A SMART HOME EMBEDDED COMPUTER SYSTEM ON PROGRAMMABLE CHIPS

by

Jonathan B. Chan
Bachelor of Engineering
Ryerson University, 2003

A thesis
presented to Ryerson University
in partial fulfillment of the
requirement for the degree of
Master of Applied Science
in the Program of
Electrical and Computer Engineering.

Toronto, Ontario, Canada, 2006

Author's Declaration

I hereby declare that I am the sole author of this thesis.
I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Signature

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature

Instructions on Borrowers

Ryerson University requires the signatures of all persons using or photocopying this thesis.

Please sign below, and give address and date.

A SMART HOME EMBEDDED COMPUTER SYSTEM ON PROGRAMMABLE CHIPS

Jonathan B. Chan
Master of Applied Science
Department of Electrical and Computer Engineering
Ryerson University, 2006

Abstract

System on Programmable Chip (SoPC) based embedded system development has been increasing, aiming for improved system design, testing, and cost savings in the workflow for Application Specific ICs (ASIC). We examine the development of Smart Home embedded systems, which have been traditionally based on a fixed processor and memory, with inflexible configuration. We investigate how more ability can be added by updating firmware without the burden of updating hardware, or using a full (but dedicated) general purpose computer system. Our development and implementation of the smart home controller is based on the SoPC development environment from Altera. The development board includes all the necessary parts such as processor, memory, and various communication interfaces. The initial implementation includes a simple protocol for communication between home appliances or devices and controller. This protocol allows data transfer between home appliances or devices and the controller, in turn allowing both to support more features. We have investigated and developed a home resource management application. The main resources being managed in this project are hot and cold water, electricity, and gas. We have introduced a number of expert rules to manage these resources. Additionally, we have developed a home simulator, with virtual appliances and devices, that communicates with the home controller. The simulator interacts with the SoPC based smart home embedded system developed in this project by generating messages representing a number of smart appliances in the home. It provides a useful testing environment for the smart home embedded system to verify its design goals.

Acknowledgment

I am grateful for the support of my supervisor, Dr. Gul Khan, for his guidance and patience. Additionally, to Ryerson University, CMC (Canadian Microelectronics Corporation), and NSERC (Natural Sciences and Engineering Research Council) for providing the necessary hardware, system development environments, and financial support. This work is also dedicated to friends and family who have been supportive during the time of this research.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 From ASIC to SoPC

Modern digital systems have come a long way from discrete analog and digital components to integrated circuits, to multi-chip IC components, and finally to single-chip Application Specific IC (ASIC) components.

Many application specific designs call for the use of a microprocessor to perform various calculations and tasks. An example of this is the increasing popularity of portable media players that get data from a data store across a common peripheral bus and must manipulate the data before passing it onto other components. Another is to monitor a process, periodically reading sensors and processing the data. Rather than designing and debugging a custom microprocessor, a more cost effective method is to incorporate existing designs such as ARM processor cores. According to ARM's milestones [1], they released the first embeddable RISC core using ARM6 design in 1991. ARM continues to license processor cores today, and are one of the most common embedded processor cores available. Another advantage to this method is many system interconnections can be incorporated onto the chip, removing many large bus traces from a finished system board design. This may help to reduce noise pickup, and latency, and recover physical spaces which may be extremely valuable in some installations. The other problem with fixed systems is glue logic, such as hardware memory mapping which often can't be changed after a system has been deployed.

Embedded engineers are pushing more and more components into a smaller chip space, faster and lower power components, all while keeping the system development time as short as possible. The advantage that ASICs bring to the table are space savings, often high speed operation, and high reliability but all of this at a huge cost of time and money in the development cycle [2, 3]. Along similar lines, modern homes have been coming with more built-in convenience, adding systems and features to help users manage their home, all in the push for more efficiency, cost savings, and stress reduction - often caused by the development cycle of modern digital systems.

System on Programmable Chip (SoPC) platforms provide an accelerated design cycle by providing some existing core design as well as enable prototypes to be quickly made and re-designed as needed. The processor cores can later be incorporated into an ASIC. SoPC are not only great for prototyping and general experimentation, but they're appropriate for designs which may be incorporating new technologies and ideas, such as communication protocols. They're also good for applying different models without changing physical hardware. That helps reduce production and parts sourcing costs. These designs may not return to manufacturing on ASICs, but will retain final designs on FPGAs or other programmable logic devices, allowing for future hardware updates in addition to software. This kind of system is our focus in this project.

1.2 Smart Home Control

Home control has been around for many years, and remains a niche market luxury item. Most homes still do not have smart installations although more and more larger homes will have some provision for running household networking or other auxiliary wiring. Many systems today concentrate on running whole-house radio or smaller automation systems for home theatre spaces.

The technology and protocols used were at one time proprietary although they are becoming more open and standardized. Still there is a lot to be desired in the open-ness of any particular system. There are many standards in use today and your geographical location will dictate the most common standard in use or available in that area of the world. There are more than eight each standards and protocols for home control available today [4].

A smart home is a living space that has been augmented with technology in order to assist the occupants in some manner. There are many definitions of what a smart home is, as well as many considerations done when designing one. The home is a part of daily living and should take an active part in providing comfort, utility, and safety for its occupants.

A Smart Home can have many different definitions depending on who you talk to. A 2004 study showed that home technology leading to greater safety and environmental benefits were favoured. This benefits everybody in the future, where costs of living are rising, and environmental issues are brought to the forefront [5]. Other uses include pure luxury or unloading some of the burden of home care and safety onto a mechanized (or in this case, computerized) system.

Homes are a personal space, and the occupants must "get along" with it, or in a more common case, the home will be changed according to the wishes of its occupants. We often want to support family and friends in a familiar comforting surrounding. It invokes a feeling of safety, peace and nostalgia, having lived many years in a home. Having said so, the home is also a place to get work done. It is a place to study, research, and do chores in. Security is an issue that we often have on our minds as a break-in and theft often leave us feeling extremely vulnerable, violated, and that the home is no longer a safe haven.

People are specific about what they want to accomplish with the technology in their homes, living, or work spaces. We introduce technology for minimizing the impacts of conditions on getting work done. With the home control platform, the home will provide information to the occupants, if they wish to see it. It will also manage the resources of the home in order to provide a certain quality of service. As a base system, this will allow more advanced features to be built on top as embedded software (firmware) features or add-on modules.

1.3 Motivation

In North America, X-10 appears to be the most well established control protocol. At the very least, it is easy to find X-10 equipment and references to such equipment [6]. One only has to do an internet search for "X-10" or "X10" and will find many resources. There are software and hardware automation modules available for the X-10 system, but they are restricted to sixteen basic functions that the individual modules understand. The connection to a PC is a serial connection, something that's becoming increasingly rare in new computers. X-10 works by sending serial data over a carrier on household power. It has been in the market for a very long time and will continue to do so. Being a legacy system, it has its share of advantages and disadvantages. The data protocol is a simple fixed 5-byte coding. The system cannot support arbitrary information and new multifunction devices.

