UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA
DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Mobile KNX
Design, development and analysis of a mobile network bridge for Domotic systems

Student
Andrea Rampin

Supervisors
Prof. Matteo Bertocco
Prof. Martin Wonders

Second Marker
Prof. Becky Strachan

14th April, 2012

2011-2012

Abstract

In the last few years, the need to easily control the electrical devices has been manifested with more persistence. This is due to the increasing of energy consumption and the introduction of new concepts, such as green homes and building automation systems. To find a solution to this demand, several companies have focused on the development of infrastructure for remote management system of automation building applications. Based on well-defined protocols and systems for automation building structures, in the current market there are several applications that permit to clients to meet this request. However, it is still possible to improve their quality pushing on automation. Therefore, a central system for automatically processing and sharing the configuration data has been developed, leaving to mobile devices only the task to render it, in order to allow a client to control all the "smart" devices in an automation building system. This project has focused on developing an application that would provide a system of transaction between mobile and automation building world. In addition, a mobile application has been developed for several mobile operating systems and architectures. The work was completed with a testing phase, without the use of simulators, focused on ease of use and speed of access to its devices, which has shown the validity of this idea. The results obtained have given a new set of real data on which basing future developments and also a basis for a different approach to the problem.

Acknowledgments

I would like to spend some words to thank those who accompanied me along this experience. A special thanks is dedicated to my parents, Luciano and Antonella, and my sister Valentina, who were always close to me especially during the moments of difficulty. I want to thank Roberto and Gianluca, who were and will continue to be good colleagues and friends. Thanks also to all my friends for putting up with me during these past years, in particular my ex flatmates, who have made my student life more enjoyable. I would like to thank Javier and Larry for the good time I had with them in England and Matteo Conti who, as a good friend, helped me with some great advices. A special thanks goes to Prof. Matteo Bertocco, Prof. Roberto Turri, Prof. Martin Wonders and Prof. Rebecca Strachan, who gave me the possibility to realize this project, whilst completing my university career in another country.

Luck should not be waited, it must be created

Delia Agnolazza - 2006

Contents

Chapter 1

Introduction

In the last years, the continuous increase of energy consumption and research of new markets have permitted the emergence of new concepts such as green home and building automation systems. [22] [25] [78]

The word *green home* is used to describe a type of building designed to be environmentally friendly and sustainable, focused on efficient use of energy, water, and building materials. This concept implies buying more energy-efficient and smart appliances and using specific building materials, for example, components more efficient in keeping both cool and heated air inside the structure. *Building automation systems (BAS)* describes functionality provided by a computerized control system, installed into a building, using an intelligent network of electronic devices designed to monitor and control mechanical and electrical systems. [47] [76]

Each of these concepts offers great possibilities, but their combination provides a new efficient and completely managed building system controllable using a mobile device hundreds of kilometers away. Idea reached using innovative technologies, which permit to establish a connection between appliances in a building and an *application system core*, device that may communicate with a mobile device using well-established technologies as GPRS service [26]. In order to reach this aim, the first step is to find a system of interaction between an appliance or, for example, a simple bulb and a mobile device. To address this first problem a technology developed by *Konnex Association (KNX)* [55] (see appendix D) can be used. It provides an international communication open standard for home & building electronic systems, designed to be independent of any particular hardware platform and based on *European Installation Bus (EIB)* [24] (see appendix E). EIB was used as communication stack, enlarged with physical layers, configuration modes and application experience obtained by merging *BatiBUS* and *European Home System (EHS)*.

Thanks to new and innovative devices and the work done by companies such as Apple [4] and Google [33], it is now possible to use the last generation mobile devices as a complete and powerful computer. Therefore, it is possible to develop complex applications and take advantage of wi-fi connection [35], as well as, GPRS service to create connections between a mobile device, as iPhone [8] or BlackBerry [72], and another component such as a dedicated computer or a web server.

Merging building automation systems with mobile technologies and make the product really useful for the customers presents a problem related to the configuration. Ask to create and configure every connection between the customer's mobile devices and his appliances creates a friction to the adoption of this innovative infrastructure. Thus, the main idea of this project is to realize a *Bridge* between mobile and home automation world, developing a melting point interfaced with both systems. The point of interconnection will be used to share a set of configuration rules when a new device requires to access to the home system.

The aim of this project is to merge a building automation system with mobile technologies with a particular emphasis on automation, in order to create a new, useful and easy-to-use product for customers and to understand if a KNX infrastructure is suitable for use in residential installations. Features provided by *KNX infrastructure* and *mobile frameworks* [2] [44] [45] will be used to develop an application suitable for mobile devices. A dedicated station will be used to realize the interconnection point, which can be equipped with *GNU Linux* [77] as operating system. In fact, Linux provides an open, stable, flexible and highly configurable operating system that can run with few resources, in manner to create a melting point with an ultra-low energetic impact on the energy consumption of the building. Using an additional device gives the opportunity to expand this project to possible future developments. They might be the creation of a product, stylistically attractive and dynamically interfaced with a mobile device, that can reproduce music and manage the lights based on the type of music or read upon request e-mails, weather forecasts or financial news as well.

Chapter 2

Building Automation & Control System

The aim of the building automation market is to increase the user comfort and decrease the operating costs. To this reasons, building automation systems have optimized control schemes for heating, ventilation and air-conditioning systems (HVAC) [96], lighting and shading. Improvements in energy efficiency and automation permit to the management of the environmental conditions and reduction in the costs, providing access to all the building service systems via a centralized monitoring and control center. This allows to detect, to localize and to correct, at an early stage, abnormal or faulty conditions easily, especially when access to the site is offered through a remote connection.

Building automation (BA) concept is based on control of building services. It is historically correlated to the automatic control of HVAC systems, which have been subject to automation since the beginning of the 20^{th} century. Initially, controllers were based on pneumatics, which were then replaced by electric and analog-electronic circuits and finally microprocessors were included in the control loop. This concept was denominated as direct digital control [65]. This term is still extensively used for programmable logic controllers (PLCs) intended for building automation purposes.

In the 1970s [1], the oil price shocks, resulting in steep increases of energy prices, triggered interest in the energy saving potential of automated systems, whereas only comfort criteria had been considered before [9]. Therefore, the term *energy management system* appeared and highlighted automation functionality related to power-saving operation such as optimum start and stop control. Further, supervisory control and data acquisition systems for buildings, referred to central control and monitoring systems, were introduced; providing the possibility to handle each piece of equipment remotely over a whole building and allowing, in this manner, remote detection of abnormal conditions.

These innovations demanded to develop distributed control applications

[1]During the 1973 oil crisis, an embargo policy by the Organization of Petroleum Exporting Countries made world oil prices quadruple for a five-month period, then settle at a 10% increased level.

based on a continuous communication. Actual and actuating values need to be transferred between sensors, controllers and actuators. Based on this requirement, key criteria regarding quality-of-service (throughput, timeliness, dependability and security) should be evaluated. As for *throughput*, building automation applications generally do not create high traffic load at the field level due to the absence of high-speed control loops. Considerable amounts of traffic can accumulate when data have to be collected from a ubiquitous large system to a central location from all over a large system. However, data do not need to be entirely available in real-time. In fact, it is acceptable to update, with the central control system, the value (on/off) of a light every two minutes or for example a monitoring system might summarize the heating or cooling loads in a specific zone of the HVCA for future reviews by an operator.

The previous example has already denoted that *timeliness* has different concern at different system levels. Essentially, real-time data is only exchanged on the field where, moderate requirements are applied to all time constraints (periodicity, jitter, response time, freshness/promptness, time coherence [79]). However no special mechanisms (for example dynamic scheduling) are necessary for handling these slightly restrictive constraints including more demanding applications.

Dependability (robustness, reliability, availability, safety) is referred to the capacity of a network to detect transmission errors, to recover from errors or other equipment failures and to respect time constraints. Guaranteed performance (still with relaxed timing requirements) is only important in case of life-safety applications where the lose of control might have serious consequences. However, a certain amount of fault tolerance is desirable on the field. A single failing unit should not bring down the whole system. The network should also provide appropriate noise immunity, in particular robustness is desirable especially at the field level, where cables are laid in the immediate vicinity of the mains wiring, though the environment of building automation networks is not particularly noisy, predominantly in office buildings.

Communications *security* level is provided by the variety of proprietary. In the last years, security concerns are increasing rapidly, this is due to the fact that more sensitive systems like access control and intrusion alarm systems are being integrated. Furthermore office networks are used to transport automation system data and also remote access is a standard on present-day systems (as will be discussed in more detail below). Protection against denial-of-service or other types of attacks becomes more of an issue if buildings get more dependent on automation systems. In any case, the security often focuses on authentication, only trusted entity should be able to understand if a door is unlocked otherwise this information could be used for malicious purposes. [62]

2.1 Basic System Model

In a distributed system based on control application, a number of nodes (sensors, actuators and controllers) are connected over a network and they can communicate through a specific protocol. The data transported on the network contain values from specific devices, such as sensors or switches, which are sent to actuators in order to perform a defined action (for example, using a switch it is possible to send a command to a well-defined actuator which may turn on or off a light).

To the application developer, the network is composed by a set of elementary elements, called *data points* (or simply *points*). These data points are the logical representation of the underlying physical process, which permits to get access and thus to control network nodes. At the logical level, each data point represents a single data application that can correspond to an aspect of the real world, for example the value associated to a switch (on/off). Data points are connected through a directed graph, which might be entirely different from the physical connections between the nodes. However, the application is based on it and a set of processing rules, which describes the interactions between the nodes at the time when a point value is changed.

The main characteristic of a data point is its *present value*. The digital representation of this value is determined by the basic point type such as integer, floating point, boolean or enumeration types. Furthermore, every data point is associated with additional parameters which are important for the control application process. Data point type is an attribute that adds a semantic meaning to the currently value of a data point, it describes the engineering unit of the value. It is strongly correlated to the specific application, for example "Temperature", and it permits to obtain a system of compatibility between two or more data points. In addition, there is another attribute which indicates the minimal increment that can be represented; moreover it is related to other parameters such as resolution or minimum and maximum value. Therefore, it is possible to understand which is the observable value range, however sometimes the resolution is different from the real accuracy thus not all the real values can be represented. The last important attribute associated to a data point is the *location point*. It is often correlated to a string that is related to the network topology, for example a possible string might be "Building/Room/DataPoint" and a relative value could be "LabD300/Room1/WallSocket". Using this attribute it is possible to realize a logical structure with which to group data points in order to describe specific functions. [55] [62] [13] [58]

2.2 Network Architecture

A typically structure of a building automation network is composed by several independent networks and each of them is controlled using a dedicated network controller. All of these networks are spread around the building that usually is composed by more than one floor. Therefore in each floor there is a control network with relative sensors and actuators. In order to control and remote maintenance the entire building using a central station, it is necessary to connect together all the floors using a backbone channel.

As in most cases, also in this context, the whole system may be composed by a variety of networks and each of them may have different notion of distributed application (for example different concepts of locations or services). Therefore, trying to integrate together special-purpose systems may create discontinuities. Thus, in order to fix this problem, it is possible to use special devices that permit to create an interface layer between two different networks. These devices are named *gateway*, they are used to handle the interconnection [14] between networks, in fact they can store up a mapping database between network entities on either side. In this manner, every control application, actuator, sensor on each network can communicate using an own protocol and the communications between different networks are hided behind a gateway, which creates a new level of abstraction (figure 2.1).

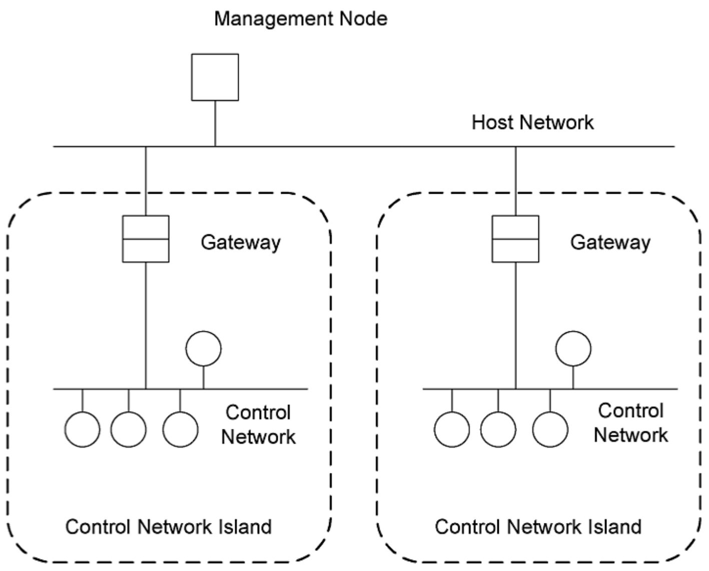


Figure 2.1: System with control network islands and a common backbone. [57]

In a building automation system, gateways typically operate on the abstraction level of data points. Therefore, a gateway should be able to mapping all the sophistications of protocols used to create the automation system; however this aim is extremely hard to achieve. While data points are really useful as point of abstraction for exchange of process-related data, some of the automation/control services, related to them, are impossible to implement using this abstraction due to the high technology-specific of those services. In fact the real problem is that the intermediate network does not support those services natively. In this particular case, the rightful approach is to transfer all the protocol layers of the control network over the intermediate (backbone) network, which basically works as communication system for the control network protocols. To realize this purpose, a method known as *tunneling* of a control network protocol over an intermediate network may be used. Using this system, the devices at the boundaries of the intermediate network, which build the ends of the tunnel between two or more control network segments, are called *tunneling routers*. [98]

The main advantage of the tunneling approach is the transparent connection between control nodes over IP networks. It is really important for two main purposes:

- separating control network segments, it is possible to make use of an high-performance backbone network, which may improve the basic characteristics of the entire communication system;
- using an intermediate network with a common and standard protocol, it is possible to locate a remote administrator with its native control tool in any within the "back communication network" (figure 2.2).

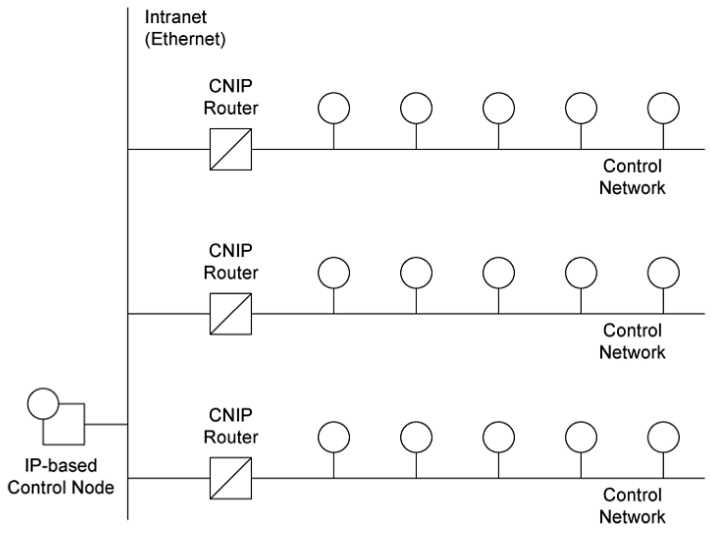


Figure 2.2: System with tunneling routers. [57]

Although gateways already provide an excellent point of flexibility, more often customers want to mix and match components from different vendors in order to avoid vendor lock-in and to give a long life expectancy to their building automation systems. To achieve this aim, all aspects of interfacing with a system should be *open*. For the purposes of this discussion, a system technology is considered open if its full specifications are made available to the general public and can be implemented at nondiscriminatory conditions. Such systems can be repaired, modified, and extended by everyone with the necessary basic qualifications without having to rely on the original manufacturer. Open does not mean "plug and play", it merely ensures that interoperability can be achieved without further involving equipment manufacturers. In addition, a system must always appear homogeneous to the end user, no matter how complex the interplay of its components may be. However, the benefits of open systems are not free in fact, the reduction in lifecycle cost, thanks to the flexibility gained, is generally considered to offset the initial additional hardware and engineering cost. [57]

2.3 Management Background Infrastructure

At the management level, system states and their representation formats are of particular interest. In order to integrate a building automation and control system with other applications, such as monitoring system or control system, a common and well-defined model of data representation and interconnection has to be developed. Therefore, in the last years lot of work has been done to achieve this aim and recently the idea of representation, of a control domain, by objects has been introduced. Variety of standards for distributed object-oriented systems were created in recent years, in fact they provide a suitable form to represent data in application layer based on gateways.

Object access protocols over IP networks are provided by Common Object Request Broker Architecture (CORBA), Java Remote Method Invocation (RMI) interface, Microsoft Distributed Component Object Model (DCOM) or Simple Object Access Protocol (SOAP) using XML notation [93] [89]. One of the first open standards for accessing process data, using an object-oriented approach which found broader acceptance by different vendors, was open process control (OPC). Services implemented on OPC are not limited to data access and exchange, event/alarm triggers and historical data access are provided as well. However, the main disadvantage of OPC is its relation to Windows-based systems, therefore arose a new trend of standardization focused on XML, which is characterized by simplicity, generality and usability data representation.

Systems such as BACnet, LonWorks or EIB/KNX should be used to cover the BA applications and get access to the devices in a building connected with a control network. These solutions have achieved considerable significance in the worldwide market (in case of BACnet and LonWorks) or in the European market (in the case of EIB/KNX) and they are often chosen by both customers and system integrators for complete system solutions.

2.3.1 BACnet

The Building Automation and Control Networking Protocol (BACnet) [13] has been developed in order to address the need to implement a building automation and control system in a different sizes and types of building. BACnet was develop to achieve this aim, it was realized ensuring a high level of interoperability in an system composed using devices made by multiple vendors.

The development of BACnet began in 1987 when an American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) could not find a suitable protocol that satisfy all the requests for a ideal standard communication protocol for building applications. The development was completed in 1995 when BACnet was published as an ANSI/ASHRAE standard. In 2003 BACnet was adopted as both a CEN and ISO standard [38], afterwards it was chosen as a national standard by the twenty eight different European countries.

While BACnet messages can be conveyed over any network, a small number of network types (Ethernet, ARCNET, Master-Slave/Token-Passing, LonTalk, and Point-to-Point) were standardized for BACnet's use in order to maximize the probability. In the 1999, Internet Protocol (IP) was recognized as particularly useful, thus *BACnet/IP* was finalized. Therefore, IP networks are natively supported by the existing BACnet network layer which allows BACnet devices to communicate using IP directly rather than via tunneling routers, as had been specified in the original standard.

The base element in the BACnet network topology is the *segment*. Segments are physical runs of cable, which can be coupled using repeaters and bridges to form a network. BACnet networks (of possibly different media types) are connected by routers to form a BACnet *internetwork*, but only one path may exist between any two devices on an internetwork. A BACnet network address consists of a 2-byte BACnet network number and a local address of up to 255 bytes. The local address is specific to the link layer medium, for example an IP address for BACnet/IP, and self-learning BACnet routers connect the individual networks route packets based on the network numbers.

BACnet represents the functionality of a building automation and control system as a set of *objects*. Each BACnet object is a collection of data elements, all of these are related to a particular function and they also correspond to the virtual representation of data points for a control application. BACnet defines 25 different object types. They include simple object types such as binary input and output, analog input and output, multi-state inputs and outputs as well as a number of more complex types related to scheduling, trending, alarming, and life safety capabilities. Any device in a building automation system may have zero, one or many objects of each object type but the "Device object" must be present in every device. It is used to represent and control all the characteristics/properties of a device. Each object may have several properties, but "object-identifier" (which represents in a unique way each object of a BACnet device), "object-name" and "object-type" have always to be present. However, this basic structure can be extended creating new objects or properties.

While objects provide an abstract representation of a building automation device, *BACnet services* provide a set of messages for accessing and manipulating this information using a client/server paradigm. Currently there are 40 application services which are grouped into five categories: Alarm and Event, File Access, Object Access, Remote Device Management, and Virtual Terminal. In particular, using the Object Access service is possible to use basic functions to read or manipulate any individual or group of property values, these functions are: ReadProperty, WriteProperty, ReadPropertyMultiple and WritePropertyMultiple.

Interoperability testing and certification programs have been pursued by both the BACnet Manufacturers Association (BMA) and BACnet Interest Groups (BIGs). The main focus of both groups has been to develop suitable software tools to test BACnet products and the procedures that can be used for specific kinds of devices. In an effort to go beyond simply verifying that a device has implemented all its BACnet capabilities correctly or just improve interoperability, a BACnet Testing Laboratories (BTL) working group was established to develop a set of guidelines for implementers to help them to avoid mere problems identified in the course of early testing or the interoperability work-shops, sponsored by BMA since 2000. The most recent addition to BACnet is the use of XML and Web services for the integration of BACS with other management-level enterprise systems (BACnet/WS). BACnet/WS is a neutral protocol and therefore it is equally applicable to non-BACnet systems (although a comprehensive mapping between BACnet and BACnet/WS services is included in the draft standard). [13] [57] [27]

2.3.2 LonWorks

The LonWorks system has been originally designed by Echelon Corp. as an event-triggered control network system. The system (described in [58]) is based on the LonTalk communication protocol, a dedicated controller (Neuron Chip) and a network management tool. In the 1999, LonTalk protocol was published as a formal standard, ANSI/EIA-709. It supports a variety of different communication media and different wiring topologies. Since it was designed as a generic control network, many protocol parameters can be decided by the designer, therefore a wide number of communication channel profiles were defined. They include a variety of twisted-pair (TP), powerline (PL), fiber optic (FO) channels and also radio frequency (RF) solutions. More recently, building backbones turn from TP-1250[2] to IP tunneling mechanisms thus (standardized in ANSI/EIA-852 [3] (also known as LonWorks/IP)) IP tunneling is supported as a standard channel by EIA-709.

The entire routable address space of the EIA-709 network is referred as a domain. Domains are identified by an ID, which the entire length can be chosen up to 48 bit corresponding to requirements (as short as possible, since it is included in every frame; as long as necessary to avoid logical interference, especially on open media). A domain can hold up to 255 subnets with a maximum of 127 nodes each. Hence, up to 32 385 nodes can be addressed within a single domain. A subnet usually correspond to a single physical channel, although it is both possible for multiple physical channels to be linked into a subnet by bridges or repeaters as well as for multiple subnets to coexist on the same physical segment. Routing is only performed between different subnets and, in particular, domain boundaries can be only crossed by gateway nodes (Figure 2.3).

Every domain can host up to 256 multicast groups and each of them can include nodes from any subnet. Broadcasts can be directed to a single subnet or the entire domain. Each node carries a world-wide unique 48-bit identification (*Node ID*) that it can be used for addressing individual nodes for management and configuration purposes, while regular unicast communication is handled through logical subnet and node addresses. The EIA-709 application layer allows generic application-specific messaging, but offers particular support for the propagation of network variables that are bound via 14-bit unique identifiers (*selectors*). The management and diagnostic services include querying the content type of the network variables, the node status, querying and updating the addressing information and network variable bindings, reading/writing memory, device identification and configuring routers.

Network nodes can be based on a chip from the Neuron series by Echelon or other embedded controllers like the LC3020 controller by Loytec. A typical network node architecture is shown in figure 2.4. The controller executes the

[2]1.25 Mb/s bus topology on TP

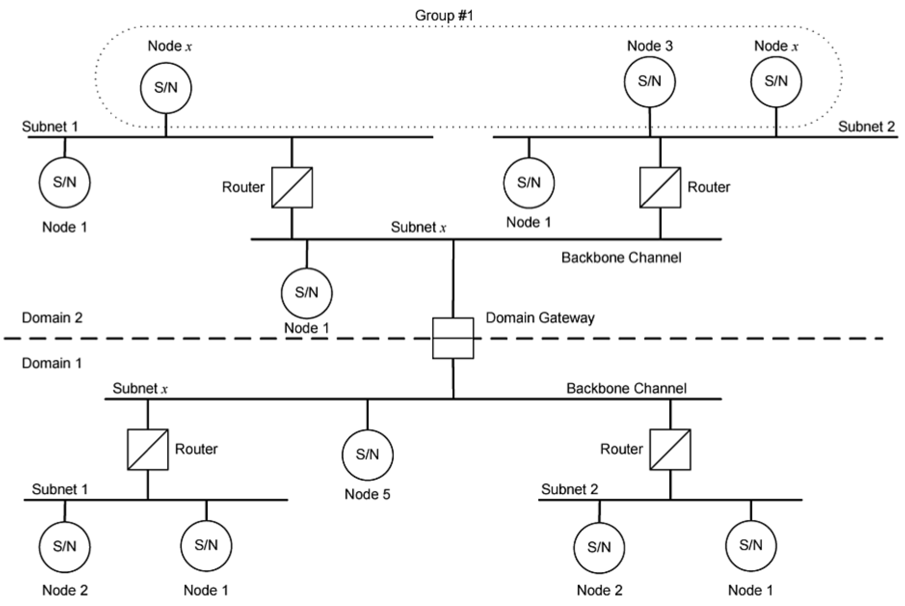


Figure 2.3: Logical segmentation in EIA-709. [57]

seven OSI protocol layers and the application program, which is interfaced with sensors and actuators connected through the I/O interface. A derivative of ANSI C called Neuron C is used to program the Neuron chips, whereas standard ANSI C can be used to program controllers such as the LC3020.

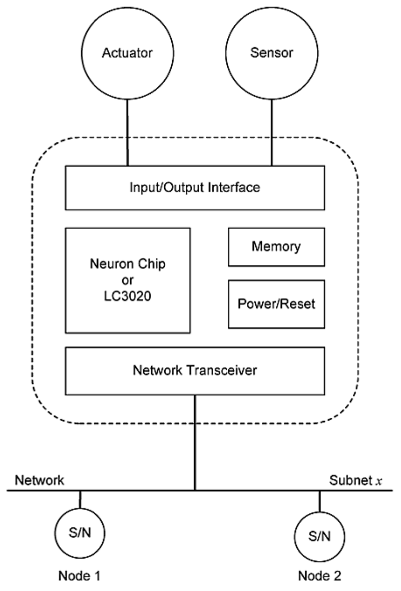


Figure 2.4: Typical EIA-709 node architecture. [57]

A variety of installation and management tools are available for EIA-709 networks. The predominant majority however is based on the *LonWorks Network Operating System (LNS)* management middleware by Echelon. For performance analysis and troubleshooting various protocol analyzers are available, including remote logging over IP networks. Modern network infrastructure components also have built-in statistics and diagnostics capabilities to allow remote monitoring and maintenance. [57] [27]

2.3.3 EIB/KNX

The EIB is a fieldbus[3] designed to increase electrical installations in homes and buildings by separating the transmission of control information from the traditional mains wiring. EIB is based on an open specification maintained until a short time ago by *EIB Association (EIBA)*. In 2002, EIB was merged with BatiBus and EHS. The new KNX standard seeks to combine their best aspects in order to create a single European home and building electronic system standard. Likewise, EIBA joined forces with the European Home Systems Association and Batibus Club International to form Konnex Association.

The main EIB/KNX medium is the twisted-pair cabling known as KNX TP1. TP1 allows free topology wiring with up to 1000 meters cable length per physical segment, up to four segments can be concatenated using bridges (called line repeaters), and the medium access on TP1 is controlled using CSMA.

EIB permits to use IP networking by the EIBnet/IP addresses tunneling over IP networks system. EIBnet/IP Tunneling is used to provide remote maintenance access to EIB/KNX installations in an easy-to-use manner and therefore restricted to point-to-point communication. EIBnet/IP Routing, instead, allows the use of an IP backbone to connect multiple EIB/KNX sub-installations. Routers using this protocol communicate using UDP multicast.

The basic building block of an EIB network is the *line*, which holds up to 254 elements. Following a three-level tree structure, sublines are connected by main lines via routers to form a zone and finally zones can in turn be coupled by a backbone line, as illustrated in figure 2.5. Network partitions on open media are typically linked into the topology as a separate line or zone through IP tunneling. Overall, the network can contain roughly 60000 devices at maximum.

To every node in an EIB/KNX network is assigned an *individual address*, which corresponds to its position within the topological structure of the network (zone/line/device). This address is exclusively used for unicast communication. Multicast addressing instead is implemented in the data link layer, therefore an additional non-unique MAC address (*group addresses*) is assigned to every node. Group address messages are routed through the whole network, therefore routers are preprogrammed with the necessary tables and it is also important to note that broadcast messages always span the entire network.

EIB/KNX uses a shared variable model to express the functionality of individual nodes and combine them into a working system. Network-visible variables of a node application are referred to group objects which can be readable, writable or both. Each group of communication objects is assigned to a unique group address that is used to handle all network traffic pertaining to the shared value in a peer-to-peer manner. Group membership is defined individually for each group object of a node and it can belong to multiple groups.

[3]Fieldbus is an industrial network system for real-time distributed control. It is a way to connect instruments in a manufacturing plant. [95]

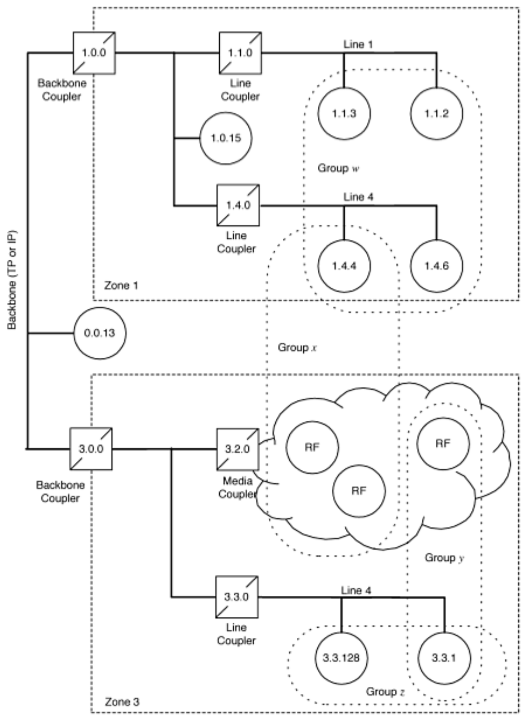


Figure 2.5: EIB/KNX network topology. [57]

For commissioning, diagnosis and maintenance of EIB/KNX installations, a single PC-based software tool called *Engineering Tool Software (ETS)* (see section E) is provided to handle every certified EIB/KNX product maintained by EIBA. ETS permits to bound group objects individually, on this, EIB Interworking Standard defines a standardized bit-level representation for the various types of shared variables. [55] [57] [27]

2.3.4 Analysis

Based the analysis on a restricted field of interest as this project (residential environments), it is necessary to figure out which is the best system that can fit better the requests for this specific case. According to this, a flexible technology able to manage a "limited" number of devices (sensors, lights, heaters..) with low investments turns out to be the ideal solution. In order to reduce costs of design, realization and maintenance of a building control system, the chosen should provide an affordable low-cost configuration tool or, at the least, an easy-to-use in order to reduce investment in training. There is also to consider the plurality of tools, different tools for vary devices instead of only one not always represent a benefit. Although a plurality of tools doesn't create an undesirable lock-in effect, different training courses are often required and sometimes problems relative to inhomogeneity of the work environment arise.

