ISSN 0280-5316
ISRN LUTFD2/TFRT--5817--SE

A Home Automation Prototype

David Eklund
Daniel Rasmusson

Department of Automatic Control
Lund University
June 2008

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
June 2008

Document Number
ISRN LUTFD2/TFRT--5817--SE

Author(s)
David Eklund and Daniel Rasmusson

Supervisor
Hans Odeberg ÅF Konsult, Malmö
Karl-Erik Årzén Automatic Control (Examiner)

Sponsoring organization

Title and subtitle
A Home Automation Prototype (En prototyp för hemautomation)

Abstract
This master thesis concerns the current and future development in the area of intelligent houses and home automation. It also describes a system for home automation developed for this thesis. Key words in the design are dynamic behaviour, flexibility, transparency and adaptability. The prototype has been built with a central, Java based, server. To this server, modules handling different communication protocols can be connected at run time. To these modules, devices can connect over the specified communications protocols and register in the server, also at run time. Devices can be for control and supervision (computer or phone GUI), but also equipment that is to be controlled (lamps, ventilation, stereo) and equipment for notification (sensors). For communication, a message system has been developed. The server uses a MySQL database and also has support for building macros with extensive command and timer functionality. Finally, the system supports users and positioning using RFID.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
50

Recipient's notes

Security classification

Contents

1. Introduction

This master thesis has been performed at the systems division of the company ÅF and the Department of Automatic Control, Faculty of Engineering, Lund University (LTH). The Systems division of ÅF provides expert consultants primarily in the fields of computer engineering and embedded systems design. ÅF has several technical consultants working with home automation and intelligent houses, specially in the area of environmental control systems.

The department of Automatic Control at LTH does research on a wide range of subjects including control theory, modelling and simulation and real time issues.

This thesis will describe existing applications and techniques for intelligent houses and expand the knowledge on how to build a system for home automation. Furthermore, to show potential uses and future possibilities, a prototype system developed within the thesis will be described.

In order to design a good system, a number of fundamental questions need answering. These questions will probably be co-dependent, but a natural starting point might be:

What is home automation?

It is to be suspected that such a wide field will have many definitions. To narrow it down, a good follow up would be trying to concretely specify the purpose of a system for home automation.

What should a system for home automation be able to do?

To be able to answer that, goals have to be specified. In this case, the goal is probably to create a system that a maximum number of people would want and be willing to pay for, i.e:

What would it take to make systems for intelligent houses become a natural part of most households in industrialised countries, and why is it not already?

Before continuing towards questions of how to design a prototype, it is necessary to know about background, competition and current development.

What systems, companies and technologies are there in the field of home automation today and what do they do?

Once these fundamental questions have been answered, a prototype is to be built. Here the knowledge collected from answering the previous questions should be used in the design, i.e.

How should a prototype be designed to fulfil all the demands on a system for home automation?

Since there is a time limit to the thesis, another question needs answering:

What tasks should our prototype be able to perform in order to show the potential of the prototype system?

These are the questions that this thesis sets out to answer.

2. Background and current development

2.1 What is an intelligent house?

The area of intelligent houses, also called home automation, building automation or domotics, is broad, but at the core quite simple. It is the idea to interconnect all (or some) of the technical equipment in a house in order to combine their functions and control them in new and easier ways. It might also include automation of equipment in a house that was previously not considered technical, such as: lamps, doors and windows.

In some cases the terms intelligent houses, home automation, domotics and building automation are used to describe slightly different things, but in this thesis we will use them as synonyms, since the prototype system is aimed at covering all possible meanings of the terms.

Here follow some definitions found on the internet:

"Home automation is the use of one or more computers to control basic home functions and features automatically and sometimes remotely. An automated home is sometimes called a smart home." [Techtarget, 2008]

"Intelligent Building: A building that integrates technology and process to create a facility that is safer, more comfortable and productive for its occupants, and more operationally efficient for its owners. Advanced technology combined with improved processes for design, construction and operations provide a superior indoor environment that improves occupant comfort and productivity while reducing energy consumption and operations staffing." [IBD, 2008]

"For some it may be something as simple as remote or automatic control of a few lights. For others, security may be the central application. Still others may choose to install advanced controllers or use voice recognition. As a very basic definition, we tend to refer to home automation as anything that gives you remote or automatic control of things around the home." [Smarthome, 2008]

"(DOMus infOrmaTICS) Information technology in the home (domus is Latin for home). Although remote lighting and appliance control have been used for years (see X10), domotics is another term for the digital home, including the networks and devices that add comfort and convenience as well as security. Controlling heating, air conditioning, food preparation, TVs, stereos, lights, appliances, entrance gates and security systems all fall under the domotics umbrella." [Techweb, 2008]

As can be seen, the definitions are quite different, but there still is some kind of consensus about the basic purposes, goals and motivation for intelligent houses.

2.2 Vision

The vision of this thesis is an open, global, market standard, used by almost all companies that make equipment for use in houses, apartments, offices, and apartment buildings. It should be very easy to get new products for the system and even easier to install them. In the ideal case, just bring it home and turn it on.

It is of course impossible to know what solutions would come of a system like this, which is one of the most important points of this thesis, but imagine coming home and opening the front door. The system automatically starts the home from the sleeping mode that is automatically activated when the last user leaves the place. It then identifies the user, and applies her custom settings. First of, the lightning is set to the level of her choice in the hall, perhaps the computer is turned on to be ready later on. Of course, music starts up and follows her through the different rooms, if she wants it to.

Perhaps the user used her phone to send a message to the house on the way home. This way the coffee might be ready, just like it is in the morning, as the machine was automatically turned on 15 minutes before the alarm that fade on the lights and the user's favourite music in the bed room. If the user has small children she can rest assured, as a warning will be heard if the baby leaves its room. This warning will of course be heard in the room that the user happens to be in at the moment.

All of these things can be controlled from LCD touch screens in the rooms, from a computer, via a web interface or a mobile phone. With sensors it is also controllable with voice commands and gestures. With a simple order in the hall, a gin and tonic will be ready in the living room after the user has taken off her shoes and wants to wind down from work. Luckily she did not forget to bring her notebook to work in the morning. The system automatically reminded her that she had not brought it on her way out. She was also quite comfortable in the car, as it was ordered to heat up 45 minutes before she left the house.

After the gin and tonic, she starts cooking, puts on some rice and goes to watch TV. The house will remind her when it is done. Since she is environmentally friendly, the lights and other equipment is automatically turned off in all rooms she cannot see at any given moment. Later on when it is time to go to bed, her bedroom is a few degrees cooler than the rest of the home, since she prefers having a thick covering when sleeping. In the winter, this is of course accomplished by automatically opening a window, instead of wasting energy with the air condition. The same goes for the lightning, which is in first hand regulated to the right level using the Venetian blinds, and only if that's not enough, further compensated with lamps.

While watching the television, a reminder is activated and a message displayed on the screen, telling her to call her mother. She orders the house to place call, the program on the television disappears and a video link is established. Normally she prefers audio only, but tonight it is nice to see her mothers face. After the call, the front door opens again; her son is home. Once more the settings are changed in the hall, since he has other preferences. She opens a voice link, so that she does not have to scream, and tell him to come to the living room. When he enters, however, it all stays the way she wants it. Her profile has a higher priority than his, something the son does not appreciate.

To show compassion, she tells the house to change the music from opera to Johnny Cash, one of the few overlaps in taste of music between the two of them. After dinner the son goes to his room and does once more have control over his environment. She tells the house to clean itself and to wash the clothes, dry and fold them during the night, and then relaxes in her favourite easy chair with a book before going to sleep.

This vision is of course far removed from our prototype, but the system design, if properly developed, would be able to support all the applications described. To implement all the functions proposed, very advanced technology would have to be used at device level though. This in turn would require development by many companies with different competence and these are only applications imagined in this thesis. There is no way of telling what solutions would be thought of if an open system like this grew truly large.

2.3 Background and requirements

For every new invention, concept or idea, there are prerequisites. In the case of intelligent houses there are many. First of, intelligent houses as we think of them, depend heavily on technology (in fact, it is basically a way to interconnect and control technology). Of special importance in this case are inexpensive and, by historical comparison, powerful microcontrollers. Computers have been around for quite a long time now, but it is only in more recent years that computers with any real power of computation have become inexpensive enough to put in simple, everyday equipment.

The first semiconductor based computers, were simple primarily because of the limitation in the number of transistors. As development continued, the number of transistors grew and the computers became more advanced. The number of instructions in the ALUs became higher with each new generation. In the early 1980s a new trend started. Using instruction level parallelism and caches, the RISC architecture focused on fewer and simpler instructions that were highly optimised. The alternative to RISC is CISC that focuses on many and advanced instructions. [John L. Hennessay, 2007]

While the CISC approach is effective, since it has specialised functionality to handle common operations, it also has drawbacks. It is hard to make these processors inexpensive, since they are so complex. The complexity raises the price for both development and construction. Furthermore, a simplistic approach might in some cases be faster, since the developers can put a greater focus on making the few, simple commands very fast. Finally, a complex processor has many transistors and thus draw a lot of power. Because all of this, the market for cheap RISC computers (primarily for use in embedded systems) has grown very large. [John L. Hennessay, 2007]

The new wave of RISC-computers utilise modern technology for manufacturing processors. Some examples are the PIC-series from Microchip Technology and the AVR-series from Atmel which are both simple in design. All the models in the PIC series, for example, have less than a hundred different instructions. The simplest models in the series have just above 30, and all of them, except branching (comparison and jump), uses a single cycle to execute. A third example of this development is the ARM architecture from ARM Limited, which is a 32-bit architecture capable of running, for example, Linux or Symbian and is used in all kinds of equipment, perhaps most prominently in mobile phones. [Microchip, 2008][Atmel, 2008][ARM, 2008]

Besides computer power now being inexpensive and power efficient, development for these microprocessors has become much simpler than before. Both PIC and AVR-processors can be programmed in C and debugged in realtime on the chip by connecting it to a personal computer via a USB-module. In most development where computers are involved, a very large proportion of the costs are related to software development, so this aspect is also important.

The development of cheap and power-efficient processors help to explaining the increasing number of embedded systems in everyday equipment. An average new car

might often contain a number of computers in the order of 50. Most new electrical equipment, such as: TV:s, stereos, and microwave owens, contains embedded computer systems. In a way, intelligent houses is the natural, next great step. When there are computers controlling all kinds of equipment, why not connect it all together in order to find new ways of improving our lives. This is what this master thesis is about. Finding new ways of combining and controlling existing equipment to improve everyday life. The term intelligent houses, or home automation, might even be a bit confusing here, since these kinds of systems are just as good for offices and apartments as they are for houses. [John L. Hennessay, 2007]

Finally another aspect is also important and necessary for the market of intelligent houses to grow; people must want them and understand the technology. Ten years ago, this might not have been the case, but today most people in western society are used to handle modern technical equipment, like computers, mobile phones, televisions with surround sound, DVD-players, digital television boxes and climate control systems for cars as well as living quarters. The step to controlling all these things, and less obvious things like lamps, water heaters, ovens, timers, alarms, baby monitors and so on, from a cell phone, an LCD touch screen on the wall, or simply by walking in to a room is not that far away.

All the things above have contributed to make the development of intelligent houses, aimed at normal consumers, possible and to create a new market for such systems.

2.4 Existing intelligent house systems

There are companies that specialise in home automation and intelligent houses and there are standards which are designed for, or can be used as, a basis for home automation. There are many ways to do this kind of automation and the solutions differ. So far, most companies and standards have aimed at automation of one or a couple of areas within the whole field of home automation. It is a way too large area for one company, standard or organisation to solve every aspect of intelligent houses and home automation.

Interconnection

Standards like USB and FireWire are high speed serial interfaces mainly used for machine to machine communication (see Figure 2.1). Classical examples are mouses and keyboards for computers, digital media like audio and video, and external hard drives. They can also be incorporated into a system for intelligent houses.

These standards face competition from short range wireless alternatives, like Bluetooth and IrDA. These can be used for much the same things as USB and FireWire, but have a lower bandwidth. Bluetooth also utilises something called profiles, which define the purpose of a connection. Examples include wireless communication with a headset or computer peripherals like keyboards and mouses. IrDA communicates with infrared light. This has the obvious problem of needing a clear line of sight.

Other possible standards for data transfer within a intelligent house system includes Ethernet, WiFi and HomePlug. Ethernet is normally associated with computer networks, both home and office networks and the Internet. For further information see section 4.3. WiFi is what people normally refer to when talking about wireless home and office networks and hot spots for personal computers. HomePlug is a standard for data transfers using power lines.

These standards are not in themselves connected to home automation, but could easily be used for this purpose. Our demonstration rig, for example, uses Bluetooth

Figure 2.1 FireWire connectors to the left and USB connectors to the right.

to connect mobile phones to the system. The phone can then be used to control the intelligent house. Ethernet is used for the computer GUI and the mp3-player. There are vast numbers of standards that could be used for communication in a system for intelligent houses. The ones represented here are only a few of the larger ones.