One benefit of data over power lines is that the infrastructure is already in place in both new and legacy homes. This avoids the cost and problems associated with retrofitting legacy homes with additional control infrastructure. In new homes, it is easier if the house had been fitted with extra cable routing capability such as piping during construction.

In this project, we propose an open system that will be upgradeable and extensible. This system supports common functions (the features that people are accustomed to and expect in a basic system) as well, it will be able to incorporate new functionality as required. The system is flexible enough to allow new technologies to be interfaced to the existing system. We achieve it by allowing any arbitrary set of commands and data descriptions from the devices. There is a simple data protocol we call "banter" that's sent within a standard

TCP/IP packet. The banter protocol is illustrated and explained further in section 3.3.2. It allows for an arbitrary data length, although fitting it in a single packet is preferred. The system acts as the gateway to the functions of the home - a home API or middleware approach. This makes the system to be somewhat autonomous while supplying functions and services to software at a higher level, to allow custom control applications to be easily written for the system.

The controller's physical infrastructure follows the same principle, allowing for adoption of new communications equipment if necessary. The initial design uses standard ethernet and TCP/IP. Serial, USB, and other connectivity options can be adapted later. Following on the heels of many researchers and hobbyists alike who have adopted many kinds of control technology, we would like this to be "hacker friendly," enabling hardware and software to be modified by anybody. This should also ease adoption of new technologies and even custom designs. It would allow for people to integrate and create a system of their own design, even using existing devices in a novel way. To help facilitate this, the system code as well as the protocols used must be open and royalty-free. Although the design as-is uses a new protocol, support for existing protocols could be written, and thus the system could be brought into standards compliance and also up to the latest "state of the art."

1.4 Augmenting the Home

Smart homes have enhanced features or functionality over basic homes. The technology is used to provide additional services such as improving its efficiency, allowing itself to be controlled or monitored, to provide and facilitate information, multimedia, or assistive services. These can largely be classified into two main categories, passive and active technology.

A few examples of passive technology are enhanced insulation, design, or materials selection to take into account the local environmental conditions, with the goal of saving energy in heating or cooling costs. This kind of technology doesn't need active participation for it to work. Design of a workspace also has an impact on the convenience and ease of use. Take the example of a kitchen. There are storage and work locations that are constantly used. Designing a kitchen that reduces the amount of walking, and allows for convenient access to all resources is an intelligent design, which is passive in use.

There is also active technology in which much research is being done today. Active technology uses electronics such as computers to help. The computers can be small low-cost embedded micro-controllers, larger embedded systems, a dedicated control computer for centralized control, or something in between. Once active technology comes into play, we start to involve what is typically known as automation.

The smart home embedded system makes decisions based on sensory inputs. It acts and reacts to what the occupants are doing. The occupants can allow the system to make these decisions or they can override the system. The users still have complete control, but only if they want to. They are no longer obligated to control every aspect of the operation or use of the home.

Augmenting an existing home can take place mainly in two phases. The first option is to install the system during construction of the house before the walls are put up. Doing so will only marginally add labour costs and materials cost, as other electrical and communications infrastructure is incorporated at a time when everything is accessible and easily routed. The second option is to retrofit an existing home. This usually means fishing the cabling through walls and ceilings and around any other obstacles, and adds a substantial amount to the labour cost. Having existing conduit, as in some newer homes will help a lot with this problem, but there may not be conduit in all the places you want or need.

Fortunately, modern home builders are starting to see the benefits of installing extra wiring during the construction phase. People are becoming comfortable with modern technology and their use, including home networking and home theatre installations. Pre-installed wiring installed helps to sell houses, so options are becoming available in new housing [4].

1.5 Home Automation

One aspect of automation is the direct control of various devices in a home. This is typically what a controller is used for and has been done successfully, for example X-10. Historically this is what all home control systems have been performing. Through various technologies, the controllers allowed for electronic centralized control of lights, electrical sockets, fans, and other devices through a simplistic on/off or dimming function. Many new controllers for the old technology are able to set alarms and timers for these devices. Newer technology has extended functions such as activation on external triggers, setting scene lighting for home theatre setups, or in the general case, one trigger for setting multiple devices.

Moving up to the next level, sensors can be incorporated into the system. These systems are starting to allow a simple state of automation. Motion sensors can activate lights and the system can deactivate them after a period of inactivity. Power savings and other conveniences can now be incorporated into the home. Feedback from light sensors can help in adjusting light levels at all the times of a day but often require tuning over a period of time. They can also be used to automate window blinds, and in combination with temperature sensors, can affect heating and cooling to passively regulate the environment.

Aside from the issue of control is the convenience that control brings, and the convenience features that are starting to emerge. Appliances may talk to a home controller to regulate its resource usage, such as water and electricity. If resources are running low, decisions can be made as to which device will have priority over a resource. Smart devices may be placed around the home to provide convenience by giving reminders, displaying information, or alerting you to dangerous conditions. These are devices that we may or may not regularly interact with, but are capable of bringing our attention to something.

There may be a need of additional services such as an internet connection, or at least a back-end information system. A smart door reminder can alert you to the weather before you leave, or remind you to bring certain items, and attend to appointments. This is now in the realm of information systems [7]. With information systems available, augmenting the environment can be achieved in more ways than simple sense-and-react. Some examples of this are a door reminder, environmental monitoring and reminders, and smart displays [8]. The information system interfaces should be placed in strategic places in order to maximize the usefulness of the system. A well placed LCD panel can serve as a replacement photo frame or painting, displaying photos when idle, and showing alerts when needed. An alternative is to hide the display behind a real photo or painting.

Going to the extreme, the whole house can be covered in hundreds or thousands of sensors, enabling monitoring of everything conceivable and possibly even detection of "unusual behaviour" without having to manually train the system [9].

Security services are a peace-of-mind option. A smart home should assist you while you're inside, and as a matter of security, it can be monitored and controlled remotely as well. It not only allows an agency to monitor your home, as in current alarm system installations, but the home can choose to contact you or some other designated person before alerting the monitoring centres. This may be good for satisfying various levels of paranoia or if a monitoring agency isn't available.

The technology involved typically includes the everyday devices such as switches, sockets, lights, and other appliances. More and more appliances now and in the future will include some sort of connectivity features such as networking, or data transfer capabilities. In addition, sensors and small control devices are added to facilitate this extended functionality. For example, motion sensors in a room can trigger the system to automatically turn the lights on for the room. There is definitely an element of synergy involved in the control of the home. Many of these can be exploited with a centralized control system [10].

Current State

Most of the experimental research systems require a high level of computing hardware, many of which are custom made for a very specific application or service. This makes the system less practical and more costly. Some examples are machine vision and language interfaces [11].

