Object Oriented Distributed Control with CAN

Ir. E.H. van de Waal
Embedded Software Engineer
Intulogic bv.

A method for distribution of intelligence is presented in which the Object Oriented Paradigm is used in software as well as in the communication protocol and application level.

Practice has shown that an optimal distribution of functionality can reduce bus-load, increase flexibility and improve the diagnostic capabilities. Some rules will be presented in order to reach this optimum. The relationship between network-management and application software will be addressed. It is also shown that efforts (and costs) for developing and maintaining software can be reduced.

It will be explained why CAN is a suitable network for distributed intelligence. Some problems facing distributed intelligence will be discussed and solutions will be given. Furthermore some methods will be given to increase the availability (up-time) of distributed systems by means of Smart CAN bridges and selective redundancy.

Introduction

As hardware costs are decreasing, a trend can be observed in the architecture of control systems. When hardware was expensive, a common approach was to concentrate all intelligence in a central processing unit, as shown in figure 1. This approach had several draw-backs:

- Wiring and testing costs were high
- Software was complex and thus difficult to develop, verify and maintain.
- Extension or modification of an existing installation was expensive

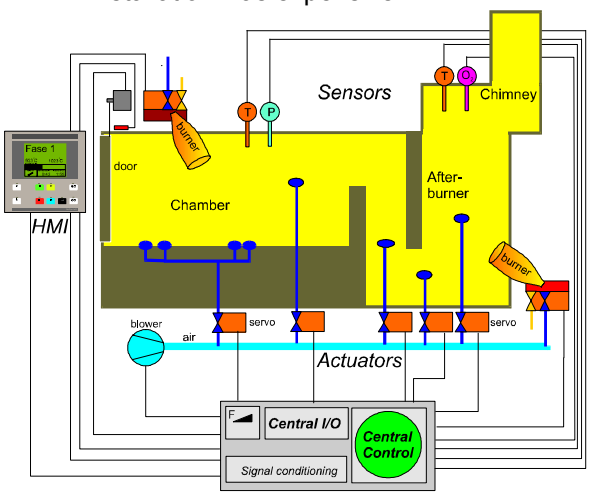


Figure 1, Centralised Control

As hardware costs decreased, it became possible to distribute intelligence, leading to the advent of DCS.

On a smaller scale, a similar move towards decentralisation is now under way. As hardware costs are decreasing, intelligence is being moved from e.g. PLC's towards individual sensors and actuators. A common method is to distribute the signal conditioning hardware in a remote-IO architecture, shown in figure 2.

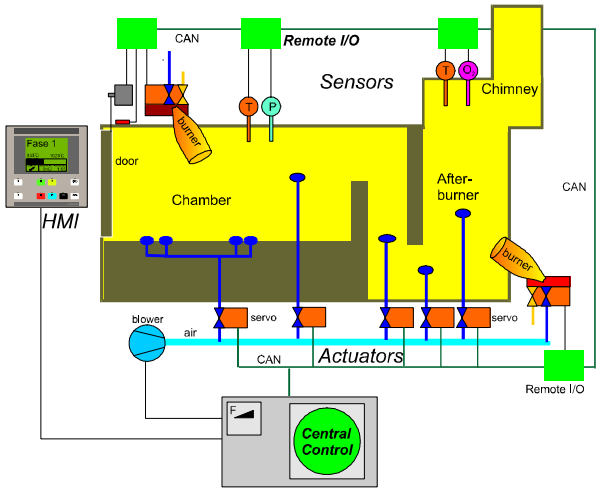


Figure 2, Remote I/O

Through using this architecture, wiring costs can be reduced significantly, but the other issues mentioned above are not addressed. There still is a considerable amount of centralised software, which is becoming increasingly difficult to develop and maintain as installations are subject to increasingly complex demands in the area's of product

quality, energy consumption, environmental impact, reliability, etc.

An architecture which addresses all of these issues, and thus minimises the combined cost of control system design, implementation, maintenance and modification, is the situation where both hardware and software are distributed, as shown in figure 3.