When deciding which system to use, an important aspect has to be avoided as much as possible in order to complete a network lock-in. To address this problem it is necessary to choose a solution that permits to select products realized by different companies but based on a similar protocol to exchange data.

An analysis on the complexity required to the system components should also be made; if the problem is restricted to houses and small offices, it is not necessary to seek technologies that permit to adopt devices equipped with complex features such as functions of analysis, logging or security at the expense of easiness of use/installation and costs.

In summary, the key drivers during the choice should be costs, easiness and completeness of basic functions. Based on this, none of these protocols is the secret to success in a building. The secret to success lies in choosing the right protocol for the right part of a building. For example, at the level of building, an infrastructure based on BACnet should be a good choice but at the room level KNX or LonWorks would give a flexible and efficient solution. In addition, talking about development tools, easiness and integration ETS gives to KNX/EIB an advantage over the others.

To conclude this short analysis, KNX/EIB was a good choice in fact it allowed me to focus on the real aim of this project without spending too much time on an initial configuration and, however, it provides all the functionality required to manage and control a standard house. [21] [27]

Chapter 3

Project Developing Tools

3.1 Calimero

Calimero is a Java library for KNX access. It was presented to the public at the KNX Scientific Conference 2005 as a part of the KNXLive! project [50]. It offers easy access to KNX system, middleware and development tools based on a Linux Live-System (Knoppix [52]).

Calimero NG API, or easily "Calimero", supports connection and management methods for all the common KNX tasks.

- Calimero library requires Java 2 platform [67].
- Modularity was improved further by grouping together features with related functionality. Therefore, interdependencies were eliminated wherever possible.
- Extensibility of functionality continues to be an important aspect of this project, thus the architecture is designed to allow to add new features or alternative services' implementations.
- Besides KNXnet/IP Tunneling, routing is provided as an additional connectivity option (Network Address Translation (NAT) is supported as specified in the KNXnet/IP documentation).
- A generic KNX network access link abstraction was introduced to accommodate further types of network access units and protocols.
- Calimero provides an API for buffering incoming KNX network messages.
- Logging is supported throughout Calimero using a generic logging API for various kinds of events such as errors, status or progress information.

3.1.1 Architecture

In order to access to a KNX network, it is necessary a bridge to fill the gap between an application and the medium. Thanks to the KNX standard, the bridge represents a layer of abstraction, it can work with an IP network interface or a serial cable connection as a bus coupler, but the application cannot perceive the difference. Therefore, the entire design of Calimero follows an architecture known as the *waist-line architecture*. The services to the high level can rely on a well defined interface abstraction that encapsulates and hides the underlying infrastructure. Figure 3.1 shows how, in this architecture, all the functionalities are divided into three layers.

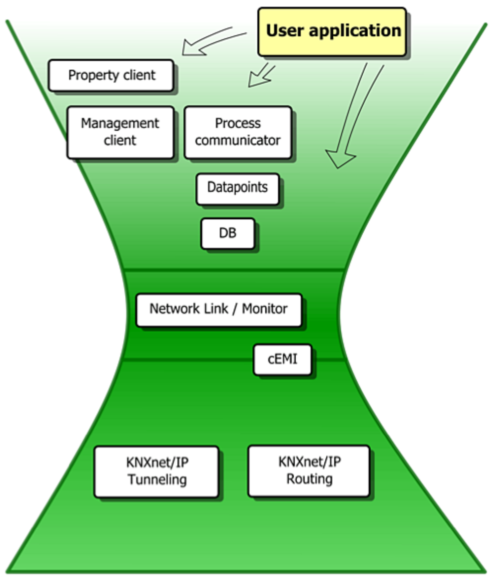


Figure 3.1: Waist-line architecture and non-strict layering in Calimero. [59]

Implementations of the basic services (the range of network access protocols) are located at the bottom. In the middle, the network link abstraction offers an homogeneous and standard interface for communication with KNX networks, hiding the chosen basic service. This interconnection, provided by a slim and stable API, is considered the “waist” of the architecture. All the user applications are located at the top level and they communicate with the KNX medium using an its abstract representation.

3.1.2 KNX network access via KNXnet/IP

KNXnet/IP is a protocol specification for working with KNX networks over an IP connection and it is implemented into the core feature of Calimero. However, client implementations of all KNXnet/IP service protocols, whose specifications are available in the KNX Handbook, are available in the KNXnet/IP bundle. This is composed by KNXnet/IP Core (discovery & description), Tunneling, Routing and Device Management (KNXnet/IP Device Management is implemented on UDP protocol). The discoverer system offers KNXnet/IP server discovery and self description; it can be used in nonblocking or blocking mode, optionally with a timeout limit for remote discovery endpoints to respond. KNX messages are handled using appropriate cEMI message types [54].

3.1.3 Process communication

Process communication refers to the exchange of process-related data, such as sensor values or actuator commands. In KNX, process communication implements the group communication, which performs the follow steps in order to send a message on the KNX network:

- The data value (or command) has to be transferred to its KNX representation and this is done by the DPT translator bundle.
- The appropriate application and transport layer control information have to be added to create a Transport layer Protocol Data Unit (TPDU).
- Control information for the lower layers has to be obtained, at least this information contains a destination group address.
- All this information must be supplied to a KNXNetworkLink, which assembles a frame and transmits it to the network.

The same procedure has to be performed, but in the other way, for receiving a message. Each of these tasks can be performed on different levels of abstraction via specific APIs available in Calimero.

3.2 Tomcat

"*Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies.*" [12]

Tomcat is developed thanks to an open and collaborative community, which works to realize an open source web server released under the Apache License [10], managed by the Apache software foundation[1].

The aim of a web server is to control, to manage and to reply requests of clients (typically a web browsers) with data processing, subjected to internal policies, in order to create answers provided by web pages. The final response due to a request is provided using special document usually created using a formatting language, known ad HTML. However, in order to realize a final answer, thus web page, a web server can use several programming and scripting languages to elaborate and extract data from different sources of information. In particular, Tomcat provides two main technologies to generate answers. Using this web server, it is possible to merge the stable and powerful Java Servlet with dynamic and easy-to-use JavaServer Pages (JSPs). These two technologies were added to the standard common gateway interfaces (CGIs).

[1]The Apache Software Foundation provides organizational, legal, and financial support for a broad range of open source software projects. The Foundation provides an established framework for intellectual property and financial contributions that simultaneously limits contributors potential legal exposure. Through a collaborative and meritocratic development process, Apache projects deliver enterprise-grade, freely available software products that attract large communities of users. The pragmatic Apache License makes it easy for all users, commercial and individual, to deploy Apache products. [11]

In addition, Tomcat can be used in conjunction with other web servers such as Netscape Enterprise Server, Microsoft Internet Information Server (IIS), Microsoft Personal Web Server etc., therefore Tomcat is dedicated to manipulation of dynamic part of a response instead the others might be used to generate static part. Tomcat, due to the fact that it is based on java, requires a Java Runtime Enterprise Environment.

3.2.1 Java Server Page

Java Server Pages (JSPs) are a server-side technology, which extends Java Servlet [41] (see section 3.2.2) system developed by Sun [69].

JavaServer Pages are realized thanks to the union between scriptlet[2] elements and markup tags, such as HTML or XML. In this manner, it is possible to separate the logic and the static part of a response. Therefore, it is possible to improve the quality of final documents, giving the right job to the correct personnel according to their knowledge. Thanks to the portability of Tomcat, JSPs are not strictly correlated to a specific platform. However, there are other systems that can, in part, compare with this technology, in fact JavaServer Pages were originally created as an alternative to Microsoft's ASPs (Active Server Pages) [63].

3.2.2 Servlet

A servlet is represented by a Java class, which is used to extend the features provided by a server, that host the application, thanks to a request-response programming model. As Java class, a servlet might process any type of requests, but they are commonly used in a web context thus they often extend the capabilities of a web server. Therefore, Java Servlet technology provides a collection of functions created especially for the HTTP environment.

The `javax.servlet` and `javax.servlet.http` packages provide interfaces and classes for writing servlets. Therefore, all servlets, that want access to specific web functions must implement these interfaces, which define life-cycle methods. In particular, using the `HttpServlet` class is possible to use methods, such as `doGet` and `doPost`, for handling HTTP-specific services.

In order to use a servlet, it has to be "installed" into a server (this step is called *Deploy*), in fact the container controls the entire life-cycle of a servlet. When a server receives a request and it is mapped to a servlet, it performs the following steps. [41]

- If an instance of the servlet does not exist, the container:
  - loads the servlet class;
  - creates an instance of the servlet class;

[2]scriptlet is a piece of Java-code embedded in the HTML-like JSP code. The scriptlet is everything inside the `<% %>` tags

  - initializes the servlet instance by calling the `init` method in it defined.

- Invokes the `service` method.

When a servlet must be removed, container calls a special method, `destroy`, which completes the life-cycle of servlet.

"Creates an instance of the servlet class", as written in the previous list, requires to create a new instance of every variables, objects, etc. and this happens for each request. Therefore, every instance has a proper "life", however sometimes it is necessary to create a point of interaction between two or more instances of servlet. Tomcat, for this purpose, provides several systems to share an object or variable [42]. Web context (`javax.servlet.ServletContext`), Session (`javax.servlet.http.HttpSession`), Request (`javax.servlet.Servlet Request`), Page (`javax.servlet.jsp.PageContext`. In particular the second solution permits an easily access to share objects using `set` and `get` methods. All the share objects are maintained into the execution environment. Thus, several instances of the same servlet can access, for example, to the same connection object. In this manner, if there is a server that can open only one connection at a time, the first instance of the servlet opens the connection and share it with all the others.

3.3 PhoneGap

PhoneGap [2] is an application framework that allows developers to use HTML [85], JavaScript [61] and CSS [83] to create apps that are presented as normal applications on the phone, but actually they run within using the *WebView* component [32] [7]. That means the apps have their own icons and operate similarly to native applications without a browser frame around them. They are distributed via the application stores, such as the Android Market and the Apple App Store, and they have access to a set of native functions to further make them work like native apps. Developers use PhoneGap because it allows them to have a common codebase for all their application code. It doesn't force developers to rewrite the sofware every time they move it from platform to platform.

3.4 jQuery Mobile

jQuery Mobile is a touch-friendly web UI development framework that lets to develop mobile web applications that work across smartphones and tablets. The jQuery Mobile framework builds upon jQuery core and provides a number of facilities, including HTML and XML Document Object Model (DOM) traversing and manipulation, handling events, performing server communication using Ajax, as well as animation and image effects for web pages. The mobile framework is separated from jQuery Core, which is an additional download of around 12KB, and in total requires around 25KB when minified/gzipped. As with the rest of the jQuery framework, jQuery Mobile is a free, dual-licensed (MIT [70] and GPL [31]) library. Basic features of jQuery Mobile include [46]:

General simplicity: the framework is simple to use. You can develop pages mainly using markup driven with minimal or no JavaScript.

Progressive enhancement and graceful degradation: while jQuery Mobile leverages the latest HTML5, CSS3, and JavaScript, not all mobile devices provide such support. jQuery Mobile philosophy is to support both high-end and less capable devices, such as those without JavaScript support, and still provide the best possible experience.

Accessibility: jQuery Mobile is designed with accessibility in mind. It has support for Accessible Rich Internet Applications (WAI-ARIA [88]) to help make web pages accessible for visitors with disabilities using assistive technologies.

Small size: the overall size of the jQuery Mobile framework is relatively small at 12KB for the JavaScript library, 6KB for the CSS, plus some icons.

Theming: the framework also provides a theme system that allows you to provide your own application styling.

3.4.1 Ajax with jQuery

Ajax is a technology based on other several technologies, each of them provides a feature set that turns Ajax into a revolutionary technology. Ajax incorporates:

- standards-based presentation using XHTML/HTML and CSS;
- dynamic display and interaction using the *Document Object Model (DOM)* [84];
- data interchange and manipulation using XML and XSLT [91];
- asynchronous data retrieval using *XMLHttpRequest*[3] [90];
- and JavaScript binding everything together.

The classic web application model works as: user actions in the interface trigger an HTTP request back to a web server. The server does some processing (retrieving data, crunching numbers, talking to various legacy systems) and

[3]XMLHttpRequest object, first implemented by Microsoft as an ActiveX object but now also available as a native object within both Mozilla and Apple's Safari browser, enables JavaScript to make HTTP requests to a remote server without the need to reload the page. In essence, HTTP requests can be made and responses received, completely in the background and without the user experiencing any visual interruptions.

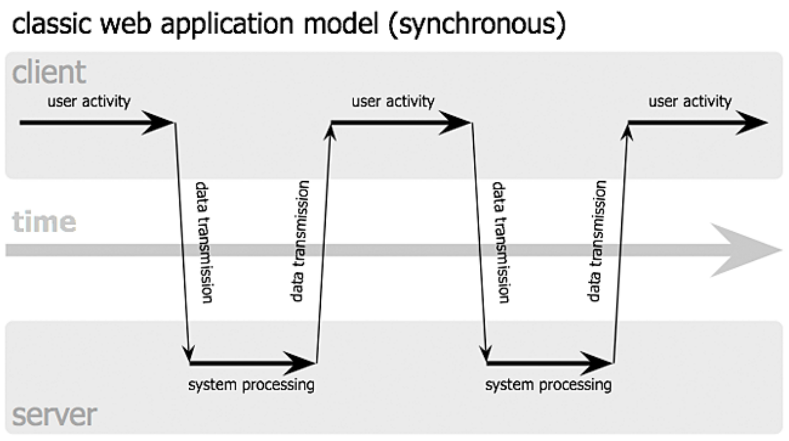


Figure 3.2: The synchronous interaction pattern of a traditional web application. [1]

then returns an HTML page to the client. It is a model adapted from the web's original use as a hypertext medium, but it does not represent the rightful way to implement web pages with the purpose of realize web applications. A web application should provide a good user experience and gives the idea, to the user, of using a real application although it runs by a web browser. Therefore, to create dynamism is possible to use Ajax. Using this technology, it is possible to interact with a server without reloading a web page. To understand its potentiality is enough thinking to a simple purchasing process in an e-commerce web site. The traditional model requires to reload a page whenever a user decides to add an item to his basket in order to run some code. Using Ajax, instead, is enough to load a web page only once and send a XMLHttpRequest in background when the user buys a new item (figure 3.2 and 3.3). In this manner, a user obtains a better user experience and also the developer/owner of the web site can save bandwidth (the KBs required by a XMLHttpRequest are less than an entire web page request) and consequently money.

An Ajax application permits to avoid the basic method of interaction on the web (send request, wait until an answer is received, send another request, ...) thanks to an intermediary application layer (Ajax engine) positioned between client and server. Therefore, when a session is opened, browser doesn't load a web page but call the Ajax engine, which is written in JavaScript, that create an independent communication with a server. This application is responsible for both rendering the user interface and communicating with the server, therefore it permits to create an asynchronously communication between client and server (figure 3.4). In this manner, client may continue to manipulate data contained into a web page without waiting each response due to changes made by him. Thus, it is possible to reach a high interaction client/server and user experience.

Using this system, every action made by a user is manipulated as a JavaScript interaction with Ajax engine, instead of an HTTP request. Therefore, the engine can decide if send a request to a server; this happens only if it needs any

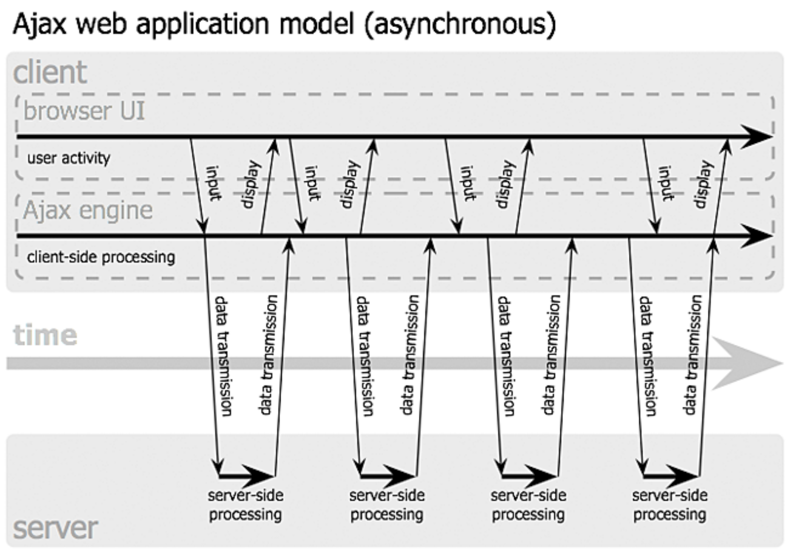


Figure 3.3: Asynchronous pattern of an Ajax application. [1]

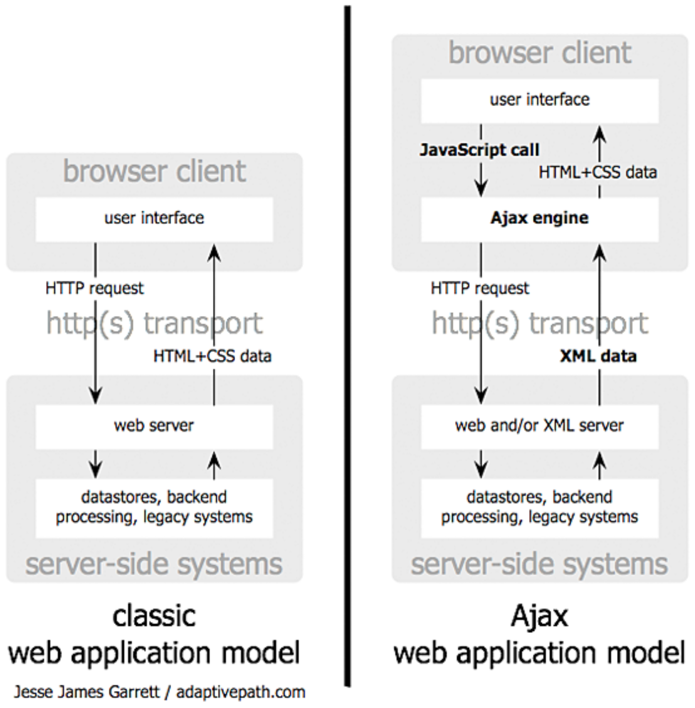


Figure 3.4: The traditional model for web applications compared to the Ajax model. [1]

data, otherwise it resolves the request by itself. In the first case, it makes an asynchronously request to the web server, usually using XML, without stalling a user's interaction with the application.

jQuery.ajax()

There are several frameworks that permit to use Ajax during the development of a web application, one of thsm is jQuery. This framework combined with the mobile version, jQuery Mobile, provide an excellent solution to implement dynamic and mobile applications. jQuery provides a set of functions created exclusively to handle, easily, the XMLHttpRequest requests and asynchronous events [43].

jQuery API offers the Ajax functionality for different types of data, in this manner it is possible to determine the information model that a developer can expecting back from a server. During the implementation developer should define this settings, however if it is not set jQuery tries to infer it based on the MIME type [86] of the response. The available types are: xml, html, script, json, jsonp (loads in a JSON block using JSONP. Adds an extra `?callback=?` to the end of your URL to specify the callback), text. The two main type, that in the last years have had an important role in the network, are JSON and JSONP. JSON is a lightweight data format (compared to XML) for the exchange of information between browser and server. The reason why developers are interested in JSON derives from the fact that, using this technology, it is possible to represent JavaScript objects using simple strings structured as:

- collection of name/value pairs (*object*) and
- an ordered list (*array*) of these elements.

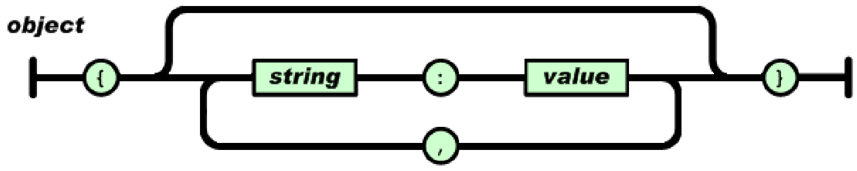


Figure 3.5: JSON object structure. [48]

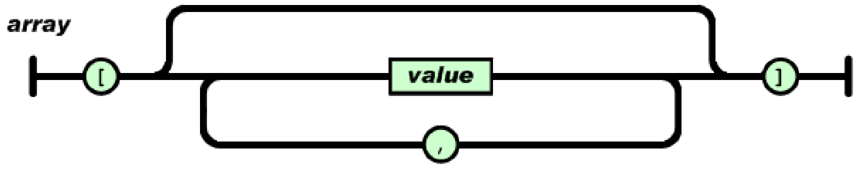


Figure 3.6: JSON array structure. [45]

To proof the simplicity with which it is possible to represent information, the next source code, 3.1, shows how describes a phone contact using the JSON structure. This helps to understand why this technology had success, data are representable in a natural structure.

Listing 3.1: Phone contact representation using JSON

```
{
     "firstName": "Andrea",
     "lastName" : "Rampin",
     "compnayAddress"  :
     {
         "streetAddress": "Ellison Place",
         "city"         : "Newcastle upon Tyne",
         "state"        : "UK",
         "postalCode"   : "NE1 8ST"
     },
     "phoneNumber":
     [
         {
           "type"  : "tel",
           "number": "+44 (0)191 232 6002"
         },
         {
           "type"  : "fax",
           "number": "+44 (0)191 227 3903"
         }
     ]
 }
```

In order to access to this information, if it is located in another domain, using for example a request sent from a mobile phone by PhoneGap, it is necessary to protect and permit the cross-domain transmission by a wrapper. Otherwise, a security error response will be received since the same-origin policy [87] prevents a data loaded from one domain from manipulating properties from another domain. In fact, the domain of the requested URL should be the same as the domain of the current Web page and this basically means that the system (usually a browser) isolates content from different origins to guard them against manipulation.

To avoid this restrictions and continuing to use only the frameworks introduced above, without adding other libraries, a rightful solution is to use JSONP. This system permits to insert all the information inside a container, which usually represents a function call, and load it dynamically as a script element. Therefore, it is important to indicate, into the request, that a JSON response, incapsulate into a JSONP structure.

This chapter has introduced, with a brief description, the technologies used for this project. In the next chapter will be presented the system at the conceptual level.

Chapter 4

System Concept and Implementation

The aim of this project is to merge together a building automation system and mobile technologies with a particular emphasis on automation, in order to create a new, useful and easy-to-use product. Features provided by *KNX infrastructure* and *mobile frameworks* [2] [44] [45] will be used to develop an application suitable for server mobile devices. A dedicated station will be used to realize the interconnection point between building automation and mobile world.

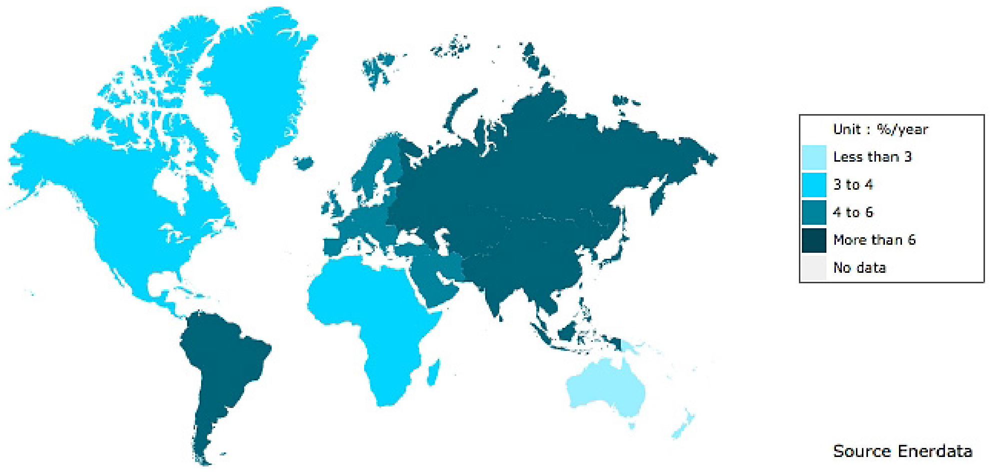


Figure 4.1: Increase energy consumption: 2009-2010 [23]

The idea behind this project is to realize the basis for a new system able to connect together: simplicity of use of smartphones, potential of an automation building network, such as KNX, and IT interconnectivity systems. However, this infrastructure is already present in the market; there are systems that allow a mobile device to connect with an automation building system. They are also trying to automate the entire process of development and management, but their level of automation and dynamism can be further improved. The argument for seeking a solution to improve the existing infrastructure is hidden behind the desire to create something, that can really permit the expansion of the automation building systems in the current market. In fact, with the increase of energy consumption[1] (figure 4.1) and the last financial crisis [97], a new vision for energy and money savings has grown.

Automation building systems provide a solution for saving energy, therefore money, (figure 4.2) [80] [73] through the control of all "passive components" inside a building (for example, they provide the possibility to switch on or switch off a power socket and therefore control an oven or a light). The only

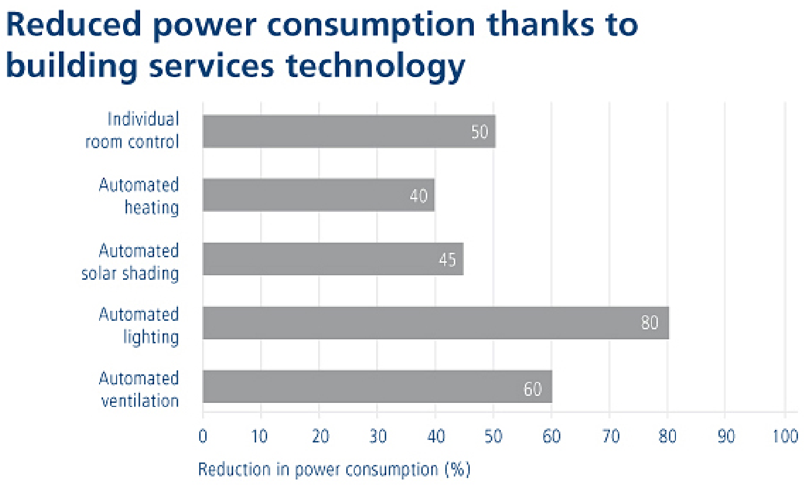


Figure 4.2: Saving energy through intelligent networking [80]

possibility for this technology to be insistently present in the market, and thus in a substantial number of buildings, is to make it easy to use and to configure. Therefore, after the initial procedure for installing the basic structure (KNX structure), final customers and also planners should be able to configure every device easily. However, KNX requires ETS as first configuration tool and, currently, it is the only manner to configure a KNX network. Therefore, an initial configuration by a planner is necessary. However, all the systems now present in the market, in order to access to an automation building network based on KNX using a mobile device, require an additional configuration step. It is used to create a relation between each smart device and a mobile device.

The final aim of this project is to find a system that improves the configuration phases. Improving this process requires to reduce the number of configuration steps. As written above, the first configuration with ETS, in case of KNX network, is indispensable; therefore, the only solution is to reduce, if it is possible, the difficulty of the second configuration step or to remove it.

First of all, it is necessary to understand how the main solutions, currently available in the market, work. Therefore to achieve this purpose, a rightful idea is to use Apple AppStore, as a source of information, to find applications based on KNX technology with a high rank. Apple AppStore [6] contains applications developed for iOS, but usually software created for this operation system are also used for other systems. However, it represents the largest container of

[1]Energy consumption soared by 5.5% in 2010, after a slight decrease in 2009, and was 4.5% above its pre-crisis level. [23]

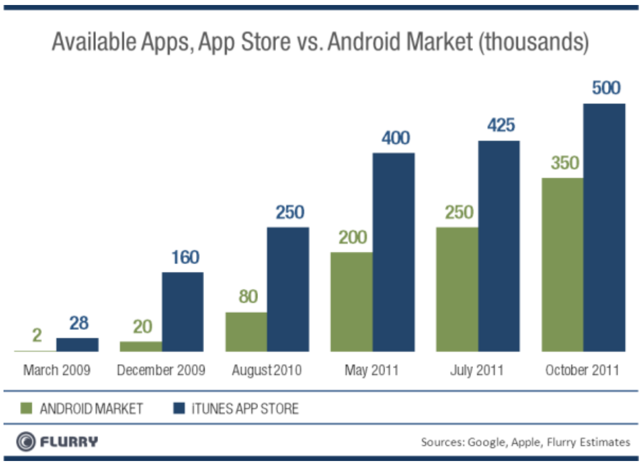


Figure 4.3: Available Applications in Apple AppStore and Android Market [28]

mobile applications (figure 4.3) [28], thus it should contain the main software developed for mobile devices and KNX technology. An alternative to this solution is to get this information via a search engine, such as Google or Yahoo!.

Some of the most popular applications for KNX network are listed below:

ayControl `http://aycontrol.com/`

OpenRemote KNX `http://openremote.org/`

houseinhand KNX `http://houseinhand.com/`

iKNiX `http://iknix.com/`

Each of them, after the standard installation, requires a configuration step in order to use a mobile phone as a controller. Therefore, a customer should know the meaning of all the parameters of the automation system network (for example device type, group address, etc.) and the relative values. This process might be done by the planner but, in any case, he should configure the same values two times.

This project has found a solution to improve the configuration process. The basic idea is to reuse, in an automatic way, KNX network settings defined using ETS. An entire KNX project can be exported, via ETS, into a file with extension `*.knxprod` or `*.knxproj`, thus a software can get it as input and, after a processing phase, share it with all systems connected to it.

