FACILITATING HOME AUTOMATION THROUGH WIRELESS PROTOCOL INTEROPERABILITY

Sune Wolff / Peter Gorm Larsen / Kenneth Lausdahl / Augusto Ribeiro / Thomas Skjødeberg Toftegaard
{swo, pgl, kel, ari}@iha.dk / tst@cs.au.dk
Aarhus School of Engineering, Engineering College of Aarhus, Aarhus University

ABSTRACT

For many years, home automation has been visible to many, but accessible to only a few, because of inadequate integration of systems. A vast number of both standard and proprietary communication protocols are used, and systems are often difficult to install and configure so professional assistance is needed. This paper describes an open universal home automation framework enabling interoperability of multiple communication protocols. The framework can easily be expanded in order to support new protocols, and due to the open source nature of the framework, this expansion can even be implemented by external vendors. We show how measures have been taken to make the system very reliable, in order to avoid a system wide crash because of a faulty protocol implementation.

I INTRODUCTION

Today, the energy consumption in private households is 22% of the total consumption in Denmark and the costs of private energy consumption have almost doubled since 1990. Hence, there is great potential to make the consumers aware of their own consumption patterns so that they will actively help to change their energy use and habits.

There is already a lot of electronic equipment in private homes with features which can help to manage and reduce energy consumption and improve comfort in the home. Unfortunately, only a few people can find ways to apply it in everyday life. This means that there is a large untapped potential for energy savings and amendment of energy-using behaviour and habits in the home without compromising the comfort of the users.

In order to exploit this potential for energy savings a number of challenges must be solved. First of all many different notions of communication are used between home automation equipment, both in terms of standards as well as proprietary protocols. This limits the interoperability between devices significantly. Secondly, the configuration of the devices such that energy savings can be achieved is challenging for average users. As a consequence costs for installation and reconfiguration can be prohibitive. The high cost of the majority of home automation devices is also a limitation at the moment, but experience from the electronics industry shows that once the quantity of products go up, the prices for these products will decline substantially.

In this paper we present our proposed solution to solve the interoperability challenge as well as minimise the efforts necessary to configure the home automation system. User-driven innovation have been used, in order to achieve a user-friendly interface for the configuration of such a system.

II MINIMUM CONFIGURATION - HOME AUTOMATION

II.A Project Overview

In 2008 the Danish Enterprise and Construction Authority decided to fund a project called “Minimum Configuration — Home Automation” (MC-HA [12]) with a budget slightly below 1M euro. The objective of this project is to develop, through user-driven innovation, a unifying concept of how different electronic solutions can be configured in the home, in such a way that they will become applicable and relevant for users. The partners in the MC-HA project are:

- Aarhus School of Engineering [9]
- The Alexandra Institute [1]
- Seluxit [18]
- Develco Products [4]

The project is based on participatory design of a multidisciplinary cooperation in the user innovation [16]. Participatory design is a new approach for a number of the involved industrial partners and they will in the process, both learn to use the new methods, while describing the applicability of them for industry value. The project will generally work with the following two kinds of users:

- a reference user group consisting of families with or without children
- two families who will live in conventional houses in Denmark and Portugal.

Based on the user driven innovation, a home automation framework has been created, called EPIC (Extendible Protocol Independent unit Controller). Once matured, this framework will be open source for anyone to use. The first version of the framework has the following focus points:

**Protocol Interoperability:** The framework must be able to handle several communication protocols in a way that is transparent to the user.

**Solid Backbone:** Since it is the idea that several third party developers will make additions to the framework in the future, it is very important that the framework core is very stable. This will ensure that the entire system will not crash due to a faulty addition.
Testing Platform: The user driven innovation will create a lot of feedback on how the home automation system should work. It is important that the framework enables rapid prototype testing of several user interfaces, to support the frequent user feedback.

II.B System Architecture

The framework developed as a part of the MC-HA project has high focus on reliability and the ability to adapt different protocols into a single system which is easy to configure for the user. An overview presenting the main parts of the system is shown in Figure 1. The system is comprised of the following main components:

Figure 1: The EPIC system overview.