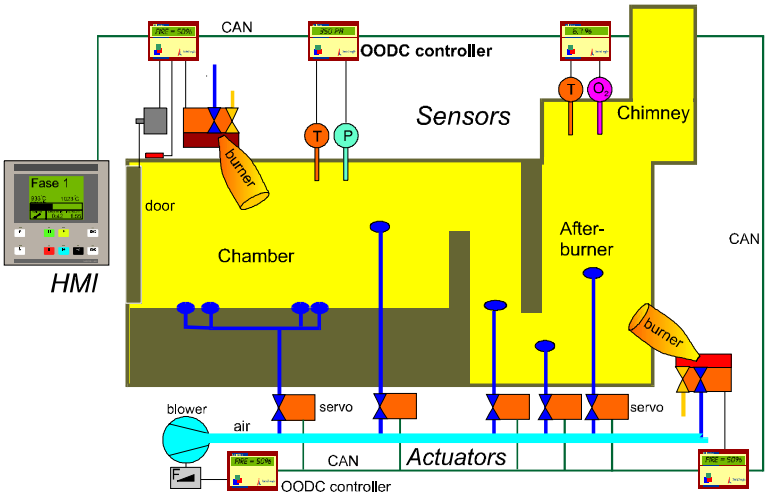


Figure 3: Distributed Control

In this paper, a method for designing such a distributed control system will be presented.

Object Oriented Distributed Control

Design Paradigm

When designing a control system, existing engineering practice is to draw a PI&D schematic of the installation, with all sensors, actuators and processing elements (e.g. PID's and SEM's), connected using lines. In essence, the designer is drawing an object oriented design of the control system, in which the processing elements could be called 'control objects'.

As application designers look at installations in an object oriented way, the implementation of the control system will be more a reflection of the installation if object orientation is used during the whole design and implementation process (e.g. Coad and Yourdon, 1993)

The designer requires an environment which allows him/her to create objects representing the objects used in the design, and interconnect and configure them. There should be no need for a paradigm shift on the side of the designer as he implements the control system.

Three elements are desired for this purpose:

- The control objects themselves
- Hardware modules on which the control objects run, and which interface to sensors and actuators.
- An computer aided environment for design, configuration and maintenance of a system.

From a designers perspective, the control objects are the most essential. The tools only support the designer and user in handling them, the hardware only gives them the facilities they need to perform their function.

The control objects correspond to the elements used in the design of the control system. With a surprisingly small number of objects, control systems can be implemented for a large variety of applications. The application specific behaviour is defined by the relation between objects but not in the objects itself. Thus, once these control objects are developed, they can be re-used without any need for modification!

OODC is a straight-forward and intuitive way to software re-use.

There is a clear analogy with the business information technology. Whilst each company has unique properties, many common processes can be recognised, like purchasing, invoicing and inventory control. ERP suppliers (e.g. SAP) utilises these common processes to minimise development and testing efforts by making standardised configurable software modules.

Such a combination of software-components, hardware-modules en designer environment is developed at Intulogic. It is meant to enable applications engineers to create and maintain there own control systems without needing to have knowledge about software engineering or network technology. It is called the *'Linked Objects Family'* (LOF^{tm})

Linkpoint Approach

Each control object is a self-contained unit, which is fully responsible for reacting in an appropriate manner to external events. An object can accept information from other objects, and/or provide information to other objects, depending on its function. For each object, several types of information can be defined, which can be seen as connection points for communication: *linkpoints*, as shown in figure 4.