4.1 System Concept

Figures 4.5 and 4.4 describe schematically the idea behind this project. The whole system is composed by:

- a Wi-Fi router connected to a broadband connection, if the user wants to have access to the network using for example a GPRS connection;
- a KNXnet/IP gateway in order to guarantee the connection between an IP based network and KNX network;

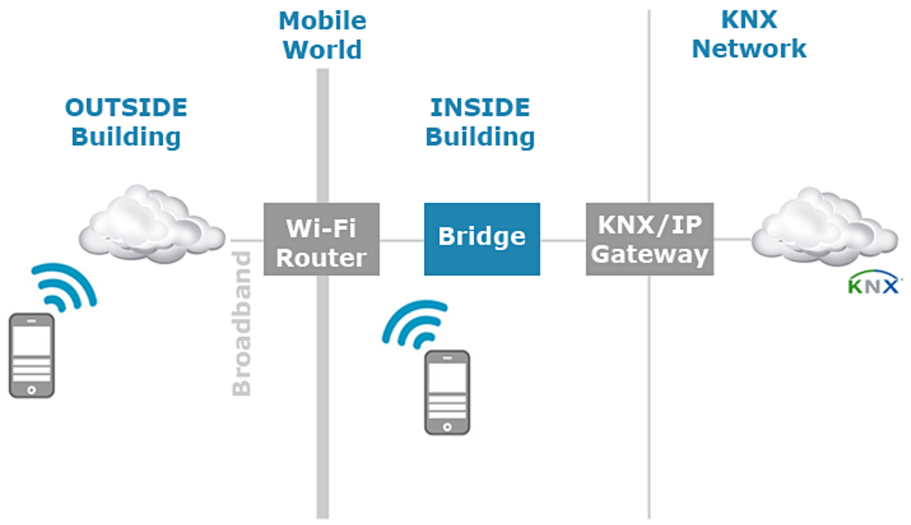


Figure 4.4: Project technical layout.

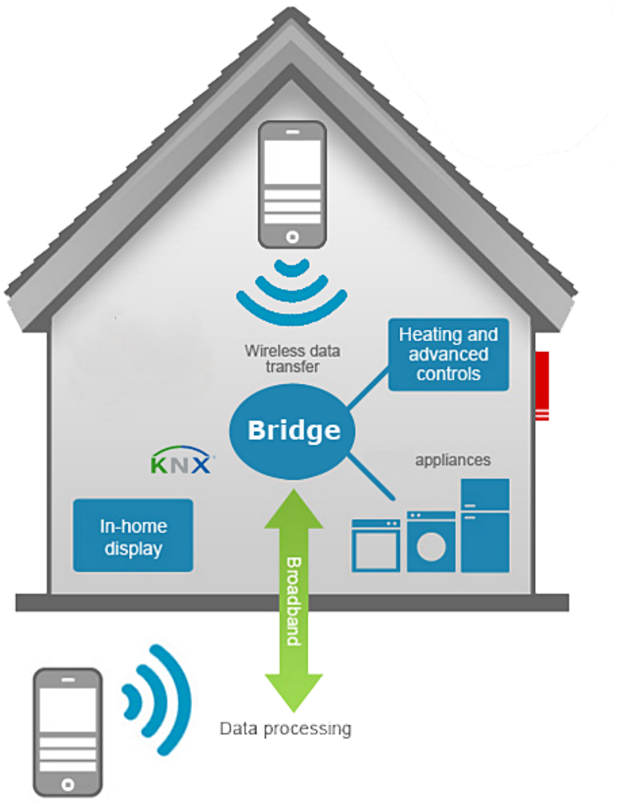


Figure 4.5: Project concept.

- a smartphone, as mobile device, equipped with application created ad-hoc for this project and
- the *Bridge* developed during this work.

Each of these components has a well-defined role and they are all important. Without a Wi-Fi router it would be impossible for a mobile device to get access to the bridge and without a KNXnet/IP gateway, it would be impossible to communicate with KNX network. Instead, by removing the Bridge the network would lose a part of automation.

The Bridge was designed and implemented with the aim of getting as input a KNX project, realized and exported with ETS. It also has to convert this file in a better form for post processing and information sharing. This means that the only workload in addition to the KNX configuration is to upload the exported project on this Bridge. Therefore, even if it represents an additional device, it significantly improves the automation level and, if it is implemented in a rightful manner, it may offer additional features, such as lights control based on music or mail reading, that make it attractive to potential customers.

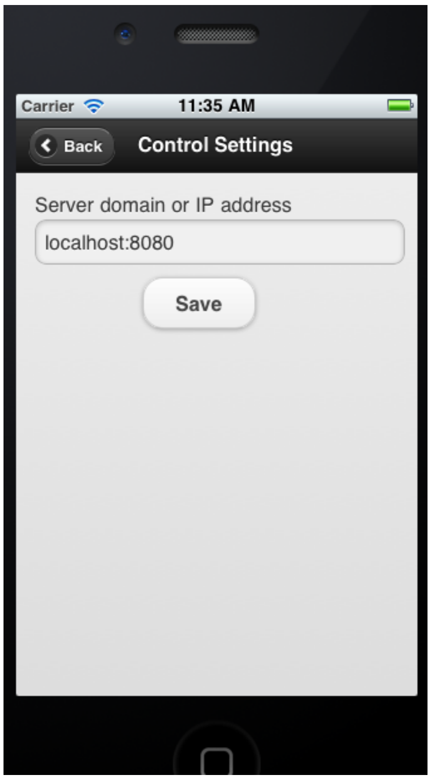
The entire installation process of a network based on KNX technology and the system realized in this project is spread over three steps. In the first phase, KNX network must be installed. Therefore, a customer should purchase smart devices (such as power sockets, switches, dimmers, etc.) and install them inside the building. To achieve this first step, specialized companies sell their knowledge and skills [53]. Once installation is finished, the planner may export the entire project related to the current building into a file with an ETS extension (`*.knxprod` or `*.knxproj`). The second step consists only in connecting the Bridge to the local area network and upload the exported project file on it. Connecting the Bridge to the whole system consists only in two steps: set its local IP address, in order to open a HTTP connection with it, and add this setting and the IP address of KNXnet/IP gateway into its config file. The last phase of this process is to install the mobile application into a device and set the Bridge IP address as server IP address, into the configuration page (figure 4.6).

In this manner, after the first configuration, any changes to the network can be easily shared on all devices. In order to modify the network, it is necessary to use ETS and, once the new settings are ready, only uploading the up-to-date project on the Bridge will give to the system the possibility to automatically share this new information. In fact, when a mobile device requires to open a new connection, it receives the new configuration and thus it will be up-to-date without any additional effort.

4.2 System Implementation

The entire project was realized by splitting it into three main layers (figure 4.7):

1. Interface layer with KNX using Calimero



Server domain or IP address: localhost:8080

Figure 4.6: Mobile application's configuration page.

2. Interface layer with Calimero using Tomcat

3. Interface layer with Tomcat using PhoneGap/iOS

BridgeCore and Bridge

The first step of the implementation was to realize the BridgeCore. Its role is to provide a set of functions with which a higher level application can interact with KNX network. In addition, it must also provide functions to open and to parse a file exported from ETS for the purpose of getting access to the configuration of KNX network. Therefore, this means that real innovation in automation is made by this element.

The main class of BridgeCore has the task of read and set, from `config.xml` (see source code 4.1), the basic settings, such as IP address and port of KNXnet/IP gateway and BridgeCore.

Listing 4.1: Configuration file of the BridgeCore.

```
<?xml version="1.0"?>
<BridgeConfig>
  <config>
    <KNXServerIP>192.168.1.254</KNXServerIP>
    <KNXServerPort>3671</KNXServerPort>
    <KNXClientIP>192.168.1.253</KNXClientIP>
    <KNXClientPort>0</KNXClientPort>
  </config>
</BridgeConfig>
```

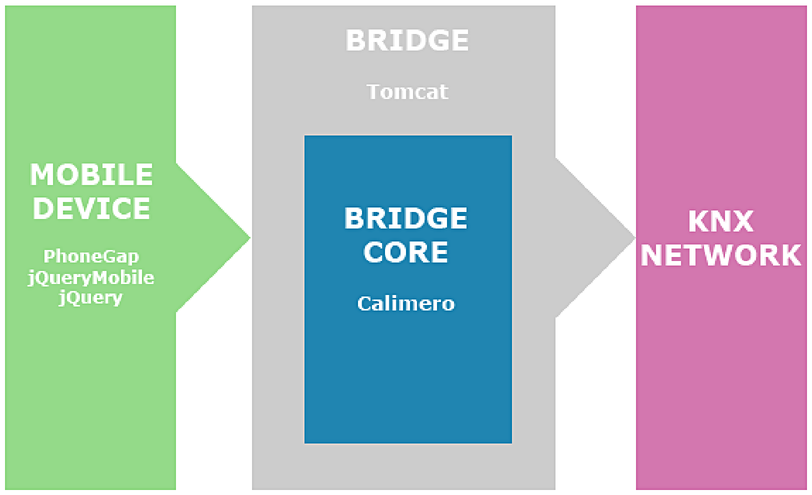


Figure 4.7: Overview of the layer structure.

Once this first phase of configuration is completed, the system starts to wait for a write command or a read command. Once a request has been received, it checks if the system has a `ProcessCommunicator`, with the aim of opening a connection with the KNXnet/IP gateway, and if it has access to the network configuration via a `DataPointExtractor`. In order to check if it has all the tools required, it launches the function `checkSystem()`. In this manner, if there is not a channel between the KNXnet/IP gateway and BridgeCore, the system automatically creates a new connection object. In addition, if the Bridge has not access to the KNX configuration, also in this case, it automatically builds a new extractor object.

The `KNXConnection`, which implements a `ProcessCommunicator`, and the `DataPointExtractor` have a really important role in the entire management and communication process. `DataPointExtractor` represents the class dedicated to control the entire stream of information between the BridgeCore and the KNX project file. Therefore, it has the task of extracting and parsing the information contained in the archive exported from ETS.

In order to get the configuration values from an exported file with the extension `⋆.knxprod` or `⋆.knxproj`, software must perform some procedures. A file exported from ETS is a compressed archive with an extension `⋆.knxprod` or `⋆.knxproj`. Therefore, in order to extract it using a standard extraction software, it must be converted to, for example, a `.zip` or `.tar.gz` file; in this case, the first was chosen. Once this translation has been completed, the system can extract the file and get its content, which is however still unusable. Thus to make it usable, it is necessary to parse the file `0.xml` (contained into a folder named, for example, `P-0497`), which contains the real useful information, with a XSL parser [34].

Once obtained the processed file, `datapoint.xml` (source code 4.2), all that remains to do is to extract its contents and remove unnecessary files. The entire process is performed thanks to four functions:

- `renameETSExportFile()`
- `extractArchive()`
- `processXmlFile()`
- `cleanSpace()`

Listing 4.2: Example of `datapoint.xml`.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<datapoints>
  ...
  <datapoint stateBased="true" name="Room 2#Socket Left bed"
     mainNumber="1" dptID="1.001">
    <knxAddress type="group">4097</knxAddress>
    <expiration timeout="0"/>
    <updatingAddresses> </updatingAddresses>
    <invalidatingAddresses> </invalidatingAddresses>
  </datapoint>
  ...
</datapoints>
```

`KNXConnection` is a class that hides behind an easy to use set of functions, the communication bundle of Calimero. Using this object, it is possible to create in one step a `KNXNetworkLinkIP`, based on the values received from `config.xml`, and a `ProcessCommunicator`. Thus, by using `connect()` it is possible to get access immediately to a KNX network via KNXnet/IP gateway. In addition, it provides all the functions for writing and reading values of a KNX network, set communication timeout, check connection status, ect.

Therefore, the BridgeCore is enough to control an automation building system based on KNX technology. It provides functions to open a connections with a KNXnet/IP gateway, to send and to receive messages, to load configuration files and to manage exceptions. However, this system is not useful for a final customer and also for an installer. In order to make it easy to use, it is necessary to provide an interface system to manage the Bridge with the minimum number of commands and information. One of the possibilities was to use a web server to get HTTP requests and to translate them to a suitable form. Thus, the BridgeCore may process this data and send the results back to the web server, which can translate them to an appropriate form for a HTTP streaming of information and retransmit the results (figure 4.8). A HTTP system, as communication protocol, permits to the BridgeCore and all systems around it to exchange messages using a channel based on a standard and well-know protocol.

The BridgeCore was written using Calimero and then Java. In order to use the code developed for the basic structure, Tomacat provides the best solution, in fact it can easily interact with application written with Java. In fact, the Servlet technology permits to Tomcat to be “connected” with a Java application. Therefore, it is necessary to adapt the code written for the BridgeCore to a servlet context.

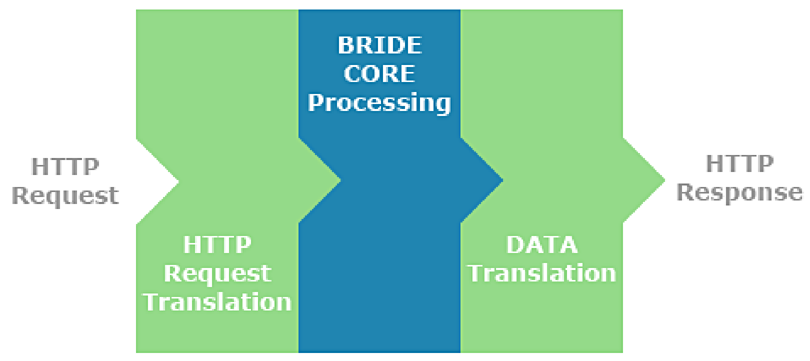


Figure 4.8: Communication process using a web server.

To achieve this purpose, it is indispensable to create a new application, which merges these two technologies. Therefore, the extension to `HttpServlet` class [40] was added to the main class of the BridgeCore. In this manner, it was possible to implement `doGet()` and `doPost()`, which respectively respond to GET and POST requests.

The function `doGet()` is extremely useful; in fact with it, it is possible, for a client (for example a browser), sending data through an uniform resource locator (URL) (source code 4.3).

Listing 4.3: Send data using a HTTP GET request.

```
http://localhost:8080/KNXBridge?check=1&connection=on
```

The example above shows how to send data using a GET request. In this case the source code behind the web page `http://localhost:8080/KNXBridge` receives as data:

1. `check = 1`
2. `connection = on`

Thus, in this manner, it is possible to send data that can be used to manage an application behind a web server. Therefore, by changing the BridgeCore, it was possible to receive and to send data from and to a KNX network, using the set of functions provided by the BridgeCore connected to a KNXnet/IP gateway.

Sending and receiving information requires a switching system with the aim of divide request sent for getting or setting values. Therefore, both of these requests have a special parameter, which distinguishes one from the other. Supposed that when a mobile device tries to get access to the automation system network the Bridge[2] is always active, there are three main actions that a mobile device may require: get configuration data, set a value and read a value.

[2]Henceforth with the term Bridge will be intended the union between web server (Tomcat) and BridgeCore.

The first action, get configuration value, is easily performed by sending a HTTP GET request at the url 4.4. This special request contains as parameters the name of the wrapper (see 3.4.1) and the request to get the KNX configuration data (`getData = 1`).

Listing 4.4: Get configuration data from the Bridge.

```
http://<BridgeCore IP address or URL>/KNXBridge/connection.jsp?
    datawrapper=?&getData=1
```

Analyzing the url reported in the 4.4, it is possible to understand where the servlet is located, which is responsible for data from the world of HTTP. In fact, behind `http://.../KNXBridge/connection.jsp` there is the rewritten Bridge Core, which is called by `connection.jsp` (in source code 4.5, tag `page="KNXBri dge"` include the BridgeCore servlet called *KNXBridge*).

Listing 4.5: connection.jsp source code.

```
<%@page contentType="application/json" pageEncoding="UTF-8"%>
<jsp:include page="KNXBridge"/>
```

When the Bridge receives this request, it calls a `DataPointManipulator` that checks if it already exists the translation in a JSONP form of the automation building configuration data. Therefore, it looks for a file named `jsonDatapoints.json` (source code 4.6), which contains all the information about the KNX configuration in a JSON form. If it exists, the Bridge just opens this file and loads its content as a `String`, otherwise it parses the file obtained, processing the project exported from ETS and loads the result as a `String`. Once this information is loaded, the Bridge can build the response, send it via a HTTP connection and create the `jsonDatapoints.json`.

Listing 4.6: Example of `jsonDatapoints.json`.

```
{
  ...
  {
    "obj":"Socket Left bed",
    "val":
      [
        {
          "id":0,
          "mainType":1,
          "subTyp":1,
          "mainGroupAddress":"2/0/1"
        }
      ]

  }
  ...
}
```

If a mobile device wants to send a command to the KNX network so if it wants to write a value, it only has to send a HTTP GET request to the Bridge.

This request is generated using three parameters: one to inform that it is a writing action, one to specify where the new value must be written and the last to provide a new value. Therefore, the request should be formed as the one shown in 4.7.

Listing 4.7: Write a new value in KNX network.

```
http://<BridgeCore IP address or URL>/KNXBridge/connection.jsp?
    set=1&id=7&value=true
```

Using the parameter `set = 1` the switcher, inside the Bridge, knows that the request is a writing action. In particular, it wants to change the value currently present in the device represented by `id = 7` with `true`, as specified by the parameter `value = true`. Once the processing by the Bridge is completed the applicant will be notified. The id used in the previous example is referred to a number that creates a relation one-to-one between a JSON representation, of a KNX device, and its abstraction inside the Bridge. This index is obtainable during the first connection, when a get configuration request is sent, and it gives to the Bridge the possibility to load a specific device, or better a datapoint, and its settings.

The last action managed by the Bridge is a reading request. As all the other types, it comes from the use of a special URL, which also contains specific parameters (see source code 4.8).

Listing 4.8: Read a value from KNX network.

```
http://<BridgeCore IP address or URL>/KNXBridge/connection.jsp?
    status=1&id=7&groupAddress=2/1/1
```

In the case, the parameter `status = 1` has the aim to inform the switcher about the type of action to take. Even in this case, the `id` parameter has the meaning of a reference code; instead, the last value, `groupAddress = 2/1/1`, represents the group address associated to a smart device in the network. A reading action requires to specify, using the group address string, which value the reader has to get from a device, because every device may be associated to more than one group address and each of them can have different value. However, not every devices are readable; in fact during the configuration process by ETS, it is possible to set a device as unreadable. In this case, it is possible to close the connection without a value and to report it to a client, thanks to the timeout event.

Mobile device application

In order to close the chain (BridgeCore, Bridge and remote access), it was necessary to develop an user interface that would allow a client to get access to the Bridge easily. Using a mobile device such as an iPhone, the only way to create a connection between it and a server, or a dedicated station, is to use a Wi-Fi or GPRS network. Therefore, the first step is to get access to a network interface, but thanks to PhoneGap, it is possible to use all the main features provided by a mobile device, such as network connection. The real role of this mobile application is to render the information that it receives from the Bridge. Therefore, it must be able to send and to receive HTTP data, to read JSON/JSONP information and to create a user interface. In order to satisfy all these requirements, the first step is to check if the mobile device is connected and if it is able to talk with the Bridge. Therefore, when the application is loaded a simple Ajax request is sent directly to the Bridge (source code 4.9).

Listing 4.9: Check if the connection is established.

```
function testConnection() {
  return $.ajax({
          type: 'GET',
          cache: false,
          url: testUrl
      });
}
```

Once the application gets the answer, it sends a get configuration request. However, if it is not able to create a connection with the Bridge, the user is notified (figure 4.9) so it has the possibility to reconfigure the application using the specific page (figure 4.6). Configuration page appears also when the software is launched for the first time. If the connection is established and

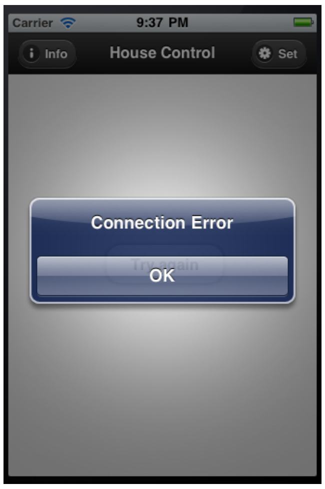


Figure 4.9: Configuration page.

the information is received, the application can render the data realizing a sequence of navigable pages. Using this series of pages a client can virtually surf into the KNX network. Only with a single touch on the screen is possible to change the page and, in this manner, have access to all devices of the managed automation building network (figures 4.10). Once a user gets access to a device section, the application creates an ad-hoc page which is different for every type of device. Therefore, using the data received from the Bridge, the application can understand which type of device is selected and, in this manner, it knows which is the best rendering form for that specific type of datapoint (figures 4.11). The last step, in order to complete the entire process, is to create the rightful system to send HTTP GET requests to the Bridge with

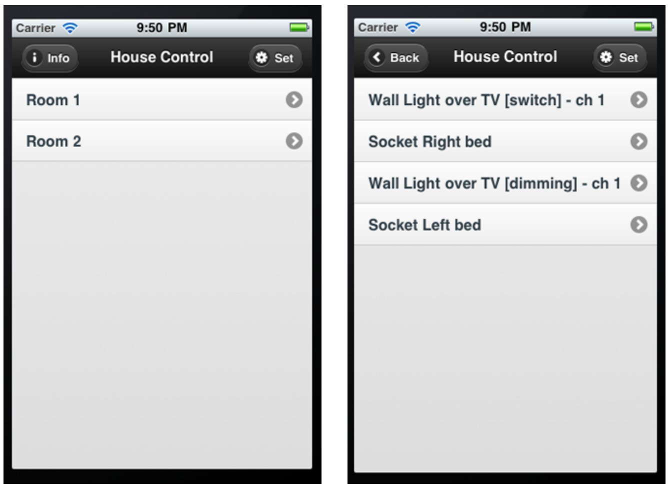


Figure 4.10: Example of navigable pages.

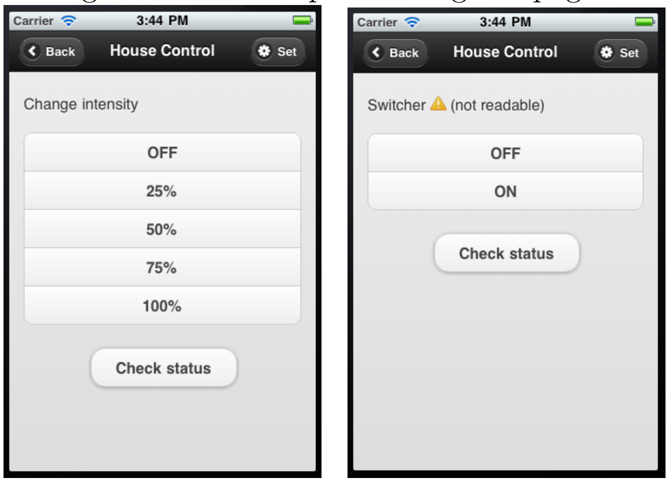


Figure 4.11: Example of ad-hoc pages.

specific parameters. The information about a given datapoint may be derived from the JSON data, therefore it is only necessary to associate the right URL (dynamically created) with the correct actuator. Furthermore, thanks to Ajax, it is possible to wait the completion of the entire process and so, at the end of it, notify the client changing the user interface (figure 4.12).

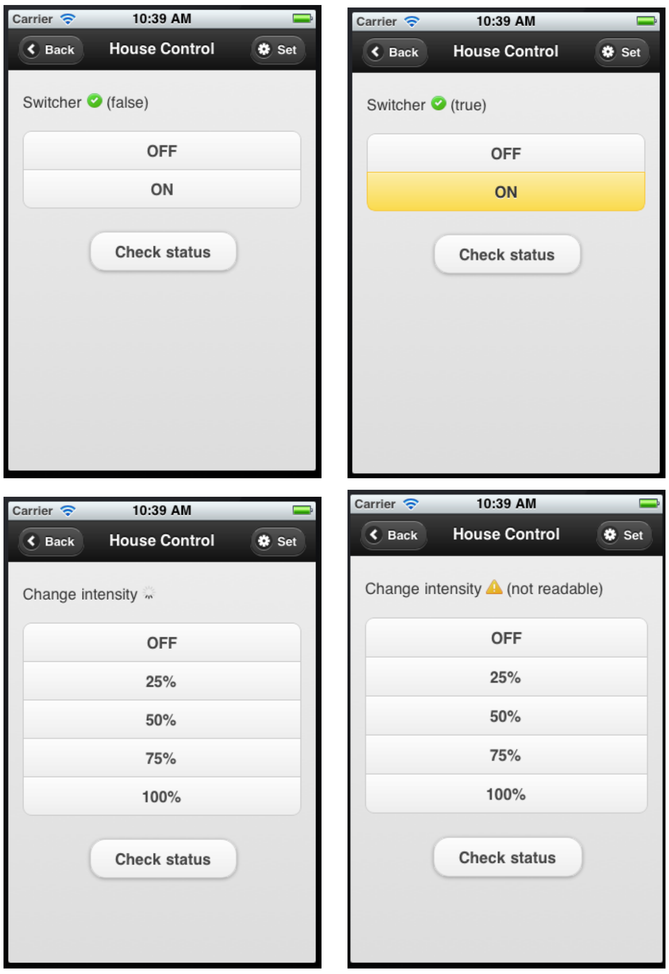


Figure 4.12: Ad-hoc device pages with related status and notification.

Chapter 5

System Testing

The entire system was developed using a real laboratory, which provides a KNX network and several workstations dedicated to the development of applications based on Konnex technology (the laboratory is located in the room D003 at Northumbria University, United Kingdom (figures 5.1)). In this man-

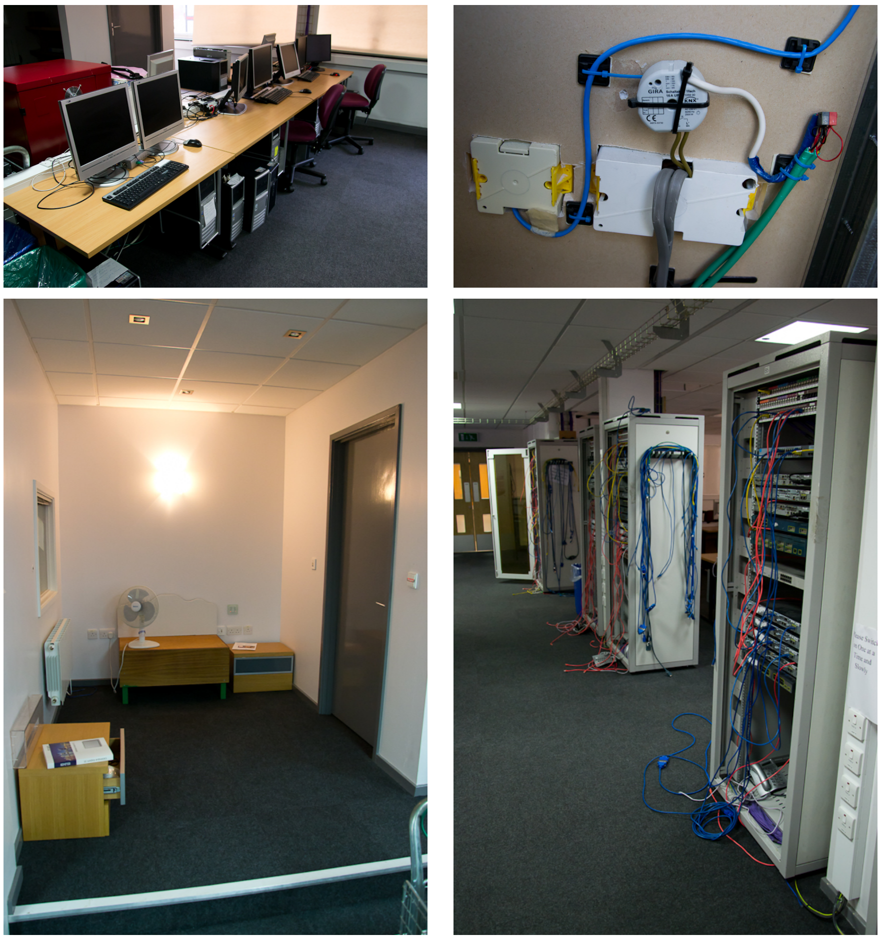


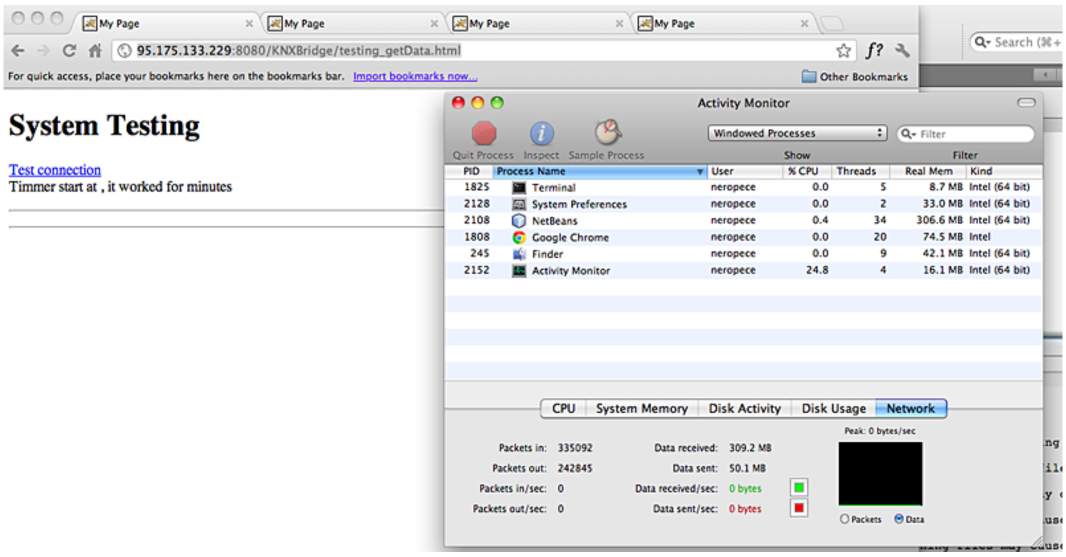
Figure 5.1: Room D003 - Ellison Building, Northumbria University.

ner, each part of this project could be tested in a real environment. Therefore, all the following tests are affected by real problems of communication and computing. However, it was possible to realize a virtualization of the whole KNX network, but most of the tests conducted by researchers are based on simulations, so it was decided to create a new benchmark based on real tests.

This project has been thought to be an application into a residential environment. Therefore, a real situation requires at least three or four mobile devices, a KNXnet/IP gateway and one Bridge connected all together using a Wi-Fi network. However, the restrictions imposed by Apple about the software distribution [5] and the impossibility to have enough mobile devices were nonetheless overcome. The mobile application, in fact, is based on HTTP requests and replies, therefore it is possible to "recreate" a real mobile device using web pages created ad-hoc for this purpose. In addition, using streaming video it is possible to saturate the bandwidth and test the behavior of the entire system even in this case.

5.1 Getting KNX configuration

The first test was dedicated to the observation of "getting data system" behavior. In this case, a dedicated local area network based on Wi-Fi technology has been created. The network was composed by an access point, a mobile phone (iPhone 3GS) and two MacBooks. The Bridge was installed in one of the two MacBooks. In order to recreate a mobile device and the basic interaction between it and the Bridge, an ad-hoc web page was created (figure 5.2).



95.175.133.229:8080/KNXBridge/testing_getData.html

For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

Other Bookmarks

System Testing

Test connection

Timmer start at , it worked for minutes

Activity Monitor

Windowed Processes

PID	Process Name	User	% CPU	Threads	Real Mem	Kind
1825	Terminal	neropece	0.0	5	8.7 MB	Intel (64 bit)
2128	System Preferences	neropece	0.0	2	33.0 MB	Intel (64 bit)
2108	NetBeans	neropece	0.4	34	306.6 MB	Intel (64 bit)
1808	Google Chrome	neropece	0.0	20	74.5 MB	Intel
245	Finder	neropece	0.0	9	42.1 MB	Intel (64 bit)
2152	Activity Monitor	neropece	24.8	4	16.1 MB	Intel (64 bit)

Peak: 0 bytes/sec

Packets in:	335092	Data received:	309.2 MB
Packets out:	242845	Data sent:	50.1 MB
Packets in/sec:	0	Data received/sec:	0 bytes
Packets out/sec:	0	Data sent/sec:	0 bytes

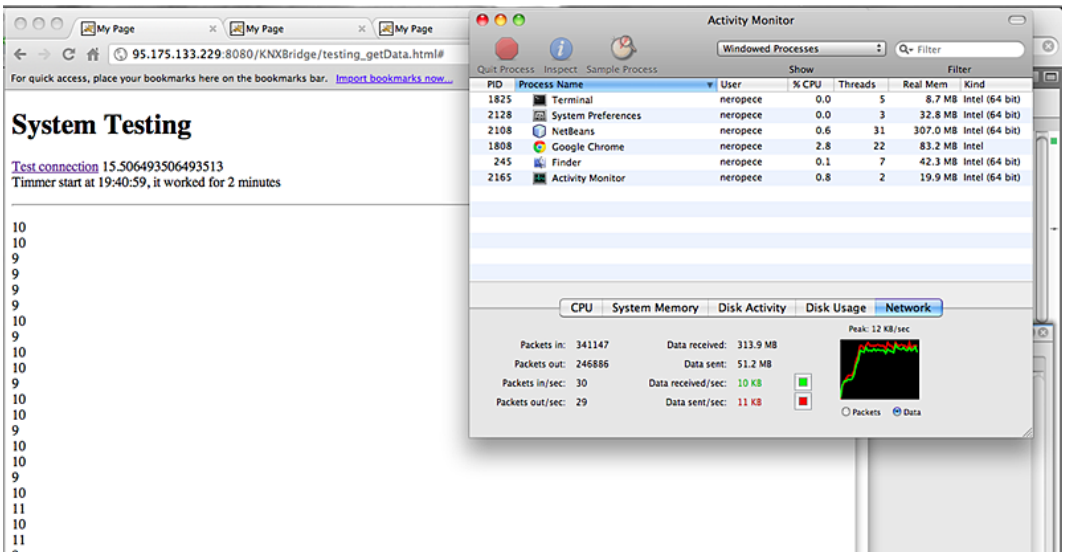
Figure 5.2: Ad-hoc web page to recreate a mobile device.

The source code 5.1 represents the main JavaScript function used to create a sequence of getting KNX configuration requests sent to the Bridge. In order to generate a request automatically, a timer, which creates a getting action every 1000 milliseconds, was used. At the end of each request, the function adds the execution time into a dedicated area.

Listing 5.1: Ad-hoc page for getting data.