Automation nets

There are many standards in the area of automation nets. Some of the more well known are Lonworks, KNX, X10 and ZigBee. These net can be used in many ways and have slightly different purposes.

Lonworks is a standard built on a protocol from Echelon Corporation and is used for control purposes like lamps, ventilation, climate control and so on. It utilises a number of standards for communication, like twisted pair and fibre optics. It is fairly complex and focuses on reliability and easy maintenance. It is mainly used for automation in apartment buildings and office buildings.

KNX is maintained by the Konnex Association and uses several means of communication, including radio, Ethernet, twisted pair and power lines. K10 is an industry standard, first defined by Pico Electronics, used for communication between devices in home automation. It uses radio and power lines for communication and has become popular because of its affordability, easy installation and the number of applications available.

ZigBee is based on the IEEE 802.15.4 standard and maintained by the The ZigBee Alliance. It is a short range, wireless, low bandwidth, low power consumption, ad hoc, self healing, mesh network standard (see Section 4.3 at page 28). The standard is ideal for sensory networks and other low data rate applications where power supplies are not necessarily available. As with the standards for interconnection, there are many more standards for automation nets. These are just a few examples.

(sources needed)

Companies and organisations

There are companies and organisations all over the world that use the above presented standards, and many others to provide solutions in the area of intelligent houses and home automation. For that reason, this section will concentrate on companies and organisations that operate in the same area as ÅF, namely southern Sweden.

The purpose of this master thesis is to devise a standard that is easy to implement, dynamic, flexible and transparent. It is supposed to open the possibility for third part developers to integrate their system with this one in an easy way. This does not

only include developers of equipment like coffee makers, stereos and so on, but also developers that specialise in different subsets of home automation.

In this region there are several companies that can be considered to be within the field of home automation. Most of them, however, put their focus in the sub area of environmental systems like: heating, water, ventilation and air condition, often in combination with other systems that are particularly interesting for office and apartment buildings, for example: Light, security, fire alarms and elevators. This field is sometimes also referred to as building automation.

Examples of such companies in southern Sweden are TAC and Regin, both dealing with building automation as described above. [tac, 2008][Regin, 2008]

If a system like the one proposed in this thesis was to be developed for commercial purposes, these kinds of companies would not be part of the competition, but rather potential allies. However, there are also companies that specialise in more general systems for intelligent houses. The most common tactic seems to be to create a system, either from scratch or based on existing standards, and then keep it locked.

This approach has the obvious benefit that the company can make more money from every sold unit as the only distributor. It also has the advantage that it is relatively easy to guarantee the reliability of the systems. The drawbacks are also quite obvious. The amount of functionality will be limited, to say the least, and many inventive applications of the concept and technology will likely never be thought of.

There is one other system that has goals similar to those of this thesis. It is the automation net mentioned above, in Section 2.4, called KNX, that is completely open for its members, but besides this, the competition for a system like the one proposed in this thesis is negligible today. The solution in the thesis has other problems though. To be successful it needs considerable backing, either from a devoted community, financial capital or both. It needs to become a market standard with weight enough to make other companies and communities want to adopt to it, or it will just end up being another limited standard with a couple of modules that are neither revolutionary nor particularly interesting as a basis for future development. [KNX, 2008]

3. Design choices

3.1 Uses and targets of home automation

Since the concept of home automation is so broad, it is almost impossible to come up with a detailed specification of what a system for intelligent houses should be able to do. As has been stated before, the main purpose is to interconnect all kinds of devices in a house to provide easier control, automatic regulation and new possibilities.

One way to approach the task of building a system for intelligent houses would be to simply identify a number of tasks that could be useful and then build a system to perform these tasks. This has already been done, however, and the obvious limitations of such a system make it doubtful that it could ever bring on the kind of revolution in thinking of technical equipment that is associated with intelligent houses, and necessary for the market segment to grow large.

Therefore this thesis argues that a new way of thinking in the area of home automation is necessary if it is ever going to go beyond small systems for people interested in high tech equipment and systems related to management of larger buildings. The solution proposed in this thesis is a very adaptable and general structure that focuses on dynamic behaviour, flexibility, open interfaces and transparency, specifically targeted at easy adaptation and development for third part manufacturers. These include producers of any kind of equipment that can be controlled, existing systems of intelligent houses and companies and communities that specifically develop, for example, control applications and GUI:s for the system.

The target of this system is thus potentially large. It is inexpensive (because of the platform independence it can also be run on very inexpensive hardware), easy to set up and both usable and affordable for any normal person with an average income. It is scalable so that it can support large systems, for example in a company, an office building or an apartment building. The only limitation is the development of devices and protocol modules that adheres to the standards proposed in this thesis.

One possibility would be to make a system like this open source, in order to make it develop and adapt faster to what people actually wants. It will give birth to new ideas that no single organisation could ever come up with.

As for the test rig that has been built as a part of this thesis, focus has mainly been aimed at building end devices, control devices and protocol modules that can demonstrate the possibilities of the system architecture. The choices would not necessarily be the same if the purpose was to develop devices and modules for a commercial system, even though the ones developed probably would be marketable if redesigned for that purpose.

3.2 Base system requirements

In order to build a system as flexible and adaptable as the one proposed by this thesis, many considerations has to be taken into account. In the next section a list of possible uses for intelligent houses is presented. The base system has been designed to be able to handle all those tasks if they were to be implemented. Indeed, the goal has been to create a system that could be used for almost any kind of equipment imaginable. Furthermore, a necessary part of the design is the possibility to build control interfaces

that can handle arbitrary devices, even if the devices did not exist when the interface was built.

Even though this makes it possible to control everything in the system, the base system must also provide the possibility for custom control devices aimed specifically at control of certain types of end devices. In order to make this possibility as dynamic as possible the system supports a system for device types that specify the possible states and commands of all devices of that type. One device type could as an example be music players, and any specific music player end device could be constructed to adhere to the standard of that device type.

In order to make the base system more attractive, it also contains some base features. It supports users and the possibility to connect users to control devices or for example RFID-tags. In order to provide automatic control, the base system supports the possibility to map all the rooms, and their connection to each other (typically doors), in which the home automation system operates. Devices and users can then be connected to this room representation and equipment to keep track of them can be built. In the prototype, RFID-readers, that could be put in every doorway of the house, are used to sense if an RFID-tag connected to a user or device is moving between the rooms.

The base system also contains support for the execution of macros, which provides the possibility to execute several commands in parallel or sequentially, make commands execute at certain dates and times or after a specified time interval. Furthermore commands can be executed upon the fulfilment of certain conditions, for example a user walking in to a room.

3.3 Method for design priority

In order to build our test rig and to anticipate the demands that will rise on what a system like this should be able to do, a number of possibles uses for an intelligent house was compiled using brainstorming. Here follows a list of the result:

Alarm	Reminders and alarms both at certain times and upon fulfilling certain other conditions.
Communication	Internal (and possibly external using IP-phones or something similar) communication between rooms, using some kind of control panels with microphones/speakers.
Control Panel	Control panels that can be put in several rooms in the house. Perhaps with an LCD screen and a GUI to control the home automation system, but also for example to stream and decode music for the audio system of the room and handling internal communications, alarms and so on.
Electronic equipment	This is a very general field, but aims mainly at controlling simpler electronic equipment that is normally found in a modern house, like coffee makers, ovens or washing machines.
Energy saving mode	The ability to put the house in a mode that minimises the amount of energy used, for example when going on vacation. This is similar to powering down the house, but more severe. For example the heating/air condition can be set to inactive unless some critical temperature, that risk to damage the house, is reached.
Environmental control	The ability to control and monitor systems for environmental control, such as ventilation, heating and humidity.
Front door scanner	This can be used together with localisation to tell users if they have forgotten something important, like the wallet, the mobile phone or the keys and also warn if a non-user enters the house.
Lamps	The possibility to turn on and off lamps, possibly with dimmer functionality.
Light control	Controlling the lighting of a room or area using light sensors, lamps and possibly motorised Venetian blinds.

Localisation	Technology to keep track of where users, phones, keys and so on are in the house.
Mobile phone GUI	A GUI to control the intelligent house using a mobile phone.
Music follows user	Music that is being played using the audio systems of the house should follow the user that started it.
Music from phone	Being able to stream music from a phone to a stereo, using Bluetooth or a similar wireless technology.
Music from server	The possibility to stream music from a central server and play it in the room where the user is at the moment.
Opening/closing windows	The possibility to open and close windows using actuators. Could be used in conjunction with the window sensors, the front door scanner, Powering down the house and the heating/air condition and ventilation systems.
Phone as remote	Using the mobile phone as a remote control for equipment such as television, stereo and DVD-players.
Powering down the house	The possibility to close down the house with one command, for example when going to work. This would mean that lights are turned off and perhaps also the water boiler and similar energy consuming equipment. It could also be a safeguard that makes sure that the stove and similar, possibly dangerous, equipment are turned off.
Timer	Timer to provide countdowns, for example when cocking. Could be tied to a user and notify with light/sound in the room where the user is at the time.
Turn on computers	The possibility to turn on the computer, using for example a mobile phone.
WC Smell sensor	Automatically turns on the ventilation when it gets odorous.

Window sensors	Sensor in the windows of the house to see if they are open or not. Could be combined with the front door scanner to warn users that windows are open when going out.

These possible uses for an intelligent house needed to be prioritised, both for judging which aspects of the base system that are most important and to decide which functionality is most important for the prototype. The prioritising has been realised by grading the different functions in five different, weighted areas on a scale from one to five.

The three first areas were meant to represent different reasons why intelligent houses might be attractive to consumers. They are: 1. Saving cost, environment and energy, 2. Making everyday life easier, and 3. Making the house seem luxurious and high tech.

The two last areas were meant to reflect aspects related to the test rig. They were: 4. Difficulty to implement, and 5. Relevance for the system as a whole. The weights were determined as follows:

Area	*Weight*
1	0.4
2	0.4
3	0.4
4	0.7
5	1.0

The higher weights for area four and five are to emphasise our limited time to finish the test rig and the importance of showing as much of the capabilities of the base system as possible. These weights would probably be different for a commercial system.

There is no simple way of judging what grades the different possible uses of the system should get in the five categories, without comprehensive surveys and the help from marketing experts. Therefore the grading has been made based to the judgement and experience of the authors. These were the results:

Area	*1*	*2*	*3*	*4*	*5*	*Grade*
Powering down the house	5	4	3	5	4	12.3
Lamps	4	3	3	4	4	10.8
Mobile phone GUI	1	5	5	2	5	10.8
Localisation	1	5	4	2	5	10.4
Electronic equipment	2	4	4	3	4	10.1
Environmental control	4	2	2	4	4	10.0
Energy saving mode	5	2	1	5	3	9.7
Music from server	1	3	5	4	3	9.4
Control Panel	1	1	4	2	5	8.8
Music follows user	1	3	5	3	3	8.7
Alarm	1	4	2	4	3	8.6
Window sensors	3	3	2	4	2	8.0
Front door scanner	1	5	2	2	3	7.6
Timer	1	4	2	4	2	7.6
Opening/closing windows	3	4	3	3	1	7.1
Light control	3	2	4	3	1	6.7
WC Smell sensor	1	3	5	3	1	6.7
Phone as remote	1	5	4	2	1	6.4
Communication	1	3	3	2	2	6.2
Turn on computers	1	3	2	4	1	6.2
Music from phone	1	2	5	1	1	4.9

In order to decide what standards of communication to implement, an approach similar to the one used above to determine the uses of an home automation system, is needed. First a list a possible standards was compiled. There are of course many more that could be used, but an initial limitation had to be done.

Bluetooth	Bluetooth is a protocol designed for wireless, short-range, low power transfer. It is used in many different areas such as mobile phones and laptops and can reach transfer rates up to 3 Mbit/s. [C Bala Kumar, 2003]
CAN	**Controller-Area network**, is mainly used to communicate between microcontrollers. There are different standars for CAN-busses, with both one or two wires, and transmission speeds between 50-1000 kbit/s and a range between 40-1000m. [in automation, 2008]
EIA 485	Two-wire, multidrop, serial transmission. Can reach speeds of 10 Mbit/s with a 1200 m cable length. [Axelson, 2005]
Ethernet	This is probably the most widely spread standard used for connection between computers. With 1000BASE-T, 1 Gbit/s transfer rate can be reached with a maximum cable length of 25 meters or a range of 100 meter with 100BASE-T at 100 Mbit/s. [George Coulouris, 2005]
FireWire 800	FireWire is primarily used to connect digital video cameras and other devices needing high bandwidth connection to a computer. It has a capacity of 3.2 Gbit/s and a range of 100 meters with the latest specification. [Axelson, 2005]
GPRS	**General Packet Radio Service**, is used by mobile GSM telephones to transmit data with speeds up to 144.4 kbit/s. The telephone can constantly be connected to the server and only pay for the transmitted data. [Clint Smith, 2002]
IrDA	**Infrared Data Association**, is a protocol used by remote controls and mobile phones for wireless transmissions of data with speeds ranging from 2.4 Kbit/s to 16 Mbit/s depending on the equipment used. It has a limited range up to 2 m. [Axelson, 2005]

USB 2.0 — **Universal Serial Bus**, is probably the most widely used protocol for connecting devices to a computer with a cable. It can reach transfer rates of up to 480 Mbit/s with USB2, but the cables have a restriction to a max length of 5 meters. [Axelson, 2005]

WiFi — IEEE 802.11 is a wireless TCP/IP based network standard with a maximum speed of 108Mbit/s. Widely used for computer-computer and computer-internet connections in homes, offices and public hotspots. [Houda Labiod, 2007]

ZigBee — ZigBee is a medium range, wireless, low speed, self healing mesh network standard based on IEEE 802.15.4 and managed by the ZigBee Alliance. It's routing capabilities makes its max length basically as long as needed for uses in home automation. [Houda Labiod, 2007]

To rank these standards, six categories have been chosen. Every standard will be given a grade from 1 to 5 in each of them. These categories are: 1. Easiness in installation, 2. Easiness of implementation, 3. Speed, 4. Range, 5. Availability, 6. Price and 7. Overall importance for the prototype. Each of these categories have been given a weight to reflect their importance in the prototype. For a commercial system, these wights might have been different.