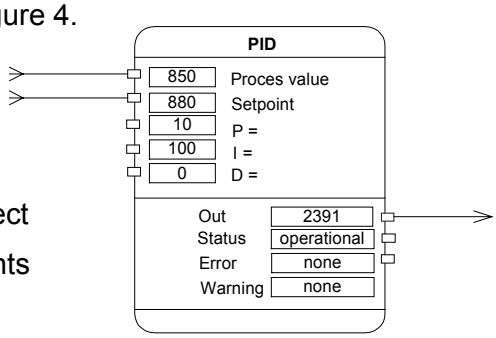


Figure 4:

Control object

with linkpoints

As each object is responsible for its own actions, but not for the actions of other objects, it follows that the responsibility of an object ends at the moment it provides its information. Whether any other object uses that information, is not its responsibility, but that of the application designer. This behaviour is commonly called the *producer / consumer* model. Information is broadcast by the *producer* and can be used by any number of other objects, the *consumers*.

Typically, a consumer does not acknowledge reception of the message. As consumers (often associated with actuators) should be intrinsically safe, they should know how to respond to error conditions (such as communication failure). Therefore, the consumer should be responsible for checking the communication with the producer. The producer (often associated with a sensor) can not respond to an error condition in a way which would make the control system intrinsically safe*.*

With the linkpoint concept, it is often not necessary for a control object to have any 'internal' parameters. Most parameters, for example the factors of a PID regulator, can be implemented as linkpoints. This gives the designer easy means to modify these, and also makes it straight-forward to implement for example a gain-scheduling scheme.

Hierarchical Structure

For correct communication, every object in a control system must be identifiable. It is possible to give each object a unique number, but such a number gives no information which might help the user to interpret network communication. A much more straightforward and intuitive approach is to relate the identifier of the object to its *type* and *location* in the control system. A *location* can then be defined as an area which is so small that there is at most one control object of each type in it. Thus, each control object is uniquely identified by its object type and location. Objects of different types which are located close to each other, can share the same location number. For example, a PID can have the same location number as the servo it controls.

However, when building large installations, a type and location number is not sufficient to keep message interpretation intuitive. A third number is needed, which holds information on the *area* in which the object is located. An *area* is defined as a part of the installation which has an intuitive boundary, and which preferably is repeated in the system. The control systems for each area can then be copies of each other, using different area numbers. Using this scheme, not only software components can be re-used, but complete system designs!

It can be useful to address several objects at the same time by using wild-cards. To use a common example, in a fully automated house, it might be useful to switch all 'light' objects in the 'kitchen' area on at the same time. This could be done by using the wildcard 'all' for location.

In the perception of the designer there can be a close relation between several objects of different types and in different locations, e.g. a complete control loop in a system. For this purpose, the concept of a *group* can be introduced. Each object can be said to belong to one or more groups, which could span many area's and locations. A group can be defined as any number of objects which have a common attribute, as perceived by the user. This attribute might relate to the function or purpose as required by the designer.

An example of such a group might be all the lights which would need to be switched on in case there is a power failure in a large building, and only limited power is available through the back-up power system. All lights would then be switched off, and only the lights belonging to group 'emergency' would be switched on, using only 2 messages which need not be changed if the lighting system in (part of) the building changes.

Wildcards can be used in any field: Group, Area, Location, or Object. Thus, it is possible to address any combination of objects and synchronise them using a single message.

This not only simplifies system design but also allows for extremely fast run time diagnostics, using commands like: "*All Objects in Group 'pressure control', do a self-test"* or *"All Servo's in Incinerator 1 what is your error state?".*

Minimising network traffic

As network bandwidth is limited due to hardware constraints, it is worthwhile to use a structure which minimises network traffic.

First of all, network traffic can be minimised by designing control objects which are 'loosely coupled', i.e. which do not depend on other objects for their internal operations. This is a consequence of proper object oriented design, and also a pre-requisite for object re-use (e.g. Carroll and Ellis, 1995). Objects that are loosely connected require few items of information to be communicated in order to carry out a specific task.

An other effective way to minimise network traffic is to filter all sensory data before it is transmitted on the network, using an intelligent software component. Rules can be built into the software to determine when new sensory information is relevant, and when it is not.

Network traffic can be minimise further by careful selection of the location of control objects. Consider the common control loop shown in figure 5. It is assumed that for every sensor reading, a PID output and an actuator position are generated. With OODC, the location of the PID can be chosen freely.