```
window.setInterval(function() {
  start = new Date().getTime();
  $('#timeStart').html(timmer.getHours() + ':' + timmer.
      getMinutes() + ':' + timmer.getSeconds());
  $.getJSON(serverUrl + '/KNXBridge/connection.jsp?datawrapper
      =?&getData=1', function(data) {
    value = (new Date().getTime()) - start;
    $('#stepexecutiontime').prepend(value + '<br>');
    avg = (avg * couter++ + value) / couter;
    $('#avg').html(avg);
    $('#timePass').html((new Date()).getMinutes() - timmer.
        getMinutes());
  })
  .error(function(data) {
    $('#stepexecutiontime').prepend(' <b>ERROR in reading</b>')
        ;
  });
}, 1000);
```

Therefore, using this web page with the two MacBooks, each of them with four browser tabs opened, and the iPhone 3GS, it was possible to recreate an environment composed by nine mobile devices (figures 5.3 and 5.4).



System Testing

Test connection 15.506493506493513
Timmer start at 19:40:59, it worked for 2 minutes

```
10
10
9
9
9
9
10
9
10
10
9
10
10
9
10
10
9
10
11
10
11
```

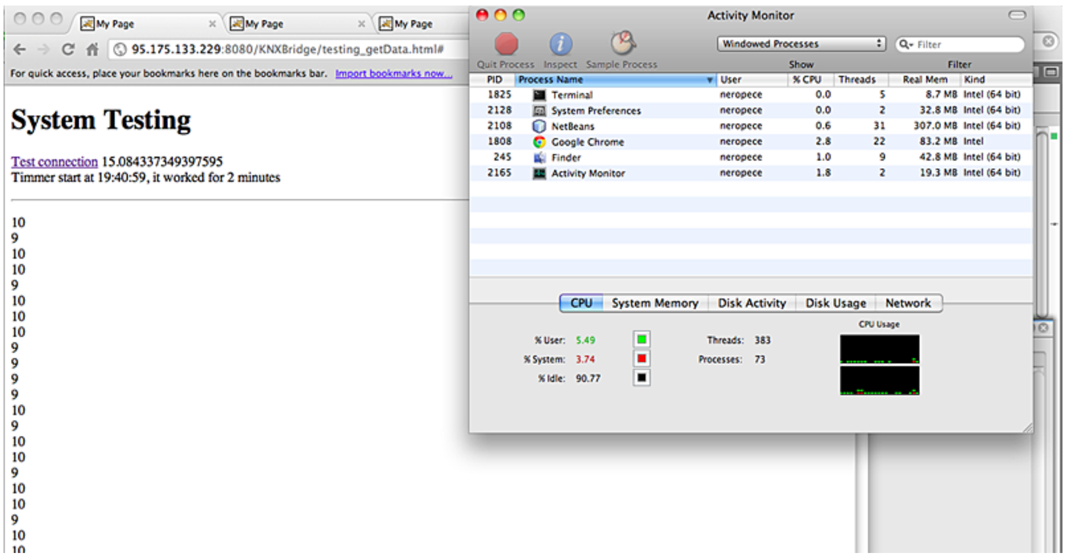
PID	Process Name	User	% CPU	Threads	Real Mem	Kind
1825	Terminal	neropece	0.0	5	8.7 MB	Intel (64 bit)
2128	System Preferences	neropece	0.0	3	32.8 MB	Intel (64 bit)
2108	NetBeans	neropece	0.6	31	307.0 MB	Intel (64 bit)
1808	Google Chrome	neropece	2.8	22	83.2 MB	Intel
245	Finder	neropece	0.1	7	42.3 MB	Intel (64 bit)
2165	Activity Monitor	neropece	0.8	2	19.9 MB	Intel (64 bit)

Packets in:	341147	Data received:	313.9 MB	
Packets out:	246886	Data sent:	51.2 MB	
Packets in/sec:	30	Data received/sec:	10 KB	🟩
Packets out/sec:	29	Data sent/sec:	11 KB	🟥

Figure 5.3: Testing web page with network load during the first phase.

Once enough testing data was obtained, a source of network and CPU overhead, represented by a streaming video and music, was introduced. The first of the two sources was generated by the browser's flash plugin, instead the second one was created using a dedicated software. Each of them received data from the Wi-Fi network using the access point (figures 5.5 and 5.6). In order to check the behavior of the system after a period of overload, during the last part of this testing phase (the last 5 minutes) the source of overhead was removed.

The chart in figure 5.7 shows the execution time of every request. It is possible to observe that the average time corresponds to 73.34 milliseconds and the majority of the requests have an execution time under this level. In



95.175.133.229:8080/KNXBridge/testing_getData.html#

For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

System Testing

Test connection 15.084337349397595
Timmer start at 19:40:59, it worked for 2 minutes

10
9
10
10
9
10
10
10
9
9
9
9
10
9
10
10
9
10
10
9
10

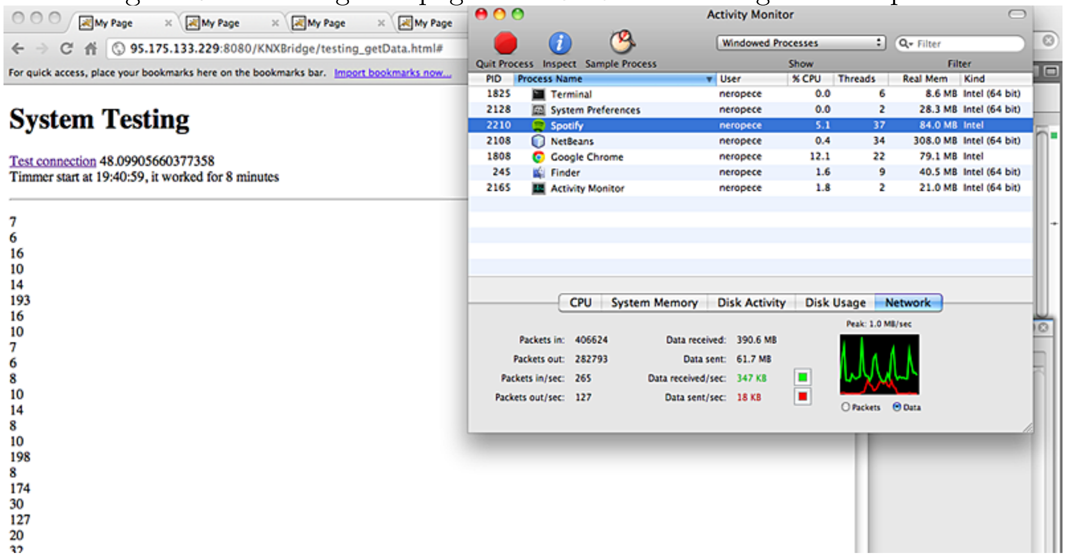
Windowed Processes

PID	Process Name	User	% CPU	Threads	Real Mem	Kind
1825	Terminal	neropece	0.0	5	8.7 MB	Intel (64 bit)
2128	System Preferences	neropece	0.0	2	32.8 MB	Intel (64 bit)
2108	NetBeans	neropece	0.6	31	307.0 MB	Intel (64 bit)
1808	Google Chrome	neropece	2.8	22	83.2 MB	Intel
245	Finder	neropece	1.0	9	42.8 MB	Intel (64 bit)
2165	Activity Monitor	neropece	1.8	2	19.3 MB	Intel (64 bit)

[✓ CPU] System Memory Disk Activity Disk Usage Network

% User: 5.49 🟩
% System: 3.74 🟥
% Idle: 90.77 ■
Threads: 383
Processes: 73

Figure 5.4: Testing web page with CPU load during the first phase.



95.175.133.229:8080/KNXBridge/testing_getData.html#

For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

System Testing

Test connection 48.09905660377358
Timmer start at 19:40:59, it worked for 8 minutes

7
6
16
10
14
193
16
10
7
6
8
10
14
8
10
198
8
174
30
127
20

Windowed Processes

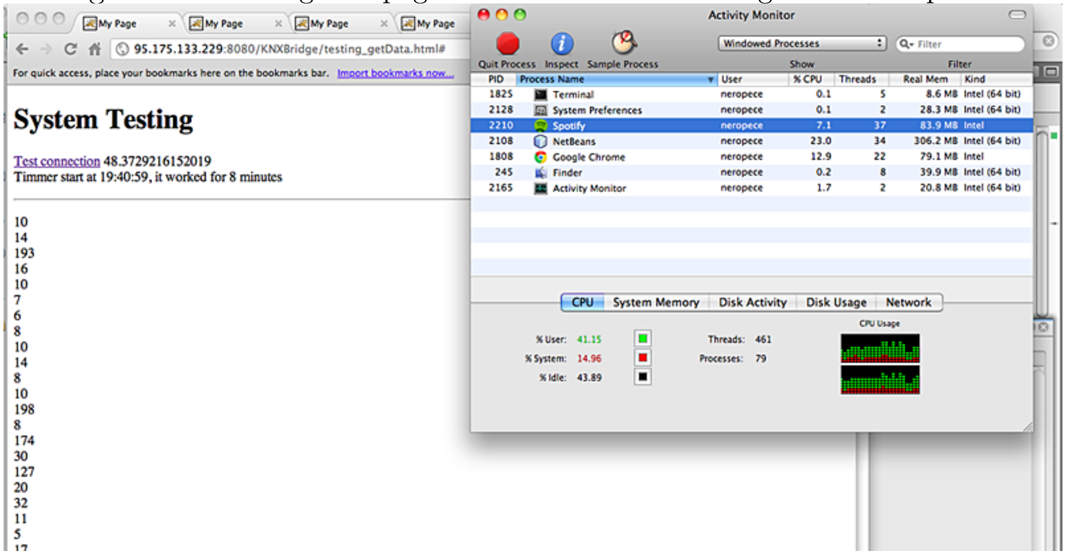
PID	Process Name	User	% CPU	Threads	Real Mem	Kind
1825	Terminal	neropece	0.0	6	8.6 MB	Intel (64 bit)
2128	System Preferences	neropece	0.0	2	28.3 MB	Intel (64 bit)
2210	Spotify	neropece	5.1	37	84.0 MB	Intel
2108	NetBeans	neropece	0.4	34	308.0 MB	Intel (64 bit)
1808	Google Chrome	neropece	12.1	22	79.1 MB	Intel
245	Finder	neropece	1.6	9	40.5 MB	Intel (64 bit)
2165	Activity Monitor	neropece	1.8	2	21.0 MB	Intel (64 bit)

CPU System Memory Disk Activity Disk Usage [✓ Network]

Packets in: 406624
Packets out: 282793
Packets in/sec: 265
Packets out/sec: 127
Data received: 390.6 MB
Data sent: 61.7 MB
Data received/sec: 347 KB 🟩
Data sent/sec: 18 KB 🟥

Packets [✓ Data]

Figure 5.5: Testing web page with network load during the second phase.



95.175.133.229:8080/KNXBridge/testing_getData.html#

For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

System Testing

Test connection 48.3729216152019
Timmer start at 19:40:59, it worked for 8 minutes

10
14
193
16
10
7
6
8
10
14
8
10
198
8
174
30
127
20
32
11
5

Windowed Processes

PID	Process Name	User	% CPU	Threads	Real Mem	Kind
1825	Terminal	neropece	0.1	5	8.6 MB	Intel (64 bit)
2128	System Preferences	neropece	0.1	2	28.3 MB	Intel (64 bit)
2210	Spotify	neropece	7.1	37	83.9 MB	Intel
2108	NetBeans	neropece	23.0	34	306.2 MB	Intel (64 bit)
1808	Google Chrome	neropece	12.9	22	79.1 MB	Intel
245	Finder	neropece	0.2	8	39.9 MB	Intel (64 bit)
2165	Activity Monitor	neropece	1.7	2	20.8 MB	Intel (64 bit)

[✓ CPU] System Memory Disk Activity Disk Usage Network

% User: 41.15 🟩
% System: 14.96 🟥
% Idle: 43.89 ■
Threads: 461
Processes: 79

Figure 5.6: Testing web page with CPU load during the second phase.

addition, using the same chart, it is evident that network and CPU overload do not affect the communication system between the Bridge and a mobile device.

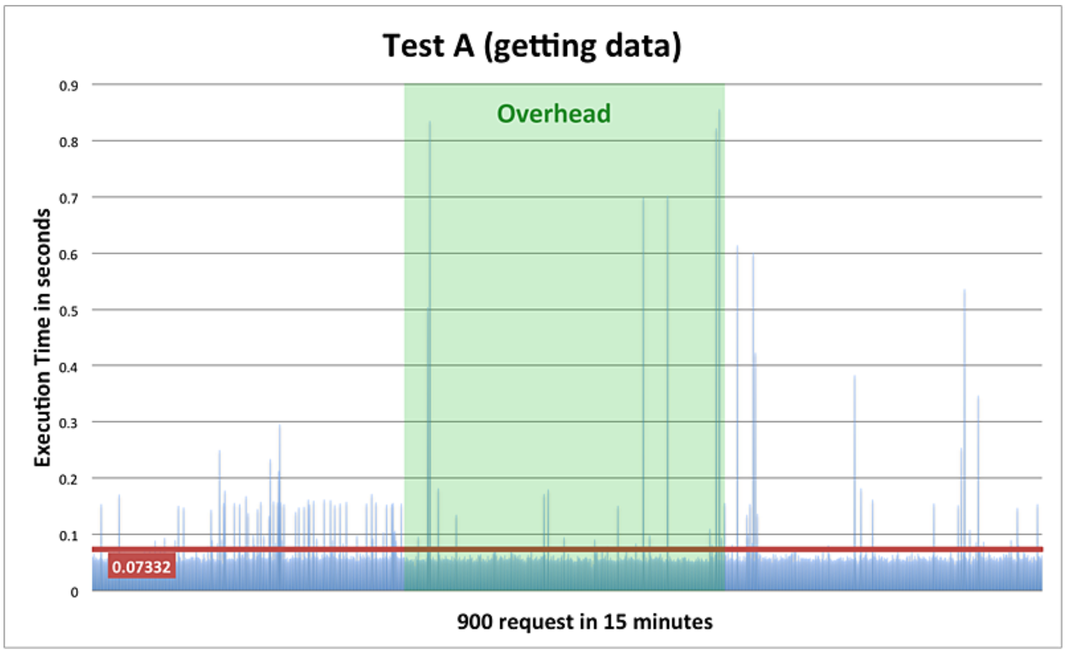


Figure 5.7: Testing chart with 900 requests in 15 minutes on an iPhone 3GS.

Therefore, the communication channel between mobile world and Bridge has a good behavior even when subjected to “external interferences”.

5.2 Setting KNX values

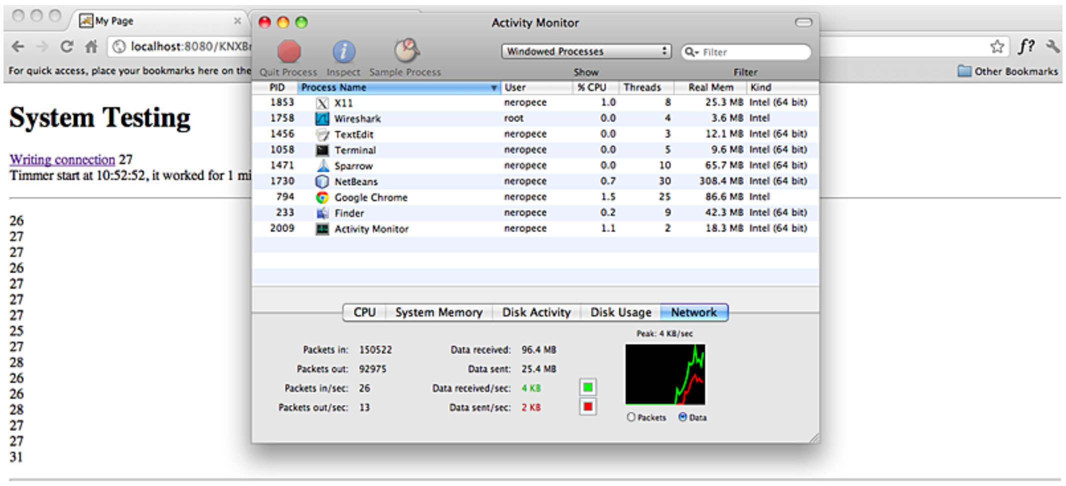
The first testing phase was dedicated to the interaction in the mobile world. The second testing part, instead, was used to verify the behavior of the entire system using writing requests. In this manner, the entire process perhaps affected also the building automation system. Even in this case, in order to recreate an environment with several mobile devices, a web page created ad-hoc (source code 5.2), which sent every 1000 milliseconds, for 15 minutes, a writing request, has been used.

Listing 5.2: Ad-hoc page for writing data.

```
window.setInterval(function() {
  start = new Date().getTime();
  $('#timeStart').html(timmer.getHours() + ':' + timmer.
     getMinutes() + ':' + timmer.getSeconds());
  $.getJSON(serverUrl + '/KNXBridge/connection.jsp?set&id=8&
     value=' + set)
  .complete(function(data) {
    set = !set;
    value = (new Date().getTime()) - start;
    $('#stepexecutiontime').prepend(value + '<br>');
    avg = (avg * couter++ + value) / couter;
```

```
        $('#avg').html(avg);
        $('#timePass').html((new Date()).getMinutes() - timmer.
            getMinutes());
      })
      .error(function(data) {
        $('#stepexecutiontime').prepend(' <b>ERROR in writing</b>')
            ;
      });
    }, 1000);
```

Using this page with the iPhone 3GS and the two MacBooks, each of them with four browser tabs opened, it was possible to recreate an environment with nine mobile devices (figure 5.8).



System Testing

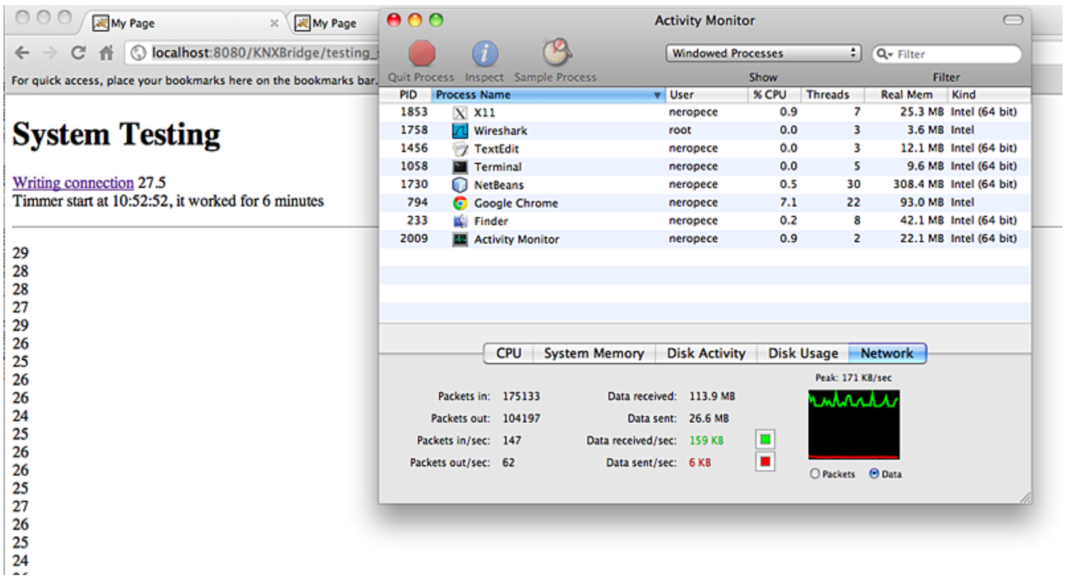
Writing connection 27
Timmer start at 10:52:52, it worked for 1 mi

26
27
27
26
27
27
27
25
27
28
26
26
28
27
27
31

PID	Process Name	User	% CPU	Threads	Real Mem	Kind
1853	X11	neropece	1.0	8	25.3 MB	Intel (64 bit)
1758	Wireshark	root	0.0	4	3.6 MB	Intel
1456	TextEdit	neropece	0.0	3	12.1 MB	Intel (64 bit)
1058	Terminal	neropece	0.0	5	9.6 MB	Intel (64 bit)
1471	Sparrow	neropece	0.0	10	65.7 MB	Intel (64 bit)
1730	NetBeans	neropece	0.7	30	308.4 MB	Intel (64 bit)
794	Google Chrome	neropece	1.5	25	86.6 MB	Intel
233	Finder	neropece	0.2	9	42.3 MB	Intel (64 bit)
2009	Activity Monitor	neropece	1.1	2	18.3 MB	Intel (64 bit)

Packets in:	150522	Data received:	96.4 MB
Packets out:	92975	Data sent:	25.4 MB
Packets in/sec:	26	Data received/sec:	4 KB 🟩
Packets out/sec:	13	Data sent/sec:	2 KB 🟥

Figure 5.8: Testing web page with network load during the first phase.



System Testing

Writing connection 27.5
Timmer start at 10:52:52, it worked for 6 minutes

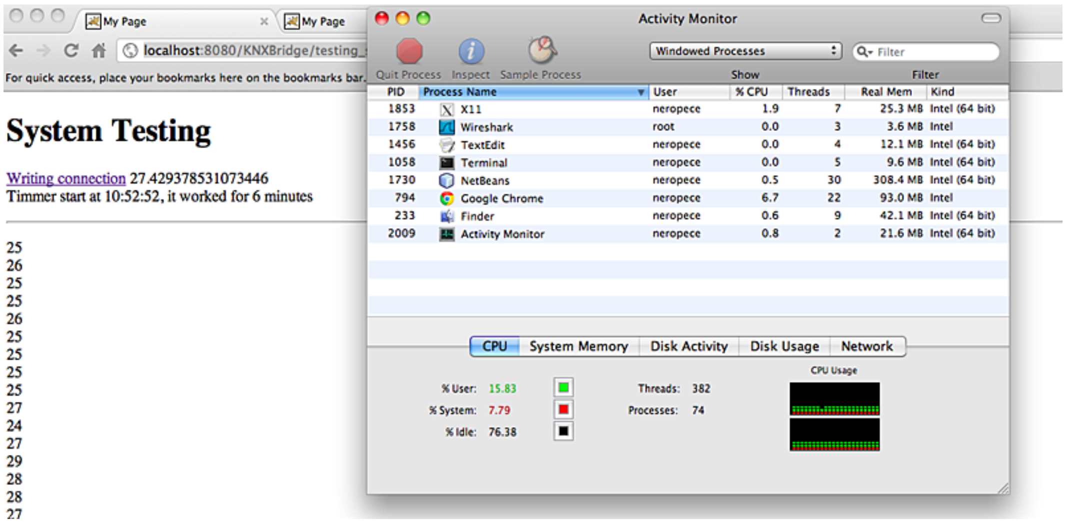
29
28
28
27
29
26
25
26
26
24
25
26
26
25
27
26
25
24

PID	Process Name	User	% CPU	Threads	Real Mem	Kind
1853	X11	neropece	0.9	7	25.3 MB	Intel (64 bit)
1758	Wireshark	root	0.0	3	3.6 MB	Intel
1456	TextEdit	neropece	0.0	3	12.1 MB	Intel (64 bit)
1058	Terminal	neropece	0.0	5	9.6 MB	Intel (64 bit)
1730	NetBeans	neropece	0.5	30	308.4 MB	Intel (64 bit)
794	Google Chrome	neropece	7.1	22	93.0 MB	Intel
233	Finder	neropece	0.2	8	42.1 MB	Intel (64 bit)
2009	Activity Monitor	neropece	0.9	2	22.1 MB	Intel (64 bit)

Packets in:	175133	Data received:	113.9 MB
Packets out:	104197	Data sent:	26.6 MB
Packets in/sec:	147	Data received/sec:	159 KB 🟩
Packets out/sec:	62	Data sent/sec:	6 KB 🟥

Figure 5.9: Testing web page with network load during the second phase.

As for the first testing, after 5 minutes, a source of network and CPU overhead represented by a streaming video, generated by the browser's flash plugin, which received data using the Wi-Fi network (figures 5.9 and 5.10), was introduced. During the last part of this testing phase (between the 10th and 15th minute), the source of overhead was removed in order to check the behavior of the system after a period of overload.



localhost:8080/KNXBridge/testing

For quick access, place your bookmarks here on the bookmarks bar.

System Testing

Writing connection 27.429378531073446
Timmer start at 10:52:52, it worked for 6 minutes

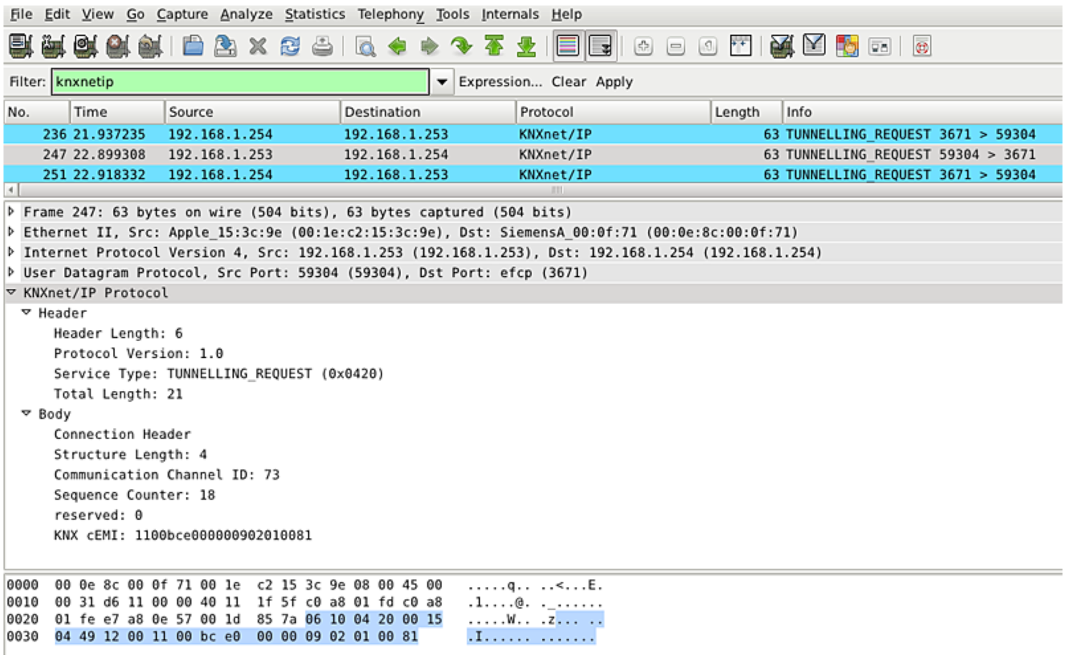
```
25
26
25
25
26
25
25
25
25
27
24
27
29
28
28
27
```

Activity Monitor — Show: Windowed Processes

PID	Process Name	User	% CPU	Threads	Real Mem	Kind
1853	X11	neropece	1.9	7	25.3 MB	Intel (64 bit)
1758	Wireshark	root	0.0	3	3.6 MB	Intel
1456	TextEdit	neropece	0.0	4	12.1 MB	Intel (64 bit)
1058	Terminal	neropece	0.0	5	9.6 MB	Intel (64 bit)
1730	NetBeans	neropece	0.5	30	308.4 MB	Intel (64 bit)
794	Google Chrome	neropece	6.7	22	93.0 MB	Intel
233	Finder	neropece	0.6	9	42.1 MB	Intel (64 bit)
2009	Activity Monitor	neropece	0.8	2	21.6 MB	Intel (64 bit)

```
% User:   15.83     Threads:   382
% System:  7.79     Processes:  74
% Idle:   76.38
```

Figure 5.10: Testing web page with CPU load during the second phase.



File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply

Filter: knxnetip

No.	Time	Source	Destination	Protocol	Length	Info
236	21.937235	192.168.1.254	192.168.1.253	KNXnet/IP	63	TUNNELLING_REQUEST 3671 > 59304
247	22.899308	192.168.1.253	192.168.1.254	KNXnet/IP	63	TUNNELLING_REQUEST 59304 > 3671
251	22.918332	192.168.1.254	192.168.1.253	KNXnet/IP	63	TUNNELLING_REQUEST 3671 > 59304

```
▷ Frame 247: 63 bytes on wire (504 bits), 63 bytes captured (504 bits)
▷ Ethernet II, Src: Apple_15:3c:9e (00:1e:c2:15:3c:9e), Dst: SiemensA_00:0f:71 (00:0e:8c:00:0f:71)
▷ Internet Protocol Version 4, Src: 192.168.1.253 (192.168.1.253), Dst: 192.168.1.254 (192.168.1.254)
▷ User Datagram Protocol, Src Port: 59304 (59304), Dst Port: efcp (3671)
▽ KNXnet/IP Protocol
  ▽ Header
      Header Length: 6
      Protocol Version: 1.0
      Service Type: TUNNELLING_REQUEST (0x0420)
      Total Length: 21
  ▽ Body
      Connection Header
      Structure Length: 4
      Communication Channel ID: 73
      Sequence Counter: 18
      reserved: 0
      KNX cEMI: 1100bce000000902010081
```

```
0000  00 0e 8c 00 0f 71 00 1e  c2 15 3c 9e 08 00 45 00   .....q.. ..<...E.
0010  00 31 d6 11 00 00 40 11  1f 5f c0 a8 01 fd c0 a8   .1....@. ._......
0020  01 fe e7 a8 0e 57 00 1d  85 7a 06 10 04 20 00 15   .....W.. .z... ..
0030  04 49 12 00 11 00 bc e0  00 00 09 02 01 00 81      .I...... .......
```

Figure 5.11: Wireshark used as network sniffer.

In this case, in addition to the first testing phase, Wireshark [94] was also used in order to analyze the network traffic between the Bridge and the KNXnet/IP gateway. The figure 5.11 shows a UDP packet; it contains, as payload, the cEMI message for the KNX network. Analyzing it, it is possible to understand that it is a writing message for a group address and, in particular, it carries the command `light_on` (analyzing code 5.3).

Listing 5.3: Low level data analysis (figure 5.11) [56].