Core The core is the main software module which controls devices and holds abstract representations between devices in the system (Explained more thoroughly in Section II.C). Here the relations between devices are represented as Tasks and evaluated upon state change of a physical device. The core has two interfaces which are exposed a Subsystem Manager and a GUI Manager. The Subsystem Manager interfaces to the Sensor System which is responsible for a single protocol. The GUI Manager interfaces the Graphical User Interface (GUI). Both interfaces are exposed through Windows Communication Foundation (WCF) [10] which easily provides endpoints of types such as .Net remoting, ASP.NET Web services (ASMX) etc.

Sensor System The sensor system is a small sub component which connects to the core. It is responsible for managing a specific protocol such as ZigBee [11] or Z-Wave [6] through the Coordinator. Another important responsibility of the Sensor System, is the transformation between a specific device representation on the protocol level and the abstract device representation used in the Core. Each device is described as a device container with a collection of features where a detailed description is given specifying what the given feature provides and how to interact with it. The implementation made for ZigBee and Z-Wave is implemented with use of System.AddIn [19] for loading the interface description of the features which a specific device provides. This is done through the Device Creator. This makes the extension of new features easy and introduces a higher level of security and reliability.

GUI The GUI component is constructed as a Windows Presentation Foundation (WPF) [17] application connected to the core through the WCF Factory. To be easily expendable and reliable System.AddIn is used to construct a stable plug-in platform. The System.AddIn technology is able to handle a wide range of problems commonly encountered when building a GUI. This means that new views of the system easily can be created and both the security and reliability of the application can be preserved even when third party vendors provide new views. The base functionality includes presentation of devices in a house along with the device information. The highly important Task management, where different devices of the system can be related to each other at the abstract level, is also presented here. To aid the pairing of devices into Tasks a range of predefined device relations is used in order to automate the process.

II.C Protocol Interoperability

One of the main objectives of the EPIC framework is to ensure interoperability between several communication protocols in a way which is transparent to the user. This will provide the user with a broader choice when it comes to purchasing home automation devices, and hence make home automation more accessible to a wider part of the population. This, in turn, will help to lower the general strain on the environment, by reducing the carbon emission.

In order to support several coexisting protocols in the EPIC framework, the different subsystems and the system core needs to be decoupled. The system core is operating on an abstract representation of home automation devices with a range of services describing the unit and how to interact with it. This abstract representation is based on the highest common denominator of several analysed communication protocols. When the system needs to send a command to a device, an abstract message is produced based on the features supported by the device. This message is sent via the Subsystem Manager to the Coordinator of the concrete subsystem, where the message is translated into the protocol specific equivalent of the abstract message. Using the protocol API, the newly translated message is sent to the hardware interface and out on the wireless network. When a message is received on the hardware, the opposite translation is carried out, in order to create a message the system core can understand and operate on. This transformation must be done near the protocol since the transformation in both directions needs detailed protocol information.

Figure 2: Protocol interoperability example.
Figure 2 shows an example of the transparent protocol interopera-
tibility. A ZigBee motion detector device reports that is
has sensed movement. The message sent received by the phy-
tical ZigBee coordinator is a generic ZigBee command, which
is passed to the ZigBee Sensor System. Here, the message is
translated to an abstract representation and passed on to the
EPIC Core. The Core evaluates all the tasks configured into
the system, and finds a relation between the state change of the
abstract device and a feature of another abstract device. A new
abstract message is created by the Core, and sent to the Subsys-
tem Manager, which identifies the communication protocol of
the destination device to be Z-Wave. Hence, the Z-Wave Sen-
lor System receives the abstract message, and translates it to a
generic Z-Wave message. Finally, the message is sent via the
physical Z-Wave coordinator to the destination end device.

II.D Reliability

Reliability is an important issue in any system, especially for
a house automation system. While the failure of such system
might not cause property loss, it might lead to severe confi-
dence loss by the part of its users. Without confidence on the
system that control their homes, the users will most definitely
not be willing to use such a kind of system.