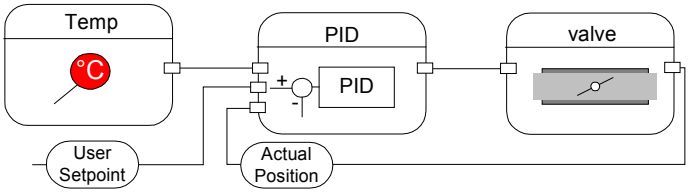


Figure 5: Minimising bus traffic

For objects which are located on the same module, the designer can choose not to broadcast the information on the network, but to let the objects communicate internally.

Then, by locating the PID with the object used to drive the actuator, network traffic can be reduced by a factor three compared with the situation where the PID is in a separate location, and by a factor two compared with locating the PID with the sensor. In most remote-I/O systems the PID is separate from both sensor and actuator, thus maximising network traffic…

OODC Implementation Issues

Network

The network is a vital component of an object oriented distributed control system. The network needs to satisfy the following requirements

- High reliability (network is essential to operation of control system).
- Sufficient bandwidth.
- Support for producer / consumer communication model.
- Deterministic

CAN satisfies all these requirements and more. Because CAN is used extensively in the high-volume automotive industry, it also has the following advantages:

- Supported by most popular micro-controller families.
- Low cost.

This makes CAN well suited for use in distributed control systems.

OODC Implementation Using CAN

The primary need for an object-oriented distributed control system is to be able to produce a linkpoint value, to be consumed by an other linkpoint. A protocol should be used which supports the designer in configuring these linkpoints. A large number of protocols exist for CAN which are capable of doing this, but most require specialist knowledge. Therefore, a more straight-forward and intuitive protocol is used for LOF.

The LOF protocol can use the following format if implemented with a 11 bits ID frame:

Information message (7 data bytes)

Id	rtr	len	0	1	2	3	4	5 & 6
xx	0	7	group	area	location	object	linkpoint	value

Here, the Id is related to the node and to the priority. It does not contain any information on the object. The scope of the id is limited to its subnet and no longer relevant if the message is transported across bridges or gateways.

The rest of the information is related to the object. The Area.Location.Object combination is unique in the whole installation.

Each node autonomously gets it id's. It can use the same id for all objects in it but also use different id's if it need to communicate on several priority categories.

For any specific installation, a database would exist which maps the numbers for Group, Object, Area etc. to a text string. For example, the user might see the following message

id	rtr	len	Group	Area	Location	Type	Linkpoint	Value
514	0	7	All	Filter	Cyclone	PID	Output	12000

which would be the new value calculated by a PID controller.

By omitting the value (resulting in a message with 5 data bytes), a request can be posted for the linkpoint to be produced. This mechanism is similar to Remote Transmit Request messages in CAN.

A draw back of this protocol is that it is not sufficient that CAN controllers filter the relevant identifiers in hardware. In this implementation additional software filtering is necessary.

Network Management

The acceptance of industrial networks is not as high as it could be in several application area's. The issues of network management is sometimes considered a factor in this.

From the point of view of application engineers, network configuration tools at the best solve or simplify problems with would not have been there in the first place if no network were used at all.

CAN has proven to be an ideal network for OODC since it is possible to use it without any network considerations of application engineers.

There is no master in CAN and there is no need for giving nodes an address.

Although the nodeguarding mechanisms in remote I/O systems are typically considered network management tasks, it is not required for OODC. In OODC the consuming object can take appropriate action if vital information is not available. This is an application related issue, not a decision what can be made on a system level.

However, for communication using a CAN network, it must be guaranteed that no nodes will ever try to transmit a message with the same message ID.

For this purpose, Intulogic's Autonomous Identifier Distribution and Allocation algorithm (AIDA) is used. Each hardware module autonomously requests an CAN ID during start-up, in different priority categories if necessary.

This allows hot-swapping of defect modules and extension of control systems during op-

eration by adding new modules, without having to update a centralised control unit.