```
06 10 04 20 00 15 04 49 12 00 11 00 bc e0 00 00 09 02 01 00 81

Some information from a low level analysis

06 = Header Length
10 = KNXnet version (1.0)
04 20 = Service type descriptor (TUNNELLING_REQUEST)
...
09 02 = 0000 1001 0000 0010 = 00001/001/00000010 = 1/1/2
      1/1/2 = group address of "wall light" into room 1
      KNX group addresses are structured with main/middle/sub
         levels (5/3/8 bits respectively)
...
00 81 = light_on
```

Even in this second testing phase it was possible to save the execution time of each request and to represent it in the chart 5.12. It shows that, even if there are any cases with a value higher than the average execution time, most of the requests have a good response time. However, every value that represents the sum of the time used to send a UDP package (contained a KNX message) to the KNXnet/IP and to receive an acknowledgment message. Therefore, they do not represent the execution time used to complete a request into the KNX network. This depends on the fact that it is not possible to get access to this data in a real world.

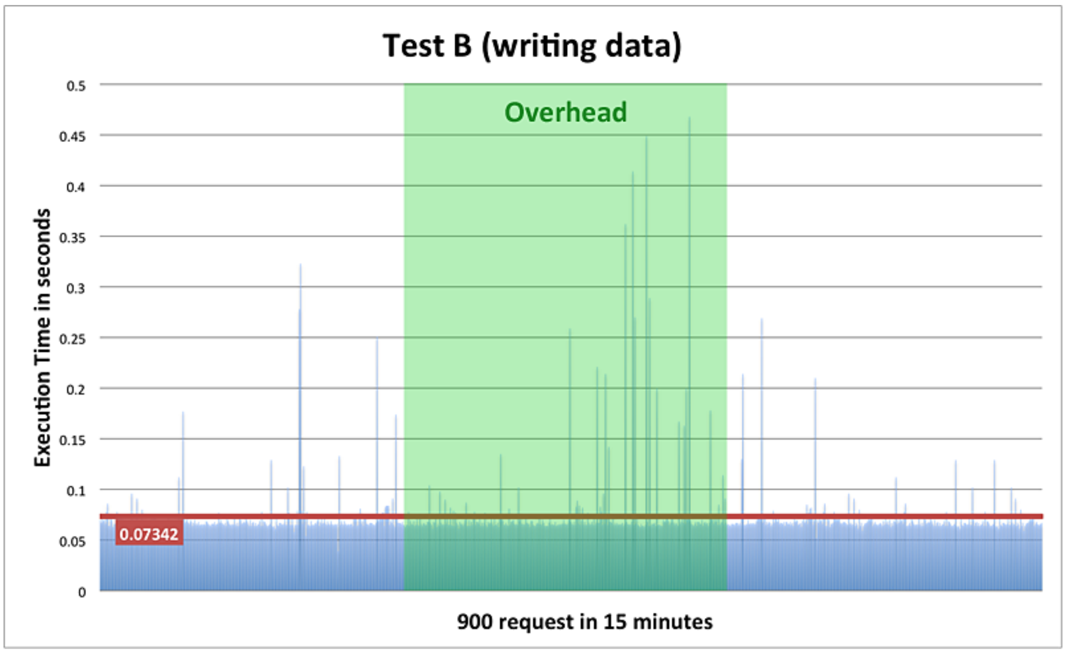


Figure 5.12: Testing chart with 900 requests in 15 minutes on an iPhone 3GS.

5.3 Getting configuration and setting values

In the sections 5.1 and 5.2, it was made a single type of action at time: writing or reading. In the third testing phase, it was created a combination between this two requests. The writing action, as in section 5.2, has been used to set a `boolean` value (`true` or `false`), in order to switch on or off a light. The reading action, instead, was represented by a reading status of the device interested by the writing request, as shown in the source code 5.4.

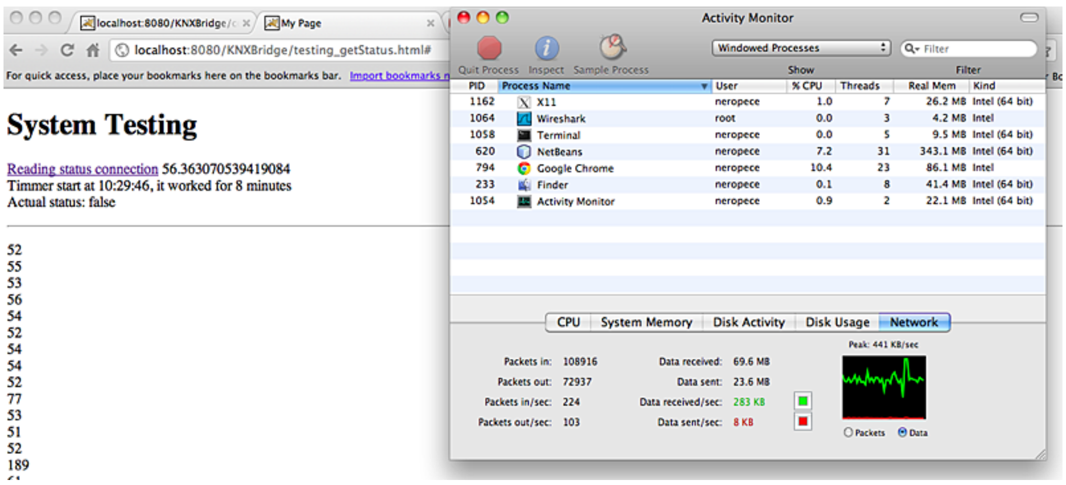
Listing 5.4: Ad-hoc page for getting device status.

```
window.setInterval(function() {
  start = new Date().getTime();
  $('#timeStart').html(timmer.getHours() + ':' + timmer.
      getMinutes() + ':' + timmer.getSeconds());
  $.getJSON(serverUrl + '/KNXBridge/connection.jsp?status&id=8&
      groupAddress=1/1/2')
  .complete(function(data) {
    value = (new Date().getTime()) - start;
    $('#stepexecutiontime').prepend(value + '<br>');
    avg = (avg * couter++ + value) / couter;
    $('#avg').html(avg);
    $('#timePass').html((new Date()).getMinutes() - timmer.
        getMinutes());
    $('#status').html(data.value);
  })
  .error(function(data) {
    $('#stepexecutiontime').prepend('<b>ERROR in writing</b>');
  });
}, 1000)
```

Even in this case, it was followed the same procedure realized for the first two testing phases: 5 minutes without overhead, 5 minutes with overhead and, again, 5 minutes without overhead (figures 5.13 and 5.14). All the data are reported in the chart 5.15; also in this case, most of the requests are completed in an execution time lower than the average and the communication system is not affected by the addition of noise sources such as streaming video.

5.4 Getting configuration, setting and reading values

This last test was conducted for the purpose of creating a combination of possible requests; it contains all the possible actions that can be performed with this system. As in all the other testing phases, KNX network, Bridge and KNXnet/IP gateway are subject to requests for 15 minutes. For the purpose of generating different types of interaction, three web pages created ad-hoc has been used (see sections 5.1 (source code 5.1), 5.2 (source code 5.2) and 5.3 (source code 5.4)). During the entire process, an iPhone 3GS was used to send writing requests, however in this case, to generate reading requests for a device's status and getting KNX configuration two browser tabs have been used



System Testing

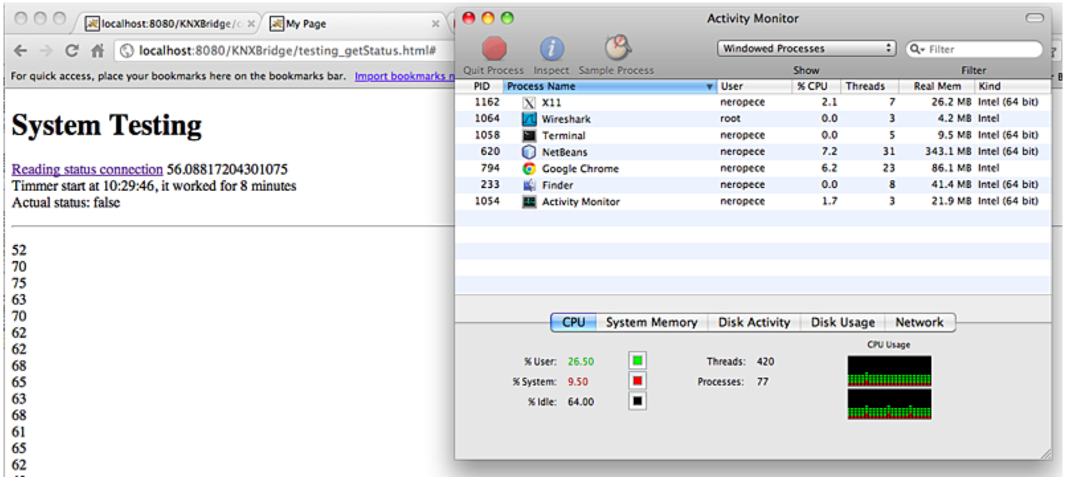
Reading status connection 56.363070539419084
Timmer start at 10:29:46, it worked for 8 minutes
Actual status: false

52
55
53
56
54
52
54
54
52
77
53
51
52
189

PID	Process Name	User	% CPU	Threads	Real Mem	Kind
1162	X11	neropece	1.0	7	26.2 MB	Intel (64 bit)
1064	Wireshark	root	0.0	3	4.2 MB	Intel
1058	Terminal	neropece	0.0	5	9.5 MB	Intel (64 bit)
620	NetBeans	neropece	7.2	31	343.1 MB	Intel (64 bit)
794	Google Chrome	neropece	10.4	23	86.1 MB	Intel
233	Finder	neropece	0.1	8	41.4 MB	Intel (64 bit)
1054	Activity Monitor	neropece	0.9	2	22.1 MB	Intel (64 bit)

Packets in:	108916	Data received:	69.6 MB
Packets out:	72937	Data sent:	23.6 MB
Packets in/sec:	224	Data received/sec:	283 KB 🟩
Packets out/sec:	103	Data sent/sec:	8 KB 🟥

Figure 5.13: Testing web page with network load during the first phase.



System Testing

Reading status connection 56.08817204301075
Timmer start at 10:29:46, it worked for 8 minutes
Actual status: false

52
70
75
63
70
62
62
68
65
63
68
61
65
62

PID	Process Name	User	% CPU	Threads	Real Mem	Kind
1162	X11	neropece	2.1	7	26.2 MB	Intel (64 bit)
1064	Wireshark	root	0.0	3	4.2 MB	Intel
1058	Terminal	neropece	0.0	5	9.5 MB	Intel (64 bit)
620	NetBeans	neropece	7.2	31	343.1 MB	Intel (64 bit)
794	Google Chrome	neropece	6.2	23	86.1 MB	Intel
233	Finder	neropece	0.0	8	41.4 MB	Intel (64 bit)
1054	Activity Monitor	neropece	1.7	3	21.9 MB	Intel (64 bit)

% User:	26.50	🟩	Threads:	420
% System:	9.50	🟥	Processes:	77
% Idle:	64.00	⬛		

Figure 5.14: Testing web page with network load during the second phase.

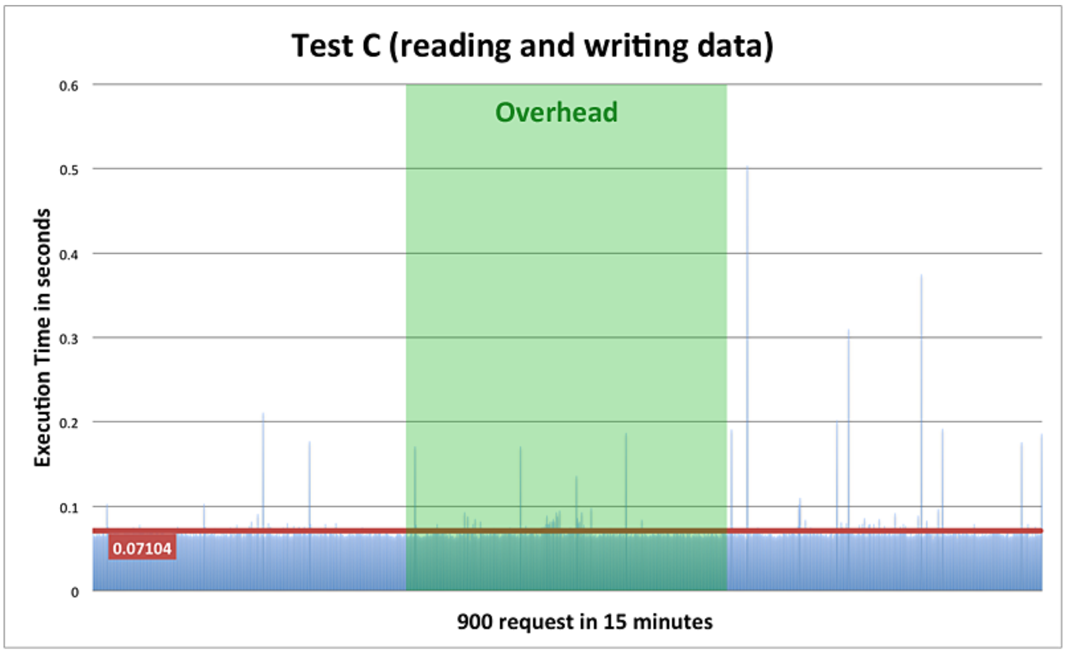


Figure 5.15: Testing chart with 900 requests in 15 minutes on an iPhone 3GS.

instead. All the components were connected together using a Wi-Fi network provided by an access point connected to the same local area network of the KNXnet/IP gateway and Bridge (figure 5.16).

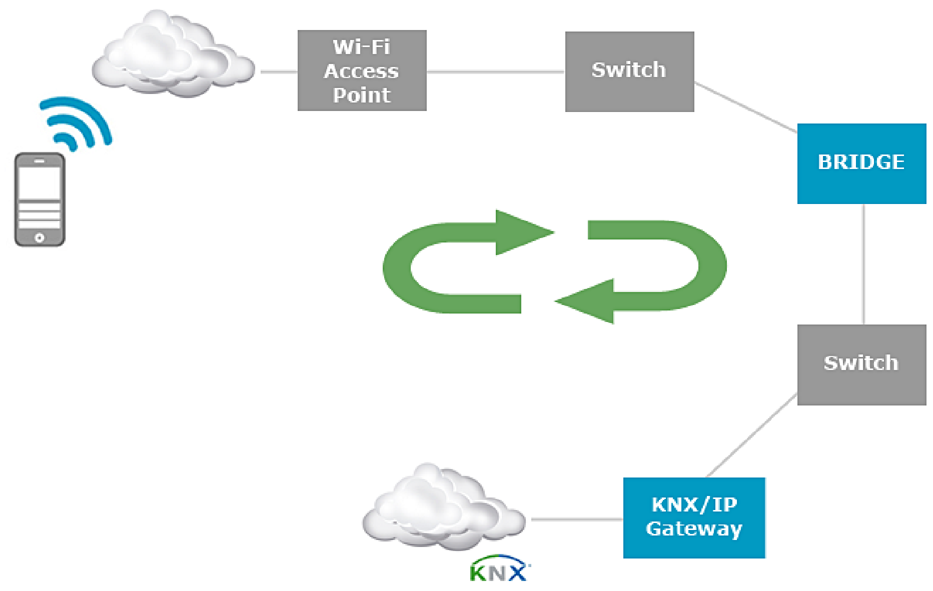
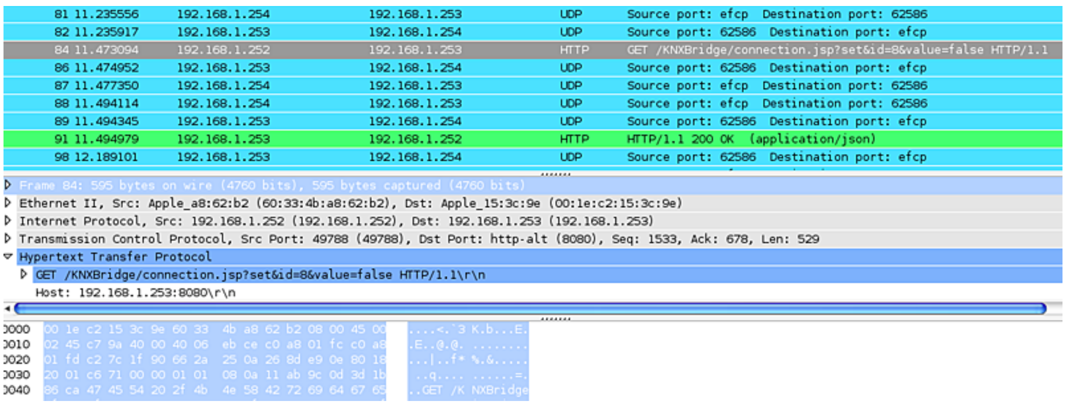


Figure 5.16: Network infrastructure.

The data collected for each device, used during this test, are represented in the following charts (figures 5.18, 5.19 and 5.20). Even in this case Wireshark was used as a network sniffer. In the figure 5.17 there is a screenshot where it is possible to observe a HTTP request sent from the iPhone 3GS (IP address `192.168.1.252`) to the Bridge (IP address `192.168.1.253`) and its relative JSON answer, which confirms the correct execution of the requested action.



```
 81 11.235556   192.168.1.254   192.168.1.253   UDP    Source port: efcp  Destination port: 62586
 82 11.235917   192.168.1.253   192.168.1.254   UDP    Source port: 62586  Destination port: efcp
 84 11.473094   192.168.1.252   192.168.1.253   HTTP   GET /KNXBridge/connection.jsp?set&id=8&value=false HTTP/1.1
 86 11.474952   192.168.1.253   192.168.1.254   UDP    Source port: 62586  Destination port: efcp
 87 11.477350   192.168.1.254   192.168.1.253   UDP    Source port: efcp  Destination port: 62586
 88 11.494114   192.168.1.254   192.168.1.253   UDP    Source port: efcp  Destination port: 62586
 89 11.494345   192.168.1.253   192.168.1.254   UDP    Source port: 62586  Destination port: efcp
 91 11.494979   192.168.1.253   192.168.1.252   HTTP   HTTP/1.1 200 OK  (application/json)
 98 12.189101   192.168.1.253   192.168.1.254   UDP    Source port: 62586  Destination port: efcp

▷ Frame 84: 595 bytes on wire (4760 bits), 595 bytes captured (4760 bits)
▷ Ethernet II, Src: Apple_a8:62:b2 (60:33:4b:a8:62:b2), Dst: Apple_15:3c:9e (00:1e:c2:15:3c:9e)
▷ Internet Protocol, Src: 192.168.1.252 (192.168.1.252), Dst: 192.168.1.253 (192.168.1.253)
▷ Transmission Control Protocol, Src Port: 49788 (49788), Dst Port: http-alt (8080), Seq: 1533, Ack: 678, Len: 529
▽ Hypertext Transfer Protocol
  ▷ GET /KNXBridge/connection.jsp?set&id=8&value=false HTTP/1.1\r\n
    Host: 192.168.1.253:8080\r\n