Moreover, the current home systems incorporate a lot of older standards and technology, limiting the capabilities of the system. Our design focuses on being practical and usable. In order to fulfill this, the system must have expansion capabilities, therefore the controller must provide a data service. It cannot hold onto the old standards with little to no feedback and simplistic hardwired control. Data can change and can be updated to incorporate newer system capabilities.

The proposed specification aims to cover some of the broader issues as well as some specific needs in the design for the home controller. The data capabilities must be present in order to provide the usability and re-usability, lowering the total cost involved in employing the system.

Design of Smart Home Control System

The home control system is service oriented, providing lightweight data to a centralized controller described in this report. This is in contrast to full service oriented architectures (SOA) involving de-centralized services [12]. Although SOA lends itself to de-centralized operation, the presence of a centralized controller can simplify the devices and provide co-ordination. A fully decentralized system is difficult to manage and control due to distributed configuration components (configuration for individual devices, etc.) and independent services being provided. Management and security can be more easily implemented and configured from one central device which all other devices must interact with. This central server can be a more traditional home control system server with services implemented, or it can itself act as a decentralized device that provides exported data and control services, essentially a data-store device for consolidation.

A decentralized system doesn't have physical infrastructure to connect all the devices, thus they must all be able to route and re-route to find the devices that they want to communicate with. With the centralized system, the infrastructure is complete, connecting all devices to a central location. In application, the centralized system is actually a hybrid, with many "mini-controllers" located to consolidate devices within a specific area to minimize physical connectivity problems. A fully centralized system is shown in Figure 1.1.

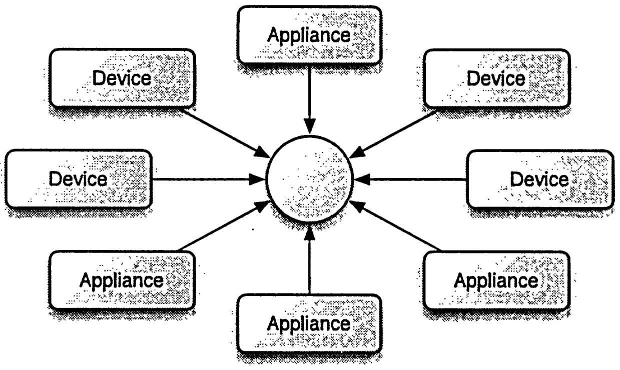


Figure 1.1: Fully Centralised System.

The data services allow for the devices and appliances to be controlled in a very open fashion, not restricted to a rigid bit-value system. Freeform input can be used and stored in the controller, to be read back to the device. Each device simply needs to understand its own command set, and will be able to update the controller on what commands it's able to process.

There is an information system that allows logging of events in the system, as well as being able to route messages and events to different devices. This allows for the information display type of devices. Resource management is built on top of this infrastructure, using message passing between the devices/appliances and controller to ration out finite home resources.

To summarize these ideas, our smart home controller prototype uses the centralized configuration, with direct communication between devices and appliances to the controller. Due to the growing use and high availability of Ethernet cables and equipment, Ethernet is the initially supported medium. We implement the "banter" protocol, which is a simple handshaking and data transmission protocol that supports arbitrary data lengths. The protocol is explained further in section 3.3.2. This allows the controller and connected devices to implement any feature, and not be forced into a rigid bit-value system. The main feature of our prototype is resource management, which can be considered a building block that allows other features to be implemented in the future.

1.6 Contributions

In this project, we investigate smart home control embedded systems and home resource management. Using a development environment from Altera, we implement the smart controller hardware with a NIOS processor core and peripheral components. The firmware is written in C and built with supplied NIOS build tools. The aim of smart home control is to improve the quality of living. An embedded resource management application is developed for regulating the usage of common home resources such as hot and cold water, electricity, and gas to improve the quality of service of those resources. Verification of the system is assisted by creating a home simulator system in absence of physical smart appliances and devices. The simulator allows for the continual development and testing of the smart home embedded control system.

Chapter 2

Smart Home Technology: Overview

Home automation was at its peak in the mid-80s but has died down twenty years later. The technology today certainly doesn't feel twenty years advanced, as much as computers have continued to evolve and make themselves an ubiquitous part of home and work life today. During this time, the focus and scale of "home automation" or "home control" has encompassed the basic systems and functionality to large commercial or "professional" installations. Many standards have been established but continue to be different in different parts of the world.

The industry hasn't been dormant but also hasn't visible unless you go looking for it. Much of the focus with commercial systems, as an example, is to make large office buildings more energy efficient, providing green technologies to lower operational costs, as well as be more environmentally friendly. Consumer level control seems to have remained more or less unchanged since the introduction of X-10. A summary of existing hardware infrastructure and control protocols concerning the smart home systems is presented in this chapter.

2.1 Simple Hard-wired Control

Any standard home we see today is made with simple wiring connections. Control is at almost every point in the system. Light switches control room lighting and wall sockets, and individual devices such as a lamp, have their own control. They are all independent and will not interfere with any other device's operation, barring a device tripping the circuit breaker.

This is the basic implementation used in homes everywhere. There may be 15-25 circuits in an average house, with many devices hanging off each of them. There are disadvantages to this approach and when a breaker trips, all devices connected to it will lose power. If one needs to do maintenance on any section of the physical circuit, power to the complete circuit must be cut, losing other devices while work is being done. Most of the installation of the circuits don't follow any hierarchy, and a single circuit breaker often controls power to many unrelated areas of the house. Even X-10, explained in the following section, is only a slightly updated wired system where only simple control is offered. This is relatively inexpensive and is one of the reasons why it is so widely employed.

2.2 Power Line Control

The protocols in the following subsections carry the protocol on the power lines. This is the way it was done for most early protocols and is still a popular option today. There are many advantages and disadvantages of this system. The main advantage is that it can be used in existing homes or buildings without the installation of additional wiring exclusive to the control system since presumably power is available everywhere. This allows for immediate use (instant gratification) of devices connected to power. It avoids the hassle of installing separate infrastructure to carry the control signals. This is the way most early protocols have been worked out.

2.2.1 X-10 and Insteon

X-10 is a legacy home control protocol used over power lines [6]. The control modules are plugged into the existing household power line that receives and sends control signals over the power line. It is then bridged to a standalone or computer-assisted controller. There is a limit of 256 devices on the system (combining the 16 house codes with 16 device codes) and the control signals are subject to any interference on the power system. Early switch mode power supplies, such as found on very early desktop computer systems were notorious for feeding noise into the power system, interfering with many FM-based communications equipment. Modern supplies tend to be more quiet.