System Testing

Testing is an important part of control system implementation. Software components need to be tested exhaustively during development, and installations need to be tested thoroughly before commissioning to ascertain that all modules are implemented correctly. However, good testing is rapidly becoming extremely difficult, as the number of possible states increases exponentially with the number of inputs and outputs in a system. It is imperative that a modern design method allows for easy testing.

OODC greatly simplifies testing at all levels. Software components are of small size, making them relatively easy to test. Also, existing software can be re-used without any modifications, and thus without need for re-testing them in a new control system.

The linkpoint concept allows for easy testing of an implemented control system. At any point in the system, the user can inspect linkpoint communication, thus verifying correct behaviour of all components. Also, by simply sending a message, any event can be simulated, and behaviour observed.

As all messages are broadcast, logging of system behaviour is straight-forward. With OODC, it is not only possible to log sensor / actuator data, but it is also possible to log all information which would be 'internal' if the software were not distributed. The source of any problem can thus be traced quickly.

Trouble shooting

The characteristics of OODC which aid the user in testing, are also of tremendous benefit in trouble-shooting. By observing linkpoint communication, and communicating with the control objects, the cause of a problem can be found very quickly. When the problem is found, the user can decide whether to replace defective hardware, or construct a work-around. Through the flexibility of linkpoint communication, a work-around is often quite easy to implement, for example by disabling certain parts of the control system, simulating other parts of it, or overriding faulty measurements during operation. Of course, it is up to the service engineer to decide whether it is safe to do so.

Tools for OODC

Due to the close relationship between the original design and the actual control system, it is not difficult to built tools which make the conversion from design to implementation almost trivial. A tool-set for OODC should support the following facilities:

Creation of a control system through computer aided design: the ability to create control objects, assign appropriate area and locations and link them. An example is shown in figure 6.

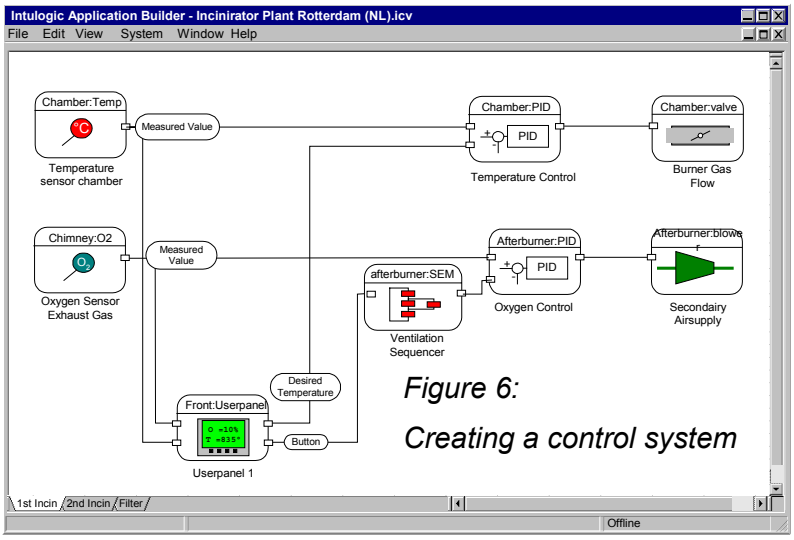


Figure 6:

Creating a control system

Configuration of a control system: the ability to place control objects at hardware modules, configure the linkpoints and set parameters. An example is shown in figure 7.

