Towards Plug and Play in Home and Building Automation Networks

Georg Neugschwandtner
Automation Systems Group, Inst. of Computer Aided Automation, TU Vienna
Treitlstrasse 1-3, A-1040 Vienna, Austria
gn@auto.tuwien.ac.at

Abstract

High configuration cost hampers the adoption of powerful home automation solutions. Automating the necessary configuration steps will alleviate this problem. The article defines the task of "plug and play" in the context of a distributed control system built from smart sensors and actuators and the network management services involved. It gives an overview of existing approaches and outlines the requirements for a comprehensive plug and play solution as well as work-in-progress toward this goal.

1. Introduction

Home and building automation are concerned with monitoring and automatic control of building services. Traditionally, these are primarily heating, ventilation and air conditioning (HVAC) systems and devices for lighting and shading. Safety and security systems are also being integrated. Usually, the term "building automation" is reserved for the context of functional buildings, while "home automation" refers to residential installations. This distinction reflects the different characteristics of the respective domains. In functional buildings, automated solutions are adopted for their potential to reduce operational cost through improved energy efficiency, optimised management (including remote access) and detailed usage reports. In the residential domain, all these apply as well. Yet, the focus is more on the increased comfort and peace-of-mind automation can offer. It also holds benefits for the elderly ("assisted living").

Although the number of devices participating is typically by orders of magnitude smaller than in a building, the complexity of a home automation scheme is not to be underestimated. Although systems such as access control or flextime are not present, domestic appliances and consumer electronic devices have to be integrated. Despite common aspects, there is significant variation in the individual homes, devices and owners' demands. Given that a system must have a certain complexity to deliver attractive functionality, considerable configuration effort is required for every individual installation. Thus, the share of configuration cost is disproportionately high in home automation. This hampers its adoption despite significant potential benefits. It therefore appears useful to look for ways to reduce the effort required for configuration. Such improvements will also be of benefit to building automation if they are done right and can be expected to be applicable to industrial automation settings as well.

1.1. Functional blocks and their connections

Home and building automation can be considered a special case of process automation, with the indoor environment (and possibly its surroundings) the process being controlled. Especially in process automation, it is usual to describe the function of automation systems using functional blocks. Every block describes a self-contained subtask in "black-box" fashion. Blocks can also be combined to form larger blocks to build the desired application. On the other hand, blocks can also be broken down into smaller blocks until they can be implemented on a device (or a resource of a device). Ideally, the resulting blocks correspond to functions for which an implementation already exists.

The interface of functional blocks is formed by their incoming and outgoing data. They receive input data, process it according to a certain rule, and send output data. Blocks can also encapsulate the functionality of field devices. Interaction with the controlled process often appears as a data source or sink within the block. Such blocks will output data generated by a hardware data source (a sensor) or direct input data towards a hardware data sink (an actuator), both contained within the block.

Multiple blocks can be implemented on a single device or be distributed over devices. In case the controller is freely programmable, the functional block paradigm can also be applied.[1] Intelligent field devices, however, have their functional blocks predefined (and pre-programmed) by the manufacturer due to resource constraints. They remain configurable within certain limits via parameters. Depending on the complexity of the application, it is very often possible to build a system entirely without custom programming, from predefined blocks only.

The concept of distributed functional blocks makes no implications about how it is implemented. The network system technologies geared towards home and building automation which are most strongly rooted in the concept

[1] Unlike in the industrial PLC world, vendor specific approaches still are popular besides IEC 61131 for this purpose in building automation.

1-4244-0681-1/06/$20.00 ©2006 IEEE

of intelligent field devices provide special support for exchanging data between functional blocks distributed over devices.

Both with KNX/EIB [1] and LonTalk [2], node applications define communication endpoints. When a change occurs on an outgoing communication endpoint (e.g., a sensor value has changed), the node application just passes the new value to the device firmware, which takes care of propagating the change to other nodes (i.e., functional blocks). Likewise, the node application is notified by the firmware and provided with the new value whenever other devices make such a transmission and new data are available for an incoming communication endpoint. Which devices (respectively functional blocks they implement) are linked is defined at installation time. The creation of these logical links between complementary endpoints is referred to as "binding." The node application is created entirely independent of these bindings. The communication endpoints are usually used to transfer values using value-based semantics, but this is not mandatory.[2]

To allow mixing and matching devices from different vendors within a system (and ensure maintainability), the blocks defined for open systems such as LonWorks or KNX are standardised. These standardised blocks can fulfil generic (such as an analogue input) or domain specific functions (e.g., a dimming actuator).