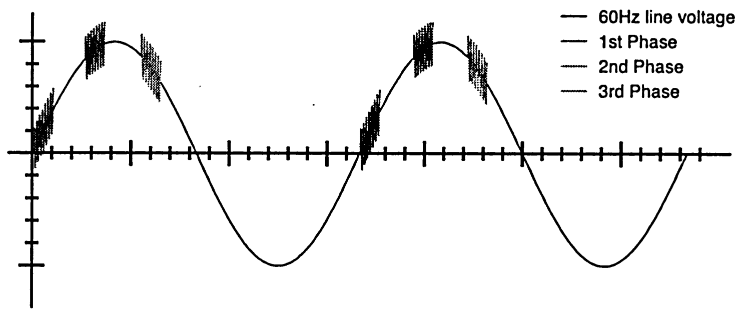


Figure 2.1: X-10 zero crossings.

The protocol requires live power because it depends on its zero crossing to transmit data. At the zero crossing, it sends data bits as bursts of 120kHz, about 1ms long. For a North American 60Hz single phase circuit, the zero crossings happen every 1/30th of a second. For three phase systems, the data pulse is repeated for the zero crossings corresponding to the remaining two phases. Thus, the three data pulses occur within a half cycle as shown in Figure 2.1. Note that X-10 hardware doesn't know or care which half of the cycle is crossing. It only uses the crossing as a synchronization marker. With the exception of the start code, all bits are complimented on the alternate half of a cycle as a simple error check. Codes are transmitted in groups of two, each taking 11 cycles. Each group of codes is separated by three quiet cycles. This results in a pair of commands every 25 cycles [6].

The computer interface is a proprietary control board connected to a serial port that is becoming uncommon in modern computer systems. USB serial adapters have been known to be less than precise on their timing such that many won't work well with the hardware controllers, though this is becoming less of a problem.

While X-10 is still prevalent, a new system is now on the market called Insteon [13]. Insteon can be installed as a standalone system or as a supplement to the X-10 system, moving the control signals off the power lines into the air. This is supposed to reduce interference problems caused by devices injecting noise back into the power system. This isn't a fully wireless system. The modules, such as lighting control don't have wireless transceivers. The Insteon system transfers control over RF back to a power line bridge interface to control devices on the wires. This hybrid approach appeals to customers already invested in the X-10 system but are looking for updated capabilities.

Insteon has two modes of operation, Insteon mode and X-10 mode, switching between them as necessary. While in X-10 mode, it communicates using the X-10 protocol. When in Insteon mode, it allows for a more complex protocol to be recognized. The Insteon protocol works on a higher 131.65KHz carrier, and allows for data to be transmitted, in addition to simple control. The burst period starts approximately 800μs before the zero crossing, lasting 1.823ms. It has a sustained data rate of 2880 bits per second, and a much higher burst rate.

2.2.2 CEBus

The Consumer Electronics Bus (CEBus) was introduced around 1989 as the next step in automation [14]. It proposed a standard based on many goals, including versatility, simplicity, low cost, compatibility, media independence, fast response, and fairness. Some goals and technical issues have been excluded, such as security - there are no provisions for secure communications [15, 16].

The CEBus standard uses four of the layers in the OSI model. They are the Application Layer (layer 7), Network Layer (layer 3), Data Link Layer (layer 2), and Physical Layer (layer 1). The complete specification is quite involved, and is not a simple four-layer model, as some of these layers have sub-layers. Control is handled via the Application Layer, a table-based system with fixed device functionality. Using the common application language (CAL), devices can be mapped to a set of pre-defined node types, such as "binary switch", "analog control", or "keypad", to name a few.

CEBus can also operate on power line and uses 120kHz bursts, which enables it to interfere with the X-10 system. There are some protocol features that enable its coexistence with X-10, but this reduces the maximum bit rate that can carried on the line. It also has the same drawbacks as the X10 system as far as the power line medium itself is concerned. CEBus however, doesn't need live power on the lines as it doesn't use zero crossings. To help combat the problem of interference, it uses a spread spectrum carrier, sweeping from 100Hz to 400Hz as it transmits bits.

There is an X-10 compatibility mode where it will generate an invalid X-10 code for any code that would otherwise be misinterpreted by the X-10 devices. This however, reduces the maximum bit rate from a nominal of 7000bps. A typical packet would be approximately 40 bytes of data, including addressing information. In addition to the power line medium, CEBus specifies many interfaces such as twisted-pair, coax, and others.

2.2.3 LonWorks

LonWorks was developed initially for large scale automation in commercial and manufacturing settings. Since its inception, it has been scaled down to fill in the gap between large scale and smaller home control environments. It is not a home control standard per se, but a standard that enables automation in a wide variety of environments [17].

It is not meant as an end-user application, with development tool kits costing a significant amount. A development kit may contain the development environment along with development hardware for a single or multiple LonWorks nodes. It is intended for hardware and systems vendors to create a system which will then be sold to end-users.

The LonTalk protocol has been standardized as EIA/CEA 709.1-B-2002. Echelon holds patents on the protocol, however the protocol specification is not directly available from them. A license for use can be purchased. Possibly due to cost and complexity, LonWorks systems tend to be commercial systems rather than smaller home systems.

2.2.4 Instabus

Siemens instabus EIB is used throughout their product line [18, 19]. The most visible being their DELTA line of wall-plate switches and controls. Instabus uses a two-wire system, European Installation Bus (EIB), for physical connection and communications. This makes the system relatively inexpensive and easy to install, as it would be similar to pulling telephone wire.

2.3 System Architecture and Intelligence

There are many architectures available for use in home automation. They have different characteristics and the choice can be decided based on cost, available infrastructure, features, reliability characteristics, etc.

2.3.1 Distributed Service-Oriented Architecture

In a distributed service oriented architecture, the devices are not controlled via a centralized controller and may not be connected to one. Each device is capable of talking directly to another device, and is able to save a configuration based on services that the devices export to each other.

There are some advantages to this architecture. Due to the distributed nature, there is no server communications or processing bottleneck. Each device processes its own tasks and communicates to a number of other devices on its own channel. Devices are largely independent (although they interoperate) so they don't rely on having an always-on server to get its tasks from. The devices are also able to be spread over a large area without large infrastructure costs simply by having enough devices close enough to communicate with. They can peer to each other to communicate to other devices that are out of direct range.

These niceties cannot be had without a cost. The devices are harder to manage since they contain their own configuration and are able to communicate and co-ordinate with other devices. The configuration is thus also distributed making large scale reconfiguration more difficult. Devices must also decide whether to pass through traffic to other devices or not. This is required since not all the devices are directly connected, so messages must be sent through devices to reach the others. That increases the communication load on each device. Moreover due to self management, each device must have a more powerful processor as well as more memory to hold its configuration. It will behave as a combination of end-user device and controller due to the co-ordination of tasks with other devices.