Area	*Weight*
1	0.4
2	0.6
3	0.1
4	0.2
5	0.6
6	0.4
7	1.0

There is no easy way of exactly determine the grades for each standard and category. That would require far deeper studies. These are only estimates.

Area	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*Grade*
Ethernet	3	5	5	4	5	4	5	15.1
Bluetooth	4	4	3	2	4	4	5	13.7
ZigBee	5	4	2	4	2	5	5	13.6
WiFi	4	5	4	3	4	3	3	12.2
IrDA	5	4	4	1	2	3	3	10.4
GPRS	5	4	1	5	2	1	3	10.1
EIA 485	2	4	1	4	2	4	3	9.9
CAN	2	4	2	4	2	3	2	8.6
USB	1	1	4	2	5	2	2	7.6
FireWire	1	1	3	3	2	1	1	4.5

Because of the time limitations for this thesis, only the three first communication standards will be implemented. These standards also show the potential of the prototype in a good way.

4. Prototype

This section explains the prototype system in detail. Both the overall functionality and its different parts.

4.1 System design

The design choices can be divided in two categories. First there is the overall design and structure of the system. Here the primary design goals have been to make the system dynamic, flexible and transparent. The system must be dynamic so that it is easy to manage for the user. Adding a new device should not require a technician to come to the home. Flexibility is important if third party manufacturers are to make their equipment compatible with the system. It is impossible to anticipate what purposes the system will be used for in the future. Finally, transparency is important largely for the same purposes. Others must be able to build on the system. If every device was made by the same manufacturer, the system would become expensive, unimaginative and very limited. For this reason the message standard has been designed to be powerful, yet easy to understand and adapt to.

In order to facilitate the fulfilment of these objectives a layered design structure has been chosen. It consists of three layers: A server layer, a communication protocol layer and a device/client layer. The server routes all communication and provides functionality to register protocols, devices, device types, users, macros and rooms in a database. Devices can also access the information in the database by sending requests to the server. Furthermore the server handles macros, localisation and some other basic functions of the system.

Outside of the server layer comes the protocol layer with independent modules, implementing different communication protocols (in our case TCP, BlueTooth and ZigBee). This layer is transparent and a third party could create new modules. All communication between modules and the server is done via TCP.

The devices (clients) can then connect to protocol modules following the standards of the communication protocol. This structure and the database makes it possible to build completely dynamic GUI:s and control units that can handle not only existing device types of the system, but future ones as well. Of course the transparency also makes it possible for a third party to create new end devices for the system. For a schematic overview see Figure 4.1.

Communication

The system uses a message protocol that looks much like XML, with tags. XML was not used, because many of the devices have very limited computational capacity and it is not realistic to implement XML parsers for them. There should, however, not be any problems sending messages to the system that fully adhere to the XML-standard. In the ZigBee communication, this message protocol is converted to a much more compact format because of limited bandwidth and computational power. The basic massage structure looks as follows:

```
<message>
<sender></sender>
<receiver></receiver>
<messageid></messageid>
```

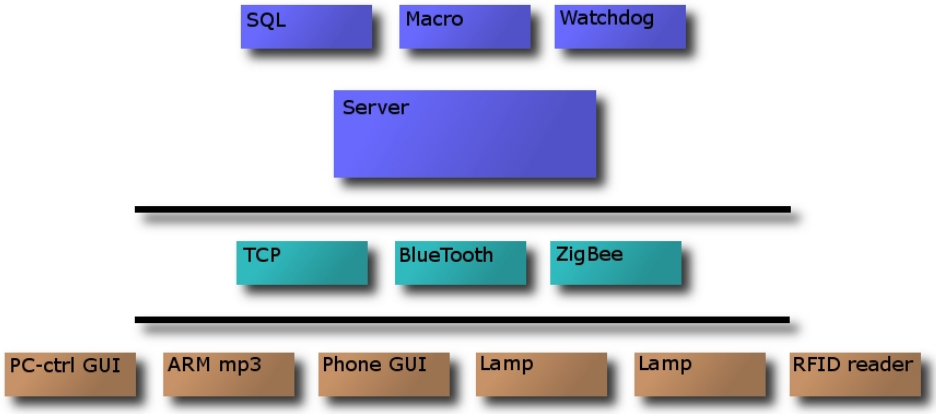

Figure 4.1 A schematic overview of the 3 layer structure of the system.

```
<messagetype></messagetype>
<timestamp></timestamp>
</message>
```

These tags must be in every message sent and their content is always an integer number. If, for some reason, the sender is unable to provide a significant value to any of these tags, -1 is to be sent instead. Depending on the message type, other tags can be added to the messages. The sender and receiver contain the unique integer number that is used to address devices in the system. The message ID is used primarily to separate answers to multiple requests from one source. The standard is to always use the request message ID in a response message. The time stamp is an integer with the time in milliseconds since 1970 and the message type facilitates identifying what kind of message it is. The predefined message types are:

Messagetype	*Number*
CommandMessage	1
DataMessage	2
ErrorMessage	3
NewDeviceMessage	4
AckMessage	6
NewProtocolMessage	7
NewMacroMessage	10
NewDeviceTypeMessage	11

A CommandMessage, for example, has two extra tags `<command>` and `<args>`. For more details on the message types and predefined ways to use messages, see Appendix A.

When a new communications protocol module is started, it connects to a TCP server socket in the server. Once a connection is established, the module is required

to send a NewProtocolMessage to the server. This adds the new protocol module to the server database and enables the server to send and receive messages from the module.

Once the module is registered, devices can connect to the module, using whatever means of communication supported. Once connected, the device is required to send a NewDeviceMessage. Since the server is responsible for giving the device an ID, this message will have the sender set to -1. The protocol module will then change that to a temporary number, and also add a tag with the protocol to the message. If the device has lost the connection to the server, and tries to reconnect, it can add its old device to the message and re-register.

If the NewDeviceMessage is sent correctly, the device will get one of two responses from the server. If the device type of the device already exist, a DataMessage with the tag datatype 0 (NEW_DEVICE_ID) will be sent from the server. A tag, newdeviceid, will contain the ID of the device. This message is also an acknowledgement that the hand shake is done and that the device now is registered in the server. After this, every message sent by the device must have its device ID in the sender tag.

If the device type of the device is not already registered in the server, an ErrorMessage with error code 6 (DEVICE_TYPE_UNKNOWN) will be sent instead. If this happens, the device must send a NewDeviceTypeMessage and then send the NewDeviceMessage again. If the messages are accepted correctly, the above mentioned DataMessage will be sent from the server and the hand shake is over.

Another common message exchange, is the one that follows when a command is issued from a control device, for example a computer GUI, to an end device, for example a lamp. The first thing that happens is that a data message of type command is created in the GUI, sent via the server that redirects it to the lamp. The lamp gets the message, executes the command and then send a state updated message to the server. The server updates the state in the database, and then sends out notification messages to all macros and control devices, see Figure 4.2.

4.2 Server

The server is the central hub of the system. All communication goes through it, even for device-device communication. This of course creates a demand that every message has a correct receiver ID, and if something goes wrong a correct sender ID for error messages.

The server has a number of purposes. It routes messages, registers protocol modules, devices, device types, users and macros in the database, handles the execution of macros and provides commands to access information from the database. It also keeps track of users and handles related functionality with the RFID localisation. The server itself is written in Java as a console application. This keeps it platform independent and Java makes for fast and easy development. (see Figure 4.3).

All communication between the different threads in the server is handled by sending messages through bounded buffers. The different message types used for communication in the system are converted to message objects in the server. The server is built around a main system thread and all messages and communication goes through it. If a message is sent to the server, the receiver device id is used to find the device in the database and determine its protocol module. The input buffer to the socket that handles the protocol module is then picked out of a sorted list and the message is delivered.

The server itself has the device id 0. If a message is sent to the sender 0 the

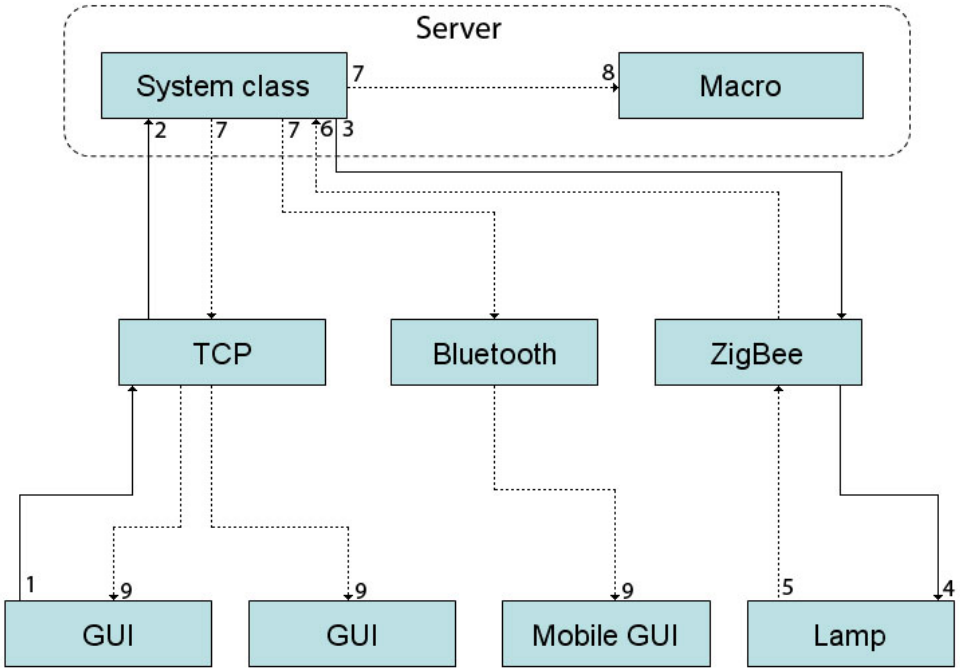

Figure 4.2 The message paths generated when a command is issued from a computer GUI to a lamp. The numbers represent the chronological order, starting with 1.

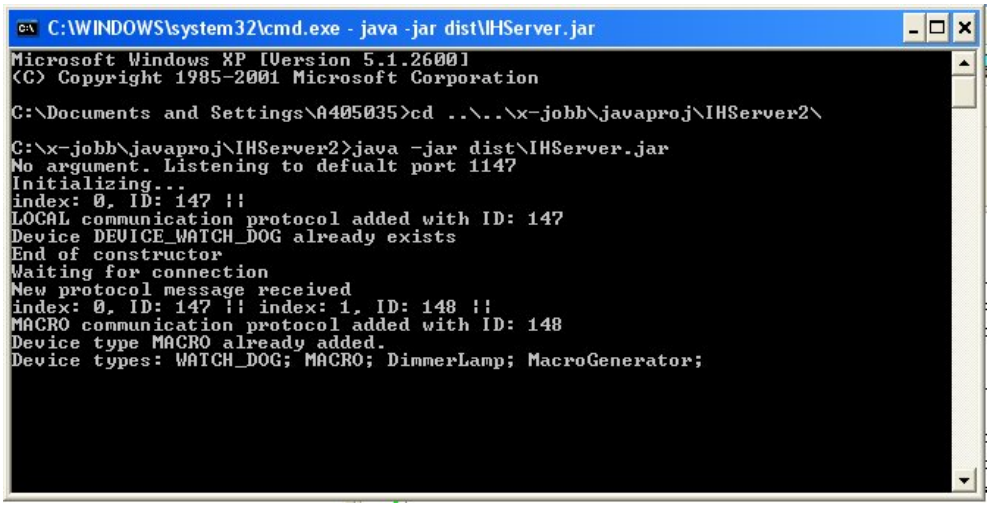

Figure 4.3 The console when starting the server.

server handles it. This is the case for database related messages, such as searching for, or managing, devices, device types, protocols, users or macros. This means that the server has a protocol module for handling messages internally between threads as well and each macro is treated as a device. See Figure 4.4 for a schematic overview.