The system was thought from the beginning to be open and
easy extendable by plugins made by anyone. These plugins
could be either new GUI views or protocols and device types.
This assumption implies that in the future, the system will grow
and eventually buggy plugins will be deployed in the system.
Something must be made that ensures that these buggy plugins
will not crash the entire system rendering it useless.

The first step to achieve the desired level of reliability was
to separate the system into a series of smaller subsystems that
would be able to minimise the loss in case of one of the subsys-
tems fails. For this matter, it was chosen that a separation must
exist between the Core and the different Sensor Systems. The
system cannot ensure that a Sensor System will not fail, but in
that case, the Core will still be up and running and it is able to
communicate to the GUI that one of the Sensor Systems failed.

The system is not only open to the creation of new Sensor
Systems by third party developers. It is also possible to extend
the current Sensor Systems as it might be useful in the case of,
for example, a new type of device being created for a proto-
col that its Sensor System does not support. These extensions,
could also potentially contain bugs and crash a Sensor System.
Therefore System.AddIn is used to add these new plugins to
the system running in separate processes. In case of the de-
tection of a faulty plugin, this plugin can simple be unplugged
because System.AddIn uses the kernel to isolate these software
components from the rest of the system. As a consequence, the
Sensor System will still be up and running in the case of a fault.

The GUI is also made to be extendable and because of the
user driven innovation, it is highly desirable that it is easy to
do so. With System.AddIn, it is possible for plugins to provide
their own representation on the GUI and the defined interfaces
allow third party developers to make it easily. GUIs are typi-
cally unstable and again, it is not desirable that the whole GUI
shuts down because one of the plugins has crashed. So also in
this case the AddIns can help providing the reliability that the
system needs.

III CHALLENGES

While working with the different wireless technologies many
challenges have been encountered. Creating an abstract repre-
sentation of devices and messages that can cover a potentially
very wide range of wireless protocols has been challenging.
An analysis of several other protocols has been made in order
to determine if these can be covered by the abstract representa-
tion chosen. Since only two Sensor Systems supporting two
protocols have been implemented this still needs to be proven
in practise though. In addition, we have discovered minor er-
ors in the ZigBee stack implementation - these have now been
corrected based on the feedback of our group.

As mentioned in Section I only a limited variety of home
automation devices are available - and these at a considerable
price. Some of the devices still need some improvement before
they are stable enough to be used by end users. To name a
few of the anomalies, some devices cannot report their state
changes at a consistent rate - instead they send messages in
great bursts.

As described in Section II.D, reliability of the system has
been greatly increased by using technologies like WCF and
System.AddIn. These technologies have the unfortunate draw-
back of adding communication overhead as well as increasing
the complexity of the final system. Even with this communica-
tion overhead the system is not slowed down significantly - it
still executes at a speed comparable to other similar systems.

IV RELATED WORK

Previous research projects attempting to solve the applicabil-
ity of home automation have been based on a single standard
or proprietary wireless protocol simplifying the lower layer
interoperaibility challenge. An example of a commercial product
like this is the Electronic House Keeper [5] which uses Z-Wave.
For the generalised case of the intelligent home automation,
this is not a very realistic scenario as described earlier in Sec-
tion I.

A number of other projects have, like MC-HA, attempted to
create a solution that combines multiple wireless interfaces like
Zigbee, Z-Wave and Waveness to create seamless connectivity.
Many of them focus on a middleware or gateway functional-
ity attempting to create some independence of the lower layers.
One example is the presentation of wireless interoperability in
home automation given in [13]. Here the solution is based on a
central server operating as a gateway and a translator between
the end-devices and a home LAN. The approach is focusing on
software based service development by introducing a technol-
ogy abstract service language (DomoML), that makes it easier
to write generic services that communicate with end devices
via the gateway.

