Standardized embedded networking in modular service robots

Holger Zeltwanger, *CAN in Automation e. V.*

***Abstract*—Service robot modules can be describe on different functional abstraction levels. The introduced highest level of abstraction specifies just a few complex key sub-systems, which communicate via a robot internal (embedded) CANopen network. On a lower level of abstraction, the functional elements of the sub-systems are specified. They also may communicate via deeply embedded CANopen networks. All these CANopen networks are just virtual networks. They may be implemented on just one physical CANopen network or if more bandwidth is required on different CANopen segments interconnected by means of bridges and routers, if desired.**

I. Introduction

The market of service robots is highly fragmented. It is still a low-volume market with some rare exceptions such as robotic vacuum cleaners and lawn mowers. The definition of the term "service robot" is somehow vague. IFR (International Federation of Robotics) defines: "A service robot is a robot, which operates semi- or fully autonomously to perform services useful to the well-being of humans and equipment, excluding manufacturing operations." Of course, this includes all the robots for serving elder and disabled people, also a lot of so-called robots to be used in applications, which are dangerous or inconvenient or boring for human beings.

But what is about agriculture robots, medical care robots, not mentioning military robots (which I exclude personally, at all, because I am a peaceful man)? Even the upcoming cooperative robots in industrial applications look more like service robots than classical factory robots manufacturing something. What is about semi- of fully autonomous driving forklifts used in logistic facilities?

I think, it is not interesting for the suppliers, if this is a service robot as defined above or a cooperative industrial robot or something else. We need to specify, if the robot is cooperating with human beings, the politically desired functional safety level for levels of modules. To be honest, I have not thought in detail how this applies to "flying" robots (drones).

We need to standardize sub-systems to be used in different robot applications, in order to increase volumes and decrease production costs. Standardized sub-systems also decrease efforts in system integration. They should be just configured and not programmed anymore. Programming is the business of the sub-system supplier. To be more precise, the task controller sub-system needs still to be programmed by the robot designer, the only one knowing what the autonomous robot should do.

I am not in favor of the term modules. I like to introduce on the highest abstraction level of robot modules the term "sub-system". On this level, we have so-to-say the key functional elements of a robot. In a first attempt, I like to distinguish between the following sub-systems; some of them are optional depending on the required functionality:

- Task controller (including human machine interface inputs in case of semi-autonomous robots)
- Movement controller (not needed for stationary robots)
- Arm controller and/or gripper controller
- Obstacle detector
- Power management controller

On this highest level of abstraction, we just demand, for example, the speed and direction of the movement. We don't care on the implementation of the control functionality. This is done in the next lower level of abstraction. The functional elements of the movement sub-system could be implemented by different kind of wheels, for example. One level lower we have the functional elements of single motion control devices and sensors for speed and position.

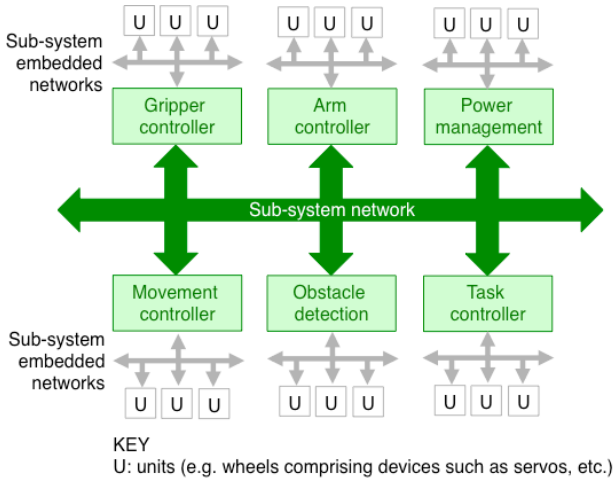

Figure 1. Embedded robot network connecting the sub-systems, sub-layered are deeply embedded sub-system networks, which may use also CANopen technology (application layer and profiles)

This three-level approach of functional elements enables the sub-system and unit suppliers to provide and adapt different technologies independent of the sub-system interfaces. The robot system designer just chooses the appropriate sub-system for his application (e.g. with different wheels); the sub-system interface does not change. It is always just speed and direction or position demand, what is needed as input parameters. The output parameters are also technology-independent: current speed and direction or current position provided as a confirmation on the application level.