In its most basic functionality, a device register its device type in the database. The entry consists primarily of its possible states and which commands and arguments the type accepts. Then the device registers itself in the database and the device ID is stored together with the actual states of the particular device. A control device (for

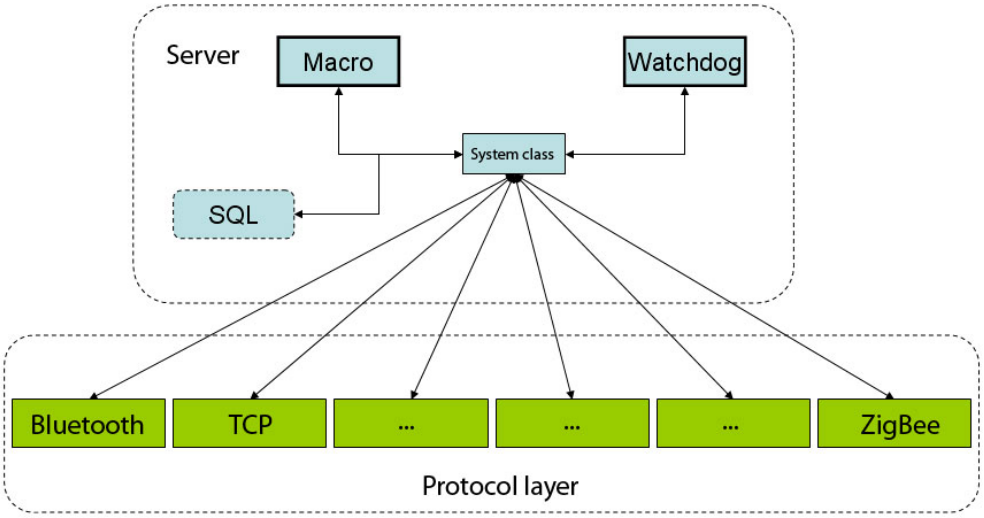

Figure 4.4 Schematic overview of the server, it's components and the protocol layer

example a mobile phone GUI) can then access the database and get information about the states and possible commands of a device and dynamically generate a control system.

The whole system is designed to be dynamic and flexible. It is impossible to think of every application for an intelligent house, and therefore, the focus has been to make it easy for a third part to create new devices, control devices and communication modules. The goal has been to make the system flexible enough to handle almost any kind of device and control.

To make the system become useful we have also decided to add some basic functionality that is critical for many applications of an intelligent house. The most obvious such feature is localisation. Many possible devices can benefit from a system for localisation of users and keeping track of where the devices actually are. For example, getting a list of every lamp in your house on a mobile phone and then being able to turn them on and off is not that useful, but if there is a macro that automatically configures the lightning in the room where the user is at the moment, it is quite an attractive feature.

Perhaps even more important, if the system keeps track of where users are, it can shut down the lightening and other systems in rooms that are empty. This of course goes back to the earlier discussion about the purpose of the system, mainly saving costs and energy, which in turn makes the house more environmentally friendly.

Macros

Macros are series of commands and other actions. In our case, a macro is a text file with a script language allowing the user to specify several device commands to be executed, setting alarms and timers and making things happen on events, for example that a user walks into a room. This is the basic structure of a simple sequential macro, without any event handling:

```
CONSTANTS BEGIN
CONSTANTS_1=DEFAULT_VALUE
..
CONSTANTS_n=DEFAULT_VALUE
END
```

```
MACRO BEGIN
COMMAND_1(ARG_1, .., ARG_n)
..
COMMAND_m(ARG_1, .., ARG_k)
END
```

The constants can be used to replace values, numbers or strings, that are used often in the macro. If a constant declaration is finished with a semi colon (;) the user is not allowed to change the value when starting the macro. Otherwise the default value is used unless the user changes it.

In the section called MACRO, the actual commands are specified. There are a number of different commands, and if a command is finished with a semi colon (;), the macro will wait for an acknowledge message before continuing the execution. Here follows a list of the commands implemented in the prototype:

CMD_DEV	**(receiver, command, arg 1, ... , arg n)** Sends a command 'command' to the receiver (device id or name) with arguments.
CMD_TYPE	**(devicetype, command, arg1, ... , arg n)** Sends a command 'command' to all devices of device type (device type name) with arguments.
WAIT	**(time)** Waits for time seconds.
WAIT_MS	**(time)** Waits for time milliseconds.
WAIT_UNTIL	**(hour, minute, second)** Waits until the next occurrence of the time specified. hour[0..23], minute[0..59], second[0..59].
WAIT_UNTIL	**(week, hour, minute, second)** Waits until the next occurance of the time specified. week[0..6] (0=sunday), hour[0..23], minute[0..59], second[0..59].
WAIT_UNTIL	**(year, month, day, hour, minute, second)** Waits until the time specified, must be later than current time. year[xxxx], month[0-11], day[1..max_nbr_days_of_month].
USR_IN_ROOM	**(user, room)** Waits until the user (id or name) enters the room (id or name).
RESTART	() Restarts the macro.

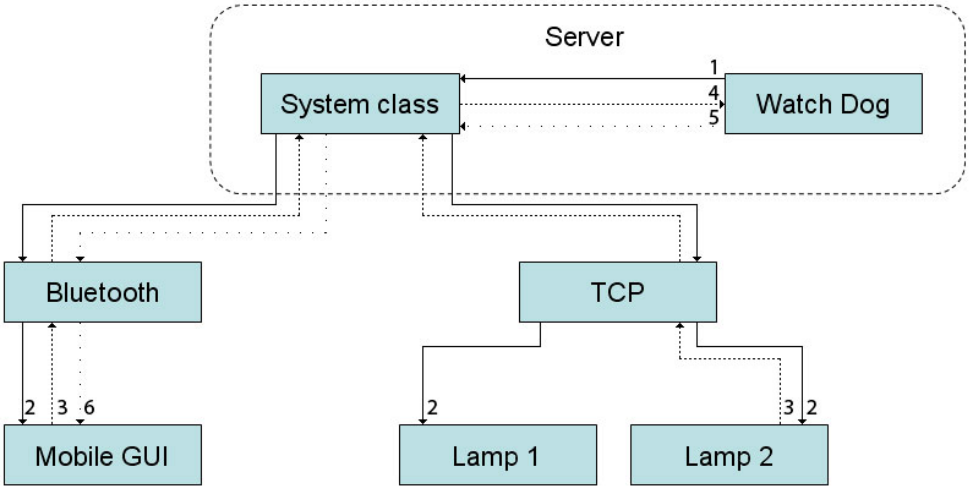

Figure 4.5 The message paths of the watchdog.

With these commands, it is possible to create custom settings for each user and room. For example individual lightening settings and music. It is also possible to create timers and alarms that are connected to both sound and other means of notification, lights for example. It also makes it possible to make large macros that affects lots of devices. For example turning of all unnecessary equipment when no one is home or put the house in a shut down mode during vacations and so on.

Watchdog

The watchdog sends out commands to all devices in regular intervals and wait for the devices to answer with an acknowledge message, see Figure 4.5. Failure to answer three times in a row will result in the device being set as passive. This means that it will not show up in GUI:s and cannot send any messages until it has re-registered.

In the ideal case, the watch dog is unnecessary, because the devices are supposed to deactivate themselves when shutting down. If the connection suddenly is broken, this might not be possible, however, and the watch dog makes sure that the database does not get flooded with old inactive entries that does not correspond to any physical devices.

Users

Users are implemented in the server SQL database with an id, a name and a room. This structure is used primarily for the localisation functionality in the system, together with macros and the room and RFID-reader representations.

As described in Section 5 on page 41, users would be of greater importance if the system was developed further.

Rooms

The different rooms of the home are represented in the SQL database with an id and a name. These are used together with the users, the RFID-reader representations and the macros. The structure is mainly used for static referencing after the initial configuration. See Figure 4.6 for the room configuration used in the prototype for demonstration.

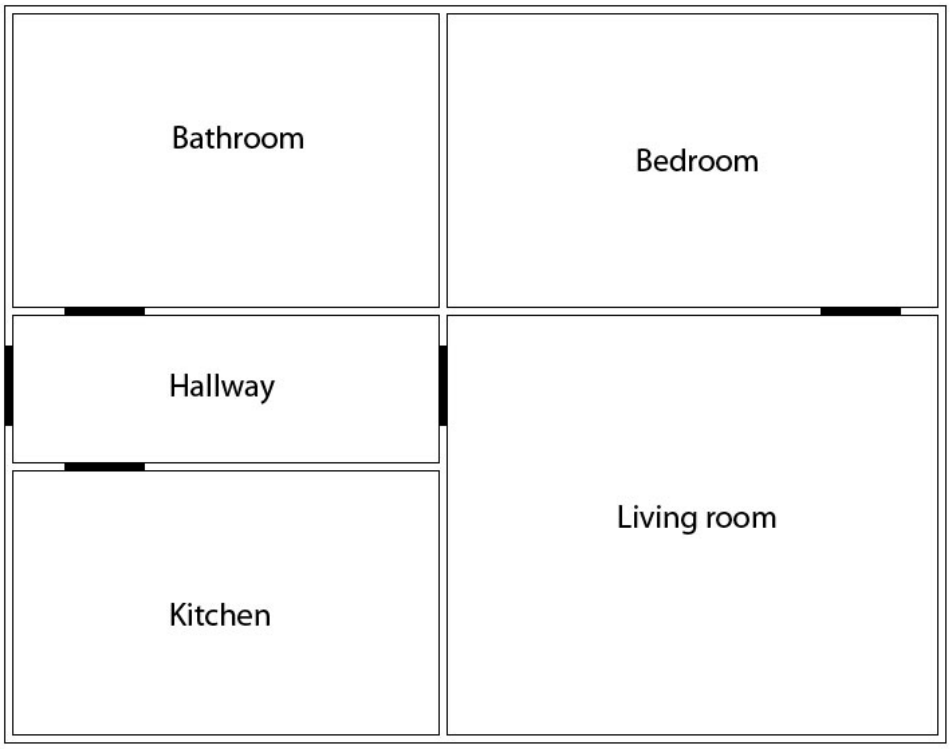

Figure 4.6 The room configuration used in the demonstration of the prototype

Doorways

The doorways are represented in the SQL database with an id, a name and the two rooms joined by it. A doorway detection device, in this case an rfid-reader, is associated with each doorway.

When a user goes through a doorway, the device notifies the server, which pulls the rooms of the corresponding doorway from the database. It then changes the room of the user and notifies the control devices and macros.

4.3 Communication Protocol Modules

The middle layer of the system is the communication protocol layer. This has been designed as a transparent layer mainly because of two reasons: diversity and adaptability. Figure 4.7 shows a schematic overview of how a communication protocol module, in this case for Bluetooth, works. Figure 4.4 shows the Communication Protocol Layer in relation to the server.

A resource heavy device, like an advanced control system, a media device or perhaps an indoors climate regulatory system might need high bandwidth communications to function, while a simple lamp or a sensor is more dependent on flexibility and easy installation. In the first case a wired (ethernet) TCP connection might be a good choice, and in the second a flexible, low cost, low power consumption, wireless network is probably a better solution.

Adaptability is important since the standards of communication are changing and developing all the time. For a system like this to be able to stay competitive, it must be able to evolve in parallel with new techniques and possibilities.

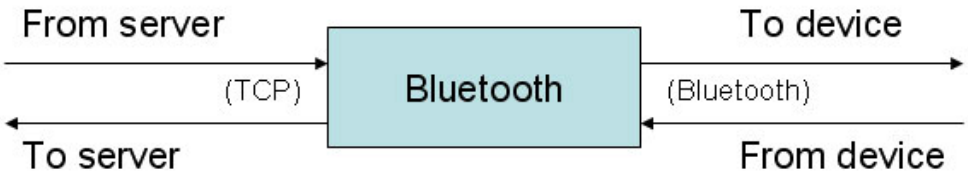

Figure 4.7 Schematic overview of a Bluetooth communication protocol module

In order to show the diversity and the possibilities of the system, three main communication protocol modules have been implemented. These are TCP, ZigBee and Bluetooth.

TCP

TCP (Transmission Control Protocol) is a protocol designed for high bandwidth and is one of the dominating standards for communication between computers worldwide today, both on the internet and in home/office networks. It has high reliability and guarantees in-order-delivery of data, which makes it ideal for file transfers. Because it is a wide spread standard, it has support on basically all larger operating systems and platforms.

In the rig, TCP is used as the protocol for Ethernet, used by the computer GUI and the mp3-player that both require a fast connection. The module itself is built as a console Java application just like the server, and for the same reason: platform independence and easy development.

In a standard system, the TCP protocol module application will probably run on the same physical machine as the server, but it is capable of running on a separate machine. When started, it opens a client socket, connects to the server via TCP and register in the server database. It then opens a TCP server socket and waits for devices to connect. When devices connect, it routes the communication between the devices and the server.