1.2. Work flow

The work flow for building such a system thus basically consists of selecting and connecting the proper blocks: 1. Define the desired top-level application (e.g., constant light; plus management level requirements such as central visualization and logging). 2. Break down the functionality: (a) determine the necessary sensors and actuators (dimmers, light sensors); (b) determine the necessary functional blocks (processing of the control loop). 3. Choose an implementation: (a) select the devices that will implement the field, automation and management functions (DDC with 1..10 V output, electronic ballast, light sensor with field bus interface, server application, ...); (b) determine where these devices will be installed and how they will be connected physically. 4. Set up the hardware: (a) obtain the devices; (b) install the medium (if applicable); (c) install the devices; (d) make the physical connections. 5. Set up the software: (a) assign identifiers (addresses); (b) assign functionality: define parameters, create and download programs, if required; (c) establish bindings. 6. Test.

Step 2 will probably leave some degree of freedom (variants), so that the precise allocation of functional blocks will not be decided until step 3. The application definition and functionality break-down can be done in a more formal way (in larger projects) or rather informally (such as in a do-it-yourself home situation). In the latter case, the software setup will also often be done in an ad-hoc fashion after installation (on-line).

[2] E.g., relative dimming in KNX/EIB uses command based semantics.

Assuming a less ad-hoc approach, either the step of assigning identifiers alone or both this step and the assignment of functionality and bindings can also be done before physical installation (off-line). Configuration and download before physical installation allows immediate testing of the specified application functions by the installer. Only performing identification beforehand allows to download the entire system configuration en bloc, without having to touch every device again, but leaving time to do the configuration while the construction process goes on. Yet, this complicates work organisation on larger projects, as every device has to go over the project engineer's desk and has to be installed in a particular location (meaning it cannot be replaced with an otherwise exactly identical part).

2. "Plug and Play"

The whole process as outlined above can be supported by automated methods. Unlike projects such as [3] or [4] which revolve around tools for the first two steps, the present paper focuses on the "software setup" step. Actually, this is what is generally associated with the phrase "plug and play" in automation networks: facilitate the configuration of the system in the commissioning phase. As an additional benefit, reconfiguration during the operation phase is made easier as well (cf. [5]). The ideal would be to take a hardware component "out of the box", add it to the system and make use of it without further configuration (cf. also [6]).

This will however not be possible entirely, if only for the fact that parameters have to be set which define the precise desired behaviour. Moreover, it will always be necessary to manually designate the device intended to perform a particular function when it is installed together with a number of functionally equivalent ones (e.g., load switches) – "plug, touch, and play" rather than "full plug and play".

2.1. Service elements

The classic "plug and play" procedure to support is as follows: a device is attached to the network (or host); it advertises its services (possibly with the help of a supervising instance); other devices and applications are notified of these events (or may discover the available services if they joined later). To support this procedure, a distributed protocol stack needs to provide certain service elements:

- Resource allocation: certain identifiers from the available name space are assigned to the newly attached device.
- Self-description: the device must be able to provide information about itself, at least a "magic number" that can be looked up in a data base. Of even greater importance than the availability of this service itself is how the information available through it is structured, as any automated configuration depends on it.
- Discovery: it must be possible to enumerate the devices that are present in the system and their capabilities, preferably according to certain filter criteria.

The attachment and detachment of devices needs to be detected and announced.

The discovery element can be split in device and service discovery. Although it is typically the services (capabilities) of a device that are of interest, device discovery may be a necessary means to perform service discovery, or be of interest for management functions.

In the PC hardware plug-and-play scenario, consumers usually choose the service to use ad-hoc (e.g., select which of multiple attached USB hard disks to store a file to). The same concept persists with classic plug-and-participate frameworks such as Jini and UPnP. The communication relationships between automation devices, however, are typically long-lasting (at least concerning process data communication): a certain switch will be used to control a particular light for a long period of time. This requires that binding, or rather some assistance with it, is considered as an additional required step towards plug-and-play.