Another approach in [3] focusses on services on a residen-
tial gateway. For interoperability, software bridges on the gate-
way are introduced to handle heterogeneous networks and de-
VICES. Being centralized, the solution suffers from the single
point of failure problem. The internal language that the applications should use to interact with the bridges is not mentioned. The non trivial problem of integrating end devices into the framework is left as an implementation issue. The key question of which market actor will create driver(s) and market the bridge(s) is unanswered. However, this question is absolutely essential based on the large number of home automation protocols, and some system vendors are not interested in creating such a bridge due to a customer lock-in business strategy. In our MC-HA approach, we solve this issue, since we base the architecture on existing wireless standards.

The Hydra middleware [8] allows developers to incorporate heterogeneous physical devices into their applications by offering a web service interface to control different types of physical devices independent of its network technology and link layer technology. The Homeport project [7, 20] is a prototype that can unite the control of different wireless systems into a common user interface. In this project the focus has been on including user-based experience to simplify the GUI. The target has been on combining existing protocols by building a service layer - a gateway - as interface between the protocols. The project focuses solely on combining several wireless networks and controlling devices through its service oriented architecture. No thought has been given to how the system should support the user in configuration of the system through a graphical user interface. We think that it is crucial to have this kind of support in order to make home automation applicable to a wider range of users.

Finally it should be noted that there are radically different alternatives which attempt new types of solutions for wireless home automation. Here the strategy is based on the idea that any wired/wireless lower layer solution is possible handling the end-to-end connectivity using the Internet protocols. A new emerged set of RFCs (Request For Comments) based on [14] offer the ability to use IPv6 for connectivity all the way to the smallest wireless battery powered sensors. Not even the traditional wireless sensor network architecture of the ambient intelligence [2] with a connecting gateway between the mobile LAN and the wireless sensor network is necessary. With this IPv6 based solution the advantage is of course that it will be extremely easy to manage and configure the HA network. The direct end-to-end connection is possible using standard IP based components. The obvious drawback is that even though it is using an arbitrary link layer the physical equipment still needs to be updated since it has to use the communication protocols supporting this. This IP based solution will be a subject for further research, see [15].

V CONCLUDING REMARKS

Automating homes and office buildings is an effective way to achieve lower energy consumption as well as better indoor climate. But in order to ease the task of configuring such systems and broaden the choice of interoperable home automation devices, a solution is needed which can handle several communication protocols as well as aid the user in system configuration.

We have presented the EPIC framework as our proposed solution to these problems. Complete decoupling between the system core and the different protocols used has been achieved, and exemplified using the two wireless protocols ZigBee and Z-Wave. When a user wishes to form relations between devices no thought needs to be given regarding which communication protocol the individual devices makes use of. The lower level communication details are made completely transparent to the user, who can setup tasks using devices of any of the available protocols.

The EPIC framework is an open source project, since it is the plan that third party developers will aid in the expansion of the framework, in order to support more protocols in the future. As a result of this strategy, it is very important that the system is completely reliable, and absolutely essential that no "single point of failure" exists. This has been achieved by separating the system core from the GUI and Subsystems. This ensures that even though a protocol has minor errors, the rest of the system can still run without any problems. In addition, System.AddIn is used in order to dynamically load the different GUI views and device features in their own application domain. By separating these unreliable parts from the core of the system, a very reliable and robust system has been created.

The EPIC framework will, as part of the MC-HA project, be installed in houses in Portugal and Denmark. After a thorough testing phase, and once the framework is deemed stable it will be made available at SourceForge or similar open source share sites - so stay tuned.

Future work includes expanding the protocol support to include several other communication protocols. In addition, the GUI will be expanded in order to support the results from the user driven innovation. The system will be thoroughly tested in small workshops as well as be installed in houses both in Portugal and Denmark. It is of great importance to test how easy the system is to configure both concerning adding new devices to the heterogeneous network as well as setting up relations between these devices. Any additions to the framework can be carried out by anybody due to the open source nature of the EPIC framework.

Figure 3: The EPIC GUI.
Acknowledgments
We would like to thank Danish Enterprise and Construction Authority for financial support for this project and all our partners in the project for their active participation.

REFERENCES