ZigBee

ZigBee is a wireless protocol stack that adheres to the IEEE 802.15.4 standard for wireless personal area networks (WPANs). It is managed by the ZigBee alliance that consists of a number of companies and organisations. It is a low speed, low cost, low power, self healing, mesh network for short range radio communication. It is also built to provide secure communication. These properties make the standard ideal for communication with simple devices and sensor devices in a home automation system. [Houda Labiod, 2007]

The network is organised around three main types of nodes. A ZigBee network always has one coordinator that builds the network. The rest of the network consists of routers and end devices. A router can send and receive data, but also route data between other nodes in the network, while an end device only receives and sends data. End devices can also be configured to go into sleep mode to conserve energy. Self healing means that if a path of communication breaks down, the network finds a new way to get the data to its destination. [Houda Labiod, 2007]

In our current configuration, the ZigBee protocol module consist of two parts. The first is another console based Java application, much like in the TCP protocol module. It connects to the server via TCP, but there the similarities end. On the other

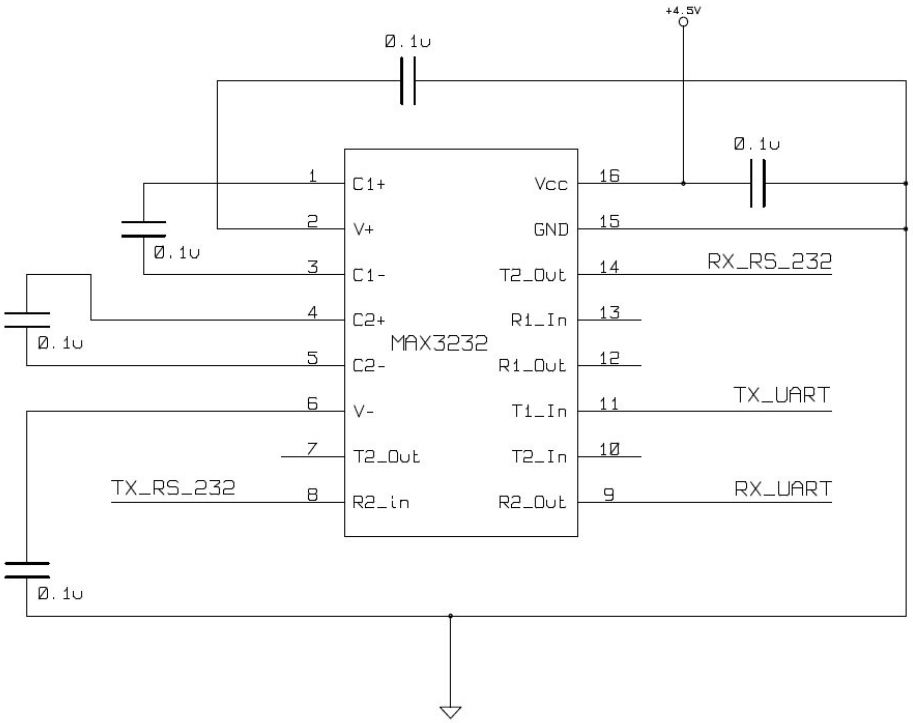

Figure 4.8 Schematic of the MAX3232 circuit that converts between the 3V signal from the AVR1281 and rs232.

end of the application, there is a serial link (rs232) communicating with a (AVR128) microcontroller with a radio link, acting as the coordinator of the ZigBee network. See Figure 4.8 for a schematic of the MAX3232 circuit and Figure 4.9 for a photo. The coordinator then routes the messages to the devices.

To facilitate the development of the ZigBee communication in the rig, a development kit called eZeeNet has been used. It consists of 5 identical modules, each with a AVR128 micro controller and a radio transceiver, see Figure 4.10. It also has a larger module with another AVR128 controller and a LCD screen, see Figure 4.11. The smaller modules can be connected on the larger one and this is used both to program the controllers via an AVRISP mkII (a USB module built by Atmel, the manufacturer of the micro controllers) and to make it possible to print things on the screen, see Figure 4.12

With the development kit, an API (application programming interface) that implements the ZigBee stack, and provides functionality for a complete ZigBee network followed. The API is in part based on Tiny OS and other functions inherent to the micro controller, such as serial communication, PWM, timers and the system clock are also implemented. Our software design is built on a heavily modified sample application that followed with the documentation.

Bluetooth

Bluetooth was first developed by Ericsson Mobile Communications, but in 1998 other companies, such as IBM, INTEL, Nokia and Toshiba joined, and created the Bluetooth special interest group (SIG), which now maintains the Bluetooth specification. It is designed for low power consumption, short range transfer between mobile devices such as cellular phones and laptops. The first devices using Bluetooth were

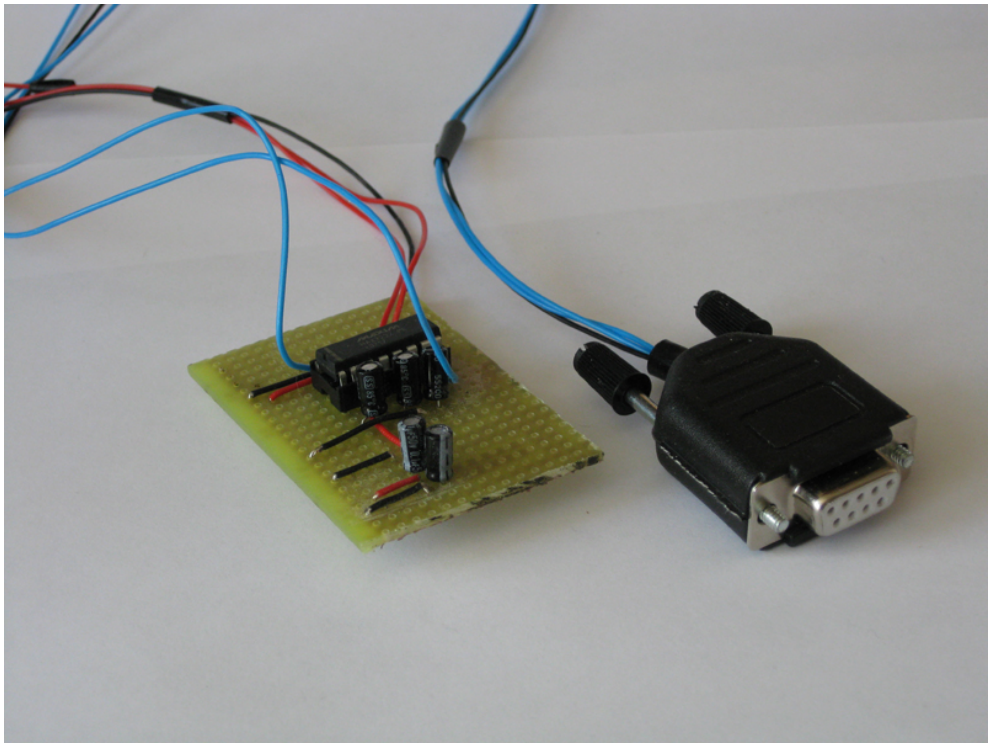

Figure 4.9 Photo of the MAX3232 circuit used to connect the computer with the ZigBee coordinator

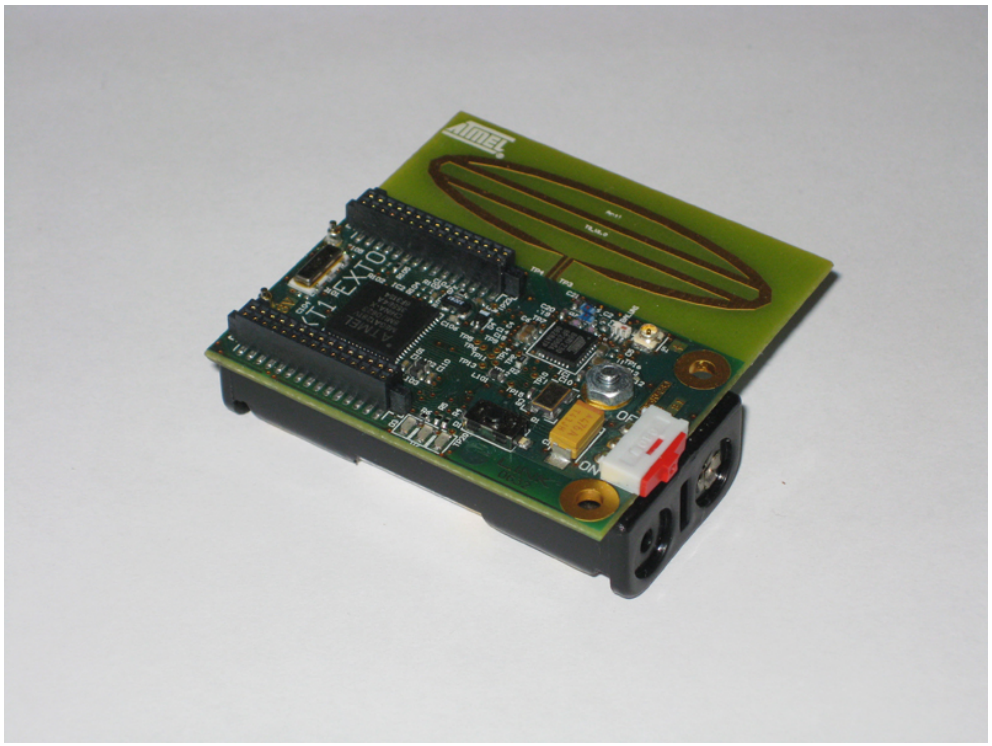

Figure 4.10 Photo of one of the AVR1281 ZigBee modules in the eZeeNet kit.

Figure 4.11 Photo of the LCD module in the eZeeNet kit.

Figure 4.12 Photo of the AVRISP mkII used to program the AVR1281 microcontrollers in the eZeeNet ZigBee kit.

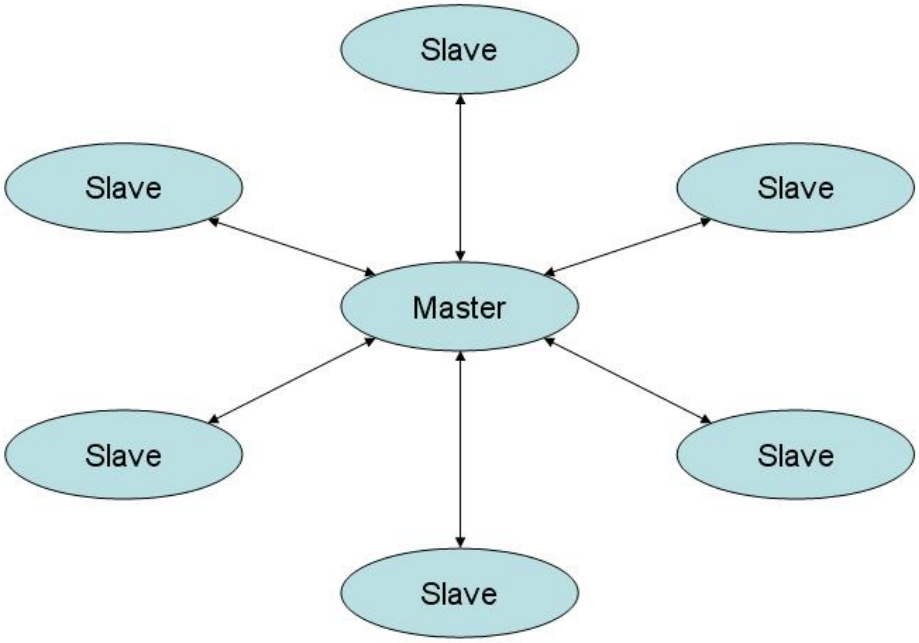

Figure 4.13 Schematic structure of a Bluetooth piconet

mobile telephones, but now the technology has spread to many areas, including gaming consoles and computer keyboards, to mention a few. [C Bala Kumar, 2003]

It is a multi-layer structure where the lowest layer defines its operating frequency of 2.4 GHz, the Industrial Scientific Medical (ISM) frequency band. The top layer (application layer) is a number of different profiles, specified by SIG, that are all are based on GAP (Generic Access Profile). This profile defines how to set up a connection between devices, discovery of devices and security. The profile used in this thesis is SPP (Serial Port Profile), which emulates a serial port connection between two devices. [Houda Labiod, 2007][C Bala Kumar, 2003][SIG, 2008]

Depending on the device and the class that is used, different range and data rate can be achieved. With class 2, that is used by computers and mobile telephones, a range of up to 10 metres and a data rate of up to 3 Mbps is theoretically possible. These numbers are specified for the current standard, version 2.1, which was released in 2007. [Houda Labiod, 2007][C Bala Kumar, 2003][SIG, 2008]

In a Bluetooth network, communication is based on a master/slave principle where a network, called a piconet, can address a maximum of 1 master and 7 slaves, see Figure 4.13. If larger networks are needed, several piconets can be connected into a scatternet, but so far, this ability is poorly developed. [Houda Labiod, 2007]

To utilise Bluetooth between the mobile telephone and the server, a module that routes messages between the two, TCP to Bluetooth and vice versa, is needed. The module is running on the same computer as the server software in the prototype, but this is no necessity, they could be run on different machines. The primary reason that Bluetooth is implemented in the prototype, is the possibility to connect a mobile telephone to the server and use it as a control device.