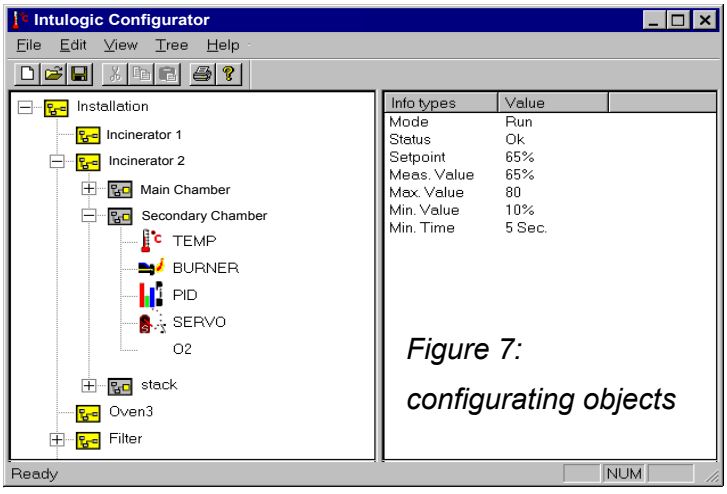
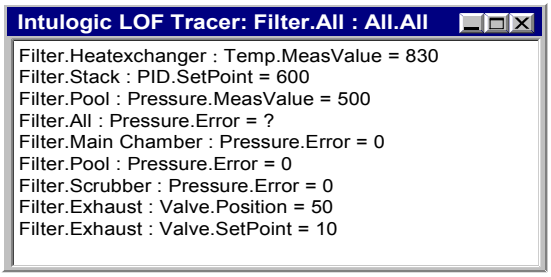


Figure 7:

configurating objects

Maintenance of a control system: Supervise a system, diagnose any problems, correct them through changing the configuration (if applicable) and determine which hardware needs maintenance, repair or replacement.



An example is shown in figure 8.

Figure 8: Monitoring the CAN bus

Supervision tools

Supervision of installations can be done with common available SCADA systems. However, since SCADA systems typically ar

e I/O or variable oriented however, there would be a need for a paradigm swift.

OODC supervision tools could also use the same databases as the developer did and response directly to the CAN messages, which is exceptional straightforward and intuitive. With very little effort it is possible to create userpanels to present system status and allow user interaction. Standard trending software can also use the shared database and allows users or application engineers to investigate trends or relations. Some examples are shown in figure 9.

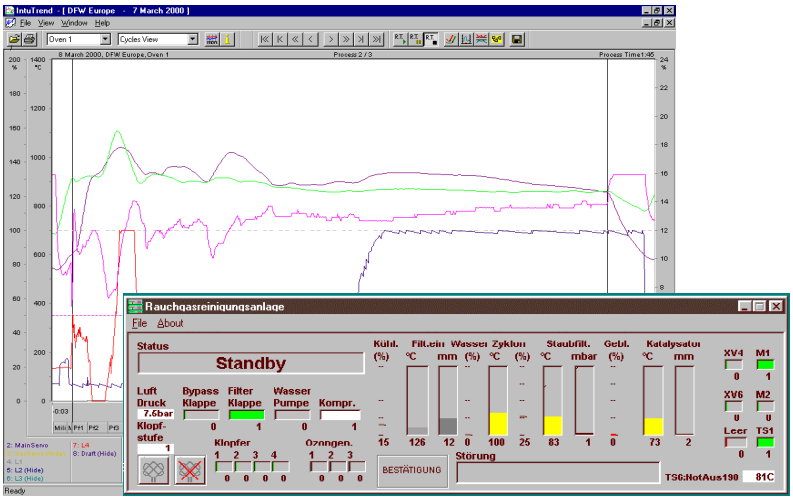
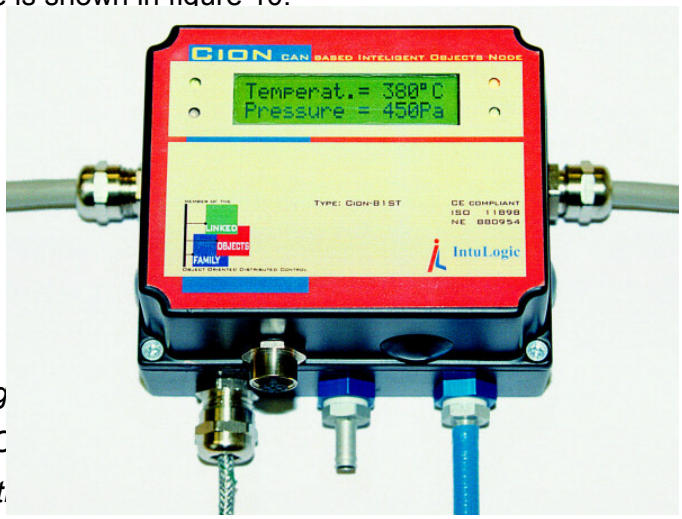


Figure 9, HMI and trending software use the same database as the application builder.