2.3.2 Ubiquitous Sensor Systems

There have been experimental systems whereby sensors are placed in all locations, measuring data such as temperature, pressure, electricity, "presence", movement, etc. Sensors (which can be hundreds, or even thousands) are usually one of a serial-linked daisy chain connection, or of massive parallel connections to a single local bus before connecting to a controller. This network of sensors can be used to gather enough data for statistical purposes and used in algorithms or rules.

With appropriate software systems backing them, they are able to determine the actions of individuals without a large amount of assistance by the end user [9]. They can also be used to model the environment and based on the activity within the environment, change its behaviour. With a large number of sensors, 3-dimensional data can even be obtained [20].

2.3.3 Machine Vision

Artificial vision systems have also been experimented with, enabling a computer to identify specific targets in a video frame. Connecting these targets to a backend database allows this identification process. The computer is then able to take actions based on what the target is doing or not doing. The target may not necessarily be a human subject but can be any application specific target.

The application for these systems is typically safety or security systems where there is either no motion expected, or to monitor a target to ensure its safety. Conditions may be set to alert a 3rd-party to take action in case of breach of security or unsafe conditions. The video output itself (or still frames) from the vision system can be used to help 3rd-parties identify and access the condition prior to taking action.

2.4 Software Infrastructure

In addition to the hardware infrastructure and connectivity, software completes the system, abstracting the hardware, implementing features, and presenting them to the user. Home control application software is often limited to scheduling and remote access for small home installations and perhaps even medium to large scale control for energy efficient office environments since they are similar in concept, differing only in scale. Software that is embedded with the hardware that is running it is firmware and is often running at all times in the background. In addition, software running on a separate computer may complete a package, supplying more end-user features, but may not always be running, or may run merely as an easy to use interface to configure a home controller.

The simplest controllers run very simple control firmware. The hardware may include only the basic interfaces, with scheduling and programming based purely in software running on a PC. These are the PC-based systems where the controller can not run without the PC present. More capable systems can have more robust firmware downloaded to flash memory so that it can run independently from a PC. Direct hardware programming interfaces (via keypad and LCD display, for example) are often lacking and can be difficult to program. A software programming interface solves this by providing the user with an easy to use interface. Often a hybrid system is used where a combination of PC and a standalone controller is used. The PC in this case is used to assist the programming of the controller and also as a data store for information systems, or to assist in more complex tasks.

2.4.1 Control, Information, and Data Systems

Many applications today are web-based control for remote access and monitoring, and simple on/off scheduling and scene selection and programming. This is accomplished by attaching a web server to the internet and interfacing the PC to the control system. User authentication takes place and it can then access and run data collection or control scripts in a secure manner. Results are displayed as a generated web page and status updates are done by simply refreshing the page or clicking a refresh button.

2.4.2 Safety Systems

Most home control applications to this date have not had safety as a major factor in its deployment. By safety systems, we mean that the one major purpose to the system is to ensure safety of some subjects in the smart home. Typically only simple home safety features are added in addition to the more common "luxury" functions, such as monitoring for flood conditions (in case where a sump pump may be used). They are mainly an alert to help prevent small problems from turning into costly repairs.

Chapter 3

Smart Home Embedded System Specifications

This chapter describes the details of the Smart Home Control Specification investigated during the course of this research project. It is a functional specification, describing both technical details as well as more nebulous details such as ideas and scenarios.

3.1 Overview

Looking at many projects, such as sensor networks and service oriented architecture, it looks as if the future of smart home control lies in data services. The controller will accept and provide data in many forms, which will be used to describe devices and services they may provide. It also allows data collection for high level services such as data mining, in order to find operational efficiencies or deficiencies, or to provide safety measures. These high level services should be separated from the basic operations of a home controller by means of an API (application programming interface) or "middleware" interface. We define middleware simply as either the intermediate services provided directly by the home controller, or as software functionality above them. These services or functions however, are at a lower level and with lower complexity than the full software services.

3.2 User Experience

The smart home occupant's user experience should be pleasant. This is a subjective area but the idea is that the system should be clear in its instruction when human intervention is needed, and stay "out of the way" when it is operating normally. Ongoing adjustments will inevitably be made, as there is no one magic configuration that will please all the people. The aim is to behave consistently and start with a configuration that will work well for everybody "out of the box."

In our society, most people carry some form of technology around all the time, but neither all the people appreciate it nor want it. Examples include cell phones, pagers, PDAs, and laptops. Some people simply want to get away from these things when away from work, while others will embrace it. The target audience would be someone willing to use the technology around them, and ideally would be curious enough to want to "hack" at the system, for customizing it and expand it as they see fit. Thus the platform must be open to accommodate such activities.

Technology isn't the only element related to a Smart Home. After all, a home isn't a home without its occupants. To that end, it should be user friendly. The occupants shouldn't feel like they have to actively use the home as they do with another device. In essence, if someone didn't want a smart home, they should still be able to live in a smart home just as they would in any other home. At best, the home should be fluidly and intuitively controlled. This is part of the adaptation that the controller must do, but there are some general guidelines that must be defined and followed in the design of the controller. Much of this can be done in higher level software.

Based on some "common sense," and various studies [21, 22, 23], the possible expectations of smart home control from the usage point of view are identified below:

1. Remind the user of pending tasks or time sensitive issues.
2. Don't hinder such a task, or daily living.
3. Allowing transparent overriding of the system. The action should speak for itself.
4. User friendly; The user should not have to fight the system.
5. Customization of user interfaces and interaction.
6. Minimize active interaction or use more passive interaction.
7. Non intrusive technology; Acknowledgement of the technology that's present while being non intrusive.
8. Context awareness; Attempt to observe what the user is doing. Don't simply "execute the next command [7]."
9. Find a balance in the technical and user elements.

Since different people will have different notions of what smart home control can be used for, the system must be able to adapt and accommodate all those notions. The system should be generic enough that it is not tied to a specific technology, and must be extensible to allow for new technology to be integrated into the system. The smart home need not be fully autonomous. Many people find it unnerving that their home maybe exhibit signs of "having a soul." After a period of usage, it becomes comforting to know that the home will adapt and adjust to the occupants.

Our modern-day quest for gadgets that make our lives easier. Historically, and perhaps going forward, automating the home has been the area for the "technologically-inclined" and not the layman. This is partially due to the complexity of the systems, which is only growing increasingly complex. The user interface has been utilitarian and quite terse rather than user-experience based. Hopefully future interfaces may breath some new life into user interface design, such as the multi-touch research [24] and HCI (human computer interaction) studies in general. This is unfortunate, as such a system could benefit everyone.

3.3 Design Goals

3.3.1 Resource repository and management

We attempt to adopt and implement some of the ideas from various projects into a smart home controller that is feasible and reasonable. We start with the centralized control topology, and implement resource sharing by smart appliances. Since a house has limited resources coming into it, the occupants and devices cannot use an infinite amount. The main resources that can be managed include water, electricity, gas, etc. since their use has a noticeable effect on others. Additionally, safety will be managed via a new risk and attention resource.