Because of time shortage and the complexity of implementing the Bluetooth protocol stack, Bluecove has been used. Bluecove is a freely available protocol stack running on JVM 1.1 or later, with a Java API provided.

4.4 Devices

The devices build the outermost layer of the design. The devices are all the parts of the system that affect the house or let the users control the system. The primary types of devices are end devices and control devices.

End devices can be anything that is to be controlled by the system or provides functionality, for example lamps, rfid-readers, environmental control systems and mp3-players. These are the sensors and the actuators of the system.

Control devices on the other hand are used to give the user control of the system. In our rig it's the mobile phone GUI (Graphical User Interface) and the computer GUI. These systems dynamically generate all the possibilities to control the system and the end devices for the user.

In theory, one device can be both a control device and an end device. As an example, the embedded Linux ARM9 board that we use for the mp3-player has a LCD display that could be used to display our computer GUI and act as a control device in a room of an intelligent house.

Computer GUI

The computer client is a Java based, platform independent application made to manage and control the system. It is used to control all devices, create macros and display the states from any device registered to the server. It is registered like a normal device but with different control and state options, and it is identified as a control unit in order to receive updates from the server.

The goal for this client is that it should be able to control every single application that is needed for an intelligent house, create macros to allow more advanced controlling and give a good overview of the different devices and their capabilities.

Upon registration the client gets a id from the server and then asks for all available devices, states and commands and then stores all necessary information locally. When a device changes a state, an update message will be sent from the server and the GUI will update its list. This is done to ensure that the GUI always have the latest information about the devices without having to make requests to the server every time a device's states are to be displayed. With only one controller it would not be an issue with the small amounts of traffic generated, but if the system grows big with many devices and control units, it becomes to slow. This is especially true if some of the control devices use slower standard, like Bluetooth. One down side with this solution is that it will take a few seconds extra to start up the client.

The user interface is dynamically created and is updated every time something changes in the system. There are three separate areas of the interface. A list of devices and macros is placed at the left side, an area for states at the top right and another area for commands at the bottom right. The list is grouped in macros and devices, and for the device group there is a subgroup for every device type containing each separate device. The top right state area is a text field displaying all states of a selected device, such as the song playing on a mp3 player or if a light is on or off. In the last area, bottom right, every command for a selected device is displayed. There is a number of command types that can be used such as short strings (up to 255 characters), long texts (used for e.g. macros and playlists), numbers and booleans (true/false). There is no upper limit for how many commands or states a device type can have. See Figure 4.14 to see when a macro is selected in the GUI and Figure 4.15 to see when the mp3-player device is selected.

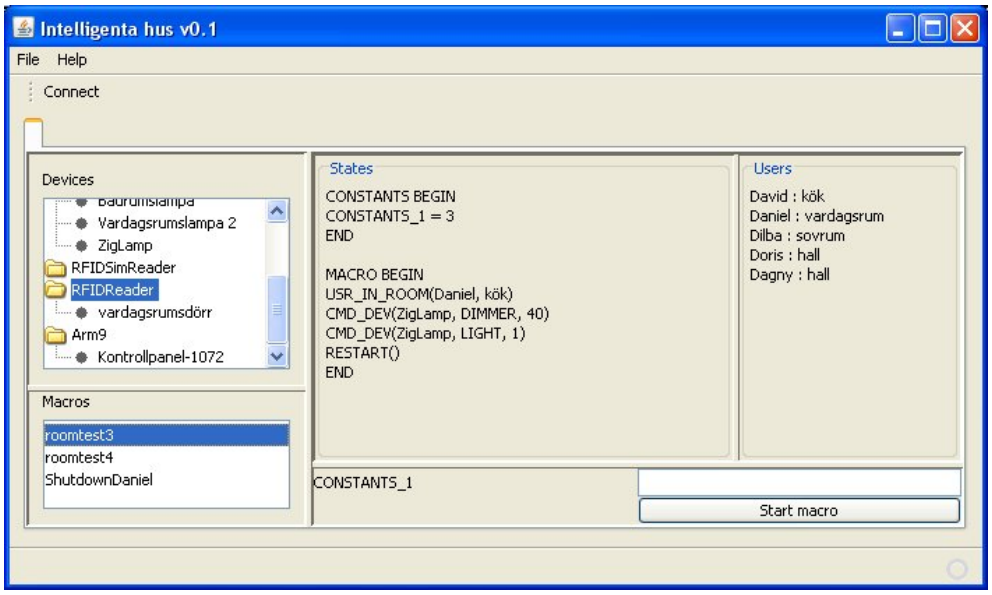

Figure 4.14 Screen shot of the computer GUI when a macro is selected

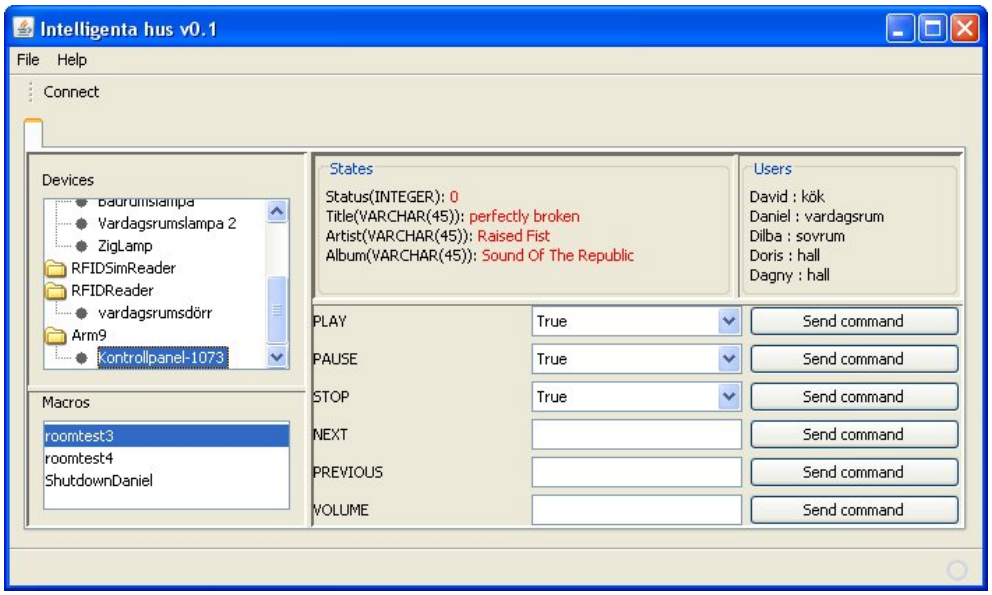

Figure 4.15 Screen shot of the computer GUI when mp3-player device is selected

Mobile phone GUI

In order to control the system, a user interface has been implemented on a mobile phone. This interface is designed to take advantage and control of the basic functionality in this system, like sending commands to devices and start macros. The menu system is designed to give a good overview of the system, see Figure 4.16, but at the same time be powerful enough to control all different types of devices.

The mobile GUI is designed to work just like the computer GUI, but it is built using the Java 2 Micro Edition (J2ME) API, and because of its limited capabilities there are minor differences between the two. Since J2ME already have APIs for the Bluetooth protocol, and nearly all mobile phones sold today supports both J2ME and

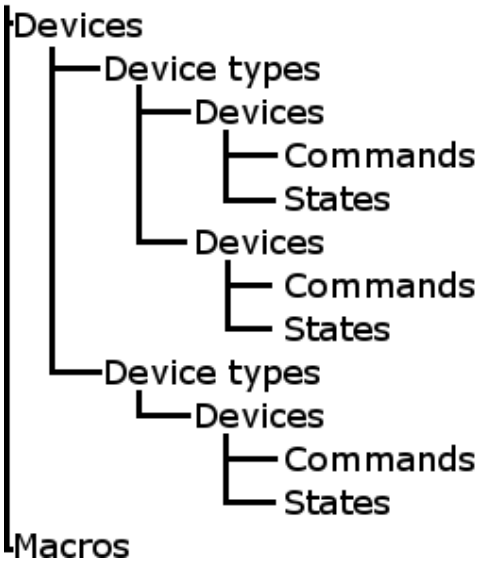

Figure 4.16 Schematic overview of the mobile phone GUI menu system.

Bluetooth, this is the protocol used between the server and mobile phone.

When a user starts the application on the cell phone, it searches for a computer that is running the Bluetooth module of the system. If found, it tries to connect with the computer and, if successful, receives a device id. Like all control devices, the GUI is registered in the server like any other device, but with the option to receive system updates to keep all system information up to date. In order to speed up the connection process in the prototype, the mobile phone skips the discovery step and always tries to connect to the same server address.

Lamps

Lamps are excellent to show the functionality of the design of the system. Lamps are fairly simple, but can show the possibilities of the ZigBee network, users, location and function as routers in the network.

Our design has two separate states. One is called LIGHT and is a boolean (true/false) telling whether the lamp is on or off. The second one is a integer (natural numbers 0..99) called DIMMER with the intensity of the lamp.

This functionality has been implemented in a AVR128 processor that is also a router in a zigbee network. It sends and receives messages from the server and control devices. The lamp is controlled by the microcontroller by a series of circuits. First, three GPIO:s of the controller are used to create a serial interface to control an DAC (MAX513). This is an eight bit converter that outputs a voltage between 0-3V, which is converted to 0-10V with the help of an OP-amplifier in a positive gain configuration. This signal, is then used to control 0-230V AC with a DC controlled dimmer module (K8064). See Figure 4.17 for a schematic of the circuit and Figure 4.18 for a photo of the K8064.

RFID readers

RFID (Radio-frequency identification) is a technology that uses passive or active tags (passive in the prototype) to transmit a signal with an identification. It also has other possible purposes like information storage, that could be useful in the further development of a home automation system. This identification can then be read by a

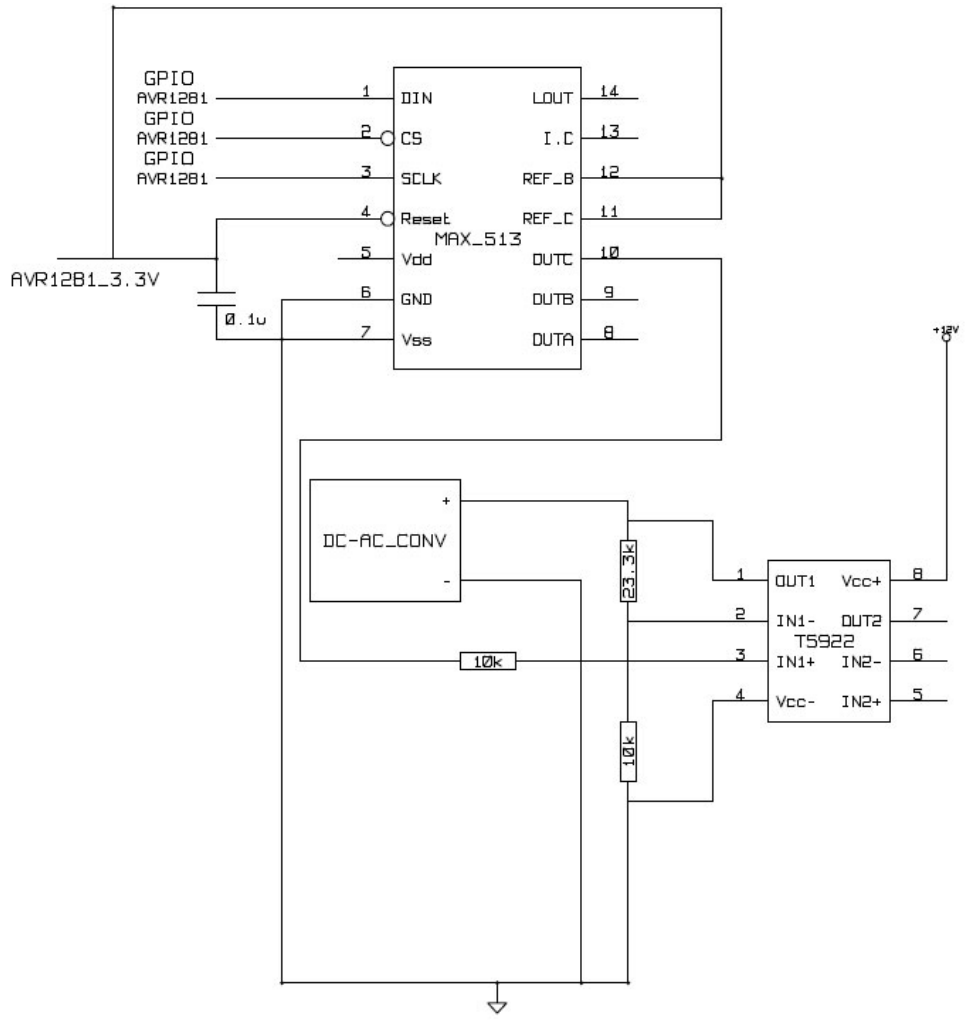

Figure 4.17 Schematic of the circuit used to control a dimmer lamp with an AVR1281 ZigBee module

reader.