0000  00 1e c2 15 3c 9e 60 33  4b a8 62 b2 08 00 45 00   ....<.`3 K.b...E.
0010  02 45 c7 9a 40 00 40 06  eb ce c0 a8 01 fc c0 a8   .E..@.@. ........
0020  01 fd c2 7c 1f 90 66 2a  25 0a 26 8d e9 0e 80 18   ...|..f* %.&.....
0030  20 01 c6 71 00 00 01 01  08 0a 11 ab 9c 0d 3d 1b    ..q.... ......=.
0040  86 ca 47 45 54 20 2f 4b  4e 58 42 72 69 64 67 65   ..GET /K NXBridge
```

Figure 5.17: Wireshark screenshot with HTTP request/response.

Using the network infrastructure represented in figure 5.16, which might be a typical situation inside a small house (three mobile devices, a Bridge, a KNXnet/IP gateway and one switch), the results obtained, based on time execution, are very encouraging. The system responses well in every situation; in fact most of the execution times are under the average level, which is $max(0.08679, 0.05657, 0.07363) = 0.08679 < 0.1\ second$. It is also possible to observe that adding sources of noise, the communication system remains stable.

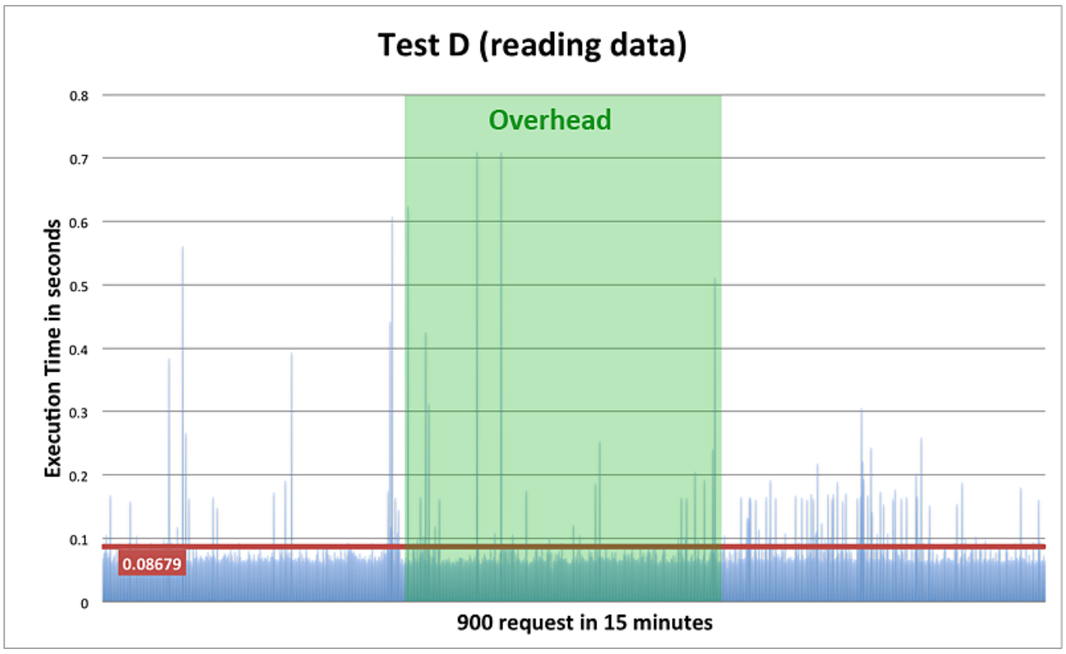


Figure 5.18: Testing chart with 900 requests in 15 minutes on a browser tab.

5.5 Results evaluation

The whole testing process focused on generating a new set of data based on real experimentation, thanks to the KNX and Wi-Fi network, provided by Northumbria University. Using the data obtained by the four testing phases, it is possible to observe that, independently from the entire structure of the mobile KNX communication system, the results are encouraging. The execution time of each component, subject to different activities, turns out to be acceptable. Using the average time of each test, it can be noticed that every request has a response time less than $max(all\ execution\ times) = 86.79\ milliseconds < 100\ milliseconds$. Therefore, thinking that for a normal user an exchange of data through a communication network, using an interface on a device, appears instantaneous if it receives a reply in less than 100 milliseconds [71]; the overall behavior of the communication system is acceptable. Thus, based on real data, without using any simulations, it is clear that a KNX remote controller based on a communication system provided by Wi-Fi network, even if with not only an access point in the middle, can be developed with the support of this technology. In conclusion, this analysis gives us a base on which to pursue this work and to seek answers to initial questions.

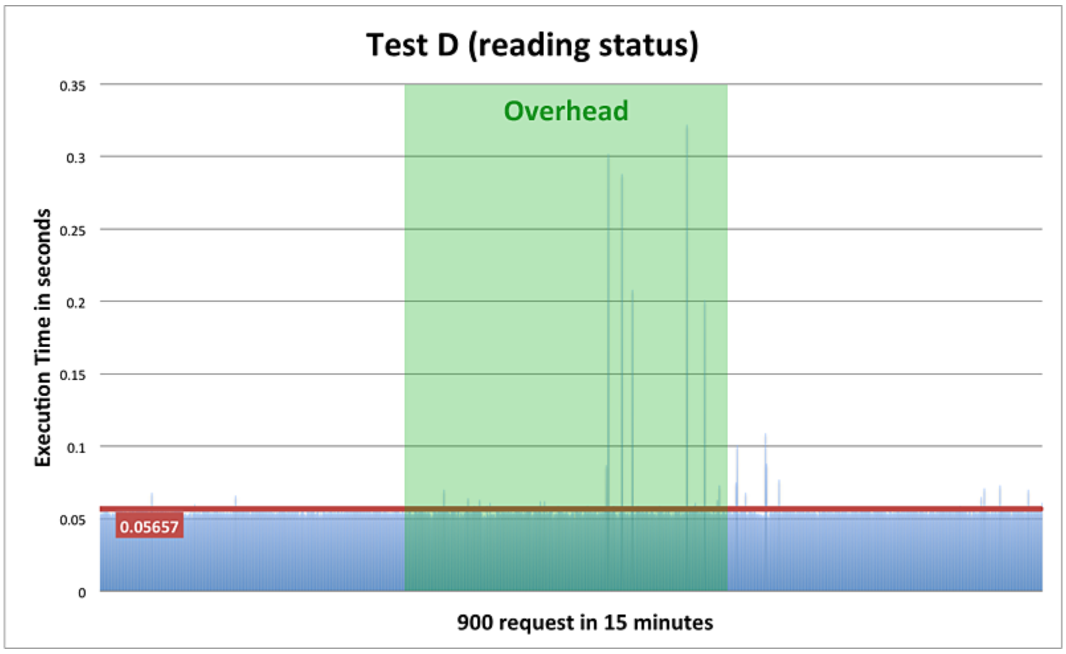


Figure 5.19: Testing chart with 900 requests in 15 minutes on a browser tab.

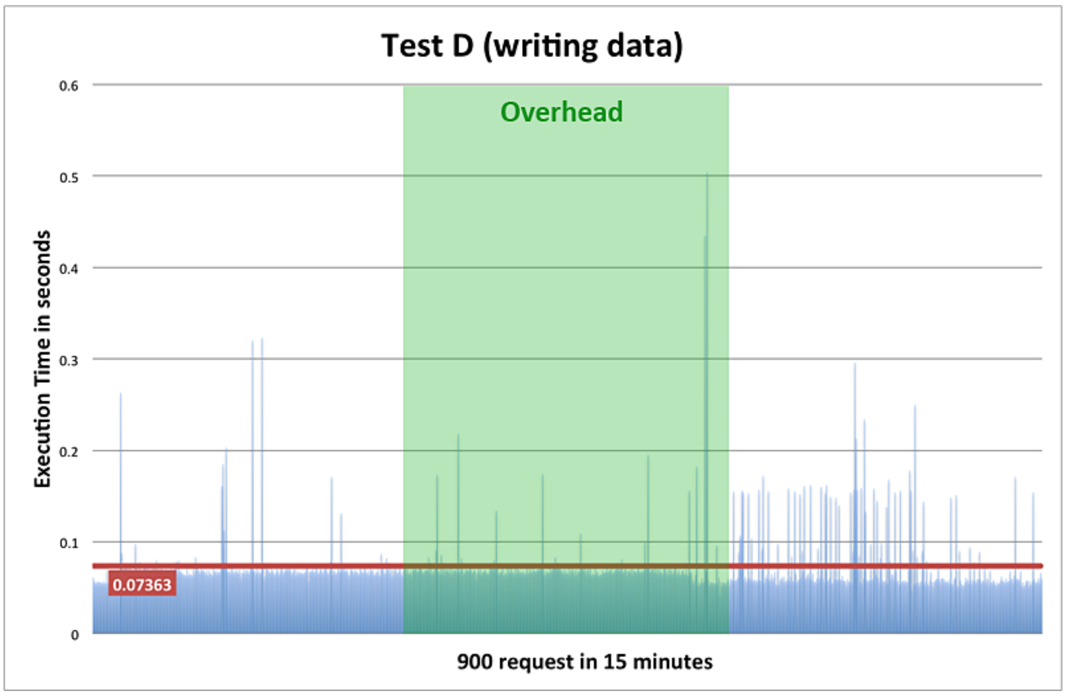


Figure 5.20: Testing chart with 900 requests in 15 minutes on an iPhone 3GS.

Chapter 6

Project evaluation and conclusions

The whole work and the final analysis were based on two main questions. We wanted to understand if the current remote management systems of a building automation technology were improvable. Furthermore, we wanted to know if the technology provided by Konnex Association is suitable for this aim.

After a first step of research, it was noticed that all the remote management systems, accessible using a mobile device, require a double configuration. First of all, once the basis for an automation building system has been obtained, it is necessary to configure it using one or more than one software. The laboratory provided by Northumbria University is equipped with a KNX network. Therefore, in this case, the first step is represented by a network configuration using ETS. This process has to be done for every type of infrastructure, such as LonWorks or BACnet. Thus, it is not possible to avoid this first configuration, which provides the basic specifications to every "smart" device.

The remote management systems currently present in the market, beyond this first phase, require also to configure a mobile device in order to control an automation building system. Therefore, the real challenge, to improve these systems, was to find and to realize an infrastructure capable of overcoming this limitation. As has been said, the first configuration step cannot be avoided, so our work has focused on this second process. Once the KNX network is configured, a planner can export the entire project from ETS. Therefore, there is a file that contains all the basic information about the configured network. The idea was to reuse this archive and to realize a system able to get it as input and automatically extracts its information. With the purpose of realizing this system, the main two ways to develop this idea were: develop an application able to receive and to analyze the exported file or implement a device able to get as input the archive and share the data in it contained with mobile devices.

The first system requires to maintain, in a well-known location the exported project, and to send it to all the mobile devices that want to get access to the KNX network. In this manner, it is necessary that a repository is always available and the application, installed into a device, has to process the archive in order to extract all the information. However, this solution requires an additional workload to each mobile device and thus the same process (extraction), with the same data, has to be done more than one time. The second possibility, instead, requires a repository as in the first case, but it can be used to handle the exported file. Therefore, using this system, the processing work is centralized and it is completed only one time even if there are several different mobile devices. Once the data are extracted, the central application may share them by a communication network. This means that each "client" only has to render the information received. During our research, it was considered also another possibility: creating a listener on the mobile device and, once discovered a smart device, permit a client to create their own project based on the data received. However, also in this case, a planner has to design an automation building network and the client must configure each device. Therefore, we would not add a sufficient automation.

Based on these ideas, it was decided to follow the second possibility. Therefore, a second research phase was made to find an interface system, which permits to exchange messages between a dedicated workstation and the KNX network. During our analysis, we decided to use Calimero, which provides a set of functions that permit a developer to control a KNX network developing application based on Java technology. We decided to use this system because it is based on Java; thus it is possible to export the application developed to several different architectures and to make the system more flexible.

Once developed the connection bridge between the KNX network and an ad-hoc device, it was necessary to realize an interface system in order to permit to a mobile device to get access to the automation building infrastructure. Thinking about the type of communication system used to connect the mobile world to the bridge, that we implemented, the best solution was to design a communication infrastructure based on HTTP requests and replies. Therefore, we had to merge an application based on Java technology and a "web world" and so it was decided to use Apache Tomcat. This web server permits to mix Java application with web pages. At the end, an application that can exchange messages, based on TCP/IP technology, with a KNX network using a KNXnet/IP gateway and with mobile devices using HTTP requests and replies was developed.

The last step of our development was dedicated to create a mobile application, which can read the information received from the Bridge. There are several mobile operating systems and architectures; for this reason we decided to find a system to make our application suitable for most of them and so to make it really flexible. Therefore, PhoneGap was chosen as a framework because it permits a developer to implement an application, based on HTML/-JavaScript/CSS technologies, for one type of architecture and operating system and export it to all the others. This development phase was realized using an iPhone 3GS and a MacBook, only because they were available and not for other reasons.

Once the whole system was completed, the next step was to test it and to give an answer to our questions. From the testing phase, we have observed that the infrastructure responds fine as execution time and it is really easy to connect a new mobile device to the network. Therefore, it is possible to say that our applications might improve the current remote management systems, even if there are some points of weakness. First of all, it is still necessary to upload the exported file to a central device, the Bridge. However, without having access to the memory and computing system of the KNXnet/IP gateway, it is not possible to merge these two devices. The second point of weakness is introduced right from the Bridge; in fact if it "falls down", it is not possible to get access to the KNX network. These are the main problems of this project, however it is still possible to fix them due to future developments.

The second question required an understanding of whether the technology provided by Konnex Association is suitable for a residential environment. Based on the articles used during this work and the experience gained in the field, it is possible to say that KNX is a good choice for an automation building system in a residential environment. ETS is a configuration tool easy to use and it is possible for a normal client to use it without any special training. KNX technology compared to other systems appears to be cheaper and it should be easy to find a reseller, especially in Europe. In addition, we had no problems with it during all the development and testing processes. Furthermore, the testing steps showed that even if a KNXnet/IP gateway is considered a bottleneck [18] [17] [19], in reality, its presence brings more benefits than problems. The main advantage is represented by the possibility to communicate with a KNX network using standard, flexible and well-defined technologies, such as Ethernet and TCP/IP.

In conclusion, the infrastructure implemented may really make improvements to the current remote management systems of a building automation technology. It is able to get information about a KNX network and to automatically share it with several different mobile devices. In addition, a normal client only has to know an IP address (Bridge IP address), and once it has obtained, it is immediately possible to manage the entire KNX network with a simple mobile device without any knowledge of technologies behind it. This means that it is a system really easy-to-use and it meets all the basic necessities of a normal user.

6.1 Future work

There are many other functions to be developed. First of all, the system should be protected using an authentication system. Furthermore, once the exported file is uploaded to the Bridge, the project structure, rendered by the mobile device, cannot be modified. Therefore, a system to permit a client to realize an own layout, for example creating new rooms and virtually assigning the devices to them, should be provided. Another possible feature is to develop auto discovery KNXnet/IP gateways into the Bridge; in this manner it is possible to automatically locate and provide access to more than one automation building system. However, the entire system works fine even if it still provides basic functions, but this was only the first step.

Appendix A

Declariation

I declare the following:

1. that the material contained in this dissertation is the end result of my own work and that due acknowledgement has been given in the bibliography and references to ALL sources be they printed, electronic or personal.
2. the Word Count of this Dissertation is 14923
3. that unless this dissertation has been confirmed as confidential, I agree to an entire electronic copy or sections of the dissertation to being placed on the eLearning Portal (Blackboard), if deemed appropriate, to allow future students the opportunity to see examples of past dissertations. I understand that if displayed on eLearning Portal it would be made available for no longer than five years and that students would be able to print off copies or download.
4. I agree to my dissertation being submitted to a plagiarism detection service, where it will be stored in a database and compared against work submitted from this or any other School or from other institutions using the service. In the event of the service detecting a high degree of similarity between content within the service this will be reported back to my supervisor and second marker, who may decide to undertake further investigation that may ultimately lead to disciplinary actions, should instances of plagiarism be detected.
5. I have read the Northumbria University/CEIS Policy Statement on Ethics in Research and Consultancy and I confirm that ethical issues have been considered, evaluated and appropriately addressed in this research.

Appendix B

Research Proposal

Research Proposal

Aim

Design and implementation of a new system for remote control of a smart house using an infrastructure based on KNX bus line technology over a wi-fi connection.

Background

In the last years, the continuous increase of energy consumption and research of new markets have permitted the emergence of new concepts such as green home and building automation systems. [22] [25] [78] The word *green home* is used to describe a type of building designed to be environmentally friendly and sustainable, focused on efficient use of energy, water, and building materials. This concept implies buying more energy-efficient and smart appliances and using specific building materials, for example, components more efficient in keeping both cool and heated air inside the structure. *Building automation systems (BAS)* describes functionality provided by a computerized control system, installed into a building, using an intelligent network of electronic devices designed to monitor and control mechanical and electrical systems. [47] [76] Each of these concepts offers great possibilities, but their combination provides a new efficient and completely managed building system controllable using a mobile device hundreds of kilometers away. Idea reached using innovative technologies, which permit to establish a connection between appliances in a building and an *application system core*, device that may communicate with a mobile device using well-established technologies such as GPRS service [26]. In order to reach this aim, the first step is to find a system of interaction between an appliance or, for example, a simple bulb and a mobile device. To address this first problem a technology developed by *Konnex Association (KNX)* [55] (see appendix D) can be used. It provides an international communication open standard for home & building electronic systems, designed to be independent of any particular hardware platform and based on *European Installation Bus (EIB)* [24] (see appendix E). EIB was used as communication stack, enlarged with physical layers, configuration modes and application experience obtained by merging *BatiBUS* and *European Home System (EHS)*.

Thanks to new and innovative devices and the work done by companies such as Apple, Google, etc. it is now possible to use the last generation mobile devices as complete and powerful computer. Therefore, it is possible to develop complex applications and take advantage of the wi-fi connection, as well as, GPRS service to create a direct connection between a mobile device, such as iPhone or a BlackBerry, and another component such as a dedicated computer. It is also possible to sell own applications around the world, using special repository, in this way reach thousands of potential customers.

The attempt to merge Home & Building Automation with mobile technology and to make the product really useful for the customer presents a problem related to configuration. Requiring a customer to configure every connection between all their mobile devices and the appliances creates a friction to the adoption of this innovative infrastructure. Based on this, the main idea of this project is to realize a *bridge* between the mobile and home automation world and therefore develop a melting point that can interface with both systems. This point of interconnection will be used to share the configuration rules when a new device requires access to the home automation system.

The project, after an analysis of the state of the art, will aim to merge Home & Building Automation with mobile technology with a particolar emphasis on automation, in order to create a new, useful and easy-to-use product for the customer. Features provided by *KNX infrastructure* and mobile developing frameworks [2] will be used to develop an application suitable for all Apple devices. A dedicated station equipped with *GNU Linux* [77] as operating system will be used to realize the interconnection point. It provides an open, stable, flexible and highly configurable operating system that can run with few resources; in order to create a melting point with an ultra-low energetic impact on the energy consumption of the building. The use of an additional device gives the opportunity to expand the areas of development. Possible future developments might be the creation of a product, stylistically attractive and dynamically interfaced with a mobile device, that can reproduce music and manage the lights based on the type of music or read upon request e-mails, weather forecasts or financial news as well.

Objective

1. Analysis the state of the art in Home Automation and Mobile Platform technologies.

2. Explore the KNX technology, evaluating the alternatives as well, and identify the best solution for the proposed project.

3. Explore the PhoneGap framework and wi-fi connection in order to identify the best system to establish a connection between a mobile device

and a remote dedicated computer on a wi-fi connection.

4. Design and create KNX network using KNX ETS4.
5. Design and develop a communication system between KNX network and the merger point.
6. Design and develop a communication system between iPhone/iPad and the merger point.
7. Design and develop a "software bridge" to interface the KNX network with wi-fi network.
8. Analysis and Testing the infrastructure.

Appendix C

Mobile Management System Analysis

"*A smartphone is a new form of mobile internet device that combines the traditional features of a phone and a PDA*" [49]. Another noteworthy definition of a smartphone describes it as a "*mobile phone that offers more advanced computing ability and connectivity than a basic current mobile phone*" [64]. Therefore, a smartphone is described as an mobile device, which provide the mobile telephone technology and, moreover, the possibility to access with it to Internet or in general to a data network. Thus, a smartphone provides the same functionality of a traditional phone and a micro computer (figure C.1). In addition, not as a traditional phone, which is a final goods and, thus without possible evolutions once in the market, a smartphone allows to a user to install and delete hundreds of applications improving its user experience. Since smartphone allows access to Internet, users of this type of mobile phone are creating an era of ubiquitous information.

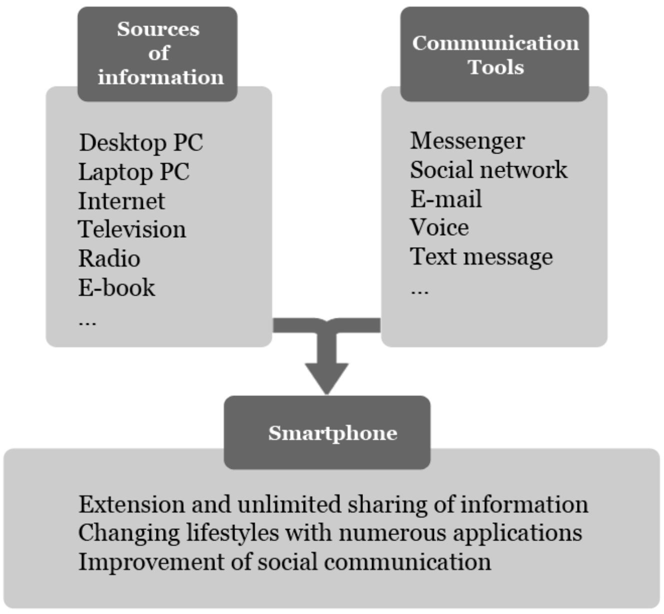


Figure C.1: The concept of the smartphone

Thanks to research and development of companies, the new smartphones are more powerful. They provides more processing power, storage capabilities and several communication mechanisms and multimedia facilities. A significant increase in mobile users leads to expectation of more commerce mobile applications with greater improved features [66]. It was predicted by Adams [15], the smartphone was likely to become a "personal trust device" (PTD) and a powerful electronic wallet.

The continuous improvement in user experience and the massive increase in interaction systems allow people to have access to enormous amount of information. Users can get access to almost any type of information, such as information on the weather, locations, food, attraction, etc. without constraints of time and place. This new mobile generation has strong access to information and networking capabilities so the producer-driven consumption pattern is rapidly being replaced by a consumer-based consumption pattern. Accordingly, the need for customized information is expected to keep pace with change in the IT environment [75]. However, at the beginning, the smartphone world failed because it is based on the concept of "always connected" and, obviously, "easy to use". Therefore, it tried to overcome slow and expensive carrier data network and also to go beyond a simple and poor user interfaces, browsers, etc.. In 2007, iPhone changed everything. A new proliferation of mobile and multimedia device started, Apple was able to convince carriers to offer flat-rate data plans to consumers, importing the Wi-Fi technology, into the mobile world, and a complete mobile browser.

Trying to understand the impact that this new type of device had in the last years, it is sufficient to examine the next graphs.

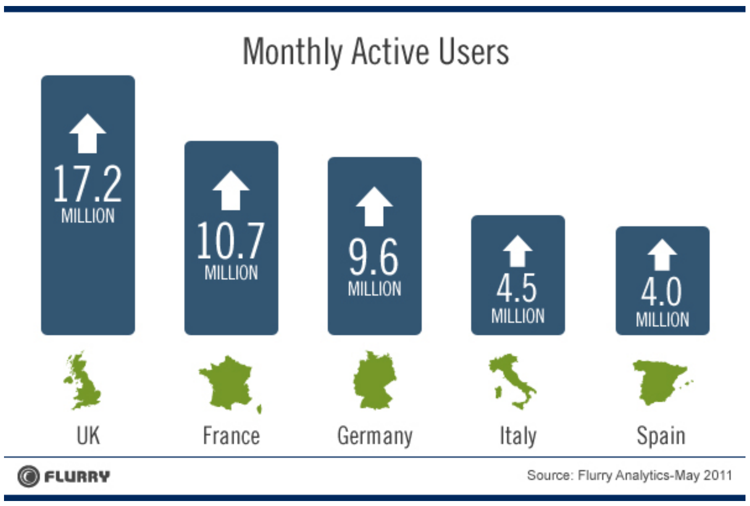


Figure C.2: People actively using apps on their smartphones in May 2011

The figure C.2 shows the number of people actively using applications on their smartphones in May 2011 across the top five European markets. Combined, the top five European markets (UK, France, Germany, Italy and Spain) actively use apps on 46 million devices each month. With a combined population of just over 240 million, for ages 13 and over, the addressable market through smartphone apps averages approximately 20% of the largest, most affluent European countries. Additionally, with a month-over-month growth rate (Compounded Monthly Growth Rate, CMGR) of more than 10% over the last two years, market share of smartphones will increase, over the next 12 months, more than double.

The figure C.3 shows the smartphone app audience as a percentage of each top European country's population, again ages 13 and over. The UK leads in penetration with a whopping 33% of its population using apps on smartphones per month. France places second with a sizable 20%, next followed by Germany, Spain and Italy coming in with 14%, 9% and 10%, respectively.

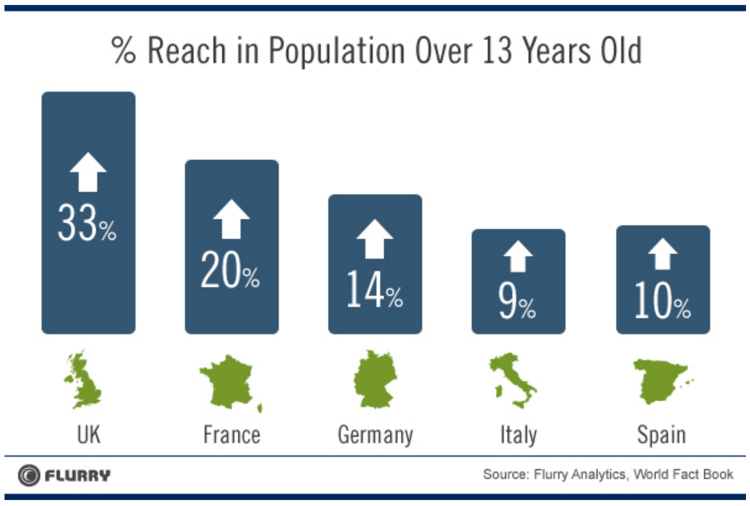


Figure C.3: Smartphone app audience as a percentage of each top European country's population

Based on data showed in the last chart, using the figure C.4, is interesting to understand the pace of smartphone adoption in these countries.

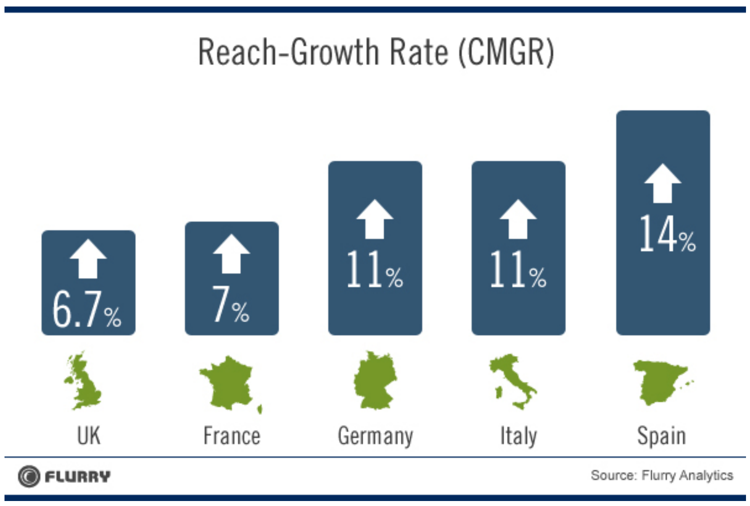


Figure C.4: Pace of smartphone adoption by country

With growing adoption of iOS and Android-based smartphones, the imminent release of Nokia phones based on Windows Phone 7, and the fact that the majority of consumers actively use applications, the growth of this mass market media channel will continue until near total smartphone market saturation.

To underscore just how aggressively this channel is growing, if the growth of smartphone adoption continued at their current rates, all five countries would have full smartphone penetration in just over two years. [30]

It is interesting to compare the smartphones penetration in Europe with all the other continents (figure C.5). The comparison shows how the adoption of this device, even if with different intensity, is present in all over the world. It permits to have a global point of view in order to understand why, in the last years, embedded systems and relatives operation systems/mobile applications have become the major source of income for ICT companies.

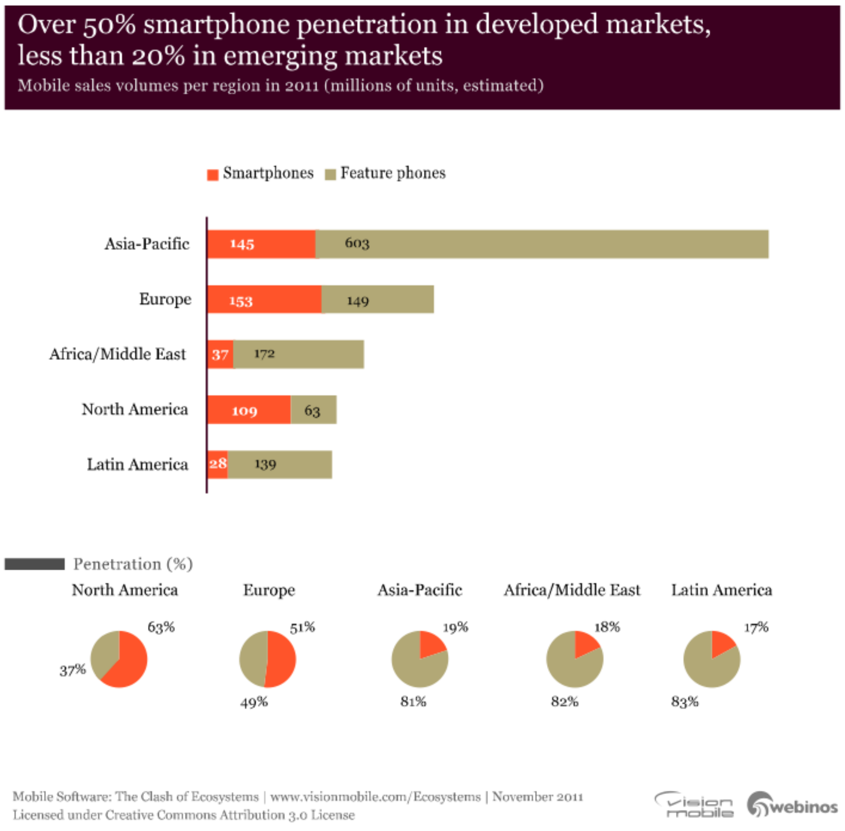


Figure C.5: Global smartphone penetration (millions of units).

Since when Apple released the iPhone, Apple was the benchmark for all the smartphone device specs, user experience, application ecosystem, browser capabilities and content. Therefore, the currently leader in the mobile market, Nokia, was too slow to react to Apple's application-led and developer-led strategy. However, the gap between these two companies was quickly filled by a large group of mobile device and software makers, such as the Google, Samsung, HTC, etc.. Today, the two main companies that pull the market are Apple, with its iOS and mobile devices, and Google, with Android which was adopted by several companies as operating system. [30]

The reason why Apple and Google are able to pull alone the smartphone market is the increasing significance of the operating system. This element of a mobile device represents the point of connection of every features provided by the device where it is installed. It controls all the hardware functions with total access to input, output, memory and it also provided the indispensable layer between hardware and applications. As a matter of fact, handset sales are driven not by hardware features (“what the handset can do”) but the user interface and applications available (“what you can do with the handset”). iOS and Android are leading the market with over 500,000 and 300,000 applications (October 2011), respectively, which offer incredible user experience and a wide range of functionality. [81]

The leadership of Google and Apple forced companies as Microsoft, Facebook and other mobile operators to find a manner to reduce the power of Apple’s iOS and Google’s Android platforms. HTML 5 had the potential to become a common bridge system across smartphone platform. It is the only common app technology supported by Android, iOS, new versions of BlackBerry OS and Windows Phone platforms. [92] [82] [29]

Appendix D

Konnex Association Technology

The demand for building control systems is continuously increasing (figure D.1). In a single-family house or in an office complex, the demand for comfort and versatility in the management of air-conditioning, lighting and access control systems is growing but, at the same time, energy efficiency is becoming increasingly important. Convenience and safety, coupled with lower energy consumption, can be achieved only using intelligent system for controlling and monitoring of all products involved. However, this implies more wiring, running from the sensors and actuators to the control and monitoring centers, with a consequent higher design and installation effort, possible increase fire risk and soaring costs.

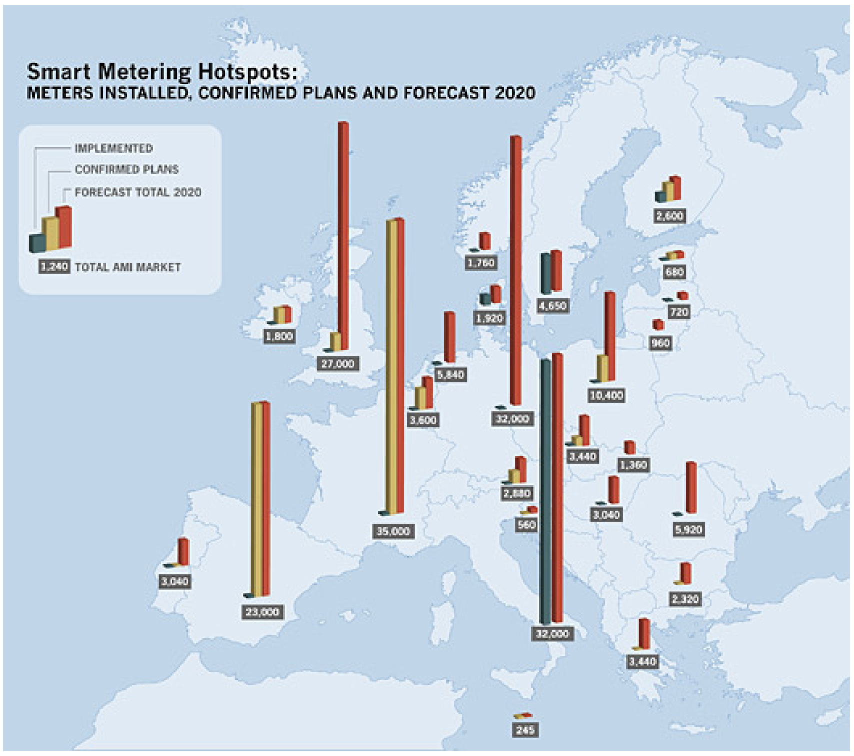


Figure D.1: Control Systems installation forecasts

In order to transfer control data to all building management components, without an excessive use of wiring and problems due to isolated devices, it is necessary a systems that ensure a common communication language. Thus, a system where manufacturers and application domains are independent. KNX developed this system thanks to twenty years of experience in the market, working on others predecessor systems such as EIB, EHS[1] [39] and BatiBUS[2] [20]. Using the KNX medium (Twisted Pair, Radio Frequency, Power Line or IP/Ethernet) all the bus devices are connected together, in this manner they are able to exchange information; moreover, they can either be sensors or actuators, which are needed for control the building management equipment (such as lighting, blinds/shutters, security systems, energy management, heating, ventilation and air-conditioning systems, signaling and monitoring systems, interfaces to service and building control systems, remote control, metering, audio/video control, white goods, etc).

Konnex Association

In May 1999, members of the following associations founded *Konnex Association*:

- BatiBUS Club International (BCI)
- European Installation Bus Association (EIBA)
- European Home Systems Association (EHSA)

The main objective of this association is to promote a new and commonly-defined "one single standard" for fieldbus applications in homes and buildings. This standard (KNX) is based on the well-established technology EIB, which was enlarged with configuration mechanisms and physical media obtained by the experience of BatiBUS and EHS.

[1]An open communication protocol dedicated to Home Automation is European Home Systems (EHS). EHS specifications have been defined in order to provide a communication and share system between home appliances. The EHS communication model follows the structure of the Open Standard Interconnection (OSI) reference model. EHS specifies the physical layer, the data link layer, the network layer and the application layer. Due to the fact that the message length is limited, the dialogue session is short and the application layer manages the command language, the transport, session instead the presentation layer has been omitted. The application layer translates the application language in data frame able to move over the network. The Data Link Layers, separated in two sub-layers Medium Access Control (MAC) and Logical Link Control (LLC), handles the bit stream conversion, the rules for accessing the network, the recognition of frames and provides acknowledgement and repetition mechanism. The network layer is identical for all the EHS units and manages data related to the route (for example the addresses to reach a unit over several sub-layers).

[2]Original standard proposed by Merlin Gerin [74]. A relatively simple low cost protocol not relying on dedicated chips. Connection structure allows any arrangement of cables making it extremely flexible in matching building requirements. A large number of devices may be connected to the network using simple guidelines to maintain reliability of exchange. The same guidelines ensure that the time taken for messages to arrive at their destination remains imperceptible to the user. The media used is a twisted pair cable with a 15 V line voltage and addresses are assigned as part of the commissioning process. Predominately used in France and described in the French Standard NFC 46620, it provides layers 1, 2 and 7 of the OSI Model [36].

Konnex Association is an international non-profit organization governed by belgium law. It is subdivided into various components. The KNX General Assembly (the highest legal authority of the association), which has al least one meeting per year to approve the activities undertaken and the budget for the coming year. The KNX Executive Board (KEB), elected amongst the members of the General Assembly, is responsible for the association's strategy, its standard and the budget proposal. The KEB is assisted by two permanent boards:

- KNX Technical Board (KTB), which coordinates all the activities regarding the development of the common standard KNX and the procedures for the certification of new KNX products.

- KNX Marketing Board (KMB), which coordinates the communication and promotional activities of Konnex Association around the KNX standard and all the activities undertaken in the different markets by the national Konnex groups.

For the daily activities the KEB appoints a team of directors, each of them is responsible for a set of activities:

- Administration and Certification

- Marketing and Communication

The objectives

The association's targets are oriented towards the development and promotion of an international communication standard for Home and Building Electronic Systems (HBES) by:

- developing, through studies and exchange of information, a single, stable and affordable system, which may improve the overall market acceptance and expansion of the KNX technology into residential and business market;

- encapsulating in one common standard, as a base for future evolution, the existing Home and Building Electronic Systems;

- defining and improving the specification of KNX protocol, adding different media, configuration modes, communication models etc.;

- standardizing system requirements;

- managing the relevant system intellectual property rights;

- setting-up an appropriate certification system to enable certification of products (hardware and software) and services in order to guarantee compatibility;

- introducing this standard into the appropriate international HBES standardization systems and actively promoting it as norm;
- managing an appropriate software tool system;
- managing an appropriate training system for professional users such as contractors, planners and installers.

KNX international standard