Holger Zeltwanger is CiA Managing Director and expert/convener of multiple ISO/IEC working groups (e-mail: headquarters@can-cia.org).

II. SUB-SYSTEM NETWORKING

A. Requirements

In general, the network used for sub-systems should be robust (physical layer) and reliable (data link layer). For battery-powered robots, a low-power mode is desired. It should be suitable also for outdoor applications (extended temperature range). Prices should be reasonable (other high-volume applications). It should be very flexible (application layer) and functional safe communication should be standardized.

The CANopen application layer and communication profile (CiA 301), internationally standardized in EN 50325-4, supports the above-mentioned requirements. It is based on CAN (Controller Area Network) as standardized in the ISO 11898 series, which is used in nearly all passenger cars. The CANopen Safety protocol (EN 50325-5) adds the necessary functional safety features.

B. CANopen basic functions

The CANopen application layer has been pre-developed within a European research project beginning of the 90ties. It was handed over in 1994 to the nonprofit CAN in Automation (CiA) users' and manufacturers' group for maintenance and further developments. Currently, CiA members are developing the next CANopen generation making use of the recently standardized CAN FD data link layer allowing higher bit-rates and larger payload (up to 64 byte).

The CANopen application layer and communication profile specifies several services and protocols. This includes:

- Network-Management (NMT): Highest prior message controlling the CANopen FSA (finite state automaton) in the NMT slave devices and Heartbeat confirming the desired state change request by the NMT master device.
- Device supervision (Heartbeat): Same communication service (see above) is used to detect "missing" devices.
- Confirmed configuration: The SDO (service data object) communication is used to configure and diagnosis the NMT slave devices. This communication service also provides data segmentation and reassembling for objects larger than the maximum payload of a single CAN message (8 byte).
- Real-time communication of process data: The PDO (process data object) communication transmits control and status data without protocol overhead. It is highly configurable and can be optimized to the application requirements.
- Synchronization: The SYNC message can be used to synchronize operations in different NMT slave devices. This is required for example in many multi-axes applications (arms, grippers, etc.)
- Emergency information: In order to indicate in real-time application and communication problems to other NMT slave devices, the high-prior EMCY service and protocol is used.
- Time management: CANopen also provides a system-time (TIME) in the accuracy of milliseconds.

Most important for the use as sub-system network is the flexibility of the PDO communication. Each PDO can be configured regarding the priority on the data link layer, the scheduling (event-triggered, periodical, cyclic and acyclic synchronized), and the content (mapping of process data). In order to avoid undesired delays and "bubbling idiots", the system designer needs to configure the PDO communication properly.

C. Development of robot sub-system profiles

Profiles for robot sub-systems have not been developed yet. This should be done in relation to the standardization activities in ISO TC299 WG6. The movement controller sub-system, for example, should support two kinds of movement control: velocity mode and position mode depending on the functionality of the task controller. The task controller demands the speed/direction or position. The movement controller preforms the desired movement and provides its status information by means of speed/direction or position. Of course, all sub-systems should support a standardized FSA, in order to allow a sophisticated functional behavior, when starting the system.

III. DEEPLY AND VERY DEEPLY EMBEDDED NETWORKING

The defined robot sub-systems may be modularized as well. I would like to call these modules "units". They also may use CANopen communication. The movement control sub-system may comprise several wheel control units communicating via CANopen. The movement sub-system translates the commands of the task control sub-system to single wheel commands in order to achieve the desired movement. This is highly depended on the used wheel technology and hidden to the robot system designer. It is a feature by the sub-system and in the responsibility of the supplier. Nevertheless, this CANopen communication can run on the very same CANopen network used for the sub-system communication. In this case, the movement control sub-system implements two profiles: One to communicate with the task control sub-system and another to talk to the individual wheel control units. Of course, if CANopen doesn't provide sufficient bandwidth, you need to run the two applications on different CANopen segments.