These readers are the basis of our system of location. The principle is that a rfid-reader is placed in every doorway of a house, and that every user has a rfid-tag (for example on their mobile phone or key chain) so that the system knows in which room all of the people in the house are and can adapt all the systems accordingly, see Figure 4.19 for a photo of an RFID-tag.

Building an RFID-reader prototype that covers a whole door would take a lot of place, be time consuming and quite expensive. For this reason, the demonstration system uses a small reader that requires a tag to be within a maximum of a few centimetres away to be read, see Figure 4.20. This is enough for us to show the principle of the technology. The reader is connected to one of the nodes in the ZigBee network, just like a lamp, see Figure 4.21

Mp3-player

The mp3-player is implemented, using the a evaluation board from Atmel. This board is based on an ARM9 microcontroller (AT91SAM9293) and includes a wide selection of connection, like RS232, USB and Ethernet. For visualisation and input, a touch

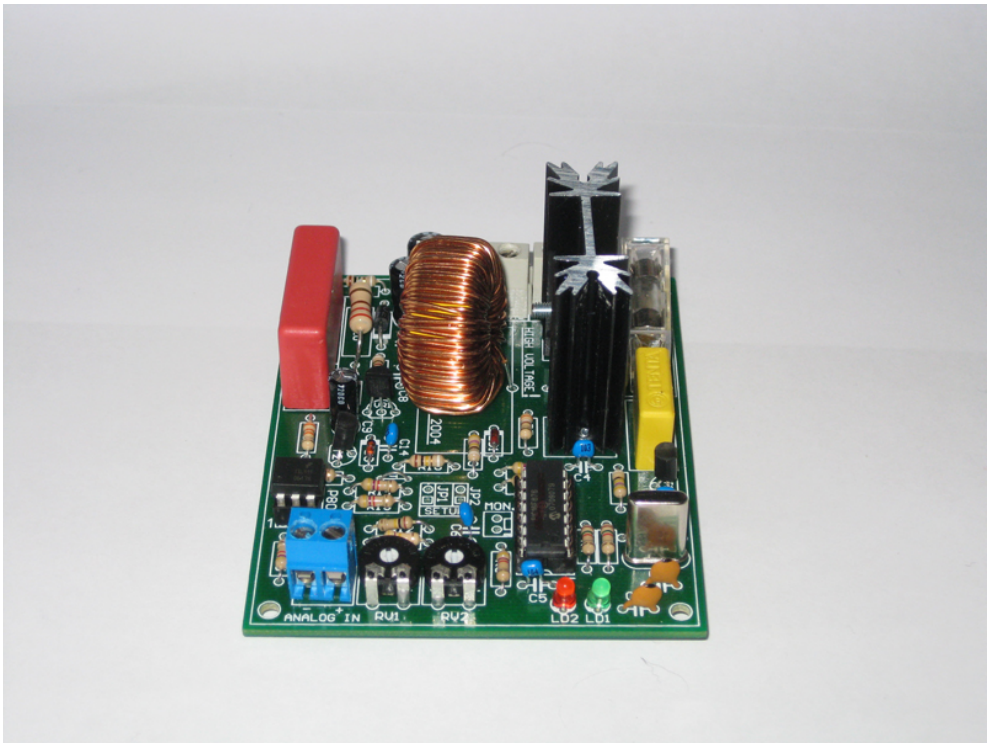

Figure 4.18 Photo of the K8064 DC controlled dimmer module

Figure 4.19 Photo of one of the RFID-tags used in the prototype

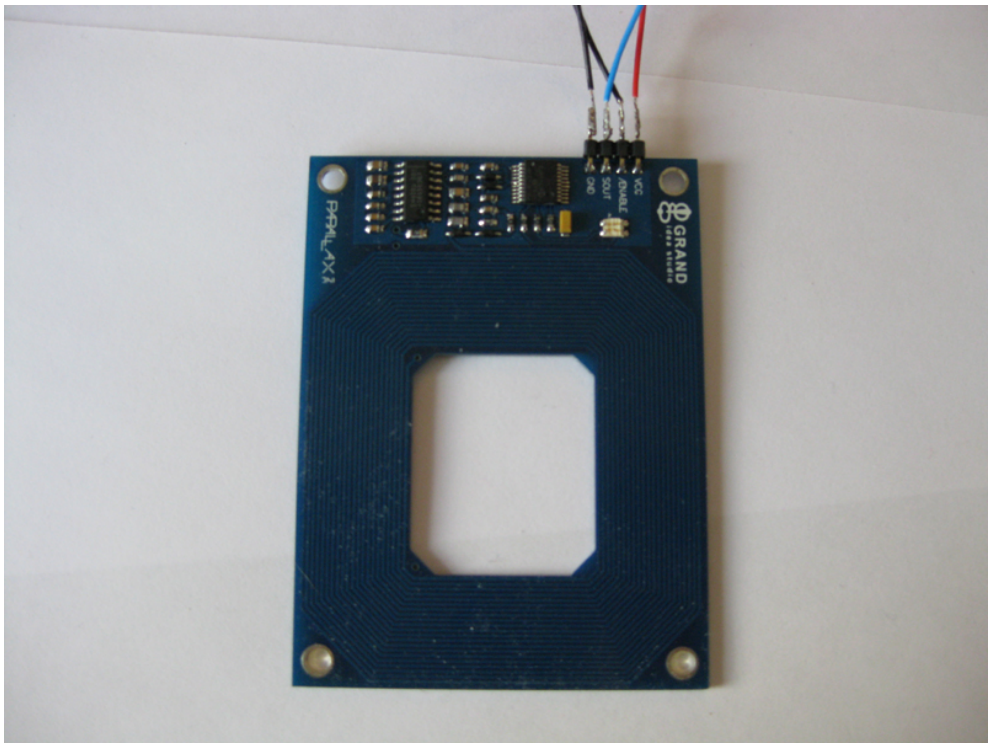

Figure 4.20 Photo of the RFID-reader used in the prototype

sensitive LCD-screen is mounted on the board. See Figure 4.22 for a photo of the evaluation board.

To be able to use both network and USB, without the need to implement protocol stacks, a linux distribution for embedded devices has been used. This distribution also includes a popular and powerful multimedia player called MPlayer. Mplayer supports most of the formats used for digital audio and video. There are two options to control MPlayer, either by a user, using a keyboard, or from a FIFO list. The list is connected using a pipe, and commands are executed in the same order as they were put in the list. This second manner of control is the one used in the implementation.

Software has been designed to handle both control of MPlayer and communication with the server. The software is built with three threads, one to route messages and create the other threads, one to handle the communication and one to execute and control MPlayer. The last thread is split in two processes, one to read from the incoming pipe, i.e. output from MPlayer, and one to write to the outgoing pipe. When the software is executed, it tries to open a connection to the TCP module, and register as a new device in the server. When a new device id has been received, MPlayer is started and instructed to open a playlist with all songs to play.

Since the evaluation board has very limited storage space the music has to be stored elsewhere. There are two solutions to this, either store the music on a USB stick or to stream the music from an FTP server. Both solutions are supported in the software.

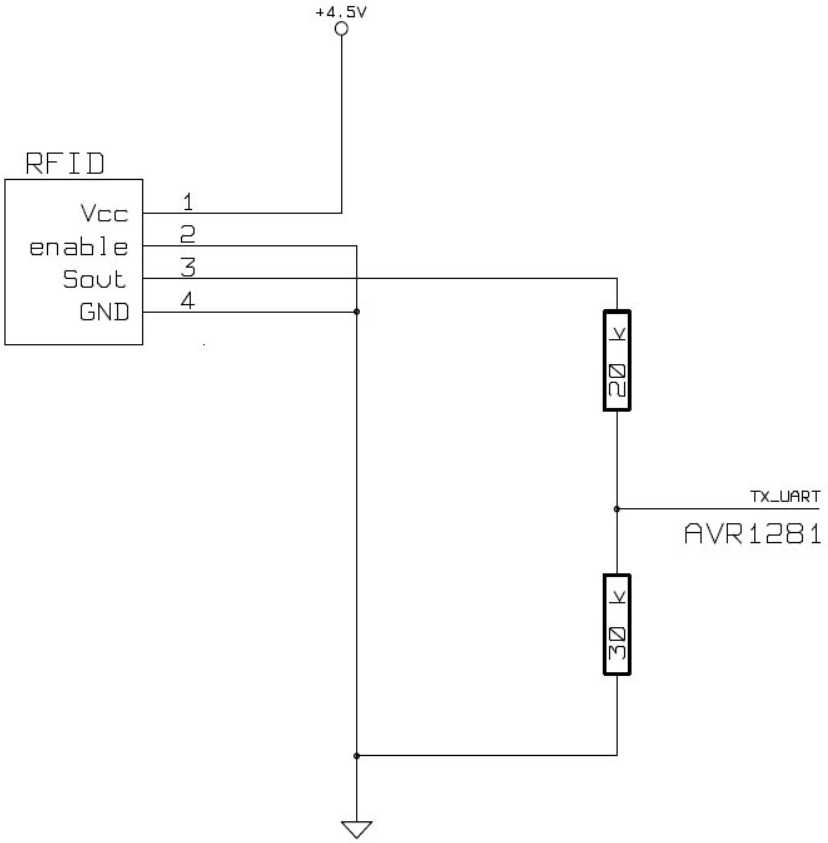

Figure 4.21 Schematic of the circuit used to connect the RFID-reader with an AVR1281 ZigBee module

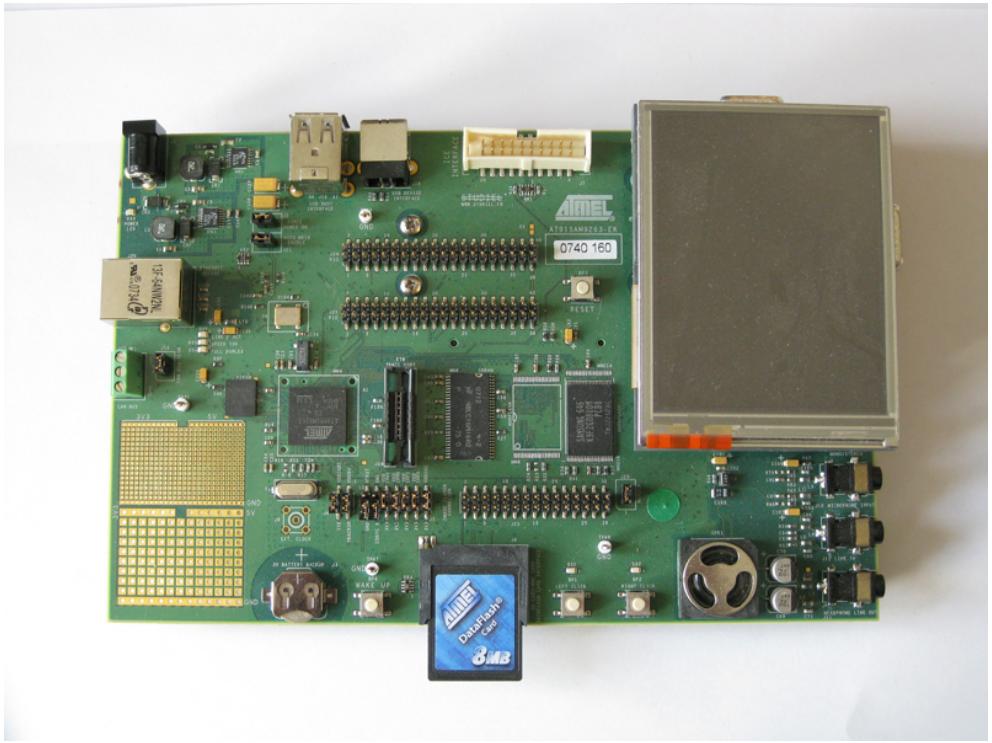

Figure 4.22 The ARM9 prototype board used as an mp3-player

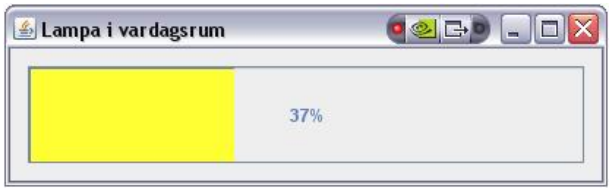

Figure 5.1 The ARM9 prototype board used as an mp3-player

5. Limitations

The analysis of background and current and future development is quite shallow due to the time limitations. In order to get a better analysis, more information gathering, surveys and interviews would have to be conducted. This, of course, also affects our design choices for the test rig, but the most fundamental assessments in that area should be correct. It is harder to tell if the list of applications for intelligent houses really reflects the market's expectations and the future of the field.

This thesis does not in any way bring up the security of intelligent houses in general nor the security of the prototype system. The prototype has been built with the future possibility to add security in mind, but is to be considered completely insecure in its current state.