A water usage example

Flow of water into the house can be measured, and it enters at only one place in the house, which is the sole source of water. A water heater can flow an amount of water in and out, but the temperature of said water will depend on its heating capabilities. After a certain run time, the water flowing into the tank will be quite cold, gradually cooling the water in the tank. Eventually its heating capacity will not be able to keep up with the demand and there will be no hot water.

The smart home controller we are prototyping tries to regulate the usage so that this situation is less likely to happen. The slower the flow of hot water, the more time the heater will have to heat its incoming flow.

Several parameters are entered into the SHCS (Smart Home Control System) regarding water flow and capacity. This forms the basis of the resource lists. When an appliance wants to use any notable amount of water, it should notify the controller by requesting the water resources - a part of the resource sign-out and check-in process, as shown in Figure 3.1. If there is enough (hot water, for example), the controller will grant permission to use it. Otherwise, the appliance is blocked or kept at standby, which can retry at intervals until the resource can be allocated to the appliance. When the appliance has finished with its resources, it checks them back in, releasing it to be used by other appliances. This needs support by a few key appliances, especially the water heater or source. It is not practical to have every single water usage source attempt to request resources. Instead, the controller can monitor real-time usage from a single source, such as a water heater and use that data in combination with its specified limits and current appliance usage to grant or deny new requests.

With any data communications, interruptions may occur. In the case of used and unused resources, a smart appliance should be careful not to go ahead and use resources that it has not successfully signed out, if possible. The controller will also have to periodically poll appliances that it sees in its usage list, to be sure that they are still operational and using the said resources. If the appliance replies negative, or does not reply after a set number of retries, the resource can be forcefully released so that new requests may succeed. The real-time data may also be taken to correlate with its resource list, to check for large discrepancies.

3.3.2 Communications protocols

A simple communication protocol is used on top of the basic TCP/IP, which is illustrated in Figure 3.2. Table 3.1 lists the possible data types supported at this time. The home controller uses the datatype to help establish what to do with the information contained in the packet.

The Status Change Message is issued by the device to the controller when updating the controller of its new status. Information Messages are sent to devices or to the controller as additional information that is not listed under another datatype. The Banter initiation message signifies the beginning of handshaking. The Acknowledgement (ACK) type is a generic acknowledgement message used either for simple receive confirmation or as a posi-

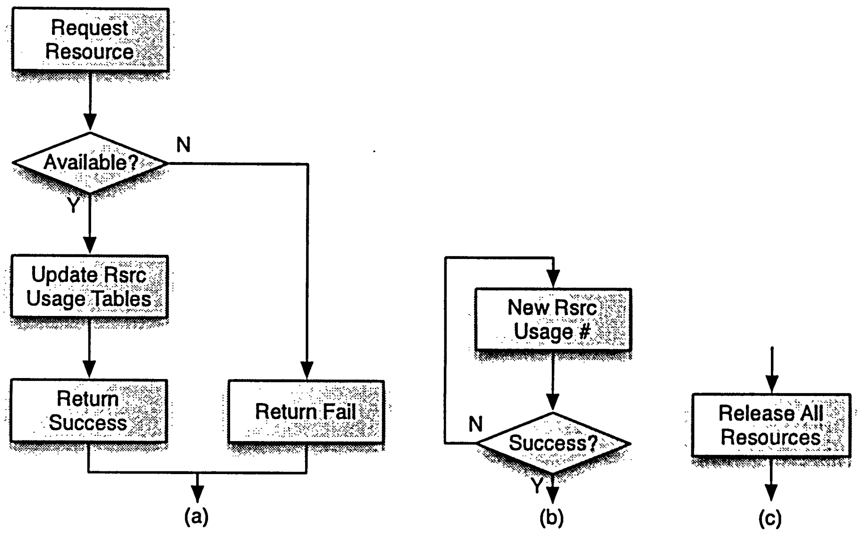


Figure 3.1: Sign-out and Check-in process.

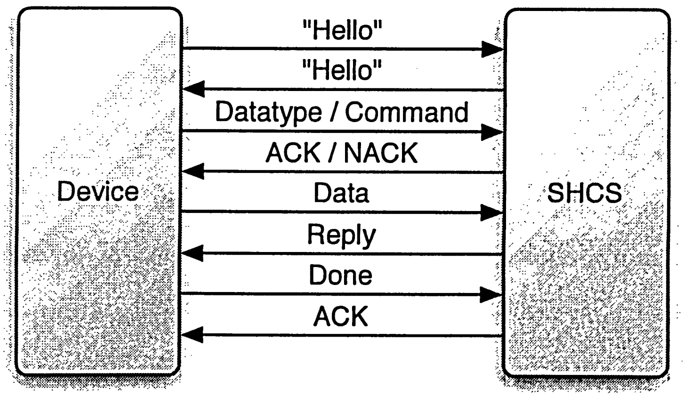


Figure 3.2: "Banter" handshake and data transmission protocol.

tive action. Negative acknowledge (NACK) is used similarly. The Data type signifies that arbitrary data is being sent. Resource Management includes request messages from the device to the controller, and grant or deny messages from the controller back to the device. Device Initialization Processes are used during the initialization and discovery phase. When the device enters its discovery phase, it will attempt to find the controller and send data under this type. The controller will then proceed with its device initialization processes.

3.3.3 Smart display

The Smart Display is named so for its use as well as functionality. It is an information driven device. The display provides a clean and usable interface for the users to see the information provided by the smart home and its controller. It also alerts the users to alarm conditions as set in the controller.

Additionally, the display may be used to change parameters of the controller. Being able to accept user input at the smart display makes the system more intuitive by providing an immediate way for the user to react with the information provided by the system. Ideally,

Type	Description
01	Status Change or Request
02	Set Informational Message
03	Banter initiation Message
04	Acknowledgement
05	Negative Acknowledgement
06	Data
07	Resource Change or Request
08	Device initialization

Table 3.1: Banter datatypes.

this would be a touch screen interface, to enable feedback in a compact self-contained unit. The simplified control loop for the display is shown in Figure 3.3.