As discussed above, binding involves designating which (otherwise identical) entities should communicate. In many situations, devices with complementary functions will be connected through more than one communication endpoint each (e.g., a push button may receive status feedback from its associated switch actuator). Since the installer typically needs to identify every connection separately (e.g., by pushing a button on both participating devices), related connections are treated en bloc by existing implementations to reduce the number of buttons (and presses) required.

2.2. Selected open commercial implementations

While its predecessor EIB did not offer any form of configuration assistance, the KNX specification describes a variety of "easy" modes with the required protocol support. Devices to be linked can for instance be identified by setting code wheels to identical positions (similar to X10) or by pushing buttons on both. Configuration can also be performed under guidance of hand-held programming devices. "Easy" binding revolves around channels, which combine the communication endpoints ("group objects") of several functional blocks, also defining which optional elements are to be used. Channels are identified using a single number. Some modes use a distributed network address assignment procedure, while others exclusively rely on message identifiers. All are limited to a single network segment. "Easy" devices can be integrated into a system managed by the standard KNX/EIB PC configuration tool, although no further "easy" configuration is possible then. Discovery functions are less developed.

Although LonTalk already provided basic device discovery and self-description, significant plug and play capabilities were only added recently when Echelon launched the "Interoperable Self Installation" (ISI) protocol extension [7]. Communication endpoints ("network variables") are combined into "assemblies" for binding. It is however possible that a communication partner binds to a subset only. The protocol is fully peer-to-peer, relying on periodic broadcasting of the allocated network resources to support devices disappearing and reappearing without special precautions. An optional domain address server can be added to allow larger ISI networks, which may also span two network segments.

DALI [8] is an example of how obtaining plug and play is radically simplified by an extremely narrow and strict application model. Since the protocol is only capable of addressing electronic ballasts (albeit with room for future extensions) through a single master controller, and device attachment requires the installer to go through an explicit configuration procedure involving the controller, the protocol is extremely compact.

Not surprisingly, BACnet [9], being geared towards large building automation projects, does not support ad-hoc configuration. It does, however, offer a kind of name service (Who-Is, Who-Has) which can map object names to binary object identifiers and network addresses to assist with system setup. In contrast, the ZigBee protocol [10] was designed for sensor-actuator networking. Despite being a lightweight wireless protocol, it supports detailed self-description and discovery. The communication endpoints for bindings between ZigBee devices are clusters of key-value pairs, whose identifiers can also be used in discovery queries.

3. Requirements

Despite the availability of commercial protocol implementations offering "plug and play" configuration, there still is ample room for improvement. The ultimate goal would be an object model and management services for near-automatic configuration fulfilling (among others) the following requirements:

- Supports "plug, touch, and play" as well as fully automatic binding (for single device instances such as washing machines).
- Supports configuration of distributed, peer-to-peer control networking schemes[3] and can perform the configuration process itself in peer-to-peer fashion.
- Allows to mix and switch back and forth between various configuration procedures (push buttons, code wheels, PC-based software or hand-held tools, ...).
- Adapts to the differences in work flow between "do-it-yourself" home installations and the construction of large buildings which is strongly based on the division of labour between different contractors.
- Allows to gradually add functionality (and complexity) only when required by the user, without the need for reconfiguring the functions already implemented.
- Also allows management functions to be handled this way (e.g., visualisation, remote services, logging).
- Is forward and backward compatible with respect to additional functionality offered by newer devices. Older devices should be able to participate even when new capabilities are being added over time.

[3] This is a key difference to master-slave approaches such as in [11].

- Should be able to cover functions from all domains, not being bound to a rigid application model. Describes devices and services (functions) using flexible semantics which can be extended as required.
- Handles non-configurable devices with only a transmitter (e.g., wireless switches).
- Allows devices to join and leave spontaneously. This is required for loose goods (e.g., vacuum cleaners).
- Does not assume an always-on Internet connection. Still, set-ups as in [12] should be possible.
- Last, certainly not least: Scales well when faced with a wide variety of functions and devices.[4]

4. Work in progress

The next step toward a system fulfilling the criteria above will be a thorough analysis of relevant systems (as those covered in Section 2.2) with respect to the mentioned aspects. It shall be investigated which design features or constraints are closely associated with the individual requirements. The semantic models supporting self description and binding, their flexibility and scalability will be of particular interest. In this context, the work of [13] and [14] shall also be taken into account.