OODC Hardware

The number of I/O points per unit needs to be sufficient but not too large. In order to keep complexity low, it is advantageous to have about 2-4 objects per HW module. An example is shown in figure 10.



Fig

OC

wit

and high CPU performance,

connecting directly to sensors and actuators

Interface to sensors / actuators

Several industrial standards exist which can be used to connect OODC hardware to sensors & actuators, e.g. 4-20mA current loop, HART, RS-485, etc., which the OODC hardware modules should be able to use. To achieve maximum flexibility, these interfaces could be implemented as add-on modules, which are placed on a standard module with basic I/O and CAN interface. The control objects linked to these sensors and actuators should then be placed in the same module.

Currently, there are a large amount of sensors and actuators available which connect directly to a CAN network using one of the popular protocols CANopen, DeviceNet or SDS. It is possible to intermix these devices, as all the mentioned protocols use the same physical layer. It is possible for the linkpoints of the control objects to be implemented in such a way that the control objects communicate directly with these sensors / actuators. (e.g. A transmit PDO of a CANopen sensor could be a linkpoint for a PID controller)

In order to avoid that all objects need to understand several protocols, protocol interpretation should be done by the real time operating system. Thus, the protocol being used is transparent to the object.

Bridges & sub-networks

Due to limitations of the network used, it is often necessary to sub-dived the network into several sub-nets. When using location / area addressing, it is often convenient to use a single sub-net for a single area.

OODC Software

As with the hardware, OODC software should be tailored to relieve the designer of as much overhead as possible. Thus, OODC software should be centred around the control objects, and their communication through linkpoints.

Design of Control Objects

In the design of control objects, it is important to manage the level of complexity such that the objects correspond to the user's intuitive perception of the objects. It was found that the short-term memory of a person can handle 5 to 9 items (Miller, 1956). Thus, the user should be present with at most 9 main objects, which can have at most 9 different modes of operation and linkpoints. In this way, the user is able to use the control objects intuitively, without constantly needing to refer to manuals.

Also, objects should be designed to a common template. Every object should have an error output for reporting fault conditions, and an input with which it can be switched on and off. Common names for linkpoints should express common functionality. Thus, complexity is reduced further.

Because the object might need to be intrinsically safe, it must be able to monitor whether all input signals behave as expected. All objects which provide information must have intrinsically safe default values for these outputs.

The use of standard control objects instead of tailor-made software, greatly simplifies the development process. Each object can be designed, implemented and tested individually. As complexity for each object is low, development costs are low. The code for these objects need not be changed during the lifetime of the product, but can be re-used continually. Objects designed for specific tasks can be re-used in future projects.

OODC: Perspective for the Future

Ethernet and OODC

At this moment there is a lot of attention for the use of Ethernet in control systems. The large bandwidth and long distance of Ethernet combines well with the high reliability and deterministic performance of CAN. For the purpose of supervision, remote operation and data logging, Ethernet could replace CAN on the upper end if real-time constraints are not stringent. A likely mixed configuration is shown in figure 11. Often real time communication between several controller areas is not required. Then, the CAN bridges could be removed.

CAN and Ethernet combined can easily cover al needs from the ERP top of the CIM pyramid to the Actor/Sensor bottom of it.