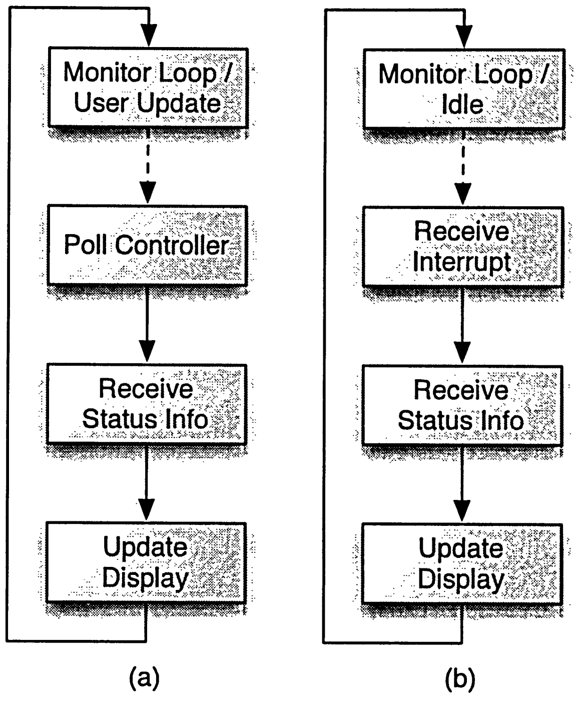


Figure 3.3: Smart Display (a) Pull from server, (b) Push from server.

3.4 Architectural Components

The Smart Home Control System consists of hardware and software components. The controller itself is an embedded SoPC that is attached to the various devices in the home through a standard Ethernet network. The controller's behaviour is programmable through software running on a workstation, also attached to this Ethernet network. A self-programming hardware interface would not be feasible when a system expands beyond a certain size. Providing a web interface from the controller would be better because of the dynamic nature of the variables and devices to be configured, and the visual feedback that can accompany the web interface. The smart display, mentioned in Section 3.3.3, is a possible configuration interface as well.

During the development and simulation phase, software will be used to emulate the data coming from various devices in the home.. This is the Home Simulator component. Both the prototype and a final system will communicate through Ethernet.

3.4.1 Physical Architecture

The main controller is an Ethernet-connected system, using standard Ethernet star topology. The firmware is our custom control software running on an embedded NIOS processor. The prototype board includes an Ethernet interface as well as serial, JTAG, and parallel interface for general purpose I/O (GPIO) including LCD and LED indicators.

The Stratix FPGA accommodates the controller hardware in the form of SoPC modules such as the NIOS processor, avalon bus, and various support hardware for timers, memory, LAN, serial, and parallel interfaces. The firmware can be loaded from flash and then run from SRAM.

The smart home controller is connected to various devices in the home through an Ethernet network. Standard TCP/IP will be used for communication, with software protocols layered on top. These devices talk directly to the controller. The network cabling is expected to be run as part of the construction phase during running of other cables. It can also be retrofitted to old installations using standard methods. Wireless communication can provide the Ethernet medium as well. The home control's Ethernet network is intended to be a completely separate system than the standard home data network. This is for security reasons as well as quality of service.

A sensor network is deployed, but not in a uniform or saturated distribution in the home. The sensors are used to collect specific information, thus minimizing the number of sensors. This sensor network collects data in specific areas and transmits its data to the area's node.

This node is the device that the controller sees and will communicate with.

Targeted sensor network

We are using a targeted set of sensors to monitor specific items of interest, rather than thousands of sensors throughout the structure (ubiquitous sensor networks). The sensor network can be controlled by low level software at the point of interest and relayed back to the smart home controller. This is one of the purposes of having a smart area hub. The hub will act as a device but will be able to convert data such as those from an analog sensor, or even digital sensor and relay them to the smart home controller.

3.4.2 Topology

The two main topologies that we're dealing with are Peer-to-Peer and Centralized configuration. The physical infrastructure can effect our choice, but ethernet networking is flexible enough to support both topologies without too much difficulty.

Peer to Peer

In a peer to peer configuration, all devices are capable of talking to other devices, including but not limited to the smart home controller. This allows maximum flexibility in the home network by allowing more than one pathway for communicating. Devices and appliances are capable of talking to each other, regardless of the presence (or absence) of a main controller.

With a large number of communication hops, this will add some latency, but in the best case it is able to talk directly to its target device. This adds complexity and expense in terms of device-side hardware and software because a device must be able to understand all other devices. They must remember which devices that it needs to deal directly with, and pass on communication to other devices when it's not the intended target, if the target it not "within range" of the originator.

Peer-to-peer becomes a little easier with a wireless system, as the messages don't have to be routed through a physical system. They can hop to any device that can talk with any other device. This allows stringing devices a longer distance from any point in the environment. This may happen if there are obstructions which make a direct connection impossible or impractical.

Centralized

In a centralized configuration, devices can only talk to the main controller, unable to talk amongst themselves. This makes the controller become a communications bottleneck. As long as we aren't being swarmed by a lot of messages, this may work. This simplifies communication because the devices only need to know how to communicate with the controller, which can be called a server. The server will take care of interpreting the information and sending messages to other devices as needed. This centralizes the configuration and interaction of the devices.

This configuration reduces the minimum required processing power of each device, and thus reducing the cost. This is preferred because potentially hundreds of devices in the home will need to be integrated with the system. The devices are mass produced, so every cost reduction here will be substantial when scaled up.

Hybrid

A hybrid topology blends the two approaches. Appliances, devices, and sensors don't all physically connect to a controller, but instead they will connect to a much closer "area node." Any connection in the vicinity can be connected to the area node. Communications for appliances and devices is transparent, allowing them to communicate with the controller as if they had a dedicated connection. The sensors have their own interface on the area node, as they will most likely be analog sensors. Analog sensors require circuitry to convert their data to digital data, which the area node can then send to the controller. This simplifies wiring greatly as an area with many devices only requires one connection to the home controller.

3.4.3 Software Architecture

Home control programming software running on a PC, Macintosh, or any other workstation, speaking to controller using standard TCP/IP protocol. It should be written in a manner similar to that of a realtime operating system, although one is not yet in use. This will help the transition to one, as needed.

There are several software layers present. The low level drivers talk directly to and from the embedded hardware. Sockets and higher level functions are provided above that, and any custom protocols and user functions are written on top of those.

"Synergistic" or "Heuristic" control

Another term for synergistic control is heuristic control, whereby things work together in harmony. This means that nothing may disrupt the operation of another, much like the FCC notices you see on many electronic devices. To paraphrase, the device should accept any interference (or interruption in its operation) but must not cause any disruptions to other devices.

In order for devices or appliances to work together, many scenarios must be defined. These are behaviours designed into the system to provide a somewhat heuristic way of self-management without direct user action. There are two heuristic behaviours that have been studied for implementation in our system.

In the first case, the system can monitor user actions and use mathematical methods to see that 99% of the time, the user does x-action, followed by y-action. After it thinks it has learned the user's behaviour, it can then automatically do y-action when it sees the user do x-action, and it will be fine 99% of the time. In the second case, the controller will follow a rule that is made by the user. It will always follow this rule. It may simply be *do* y*-action when user does* x*-action.*

The second case is most likely to be implemented because it is easier for the user to tell the controller how to behave, rather than have the controller guess. This provides absolute feedback because the rules are known, which don't change unless modified by the user. It is also a lot less likely to fail in such a way as to confuse the user. This follows the reasoning that the user should feel in control, even when not directly managing the details.