This analysis shall establish criteria to judge a new design against. Based on it, a generic, resource efficient, scalable plug, touch, and play concept shall be designed and formally specified. In case particular requirements should be found to be in conflict, a coherent subset shall be chosen. The concept shall be validated by simulation and implementation on top of a proven control network technology. Still, its design shall be as network technology independent as possible.

A first informal sketch of the model that appears most promising at the moment shall be given in the following. Starting out from the requirement to be able to gradually add functionality, it appears an obvious choice to require values which directly relate to actual system inputs or outputs (also known as "hard (data) points") to always be network-visible, rather than encapsulated within a functional block. If this is combined with a strictly value based semantic for process data exchange, the process image is naturally available over the network at any time. Additional devices can be added without endangering consistency. For example, consider a push button switch which toggles a light. If it stores the assumed state of the light internally, it will get out of sync when the light is turned off via a newly added additional switch. This cannot occur when it always bases its action on the actual state.

Nevertheless, the propagation of value changes over the network shall remain event based. This is more efficient given the traffic pattern in home and building automation. For the same reason, the explicit separation of data and events as in IEC 61499 does not appear necessary. The required precision of timing and temporal coherence of events is low. Multicast support shall cater for the cases where the latter is of concern. To conserve bandwidth, a subscription scheme for event notifications shall be provided where consumers shall be able to define the criteria for generation. This could among others be a value or time difference, or a combination. This service could be loosely modelled on BACnet intrinsic reporting (though with multicast support).

In general, communication relationships shall be required to be one way (that is, communication endpoints either incoming or outgoing), and situations where multiple outgoing communication endpoints are bound to a single incoming endpoint ("fan-in") disallowed for simplicity and clarity. The same transparency as for hard points shall be required for system internal values without immediate relation to the real world ("soft points")–set points, but also schedules and similar. Thus, the actual task of control processing can be flexibly assigned to whatever device appears best suited.

[4] E. g., it would be unacceptable to require each device to carry an almanac describing the capabilities of all other devices on the market.

Acknowledgment The author would like to thank Peter Ferstl and Wolfgang Kastner for valuable input.

References

[1] *Home and Building Electronic Systems (HBES)*, European Std. series 50090.

[2] *Open Data Communication in Building Automation, Controls and Building Management - Control Network Protocol*, European Std. series 14908.

[3] C. Schwab, M. Tangermann, A. Luder, A. Kalogeras, and L. Ferrarini, "Mapping of IEC 61499 function blocks to automation protocols within the TORERO approach," in *Proc. INDIN 2004*, June 2004, pp. 149–154.

[4] M. Kirchhof, U. Norbisrath, C. Skrzypczyk, "Towards automatic deployment in eHome systems: Description language and tool support," in *Proc. CoopIS/DOA/ODBASE 2004*, LNCS, Springer, vol. 3290, pp. 460–476, 2004.

[5] T. Sauter, "Integration aspects in automation - a technology survey," in *Proc. ETFA'05*, 2005, vol. 2, pp. 255-263.

[6] J. P. Thomesse, "Fieldbuses and interoperability," *Control Engineering Practice*, vol. 7, no. 1, pp. 81–94, 1999.

[7] *ISI Protocol Specification*, Echelon doc.# 078-0300-01C, 2005. [Online.] Available: http://www.echelon.com/isi

[8] *A.C. supplied electronic ballasts for tubular fluorescent lamps – Performance requirements*, IEC Std. 60929, 2003.

[9] *Building Automation and Control Systems (BACS)–Part 5: Data Communication Protocol*, ISO Std. 16484-5, 2003.

[10] *ZigBee Specification 1.0*, ZigBee doc.# 053474r06, 2006. [Online]. Available: http://www.zigbee.org

[11] S. Pitzek and W. Elmenreich, "Plug-and-Play: Bridging the semantic gap between application and transducers," in *Proc. ETFA 2005*, Sept. 2005, vol. 1, pp. 799–806.

[12] M. Wollschlaeger and D. Hasler, "Uniform identification and maintenance functions for PROFIBUS devices as an example for Web-based information systems in automation," in *Proc. WFCS 2004*, Sept. 2004, pp. 385–388.

[13] G. Stein, "Composite Ports for an architecture-oriented assembling of components," in *Proc. WFCS 2004*, Sept. 2004, pp. 119–124.

[14] S. Deter, "Fieldbus device description using tag-based trees," in *Proc. AFRICON'02*, 2002, vol. 1, pp. 263-268.