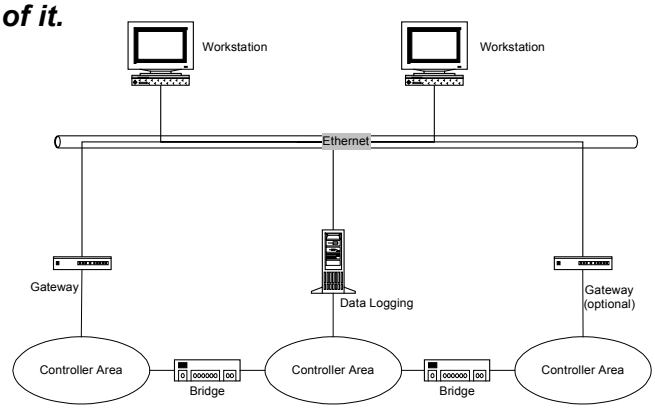


Figure 11: CAN combined with Ethernet

Increased Availability

Another trend is an increased demand for high availability systems. With OODC, it is relatively easy to increase availability trough built-in redundancy, to prevent a single point of failure from leading to system degradation.

One possibility is to implement a redundant CAN network but this is expensive. A more

cost effective method is using a 'smart-bridge' as shown in figure 12.

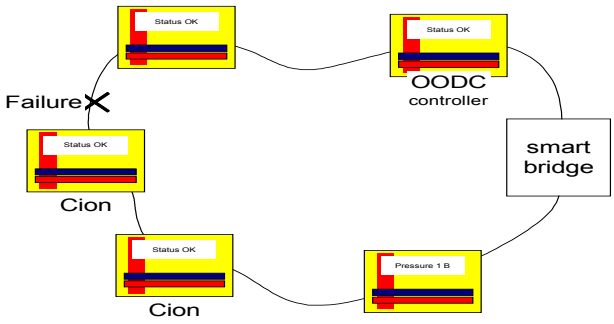


Figure 12: Redundancy through smart bridge

The bridge monitors both ends of the CAN ring. If a difference occurs, e.g. as a result of a cable failure, it will start repeating messages between the two remaining legs of the CAN network, maintaining system integrity. As network termination would no longer be optimal, it might be necessary to switch to a lower baud-rate.

Another measure is to have redundant sensors / actuators. When a fault condition is recognised, the actuator / sensor at fault is disabled, and the back-up takes over.

Another technique is to have multiple control strategies. When a sensor / actuator vital to one control strategy fails, an alternative strategy which does not depend on this device can be selected, either automatically or by an operator.

OODC: The Way To Go

It was demonstrated that through the use of OODC, both the costs for the supplier, as for the user are reduced compared with other techniques. The LOF implementation of OODC makes it straight-forward and intuitive to:

- Design and implement a control system
- Test a control system
- Maintain a control system
- Diagnose and trouble-shoot problems
- Re-use existing software and even complete control solutions

A protocol which makes full use of these advantages was presented. The protocol has the following characteristics:

- No network management necessary
- Hot-swap of modules possible
- Messages are easy to interpret

However, there are also some drawbacks:

- CAN messages are longer than the actual process data.
- Additional filtering of messages in software is necessary, causing CPU overhead.

These drawbacks can easily be overcome by modern micro-controllers.

This makes the LOF method useful for application areas where systems should be available, adaptable and scalable.

The most implementations have been in process-control applications. e.g. Incinerators (since 1994), dairy product industry (since 1995), exhaust gas cleaning installations (since 1996).

Other area's where the LOF method can make a difference are building automation and logistic systems.

References

Coad and Yourdon (1993): Coad, P. and Yourdon, E.: *Object-oriented Analysis, 2nd edition*, Prentice-Hall, 1993.

Carroll and Ellis (1995): Carroll, M.D. and Ellis, M.A.: *Designing and Coding Reusable C++,* Addison-Wesley, 1995.

www.intulogic\aida www.intulogic\LOF

Miller, G.A. (1956). The Magical Number Seven, Plus or Minus Two. *Psychological Review*, *63*, 81-97.

Intulogic bv.
Bisonspoor 1218
3605 KZ Maarssen
The Netherlands
Phone: +31-346-554411
Fax: +31-346-553711
E-mail: info@intulogic.nl
Web: www.intulogic.nl