggyback board.

3.2. *Motorcontrol Board*

As the main actuation principle in our robots are electronic motors the first piggyback board we developed was a power amplifier able to drive three brushed DC motors. We named this piggyback board 3way-brushdriver. On the 3way-brushdriver we integrated three H-Bridges which are driven by a 3-phase brushless DC motor controller chip each. This motor controller chip can be configured to drive brushless or brushed motors. To drive brushless motors the hall-inputs to the driver must be connected to the hall-sensors of the motor. To drive brushed DC motors the hall-inputs are simply tied to ground. We chose this motor controller chip so that it can be interfaced by software in the same manner if we need to design a piggyback board for brushless DC motors in the future. In both branches of the H-Bridge we integrated a shunt via which we can measure the motor current for each motor. As the motors are driven by PWM signals we had to use an OP-AMP with a high gain-bandwidth-product to amplify the signals for use in the AD-Converter.

As we wanted to keep the UCoM as small as possible we decided to put all interface connectors except the ones that are directly wired to the DSP like CAN-Bus, serial communication interface and JTAG to the piggyback board. Thus the piggyback board is responsible for supplying the UCoM with the 5 V input voltage. The 3way-brushdriver has a connector for input of the aforementioned 5 V, 24 V as power supply to the motors and a common Ground.

Further connectors on the motor control board are six small connectors

Fig. 2. Robots in which the presented computer architecture is actually used: ARMAR III (left) and LAURON IV (right)

for the quadrature coded encoder signals. Each of these connectors has six pins, two of which carry supply voltage of 5 V and ground for the encoder, two are the quadrature channels A and B. The remaining two are an index signal and a pseudo absolute code. A schematic overview of the dataflow on the UCoM and to the piggyback board can be found in Fig. 1.

3.3. *Software Components on DSP*

To extend the modularity down to the code for the DSP each UCoM is treated as one module in the MCA2 software framework. Thus the UCoMs can be seamlessly integrated into the software running on the linux PC via a MCA-driver interfacing to the CAN-Bus. A number of different basic application programs for the UCoM were developed so that they can be used in most of the robots with only small adaptations. In the humanoid robot ARMAR-III[12] we only use one generic program for all different joints. This generic program can be adapted via a configuration file that is evaluated at system startup. Other basic application programs that are implemented are: direct_pwm, a p_controller, a pid_controller and a Speed/Position Controller. Further control programs including for example time discrete and torque control algorithms are in development.

To download these programs to the UCoM a bootloader that accepts data via CAN-Bus is used. This is very convenient if programming is done frequently in the development phase.

3.4. *Functions on FPGA*

The FPGA can be seen as an extension to the DSP. It equips the DSP with a large number of general purpose IOs and processes data that is exchanged between the DSP and the piggyback board. As stated above we use the memory interface to communicate between DSP and FPGA. The function blocks in the FPGA are programmed in VHDL. There are some blocks that are common to all designs independent of the plug on board: An Address Decoder, a version supervision sytem, Initialization and a Watchdog.

For the introduced 3way-brushdriver the following function blocks were already implemented into the FPGA: A Quadrature decoder, Pseudo Absolute Decoder, Serial Synchronous Interface and a Motor Control register. Further modules can easily be integrated into the FPGA as they are needed.

4. Summary and Outlook

In this paper we presented a modular concept to control robots which we applied to our robots ARMAR III and LAURON IV (depicted in Fig. 2). This concept includes a control architecture from which the used computer architecture was derived, a modular software framework and the development of a hardware architecture that can be mapped into the computer architecture. The focus of this paper was on setting up a modular system that can be used in a variety of robots so that not only software components can be reused but that also the hardware is interchangeable. It was laid out how this goal was achieved and especially the development of the UCoM and the 3way-brushdriver were described. Some examples of our robots in which these control devices are successfully used were presented above. In ongoing research we will implement this concept in robots that are going to be built. Most likely further piggyback boards will be designed for that purpose.

5. Acknowledgement

This work is partially supported by the German Research Foundation (DFG) within the Collaborative Research Centre SFB 588 (Humanoid Robots - Learning and Cooperating Multimodal Robots).

References

1. J.-Y. Kim, I.-W. Park, J. Lee, M.-S. Kim, B. kyu Cho, and J.-H. Oh, "System Design and Dynamic Walking of Humanoid Robot KHR-2," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, April 2005*, 18-22 April 2005, pp. 1431–1436.
2. T. Sugihara, K. Yamamoto, and Y. Nakamura, "Architectural design of miniature anthropomorphic robots towards high-mobility," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, 2-6 Aug. 2005, pp. 2869–2874.
3. J. Butterfass, G. Hirzinger, S. Knoch, and H. Liu, "DLR's multisensory articulated hand. I. Hard- and software architecture," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 3, 16-20 May 1998, pp. 2081–2086vol.3.
4. R. Bischoff and V. Graefe, "HERMES - a versatile personal robotic assistant," *Proceedings of the IEEE*, vol. 92, no. 11, pp. 1759–1779, Nov. 2004.
5. K. Regenstein and R. Dillmann, "Design of an open hardware architecture for the humanoid robot ARMAR," in *Humanoids 2003, International Conference on Humanoid Robots, Conference Documentation, October 1 - 3, 2003, Karlsruhe and Munich, Germany*, October 2003, p. 3.

6. T. Asfour, D. Ly, K. Regenstein, and R. Dillmann, "Coordinated Task Execution for Humanoid Robots," in *Experimental Robotics IX*. Springer Berlin / Heidelberg, 2006, vol. 21, pp. 259–267.
7. K. U. Scholl, J. Albiez, and B. Gassmann, "MCA - An Expandable Modular Controller Architecture," in *3rd Real-Time Linux Workshop, 2001, Milano, Italy*, 2001.
8. FZI, "Modular Controller Architecture Version 2." [Online]. Available: http://www.mca2.org/
9. K. Berns, J. Albiez, V. Kepplin, and C. Hillenbrand, "Airbug - Insect-like MachineActuated By Fluidic Muscle," in *Proceedings of CLAWAR 2001, Int. Conference on Climbing and Walking Robots*, 2001.
10. B. Gassmann, T. Bär, J. Zöllner, and R. Dillmann, "Navigation of Walking Robots: Adaptation to the Terrain," in *Proceedings of CLAWAR 2006, 9th International Conference on Climbing and Walking Robots (CLAWAR)*, 2006.
11. Freescale, *DSP56F803 Data Sheet*. [Online]. Available: http://www.freescale.com/files/dsp/doc/data_sheet/DSP56F803.pdf
12. T. Asfour, K. Regenstein, P. Azad, J. Schröder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann, "ARMAR-III: An Integrated Humanoid Plattfrom for Sensory-Motor Control," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, 2006.