Primarily because of the complexity and the size of the base system of the prototype, the number of protocol modules and devices is somewhat limited. To compensate for this, the rig features a number of simulations of devices that help to further demonstrate the possibilities of the system, see Figure 5.1.

To save costs and time, our RFID-reader device only has a small reader, with a range of a couple of centimetres. In a real system these would be scaled up to cover whole doorways. Finally, some kind of sensor to judge direction would probably be necessary for robustness.

The possibilities of the macro functionality is somewhat limited. If a full scale system were to be built, the possibilities of macros could be developed much further. Functionality to generate macros graphically with some kind of drag and drop interfaces in the GUI would also help to make the potential of the system more reachable for the users.

The GUI also lacks the functionality to add users, rooms and doorways. This is done manually with database calls in the prototype. It would also be necessary to add some kind of priority levels to the users to determine who controls the systems in the case of a conflict.

The Bluetooth module in the prototype uses the transceiver of the server computer, which is class 2 and has a range of about 10 metres. Furthermore, the Bluetooth standard limits the number of possible devices using the module to 7.

6. Result and Analysis

This thesis set out to analyse the background and necessary conditions for success in the area of intelligent houses. Furthermore, an analysis of the current state of the market and development was to be made, and future trends to be extrapolated. Finally a prototype was to be created to show the possibilities in the area.

The background concludes that both the technology of the market and the potential customers should be ready for intelligent houses as a natural next step in the development. This conclusion is inherently impossible to verify. In the current development, two trends could be discerned. The first one was a rather large market in the area of larger scale automation of office buildings, apartment buildings and the like, focused mainly in fields like ventilation, heating, lightning and elevators. The other trend was focusing on the consumer market, but suffered from locked standards and incompatibility.

Based on the analysis, this thesis focused on a simple, yet powerful solution that would allow both the existing systems in home automation and third part manufacturers, not traditionally within the market, to integrate their products in one solution. To make this possible, a design with a great focus on flexibility, dynamic behaviour and transparency was chosen.

For simplicity the solution was based on a central server. This server was the first layer in a three layered structure. The second layer was a protocol layer with independent modules, connected to the server in run time using the TCP/IP standard. These modules were then used to implement different communication protocols used by the third device layer. In our case, three modules were implemented, creating support for TCP/IP, Bluetooth and ZigBee. Due to the transparency and independence of the architecture, creating new modules for other communication protocols should be easy, and possible for third part developers.

Finally the device layer contains all the parts of the system that the user would come into contact with. The devices are mainly control services, sensory devices, actuator devices or combinations of these. Communication is done through a tag based XML-like protocol which is also designed according to the principle of being powerful yet simple. The server contains a database which keeps track of all devices and their functions and provides updates and all necessary functionality to made control devices completely dynamic and able to control arbitrary devices adhering to the standards set up in the thesis. In the test rig we have two control devices: a computer GUI (TCP) and a cell phone GUI (Bluetooth), and three end devices: a lamp with dimmer (ZigBee), a RFID-reader (ZigBee) and a mp3-player (TCP). Futhermore we have programs that simulate lamps (TCP) and rfid-readers (TCP).

The system also supports simple macros for executing several commands, command all devices of one type, timers and trigger on events (a user walking into a room). This makes it possible to create functionality to, as an example, close down the house with a simple command, get notices to users when they leave the house and automatically configure the lightning and other equipment in a room when a user enters.

Overall, the system has the functionality that was intended, but there are some minor aspects that need further development. The macros have quite limited functionality and could be made more efficient. Some more devices could be developed to further show the potential of the test rig. Some parts of the system are a bit unstable. The mp3-player crashes sometimes, the ZigBee devices have disconnected a few times, the server does not handle all possible exceptions and is very vulnerable to

both failures in following the message protocol and to malicious attacks. The protocol modules does not automatically reconnect if the socket to the server closes.

Furthermore, none of the control devices supports full configuration of the system. To add rooms, users and rfid-readers, it is necessary to manually add them to the database. The ZigBee devices does not support dynamic, permanent storage of their device ID:s, which means that they either must be hard wired with an ID or get new id:s for every connection.

There are also some parts of the design that might be improved. Our message system, though fairly simple, could have been better structured and intuitive. It is not completely clear which message type to use for different purposes.

A better solution to the connection between the ZigBee devices and the server, might have been a ZigBee module that could be connected directly to the computer, perhaps using USB or something similar. The solution proposed by this thesis works, but the comm-port seems a bit out of date.

The control circuit for the dimmer lamp might be a bit of a detour. The precision, price and complexity might have been reduced by choosing an digitally controlled dimmer, instead of an analog DC-controlled one.

The start up process of the prototype is too complex. For simplicity, the protocol modules should start automatically with the server and the ZigBee coordinator must be started before the MAX3232 circuit, or the UART ports on the AVR1281 won not initialise properly.

7. Conclusion and Summary

Home automation has great potential, both from a market perspective and in its possibilities to improve people's quality of life. In this thesis we have put most of our focus on functionality that most people would probably want from a system like this. The possibilities are much greater than this though. If an open standard for intelligent houses, like the one proposed in this thesis, becomes big, custom devices for specialised needs could be made. It is not hard to, for example, imagine devices that would greatly improve life for disabled, hearing impaired or blind people.

There could also be great possibilities for interconnection of advanced technical equipment like computers, TV:s, recording devices and so on. Another area that could be used for early marketing of a system are the applications to improve safety for children and help alleviate parents.

The things that can be done using home automation are almost infinite, which reinforces the conclusions made in this thesis. An open, flexible and dynamic system is necessary to bring out the potential of intelligent houses. One interesting way to handle this would be open source, possibly aided by companies in the market who see the potential to widen their customer base if their systems could be combined with other functionality in a simple and powerful way.

In order for a system like the one proposed here to be a solution in reality, the aspect of security can not be stressed enough. The magnitude of the possibilities to infringe on the personal integrity of people with a powerful home automation system makes 1984 by George Orwell seem like children's play. Furthermore, control over such a system could make a potential malicious intruder able to physically harm or even kill people in the home. These matters are of grave importance and must be resolved if home automation is to become a natural part of life.

We have reached the goals that we had set for the thesis and for our system. A necessary criteria for intelligent houses to grow big is an open, flexible, simple yet powerful standard that can unite all other systems in the market. This thesis proposes a starting point for such a standard. It could have been done in many ways, but this solution tries to weigh simplicity, adaptability, power, price and flexibility to make the system widely accessible and scalable.

A. Glossary

ADC	**Analog-to-Digital Converter** is an electronic component that converts an analog signal to a digital. It receives an analog signal, such as a voltage between zero and a reference value, as an input and converts it to a digital number with a certain number of bits. As an example, a 5V, 8 bit ADC would take a voltage of 0-5V and return a linearly proportional digital value between 0 and 255 as a result.
ALU	**Arithmetic Logic Unit** is the part of a CPU that performs arithmetic and logical operations. Examples include bit wise logical operations like AND and OR, and arithmetical ones like addition and subtraction.
API	**Application Programming Interface** is an interface for source code, created for application programmers to access functionality within a library, an operating system or a service of some kind.
ARM	**Advanced RISC Machine** is an architecture for relatively powerful, cost and power efficient RISC processors used primarily in embedded systems. Developed by ARM Limited.
AVR	A series of RISC microcontrollers with a fairly low data processing capacity, that are very cost and power efficient. The chips usually support a large number of peripheral functions like: UART, PWM and ADC. The series is developed by Atmel.
CISC	**Complex Instruction Set Computer** refers to a computer architecture design principle. The principle is to include a very large number of instructions in the design and make the processor fast by implementing complex operations that corresponds to functionality that is often needed when executing a program.

Console	A console (or system console) is a text based way to display administration messages and/or control a computer.
Console application	The word console application refers to a computer program that is designed for an interface with only text, for example a system console. A program like this has no GUI.
DAC	**Digital-to-Analog Converter** is the opposite of an ADC. It takes a digital value with a certain number of bits as input, and converts it to an analog value, typically a voltage between zero and a reference value.
Database	A database is a system to store data in a structured way. It uses some kind of organisation that make it easy to find and change the data, and is managed by computer software to facilitate its usage.
Embedded system	Embedded system is a term used for all kinds of computer systems that work as support for other functionality. It is characteristic for an embedded system to perform one or a number of dedicated instructions. Embedded systems are also often, at least in part, invisible to the user. Most people don't consider their new television to be a computer, even though it probably has several processors in it.
GUI	**Graphical User Interface**. A user interface is the part of a program that the user can see and interact with. A Graphical User Interface is a sub type of user interfaces that uses graphics, normally icons, menus, buttons and so on.
IEEE	**Institute of Electrical and Electronics Engineers** is an non profit organisation that promotes advancement of technology. They are well know for leading standardisation efforts on different areas including wireless communication.
IrDA	**Infrared Data Association** refers to a wireless standard using infrared light. It is commonly used to transfer data between devices such as notebooks and mobile phones.

Java — Java is an object oriented programming language developed by Sun Microsystems. It is a high level language, designed for fast development of both big and small applications and is platform independent.

Mesh Network — A mesh network uses nodes to route information. If one node goes down, the network automatically finds a new way to get the information to its destination. This makes a mesh network self healing, very robust and reliable.

Microcontroller — This term is usually used to refer to a fully functioning computer on a single chip. They have, in general, a relatively low computational power compared to, for example, desktop computers, and tend to incorporate a lot to additional functionality such as support for different communication standards. The models in the AVR-series from Atmel are typical examples of microcontrollers.

Platform independent — This is a term within the software community that refers to software (such as programs, programming languages and operating systems) that can run on several computer hardware platforms.

PWM — **Pulse Width Modulation** is a periodic signal where each period has a duty cycle where the signal is high and the rest of the period the signal is low. This creates a square wave where the average value of the signal can be controlled by modulating the pulse width. This principle is used to control the power of actuators, such as motors, or in power regulators.

RFID — **Radio-frequency identification** is a technology standard for automatic wireless identification of tags. The tags send a unique signature to the reader. RFID tags can also be used to store information.

RISC	**Reduced Instruction Set Computer** is a design principle for computer processors. Instead of having lots of instructions for every kind of operation, a RISC computer has very few, but highly optimised, instructions that are combined to do more advanced things. This makes the design of the processor simpler which makes it cheaper and more power efficient.
Serial Communication	This is a principle for data transfer where the information is transmitted one bit at a time in a sequence, as opposed to parallel transmission where several bits are transmitted at the same time, usually using several channels of transfer.
SQL	**Structured Query Language** is a computer language used to manage and receive data from relational databases.
TinyOS	TinyOS is a very small operating system, primarily used in embedded systems. It is an open source project and specifically designed for Wireless sensor networks, but that is not its only application.
Transparency	In computer software development transparency means that the parts of the system are designed to have fixed interfaces that remain intact even if the internal behaviour of the individual components change. This removes the need for individual parts (and their developers) to know how the rest of the system is built.
WiFi	WiFi is a standard for wireless networking, used mainly as a wireless alternative to Ethernet. When people normally talk about wireless networks a home or in an office, or a internet hot spot they refer to WiFi.

XML

Extensible Markup Language is a general markup specification. An XML form consists of tags which looks something like this:

```
<tag>content</tag>
```

XML is used to structure data, specially for data transfer between different systems. It uses a parser that forces every XML document to adhere to the standard in order to be read.

B. Bibliography

ARM (2008): “http://www.arm.com/.” Internet.

Atmel (2008): “http://www.atmel.com.” Internet.

Axelson, J. (2005): *USB Complete: Everything You Need to Develop Custom USB Peripherals*, third edition edition. Lakeview research llc.

C Bala Kumar, Paul J. Kline, T. J. T. (2003): *Bluetooth application programming with Java APIs*. Morgan Kaufmann publishers.

Clint Smith, D. C. (2002): *3G wireless networks*. McGraw-Hill Professional.

George Coulouris, Jean Dollimore, T. K. (2005): *Distributed systems: concepts and design*, fourth edition edition. Addison Wesley.

Houda Labiod, Hossam Afifi, C. D. S. (2007): *WiFi, Bluetooth, ZigBee and WiMax*, first edition edition. Springer.

IBD (2008): “http://www.intelligent-building-dictionary.com/.” Internet.

in automation, C. (2008): “http://www.can-cia.org.” Internet.

John L. Hennessay, D. A. P. (2007): *Computer architecture: A quantitative approach*, fourth edition edition. Morgan Kaufmann publishers.

KNX (2008): “http://www.knx.org.” Internet.

Microchip (2008): “http://www.microchip.com/.” Internet.

Regin (2008): “http://www.regin.se.” Internet.

SIG (2008): “http://www.bluetooth.com.” Internet.

Smarthome (2008): “http://www.smarthome.com.” Internet.

tac (2008): “http://www.tac.com/.” Internet.

Techtarget (2008): “http://whatis.techtarget.com/.” Internet.

Techweb (2008): “http://www.techweb.com.” Internet.