Standards and certified products are an important asset for the association and customers. These properties ensure that same product made by various manufacturers provides the expected features. In line with the common policy of the three partners (BatiBUS, EIBA and EHSA), KNX standard was designed, since the beginning, with the purpose of providing interest of customers and also ensuring a broad market for the members.

Joining three forces (BatiBUS, EIBA and EHSA) into one framework permitted to Konnex Association, on the 5th June 2000, to conclude a “Cooperation Agreement” with CENELEC[3] [16]. Therefore, Konnex Association, constituted by manufacturers, service providers and interested parties, achieved a privileged role within the framework of European standardization. Thus, Konnex Association may directly supply to the European standardization the requirements requested by all the groups involved, by the economic process, in the HBES field such as consumers, service providers and industry.

Konnex Association had in the meanwhile submitted the KNX protocol followed by Twisted Pair and Power Line media as a basis for furthering the EN 50090 series. In June 2003, the national committees voted positively, during the Unique Acceptance Procedure, on the following standard parts:

- EN 50090-3-2 corresponding to the KNX Application Interface Layer
- EN 50090-4-1 corresponding to the KNX Application Layer
- EN 50090-4-2 corresponding to the KNX Network, Transport and Link Layer (general part)
- EN 50090-7-1 corresponding to the KNX Management Procedures
- EN 50090-5-2 corresponding to he KNX TP medium The following parts are about to be voted or are planned:
- EN 50090-3-x corresponding to the KNX Interworking Model (planned)
- EN 50090-5-5 corresponding to the KNX Radio Frequency Medium (planned)
- EN 50090-8-x corresponding to the KNX Application Descriptions (planned)
- EN 50090-5-1 corresponding to the KNX Powerline Medium (in voting)

[3]European Committee for Electrotechnical Standardization (http://www.cenelec.eu/)

In December 2003, the CENELEC Bureau Technique ratified the positively voted EN 50090 parts. In this manner, the KNX specifications became the first European Standard for intelligent Homes and Buildings.

Why KNX?

The needs of customers change constantly, they require independence, compatible system solutions with a future, high convenience, low power consumption and great reliability. KNX standard can provide several interesting advantages based on these requirements:

Interoperability: it ensures that products of different manufactures can operate and communicate with each other. This permits a high level of flexibility such as future extensions.

Product quality: Konnex Association requires a high level of quality control during all stages of the product life. Therefore, all Konnex members branding own developed KNX products with the KNX Trademark if they fulfill the constraints imposed by standard ISO 9001 [37]. Besides compliance of the manufacturer to ISO 9001, the products have to comply with the requirements of the European standard for home and building electronic systems. However, Konnex Association is entitled to require from the manufacturer test reports underlying his declaration of hardware conformity.

Manufacturer independent functionalities: the KNX standard contains application profiles for many common applications in home and buildings. Under the Technical Board's supervision, several application specification workgroups make proposals for standardization of functionality (inputs, outputs, diagnostic data and parameters) in their specific application domain. To ensure a high level of cross-discipline and multi-vendor interoperability, the Task Force Interworking re-evaluate these proposals before incorporating a new application profile into the KNX standard.

Others: KNX can be developed on any type of hardware/software technology platform, even if off-the-shelf solutions are available and IPR[4] of fellow-members as contained in the KNX specifications is free-of-charge for KNX certified products

The KNX standard allows to each manufacturer a free choice between configuration mode and communication medium during the development of a KNX device.

Communication Media

KNX standard includes several communication media. Each of them may be used in combination with one or more configuration modes, allowing each

[4]Intellectual Property Rights

manufacturer to choose the right combination for the targeted market segment and application.

TP-0, (Twisted pair, type 0): this communication medium, twisted pair, bit rate 4800 bits/s, has been taken over from BatiBUS. The KNX TP0 certified products operate on the same busline as the BatiBUS certified components but they are *not be able to exchange information* amongst each other.

TP-1, (Twisted pair, type 1): this communication medium, twisted pair, bit rate 9600 bits/s, has been taken over from EIB. The EIB and the KNX certified TP1 products operate and communicate with each other on the same busline.

PL-110, (Power-line, 110 kHz): this communication medium, power line, bit rate 1200 bits/s, has also been taken over from EIB. The EIB and the KNX PL110 products operate and communicate with each other on the same electrical distribution network.

PL-132, (Power-line, 132 kHz): this communication medium, power line, bit rate 2400 bits/s, has been taken over from EHS, where it is still used. KNX PL132 certified components and EHS certified products operate together on the same electrical distribution network but they do *not communicate* with each other without a dedicated protocol converter.

RF, (Radio frequency on 868 MHz): this communication medium, radio frequency with a bit rate of 38.4 kbits/s, has been developed directly within the framework of the KNX standard.

Ethernet: this widespread communication medium can be used in conjunction with the "KNX over IP" specifications, which allow the tunneling of KNX frames encapsulated in IP frames.

The KNX device network results from the formal merger of the three leading systems for home and building automation (EIB, EHS, BatiBus) into the specification of the new Konnex Association. The common specification of the KNX system provides, besides powerful runtime characteristics, an enhanced toolkit of services and mechanisms for network management.

On the Konnex device network, all the devices are represented with standardized *data-point types* and *functional block objects* in order to form a distributed applications. Therefore, KNX system is entirely independent by each specific microprocessor platform or even architecture. A manufacturer can select any suitable industry-standard chip or opt for available KNX OEM[5] solutions such as Bus Coupling Units, BIM's, chip sets etc. Thus, KNX device networks may be flexibly adapted to present an optimal solution for each application domain and installation. Members of Konnex Association argue

[5]Original Equipment Manufacturer: products or components that are purchased by a company and retailed under that purchasing company's brand name. OEM refers to the company that originally manufactured the product.

that the Home and Building market requires open, flexible and interoperable solutions in the communication between controllers, actuators and sensors for standard applications on field bus level. The KNX standard corresponds to these needs. Its different configuration modes in combination with its different communication media make KNX the rightful field bus choice for home and building applications.

KNX Metering

Using intelligent energy meters, electricity consumers could control energy consumption to cope the rising cost of energy. KNX provide the possibility to monitor, in more detail, the energy usage but also heat, water, fossil fuels and gas consumption.

To provide more selective energy consumption patterns to the customers, it is necessary to realize systems able to monitor the ongoing energy consumption. As the fuel consumption indicators in a car, a “smart metering” provides the same system in a buildings through intelligent metering and displaying of the energy consumption. In this manner, the customer can choose economic choices, such as turning off appliances or shifting uses to cheaper tariff time zones. In the attempt to introduce smart metering, the ROI[6] or cost neutrality is of great importance. The investment is offset by increases in efficiency through on-line meter reading, billing and by cost reductions in energy consumption.

Smart homes and buildings may use KNX as controller infrastructure, combining communication media such as KNX Twisted Pair (TP) or Radio Frequency (RF). In the last years, KNX RF has become the most used medium for linking the metering applications indeed, starting with the first design concepts of the KNX RF communication medium. KNX Association worked with CEN to define, based on the M-Bus specifications, the parameters of the KNX RF Physical Layer (868 MHz-standard CEPT/ERC 70-03) and the Data Link Layer (based on the FT3-protocol IEC870-5-2).

The M-Bus is a European standard (EN 13757-2 physical and link layer, EN 13757-3 application layer [60]) for remote reading of gas and electrical meters. The M-Bus was developed to fill the need of a system to access to a remote reading of meters, for example measuring the consumption of gas or water in a house

M-Bus RF metering devices may be spread all over the building each equipped with several M-Bus as well as KNX RF devices. In such buildings, one common KNX TP network is often available or may be installed and in this way metering data is easily captured through a minimum number of couplers. To achieve this was modeled a Metering Data Collector, to be hosted in the single RF-to-wired KNX coupler referred to above. The Metering Data Collector maps a limited and well-defined subset of M-Bus metering data to a structure, KNX compliant data interface (KNX Interface Objects), in order to

[6]Return On Investment

make them accessible to the gateway and provide access to the most important data on energy consumption such as current values, minimum, maximum and average. Once on KNX, the metering data may be readily transported over KNX TP/IP and made available to an operator or service provider locally or remotely (e.g. through Internet), as shown in the figure D.2.

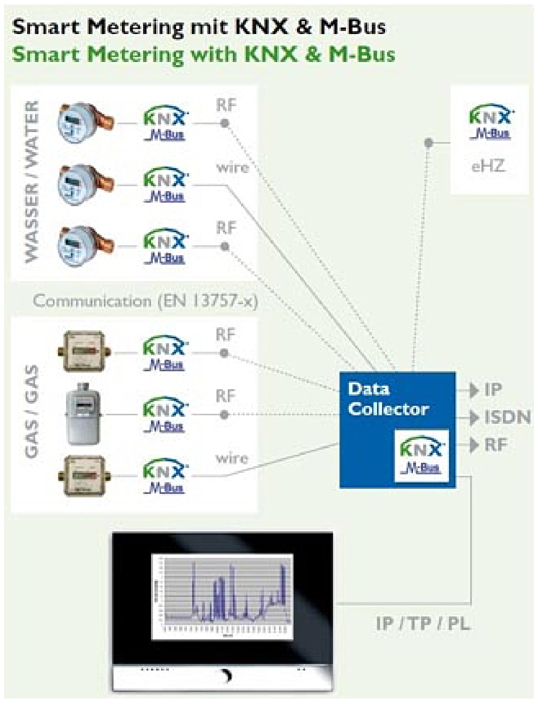

Smart Metering mit KNX & M-Bus
Smart Metering with KNX & M-Bus
WASSER / WATER
GAS / GAS
KNX M-Bus
RF
wire
Communication (EN 13757-x)
eHZ
Data Collector
IP
ISDN
RF
IP / TP / PL

Figure D.2: Data of energy consumption will be archived and access via Internet [55]

Implementation Aspects for KNX IP

The transmission of KNX telegrams via Ethernet is defined as KNXnet/IP and it is a part of the KNX Standard. IP routers are similar to line couplers, except that they use Ethernet for the main line, but based on this concept now it is possible to integrate KNX devices directly via IP on the KNX network. A KNX IP node basically consists of the following elements:

Ethernet controller: ethernet controllers are available from different semiconductor manufacturers. The Ethernet controllers, based on the KNX IP requirements, provide a bit rate of 10 MBits.

Microcontroller: a KNXnet/IP controller can be implemented on a 8-bit controller but this depends on the application, more powerful controllers might be required.

Communication Stack: communication via Ethernet requires an IP Stack with UDP protocol in fact KNXnet/IP is based on connectionless communication. The translation from KNX telegrams to UDP telegrams is established via KNXnet/IP. Therefore, the KNX application has access to the KNX stack, in order to communicate with the whole system, using a set of Application Programming Interface (API).

Choosing the proper hardware depends basically on the type of application. Hardware implementations, made especially for KNX IP devices, and the appropriate stacks are already available on market. However, for complex devices more powerful operating systems such as Linux, which basically contain an IP Stack with UDP, can be used; in this case, only the KNX stack is required.

Appendix E

European Installation Bus

The European Installation Bus [24] is an open, comprehensive system which covers all aspects of building automation. This protocol is similar to the BACnet protocol but it is manage by EIB Association. EIB embeds the protocol for home and building electronic systems with regulations for standardization of components, network management and interworking standards, etc. The EIB is designed as a management system in the field of electrical installation for environmental control and security of different types of buildings. Its purpose is to ensure the monitoring and control of processes such as lighting, window blinds, heating, ventilation, air-conditioning, load management, signaling, monitoring and alarms. The EIB system is designed to allow bus devices to draw the power supply from a communication medium such as Twisted Pair or Powerline.

Network Topology

EIB is a fully peer-to-peer network, which accommodates up to 65536 devices. The logical topology allows 256 devices on one line and, as shown in figure E.1, several lines may be grouped together in order to form an area that can be member of a domain, which is identified by a 16 bit SystemID. Each domain can contain at most 15 areas, connected together by a backbone line. Without the addresses reserved for couplers (255 x 16) x 15 + 255 = 61455 end devices can be joined by an EIB network. However, some installation restrictions due to implementation (medium, transceiver types, power supply capacity) and environmental (electromagnetic noise,...) factors can exist.

Lines or segments are connected by couplers in order to make use of systems such us repeater, bridge, router, package filter (for traffic optimization), firewall protection etc.

Media

EIB Medium Access Control is highly optimized for each medium individually and some implementations are available further optimize for a combination

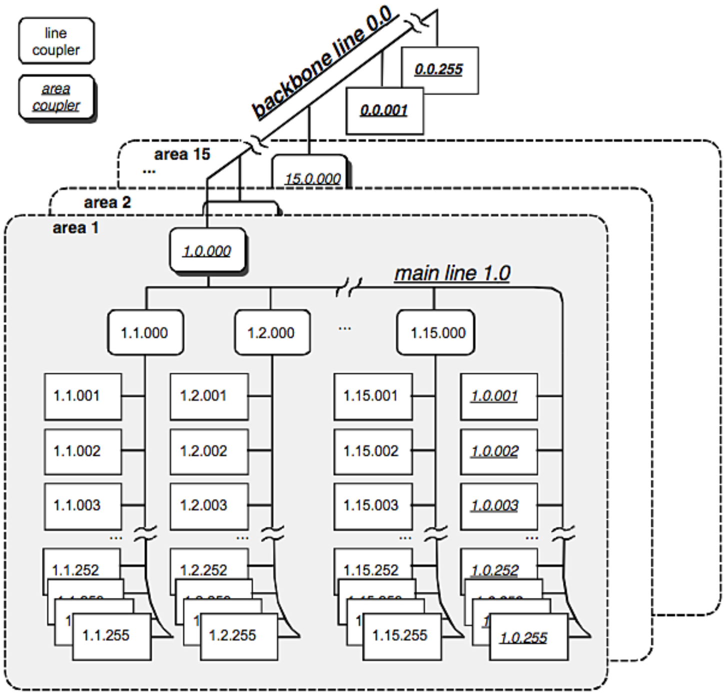


Figure E.1: The logical topology of EIB [24]

of transceiver performance and cost.

EIB.TP: on EIB Twisted Pair (TP), bit-level collision detection with dominant logical 0 ensures that in case of collision, the transmission always succeeds for one of the communication partners. The resulting elimination of retransmissions further enhances the performance of EIB TP. Together with EIB's powerful group addressing, EIB TP1 Collision Avoidance caters for extreme efficiency with reaction times 100 ms for two simultaneous transmissions.

EIB.PL: EIB Powerline (PL) uses a novel Spread Frequency Shift Keying modulation technique. With a corresponding numerical matched filter, the available BAU's guarantee adequate communication for group addressing to work reliably on PL. Medium access is controlled via a preamble sequence, with randomized retransmission slots, and it is important to underline that exists a maximum distance between 2 devices (without repeater) which is 600 meters. (Communication is influenced by electromagnetic pollution conditions in the installation.)

EIB.RF: EIB Radio Frequency (RF) lines are physically separated by a different carrier frequency. In free field conditions, the transmission distance is about 300 m. Retransmission ensures that large volumes can also be covered inside the building and this functionality is optimally distributed among the installed devices by the system itself.

EIB.net: the EIB.net specification realizes EIB on all media with a logical link layer according to ISO/IEC 802, including Ethernet. Furthermore with EIB.net/IP, EIB.net becomes routable across existing office and building networks, or even remotely through the Internet, employing the Internet Protocol.

The EIB OSI Communication Protocol

Figure E.2 shows how the EIB communication stack is structured according to the OSI seven layers model. This is also reflected in the frame structure shown in figure E.3. The physical layer and link layer obviously depend on the characteristics of the physical medium. For medium access control, EIB prescribes Carrier Sense Multiple Access with Optimized Collision Avoidance (CS-MA/CA). The Destination Address Flag (DAF) distinguishes between Group and Device Oriented telegrams.

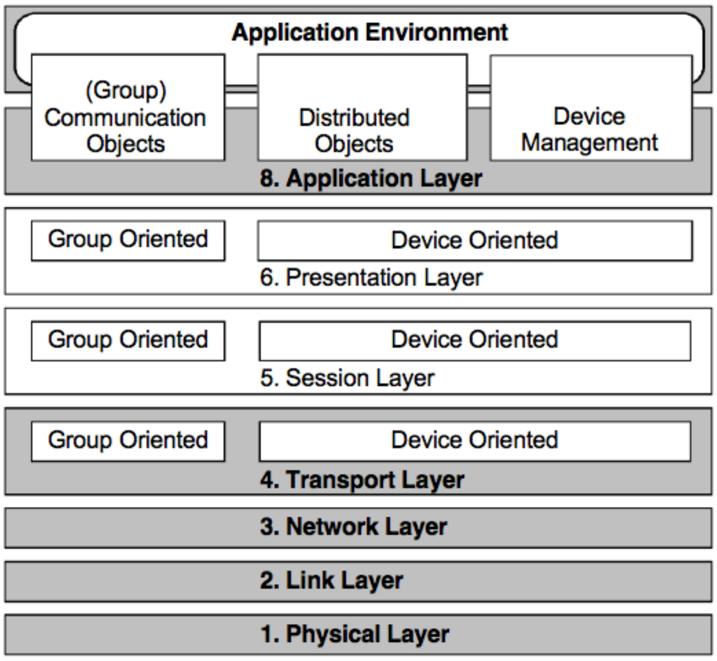


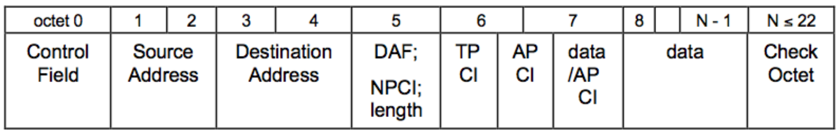
Figure E.2: EIB Communication according to the OSI reference model [24]

Through the Network Protocol Control Information (NPCI), the network layer controls the hop count. The transport layer manages logical communication relationships can be:

- one-to-many connectionless ("group" multicast)
- one-to-all connectionless (broadcast)
- one-to-one connectionless
- one-to-one connection-oriented

It provides the mapping between addresses and an abstract internal representation, the *Communication_Reference_ID* (*cr_id*).

All services are mapped transparently across session and presentation layer. Application layer implements API for client/server management of EIB networks. The Group Application Layer deals with the assignment of a group *cr_id* to a local instance of a Group Communication Object (or shared variable), for receiving (one-to-n) and for sending (one-to-one).



octet 0	1	2	3	4	5	6		7	8		N - 1	N ≤ 22
Control Field	Source Address		Destination Address		DAF; NPCI; length	TP CI	AP CI	data /AP CI	data			Check Octet

Figure E.3: EIB PDU frame structure (long frames allow N $<$ 255) [24]

EIB Protocol Data Unit (PDU) frames can carry application data formats of up to 14 bytes (extension to 230 is currently under consideration). In the next section is presented the central importance of the dedicated group-oriented facilities of the EIB Operating System.

EIB Network Management and Addressing

Network Management

To manage network resources (for example when configuring an installation) EIB uses a combination of broadcast and point-to-point communication. To address the point-to-point communication, to each device in the installation is assigned a unique Physical Address. In this manner, EIB provides some interesting features:

- A connection may be built up for example to download the complete binary image of an application program.

- It is possible to open a connectionless access to EIB Distributed Objects through `<dev>.<obj>.<prop>` addressing, as a native EIB management-level mechanism for status visualization and control.

- A dedicated master-slave fast polling mode ensures live and status check of critical subsystems.

Group Addressing for Run-Time Efficiency

EIB supports full multicast (group) addressing and this means that:

- EIB is not limited to grouping devices. Each device may publish several variables (known as "(Group) Communication Objects") individually, which can be grouped independently into a network-wide shared variables. As a bonus, properties of Distributed Objects may be also published as shared variables.

- In the description of the group-oriented EIB communication stack, a shared variable can be fully read/write bidirectional. In this manner every device can also send unsolicited multicast frames.
- EIB makes a 16 bit address space available for these shared variables. Even with the limitation of some implementations to 15 bits, this signifies that one installation may have up to 32k shared variables (or group addresses), each with any number of local instances.

The resulting scope and efficiency makes group address communication the preferred runtime mode for autonomous EIB field level communication.

Multi-client/Multi-server Management of the Object Oriented EIB Network

An EIB installation may be seen as a collection of distributed resources, which can be managed across the network. Therefore, each EIB device implements a server, which provides control over local resources (including hosting services for external CPU or memory resources) accessed via the serial Physical External Interface. A series of APCI's render these services accessible to remote clients. Through the introduction of EIB Distributed Objects, the network resources actually become Object Oriented.

Management clients typically access the network either for control or for (initial) configuration services. EIB implements a complete suite of vendor-neutral and standard PC-based configuration tools, which manage loadable applications.

Data Formats and Interworking

EIB frame can carry data formats of up to 14 bytes. The basic data formats specified by the EIB specification include:

- boolean (1bit)
- (un)signed short (16 bit)
- (un)signed long (32 bit)
- short float(16bit)
- IEEE float(32bit)
- date (24 bit)
- time (24 bit)
- control (4 bit)

Identifiers are defined for all physical values such as temperature, length, speed, field strength, energy, power, etc. Type information is used mainly at configuration time, in fact it is not transmitted for better performance. Properties with these basic data types are grouped into Distributed Objects, which is accessible by network. The EIB Interworking Standards (EIS) specify various specialized objects for all areas of building automation such as lighting (dimming control,...).

Tool Suites and Software Engineering Framework

EIB explicitly encompasses a methodology for Project Engineering, for example for linking a series of individual devices into a functioning installation. This is provided by the vendor-independent EIB Tool Software (ETS):

- Using the ETS Developer's Edition the manufacture encapsulates the remotely loadable applets in a series of abstract representations, which hide all implementation details. The resulting Component Description can be exported.
- A project engineer or electrical contractor can import the Component Description into he ETS Project Edition. All device instances can be customized to the needs of the project and logically linked by assigning Group Addresses.

ETS is built on top of a framework of software-engineering components, called the EIB Tool Environment (ETE) and its implementation is commercially available for 3^{rd} party use.

ETS4 Professional, the newest version for all application areas

The new version of ETS Professional (ETS4 Professional), available through the KNX distribution channel, is the powerful successor of ETS3. ETS4 Professional provides a set of features suited to the realization of home and building automation projects through the following phases and tasks:

- Project Planning & Design
- Commissioning
- Project-Documentation
- Diagnostics and Troubleshooting

Using ETS4 Professional, it is possible to compose solutions for all application areas for which ETS ready products are available from KNX manufacturers. Application areas include:

- Lighting control
- Shading control
- Heating, ventilation, air-conditioning
- Access & Security
- Energy management
- Comfort functions and intelligent control across all applications
- Interfacing to complementary or peripheral systems
- ...

Before planning a KNX projects, every technical information of the KNX products has to be imported into ETS. These data (included in so called product - databases) are easily accessible from the particular manufacturers, for example though a special download area in the manufacture's website or into a CD-ROM, and a simple guided procedure, provided by ETS, simplifies the importing process.

How to use ETS4

In this section there is a brief practical introduction of ETS, it is possible to observe the simplicity of use of this tools.

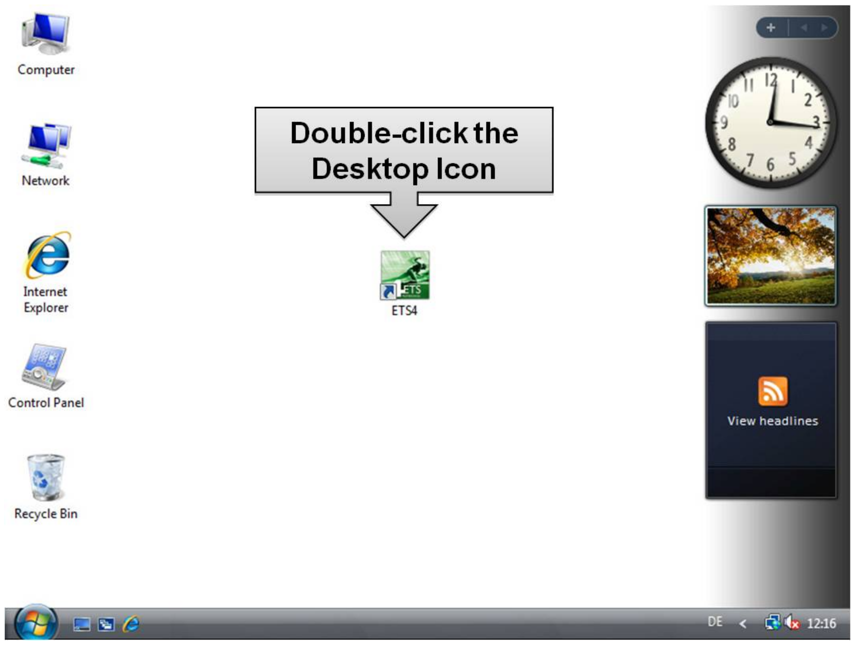
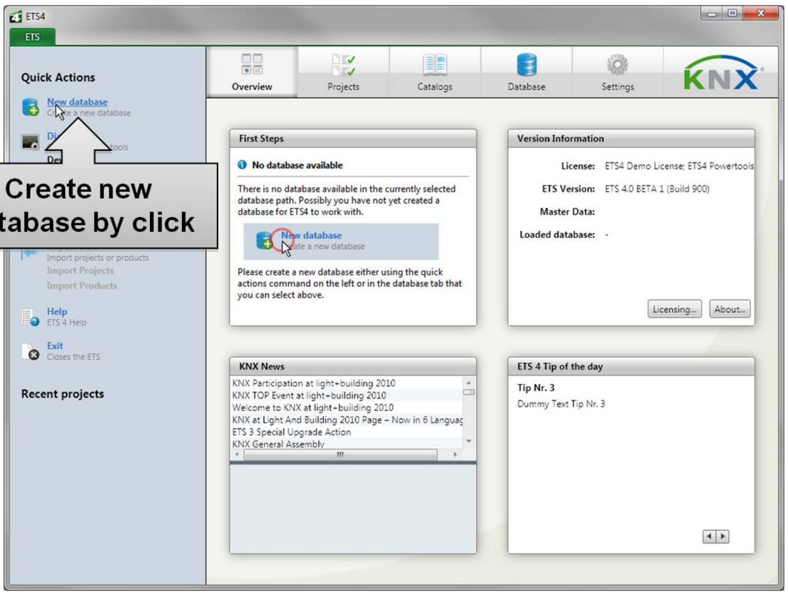


Figure E.4: [**Step 1**] When the installation is completed an ETS4 icon appears on the desktop. By double clicking it, the ETS4 will be executed for the first time. It is also possible to navigate on windows "program" folder the ETS4 and start the program from this location. [51]



No database available

There is no database available in the currently selected database path. Possibly you have not yet created a database for ETS4 to work with.

New database
Create a new database

Please create a new database either using the quick actions command on the left or in the database tab that you can select above.

License: ETS4 Demo License; ETS4 Powertools
ETS Version: ETS 4.0 BETA 1 (Build 900)
Master Data:
Loaded database: -

KNX Participation at light+building 2010
KNX TOP Event at light+building 2010
Welcome to KNX at light+building 2010
KNX at Light And Building 2010 Page – Now in 6 Languag
ETS 3 Special Upgrade Action
KNX General Assembly

Tip Nr. 3
Dummy Text Tip Nr. 3

Figure E.5: [**Step 2**] To build any KNX projects, first of all a database must be created; within this the projects are stored. Via the overview menu/quick actions the database will be created by clicking on the corresponding entry. A file dialog appears; via this dialog the possibility exists to enter the database name and also the storage location. [51]

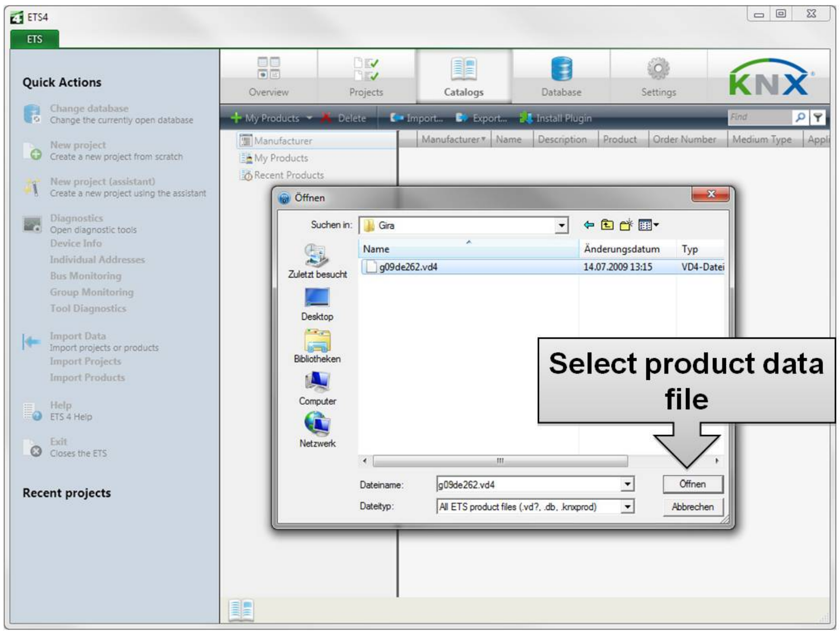


Figure E.6: [**Step 3**] Subsequent to the database creation, certified KNX products from various KNX manufacturers can be imported. Via overview tab Catalogs and a file dialog choose the product file, which can be downloaded from the manufacturers' web site. The import of file is controlled via a "wizard" who gives you several selection opportunities as single selection of a product to be imported or corresponding translation languages. After closing or rather at the end of wizard in the Catalogs tab overview all imported products are enumerated. From now on they are ready to use in project creation. [51]

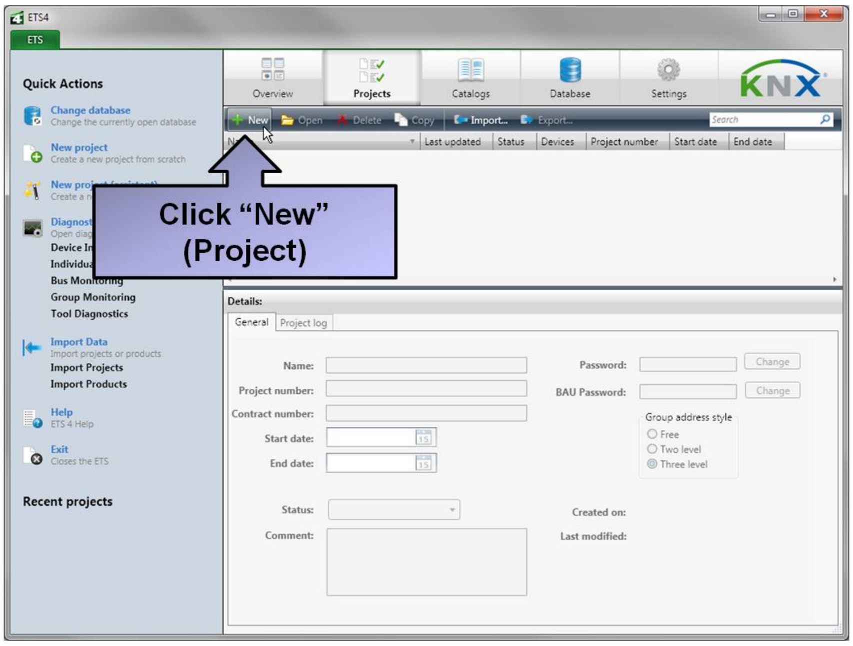


Figure E.7: [**Step 4**] Inside tab Projects and a click on New a new project will be created. The project name is arbitrary and, in the properties of Project, the used KNX media and how group addresses will be displayed. [51]

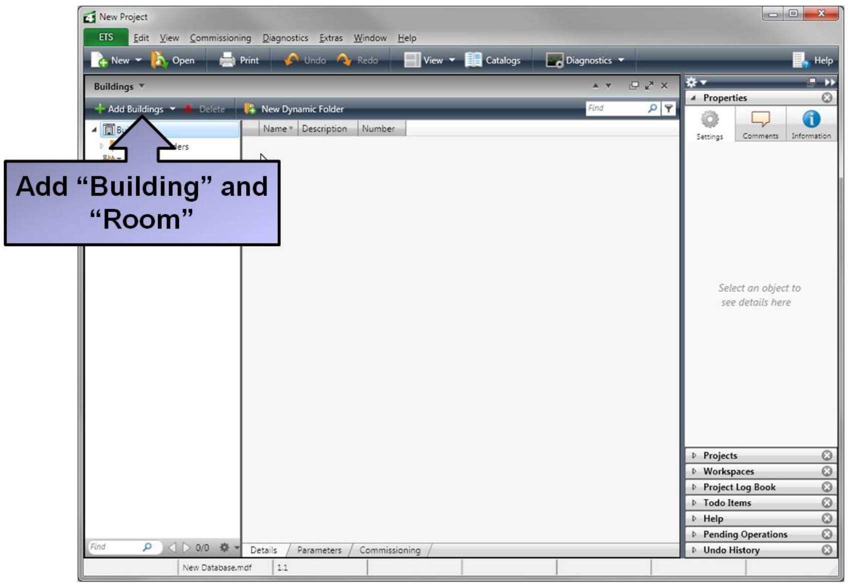
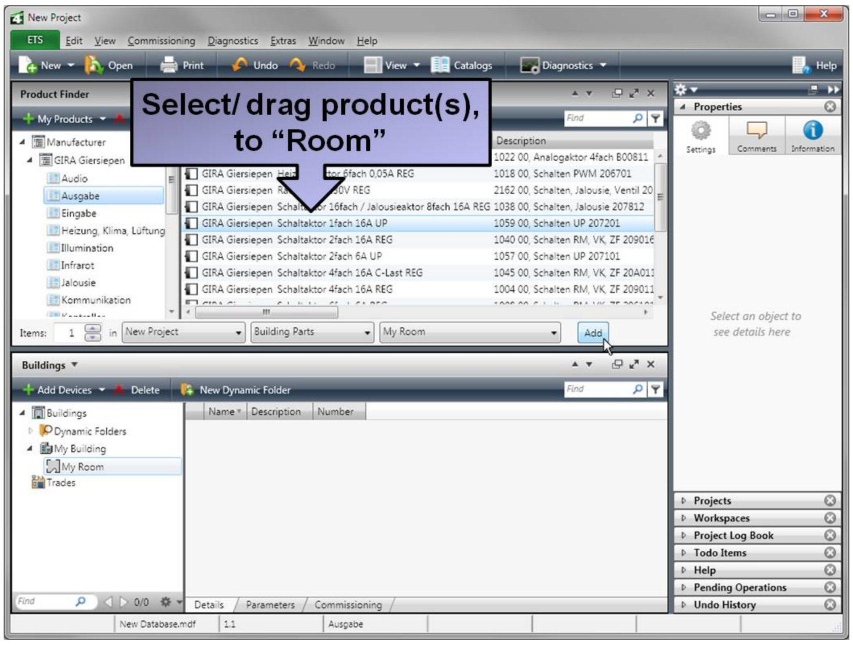


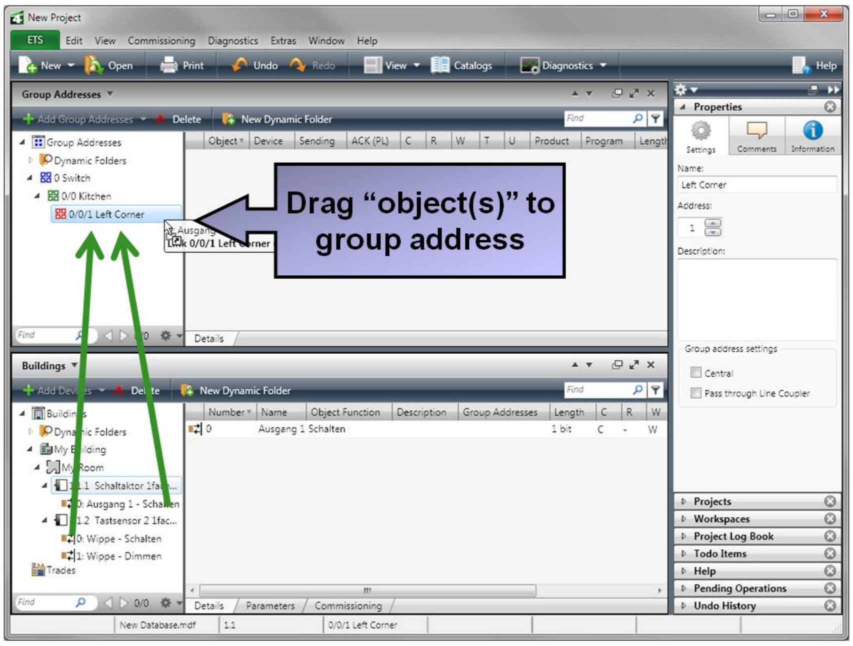
Figure E.8: [**Step 5.1**] A project consists of KNX devices and their links to each other. The devices itself are located in the installation within building parts; e.g., rooms. Hence in ETS4 in panel Buildings e.g. the building structure and also equivalent effigies of appropriate rooms must be created. Within building structure, an assignment of KNX devices in an installation takes place to the installation point. [51]


Select/ drag product(s), to “Room”
New Project
ETS Edit View Commissioning Diagnostics Extras Window Help
New Open Print Undo Redo View Catalogs Diagnostics Help
Product Finder
My Products
Manufacturer
GIRA Giersiepen
Audio
Ausgabe
Eingabe
Heizung, Klima, Lüftung
Illumination
Infrarot
Jalousie
Kommunikation
Buildings
Add Devices Delete New Dynamic Folder
Dynamic Folders
My Building
My Room
Trades
Name Description Number
Details Parameters Commissioning
Properties
Settings Comments Information
Select an object to see details here
Projects
Workspaces
Project Log Book
Todo Items
Help
Pending Operations
Undo History
New Database.mdf 1.1 Ausgabe

	Description
	1022 00, Analogaktor 4fach B00811
GIRA Giersiepen Hei[illegible]tor 6fach 0,05A REG	1018 00, Schalten PWM 206701
GIRA Giersiepen Ra[illegible]30V REG	2162 00, Schalten, Jalousie, Ventil 20
GIRA Giersiepen Schalt[illegible]or 16fach / Jalousieaktor 8fach 16A REG	1038 00, Schalten, Jalousie 207812
GIRA Giersiepen Schaltaktor 1fach 16A UP	1059 00, Schalten UP 207201
GIRA Giersiepen Schaltaktor 2fach 16A REG	1040 00, Schalten RM, VK, ZF 209016
GIRA Giersiepen Schaltaktor 2fach 6A UP	1057 00, Schalten UP 207101
GIRA Giersiepen Schaltaktor 4fach 16A C-Last REG	1045 00, Schalten RM, VK, ZF 20A011
GIRA Giersiepen Schaltaktor 4fach 16A REG	1004 00, Schalten RM, VK, ZF 209011

Items: 1 in New Project | Building Parts | My Room | Add

Figure E.9: [**Step 5.2**] In next step via the panel Catalogs the designated devices will be inserted in the previously created rooms. The used devices reflect in principle an application what you want to achieve, e.g. in child’s room, lighting including a dimming and sun blind control. [51]


Drag “object(s)” to group address
New Project
ETS Edit View Commissioning Diagnostics Extras Window Help
New Open Print Undo Redo View Catalogs Diagnostics Help
Group Addresses
Add Group Addresses Delete New Dynamic Folder
Group Addresses
Dynamic Folders
0 Switch
0/0 Kitchen
0/0/1 Left Corner
Object Device Sending ACK (PL) C R W T U Product Program Length
Ausgang
Link 0/0/1 Left Corner
Buildings
Add Devices Delete New Dynamic Folder
Buildings
Dynamic Folders
My Building
My Room
1.1 Schaltaktor 1fa...
0: Ausgang 1 - Schalten
1.2 Tastsensor 2 1fac...
0: Wippe - Schalten
1: Wippe - Dimmen
Trades
Details Parameters Commissioning
Properties
Settings Comments Information
Group address settings
Central
Pass through Line Coupler
Projects
Workspaces
Project Log Book
Todo Items
Help
Pending Operations
Undo History
New Database.mdf 1.1 0/0/1 Left Corner

Name: Left Corner
Address: 1
Description:

Number	Name	Object Function	Description	Group Addresses	Length	C	R	W
0	Ausgang 1	Schalten			1 bit	C	-	W

Figure E.10: [**Step 5.3**] After this the links (in KNX terminology “group addresses”) between the different devices will be established. For this a communication object from at least two different devices (the green arrows) will be dragged on a previously created group address in the Group Addresses panel. Result is a logical link between these two devices. [51]

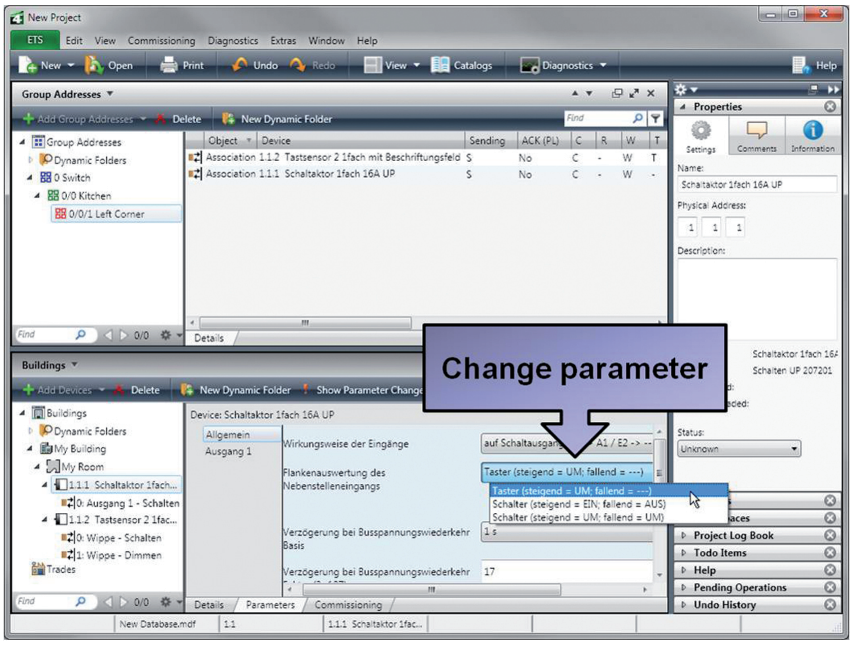


Figure E.11: [**Step 6**] The properties of a device, e.g. a lighting release delay will be adjusted in the Parameter Dialog of appropriate device. [51]

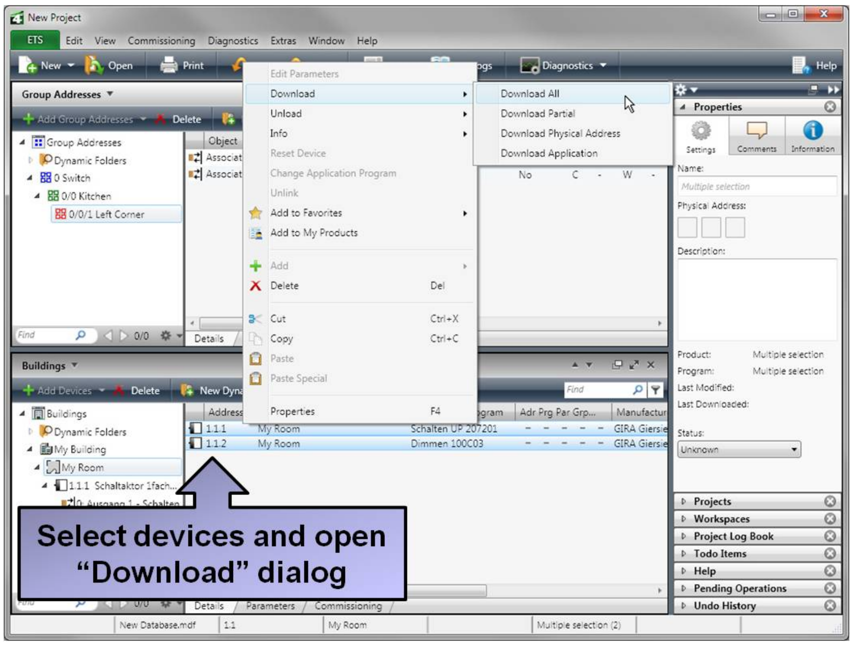
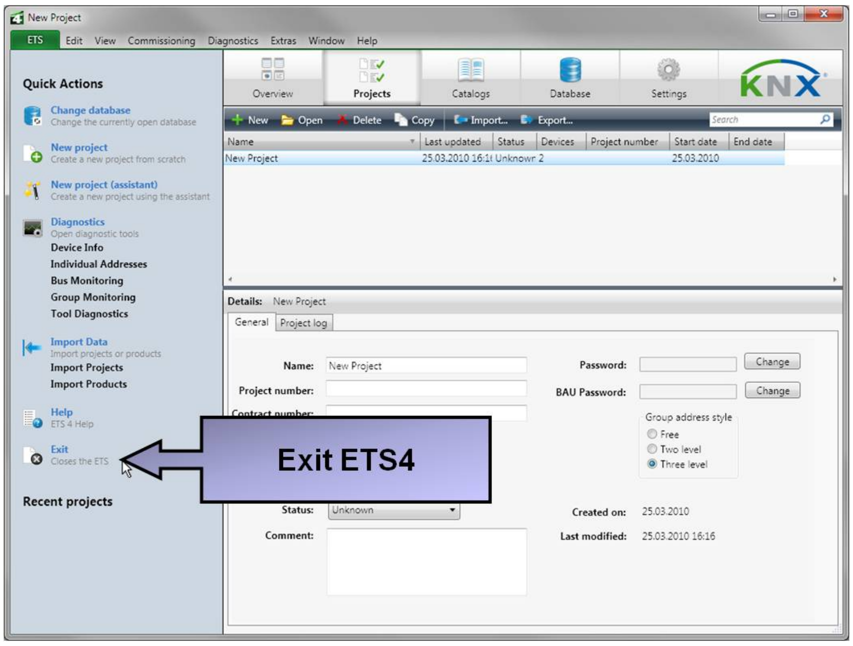


Figure E.12: [**Step 7**] When all adjustments are done, the download of parameters into KNX devices will be started. For this, all necessary devices will be marked and via mouse context or menu bar the function will be called. After finishing the download, the device status will be automatically adjusted by ETS4. This status shows the correct cycle or also errors during the download. To check this it is also possible to select a device and call its status explicitly. [51]



Name	Last updated	Status	Devices	Project number	Start date	End date
New Project	25.03.2010 16:1	Unknown	2		25.03.2010	

Details: New Project

Name: New Project
Project number:
Contract number:
Password:
BAU Password:
Group address style: Free [✓ Three level] Two level
Status: Unknown
Comment:
Created on: 25.03.2010
Last modified: 25.03.2010 16:16

Figure E.13: [**Step 8**] Before you exit ETS4 you should as a last step always perform a backup of the database (tab Database). If this is currently not necessary, you can exit ETS4 via Exit button. [51]

Appendix F

Deploying Servlets in TomCat

The process of installation of a servlet into Tomcat is called *deploying*, it requires few easy steps, since the web server is correctly configured.

Upload compiled servlet: your compiled servlets should go unpacked into `WEB-INF/classes/package/dirs/CompiledServlet.class` for example the `intranet.AdminControllerServlet` should go inside `WEB-INF/classes/intranet/AdminControllerServlet.class` The package can also be transformed into a JAR library file and simply uploaded in `WEB-INF/lib/`

Locate web.xml: the `web.xml` should be located inside the `WEB-INF/`. It is the same file that was used when you specified the security constraints for your web application.

Add servlet and servlet mappings: before the security constraint elements, and underneath the web-app, add all your servlets. Then add all your servlet mappings, the servlets and servlet mappings cannot be mixed together, because this will break the order defined in the `xsd` (source code F.1).

Listing F.1: Example of `web.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
    <servlet>
        <servlet-name>KNXBridge</servlet-name>
        <servlet-class>knx.KNXBridge</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>KNXBridge</servlet-name>
        <url-pattern>/KNXBridge</url-pattern>
        </servlet-mapping>
        <session-config>
            <session-timeout>
                30
            </session-timeout>
        </session-config>
        <welcome-file-list>
            <welcome-file>index.jsp</welcome-file>
        </welcome-file-list>
        <security-constraint>
        ...
        </security-constraint>
    </web-app>
```