A. Units and devices

Even the units may be modularized. This is the next lower level of abstraction. A wheel control unit may comprise two or more motion control devices (steered wheel), one for the movement and another for the direction. I would like to use the term "device". Of course, such motion control devices may communicate with the wheel controller by means of another virtual CANopen network using the CiA 402 drive and motion control device profile internationally standardized in IEC 61800-7-201/301. This could be done by means of a very deeply embedded network in the wheel units or "folded" on the deeply embedded movement sub-system. For simple applications, this also can be "folded" to the embedded robot network, if enough bandwidth is available.

CiA has not yet standardized a wheel control unit profile. This is something to be done in coordination with the work in ISO TC299 WG6. Available are several CANopen device profiles on the third level of abstraction. Of course, the CiA 402 profile is already suitable for not steered wheels. As mentioned-above, all these details are hidden to the robot system designer.

Other already long-term approved device profiles, which may be used for modularized units include:

- CiA 401: Modular I/O devices
- CiA 404: Measurement devices
- CiA 406: Encoder
- CiA 408: Hydraulic valves and servos
- CiA 410: Inclinometer
- CiA 418/419: Battery/charger
- Etc.

On the level of units or/and sub-systems the profiles CiA 454 (energy management system) and CiA 462 (item detection) might be considered, reviewed, and adapted. The item detection profile was originally developed for construction and agriculture machines. The term "item" is used, because the term "object" is already used for other purposes in CANopen technology.

This scalable networking approach is supported by the CANopen technology. There are also additional specifications for high-availability and redundancy (CiA 302 series). This includes for example the "Flying" NMT master and bus-line redundancy, originally required by maritime electronic systems. To avoid the unique assignment of the necessary node-IDs to each CANopen interface, LSS (layer setting services) can be implemented as specified in CiA 305.

IV. System design environments

In order to simplify system design, several tool manufacturers have developed generic CANopen device and system design tools. They hide some details of the CANopen technology to the user. This simplifies system design. Nevertheless you need some CANopen know-how. Also the use of the standardized EDS (electronic data sheets) formats (ASCII and XML) simplifies the exchange of system data between the tools. CiA also provides to its members a free-of-charge CANopen (CiA 301) conformance test tool.

In order to design the robot communication system independent of CANopen, CiA has mapped the Robotic technology component (RTC) specification by OMG to CANopen. The CiA 460 profile describes what in this paper is named task controller or any of the sub-system controllers using a deeply embedded CANopen network. In addition, the CiA 318 specifies the integration of CANopen networks into the RTC environment. This allows implementing heterogeneous network technologies. Of course, this not necessary when using just CANopen networks or those network technologies using the CANopen data modeling (object dictionary) and CANopen profiles.

V. Conclusion

When describing virtual network architectures, it is useful to standardize several functional abstraction levels similar to the OSI (open systems interconnection) model. In case of modular service robots, a three-level architecture seems to be sufficient: sub-systems, units, and devices. Using CANopen for all levels standardized profiles are advantageous for the system designer (robot), the sub-system supplier (e.g. movement controller), respectively the unit supplier (e.g. wheel controller). Of course, also the device manufacturer benefits from standardized profiles. The new work item of ISO TC299, defining service robot modules, is such an attempt to standardize future-proofed modules.

References

[1] CANopen specifications (CiA 300 series and CiA 400 series): Partly available for non-members on CiA's website (www.can-cia-org)

[2] ISO and IEC standards available on the standardization bodies' websites (www.iso.org and www.iec.ch)

[3] Holger Zeltwanger, *CANopen virtual device architectures*, in iCC proceedings 2003, Erlangen (Germany), also available on the CiA website for download

[4] Several articles in the CAN Newsletter magazine published by in Automation (CiA), Erlangen/Nuremberg (Germany); since 2012 downloadable as pdf (www.can-newsletter.org)

[5] RTC specification, OMG (Object Management Group): available on OMG's website (www.omg.org/spec/RTC/1.1)