The url-patterns are all relative to `http://tomcatserver/Context_Path/`, where `Context_Path` is defined as part of the `server.xml`. Fortunately, tools as NetBeans IDE helps the developer with to install Tomcat and also to deploy a servlet [68].

Bibliography

[1] Ajax: A new approach to web applications.
Adaptivepath
http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications.

[2] Phonegap.
Adobe Systems Inc.
http://phonegap.com/.

[3] *Tunneling Component Network Protocols Over Internet Protocol Channels*, Std. 852, 2002.
ANSI/EIA/CEA.

[4] Apple.
Apple Inc.
http://www.apple.com/.

[5] Apple developer programs.
Apple Inc.
https://developer.apple.com/programs/.

[6] Appstore.
Apple Inc.
http://www.apple.com/iphone/from-the-app-store/.

[7] ios webview.
Apple Inc.
https://developer.apple.com/library/mac/#documentation/Cocoa/Reference/WebKit/Classes/WebView_Class/Reference/Reference.html.

[8] iphone.
Apple
http://www.apple.com/iphone/.

[9] Oil crisis, energy-saving technological change and the stock market crash of 1973-74.
Sami Alpanda & Adrian Peralta-Alva
http://research.stlouisfed.org/wp/2008/2008-019.pdf.

[10] Apache license.
Apache Software Foundation
http://www.apache.org/licenses/.

[11] Apache software foundation.
Apache Software Foundation
http://www.apache.org/foundation/.

[12] Apache tomcat.
Apache Software Foundation
http://tomcat.apache.org/.

[13] *BACnet: a standard communication infrastructure for intelligent buildings.*, Autom. Construction, vol. 6, no. 5-6, pp. 529-540 - 1997.
S. T. Bushby.

[14] *Communication Gateways: Friend or Foe?*, Published in ASHRAE Journal Vol.40, No. 4, pp. 50-53 - 1998.
Steven T. Bushby.

[15] *Personal Trust Space and Devices: "Geography will not be History" in the m-commerce future*, 2003.
Adams C. & Millard P.

[16] Cenelec ratifies new set of standards on home & building electronic systems (hbes), 2004.
CENELEC
http://www.knx.org/fileadmin/downloads/07-News&Press/03-PressReleases/01-English/PR2004-03-29_EN.pdf.

[17] *Analyzing Congestion in KNXnet/IP Communication System.*
Salvatore Cavalieri.

[18] *Comparing KNXnet/IP Forwarding Rules in Congestion Conditions.*
Salvatore Cavalieri.

[19] *Estimating KNXnet/IP Routing Congestion.*
Salvatore Cavalieri.

[20] Batibus.
CWCT
http://www.cwct.co.uk/ibcwindow/ibc/fieldbus/batibus.html.

[21] Open systems - is an open protocol enough?
Andy Davis
http://www.automatedbuildings.com/news/oct11/articles/andydavis/110925014909andydavis.html.

[22] Home, green home.
Economist
http://www.economist.com/node/11999259.

[23] Total energy consumption.
Enerdata Global Energy Statistical
http://yearbook.enerdata.net/.

[24] EIBA, Brussels, Belgium. *The EIB System for Home & Building Electronics*, 1998.
Marc Goossens
http://www.eiba.ru/download/eib02_system.pdf.

[25] *World Energy use in 2010: over 5% growth*, 2011.
Enerdata

http://www.enerdata.net/enerdatauk/press-and-publication/publications/g-20-2010-strongly-energy-demand-increase.php.

[26] General packet radio service.
ETSI
`http://www.etsi.org/WebSite/technologies/gprs.aspx`.

[27] *Building automation interoperability - A review*, IWSSIP 2010.
F. Ferreira & A. L. Osório & J. M. F. Calado & C. S. Pedro
17th International Conference on Systems, Signals and Image Processing.

[28] Mobile app usage further dominates web.
Flurry
`http://blog.flurry.com/?Tag=Apps`.

[29] Trying to understand html5 compatibility on mobile and tablet browsers.
Maximiliano Firtman
`http://mobilehtml5.org/`.

[30] Smartphone apps in europe: The 8th mass market media channel, 2011.
Peter Farago
`http://blog.flurry.com/?Tag=reach`.

[31] Gnu general public license.
Free Software Foundation
`http://www.gnu.org/licenses/gpl.html`.

[32] Android webview.
Google
`http://developer.android.com/reference/android/webkit/WebView.html`.

[33] Google.
Google
`http://www.google.com/intl/en/about/corporate/index.html`.

[34] Ets4 xml to calimero ng xml.
Wolfgang Granzer
`https://www.auto.tuwien.ac.at/`.

[35] Ieee 802.11™ wireless local area networks.
IEEE
`http://www.ieee802.org/11/`.

[36] International standards for business, government and society.
ISO
`http://www.iso.org/`.

[37] Iso 9000 essentials.
ISO
`http://www.iso.org/iso/iso_9000_essentials`.

[38] *Building Automation and Control Systems (BACS) - Part 5: Data Communication Protocol.*, ISO Std. 16 484-5, 2003.
ISO.

[39] European home systems.
JAEC
`http://www.jaec.info/HomeAutomation/Protocols-buses-house/Ehs-Protocol/ehs-protocol.php`.

[40] Class httpservlet.
Java^{TM}
http://docs.oracle.com/javaee/1.3/api/javax/servlet/http/HttpServlet.html.

[41] Java servlet technology.
Java^{TM}
http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets.html.

[42] Sharing information.
Java^{TM}
http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/Servlets5.html#73895.

[43] jquery ajax.
The jQuery Project
http://api.jquery.com/category/ajax/.

[44] jquery mobile.
jQuery Project
http://jquery.com//.

[45] jquery mobile.
jQuery Project
http://jquerymobile.com/.

[46] A touch-optimized web framework for smartphones & tablets.
The jQuery Project
http://jquerymobile.com/demos/1.0a2/.

[47] *Good green homes*, 2003.
Roberts Jennifer.

[48] Introducing json.
JSON.org
http://json.org/.

[49] Smart phone shaping the future, 2010.
Ki-Duck Kown
Samsung Economic Research Institute
http://www.seriworld.org/01/wldContL.html?mn=B&mncd=0201&pagen=3.

[50] Knxlive! project.
Martin Kögler
https://www.auto.tuwien.ac.at/a-lab/home.html.

[51] Engineering tool software.
KNX Association
http://www.knx.org/knx-tools/ets4/description/.

[52] Knoppix - live linux filesistem on cd.
KNOPPIX developers
http://www.knoppix.org/.

[53] Knx - wholesalers.
KNX Association
http://www.knxuk.org/knx-members/members_integrators.asp?Section=4&Search=Sect.

[54] *KNX AN033, Common EMI Specification, v03 AS and KNX AN066, cEMI Adaptions, v03 AS.*
KNX Association.

[55] Knx association.
KNX®
`http://www.knx.org`.

[56] *KNX handbook for home and building control: basic principles.*
KNX Association.

[57] *Communication Systems for Building Automation and Control.*, IEEE - 2005.
Wolfgang Kastner & Georg Neugschwandtner & Stefan Soucek & H. Michael Newman.

[58] *Open Control Networks.*, 2004.
D. Loy & D. Dietrich & H. Schweinzer.

[59] *Calimero: Next Generation.*
Boris Malinowsky & Georg Neugschwandtner & Wolfgang Kastner.

[60] M-bus standard: En 13757-x.
M-Bus Usergroup
`http://www.m-bus.com`.

[61] Javascript.
Mozilla Developer Network
`https://developer.mozilla.org/en/JavaScript`.

[62] *Building Automation: Communication systems with EIB/KNX, LON and BACnet*, 2009.
Hermann Merz & Thomas Hansemann & Christof Hübne.

[63] Aspṅet.
Microsoft®
`http://www.asp.net/`.

[64] Smartphone vs. feature phone arms race heats up; which did you buy?, 2009.
Andrew Nusca
ZDNet.

[65] *Direct Digital Control of Building Systems: Theory and Practice.*, New York: Wiley - 1994.
H. M. Newman.

[66] A disruption analysis in the mobile payment market, 2005.
J. Ondrus & Y. Pigneur
Proceedings of the 38th Hawaii International Conference on System Sciences
`http://csdl2.computer.org/comp/proceedings/hicss/2005/2268/03/22680084c.pdf`.

[67] Java.
Java™
`http://java.com/`.

[68] Netbeans ide.
Oracle
`http://netbeans.org/`.

[69] Sun microsystem - oracle.
Oracle
http://www.oracle.com/us/sun/index.htm.

[70] The mit license (mit).
Open Source Initiative OSI
http://www.opensource.org/licenses/MIT.

[71] Proc. AFIPS Fall Joint Computer Conference. *Response in man-machine conversational transactions.*
Millar.

[72] Blackberry.
Research In Motion - RIM
http://us.blackberry.com/.

[73] *Energy Efficiency and Building Automation.*
SCL Elements Inc.

[74] Merlin gerin is now schneider electric.
Schneider Electric Global Energy Management Specialist
http://www.schneider-electric.co.uk/sites/uk/en/company/brands/merlin-gerin.page.

[75] *Analysis of the global smartphone market and the strategies of its major players*, 2011.
Hee-Chan Song.

[76] *Chasing a Habitable 'Home of the Future*, 2006.
Sydell Laura.

[77] Linux and the gnu project.
Richard Stallman
http://www.gnu.org/gnu/linux-and-gnu.html.

[78] *Rate of world energy usage in terawatts (TW)*, 2009.
Statistical Review of World Energy 2009
http://www.bp.com/liveassets/bp_internet/globalbp/globalbp_uk_english/reports_and_publications/statistical_energy_review_2008/STAGING/local_assets/2009_downloads/statistical_review_of_world_energy_full_report_2009.xls#'Primary.

[79] *Fieldbuses and interoperability*, Control Eng. Practice, vol. 7, no. 1, pp. 81-94 - 1999.
J. P. Thomesse.

[80] Saving energy through intelligent networking.
Union Investment
http://www.sustainable-realestate-investments.com/en/topics/electrical-engineering-and-building-services/building-automation-systems/.

[81] *Mobile Platforms: The Clash of Ecosystems*, 2011.
VisioMobile
http://www.visionmobile.com/research.php#ecosystems.

[82] Apple - html5 and web standards.
W3C®
http://www.apple.com/html5/.

[83] Cascading style sheets.
W3C®
`http://www.w3.org/Style/CSS/`.

[84] Document object model (dom).
W3C®
`http://www.w3.org/DOM/`.

[85] Hypertext markup language.
W3C®
`http://www.w3.org/MarkUp/`.

[86] Internet media type registration consistency of use.
W3C®
`http://www.w3.org/2001/tag/2002/0129-mime`.

[87] Same-origin policy.
w3c®
`http://www.w3.org/Security/wiki/Same_Origin_Policy`.

[88] Wai-aria.
W3C®
`http://www.w3.org/WAI/intro/aria`.

[89] Xml protocol working group.
W3C®
`http://www.w3.org/2000/xp/Group/`.

[90] Xmlhttprequest.
W3C®
`http://www.w3.org/TR/XMLHttpRequest/`.

[91] Xsl transformations (xslt).
W3C®
`http://www.w3.org/TR/xslt`.

[92] Html5, 2011.
W3C®
`http://www.w3.org/TR/html5/`.

[93] Extensible markup language (xml).
WAI-ARIA
`http://www.w3.org/XML/`.

[94] Wireshark, network protocol analyser.
Wireshark Foundation
`http://www.wireshark.org/`.

[95] Fieldbus.
Wikipedia
`http://en.wikipedia.org/wiki/Fieldbus`.

[96] Heating, ventilation, and air-conditioning.
Wikipedia
`http://en.wikipedia.org/wiki/HVAC`.

[97] Late-2000s financial crisis.
Wikipedia
`http://en.wikipedia.org/wiki/Late-2000s_financial_crisis`.

[98] Tunneling protocol.
Wikipedia
`http://en.wikipedia.org/wiki/Tunneling_protocol`.

Index