Resource Management

Resource monitoring and control is a quality of service feature. It helps to provide arbitration and feedback to devices, and allows collection of data for measuring usage efficiency. It's handled by the smart home controller and is partially fed by the sensor network. Resource tables are initialized to specific numbers, and are modified as devices use these resources. These initial values and limits will be set by the user.

The "standard" resources are electricity, hot and cold water, and gas. These are some of the common resources that are consumed and charged for by the utility companies. As appliances are used, and perform various tasks, they will require different resources. They will request a resource change to the smart home controller, and the controller will grant or deny these requests based on the existing resource usage. In addition, risk and attention resources can be used.

Risk resource management

If a particular user operates many devices, the attention of the associated user will decrease according to the needs of the device. This is one concept built on the simple resource management system. It includes two more resources, "risk" and "attention," that helps to determine if a device can safely be used. It can also be assigned to work areas to maintain the integrity of the area.

The risk resource identifies a risk level associated with the device, or operation of the device. The attention level represents the attention of a particular user. The idea behind this is to ensure that a user's attention doesn't get divided into too many places, increasing the risk of injury. This applies to running and appliance or leaving appliances and work spaces unattended. If a threshold is reached, or the attention level is not sufficient, than the system will warn the user and may also not allow the use of the device. Optionally the user will be reminded of other "high risk" devices that they are currently using.

Passive sensor network

Smart home controller software reads passive sensors at time intervals and updates internal status tables as necessary. Active sensors, such as motion detectors trigger events, which are recorded with timestamps, and is a related, but different system. The sensor network provides the passive feedback that the controller can use to make decisions.

Logging

The user may choose to log information about their home, in order to get useful history of resource usage. This may be used to help monitor energy consumption, for example, to help decide a conservation method if one is needed.

In addition to this, any add-on components may require that events be exported or logged, so they can see them and react as appropriate according to their designed purpose.

3.5 Additional Features

In future work and finalization for a usable base system, re-introduction of standard and advanced home control functionality will need to be done. Expansion module for current sensing circuits and continuous monitoring and alarms. For example, Veris Industries provides hardware components needed to monitor large and small scale electrical systems [34].

3.6 Scenarios and abilities

To further illustrate capabilities expected of the smart home control system and possible uses, a couple of scenarios involving the risk and attention resources follows below.

Monitoring and Notification for Safety

We'll use the scenario of having a small child in the house (let's call him Eric) that is old enough to easily get around and understand what's around them, but still may not have a full understanding of the consequences of their actions or some of the things in their surroundings. The parents may be up and around the house taking care of errands while Eric is, at the moment, playing in the living room.

While the idea of a tracking device is disliked, in some cases it makes sense to have an identifiable tag so that the system can act as an extra "set of eyes" watching Eric. In this case, it's simply to help keep Eric out of danger and to get the attention of his parents when he starts to do something potentially hazardous. Some dangerous places can be physically locked out such as a basement workshop, but in many cases there are many small hazards around the household. Eric may simply be roaming around and his parents should be notified if he slips away from attention for too long.

In another scenario where a senior may be left relatively unattended at home, so-called "dangerous appliances" should also be monitored for unattended use. Many fires have been started over an oven or stove turned on and then forgotten.

Monitoring and Control for Safety

Another example of risk that's often "set and forget" is the temperature of the water heater. If set too low, there is a risk of bacteria growth, and if set too high, there is risk of scalding, especially in young children. Although the thermostat for the water heater is set, it can malfunction. While the risk can be set low, as failures are not common and most homeowners probably don't actively check the temperature of their water heaters, the risk is higher for higher temperatures. Constant monitoring can be done to ensure the temperature is within the preset range.

Showers, tubs, and other areas where large amounts of hot water can be accessed would be part of this monitoring network and reminders can be made to the caregivers to be careful with the water around children. An automatic shutoff can occur if the child is left alone, for example. Users can start with a sparse system and add more monitoring and control as they see the need.

Chapter 4

Smart Home Controller

The "desktop implementation" of the system includes the smart home controller as an SoPC, and home simulation software hosted on Mac OS X. Since we do not have a "live" home environment or model home lab delivering messages to the controller, we simply do this by using a home simulator as shown in Figure 4.1. This is possible due to the network connection available at the controller. All devices including the controller itself are connected via a standard ethernet network in order to communicate. This is one of many options that can be taken and it is the default.

Development and testing are done by using the combination of hardware and software. As the controller evolves, so does the testing software. As both become more complex, the software can be forked to create a custom interface to the final smart home controller. This would be an optional software program which is able to take advantage of (or even create) the more complex features of the controller.

4.1 Controller hardware

Our smart home control system prototype consists of readily available parts including an Altera Stratix development kit. This enables us to communicate using the ethernet port,

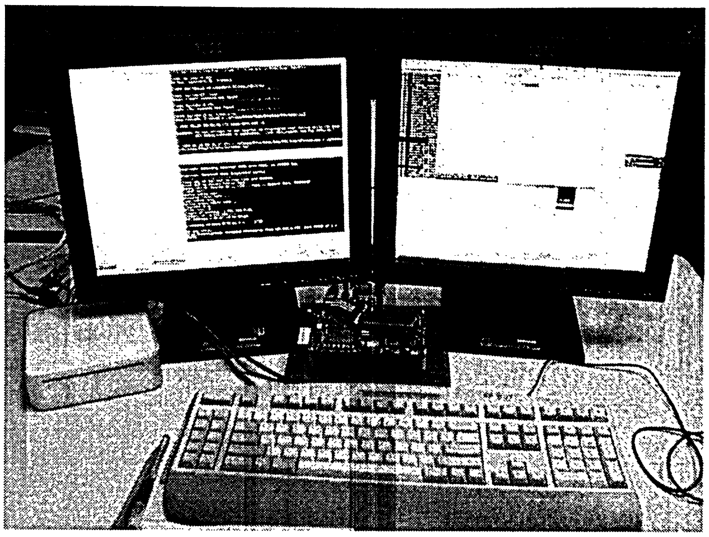


Figure 4.1: Photograph of the SoPC development workstation, board, and Mac running the simulation software.

serial port, and a 2-line LCD display. Smart home devices and appliances communicate across an ethernet network, using TCP/IP as the standard protocol. The serial port is used in this system as an output terminal for JTAG and debug output only. The LCD module at the controller is used for short status messages when appropriate.

The Smart Home Control System was designed as a dedicated embedded computer system. A typical embedded SoPC is shown in Figure 4.2. While general purpose computers can be more powerful, they are too expensive in this configuration in all the areas of cost, power consumption, heat, and reliability. The basic decisions for key considerations are outlined below.

- Hardware cost

  The initial cost of hardware is lower for the embedded system because it includes the basic necessities and nothing more. An SoPC (FPGA) and few support chips are all that is required. Creation of a custom PCB for final design may incur some costs, but

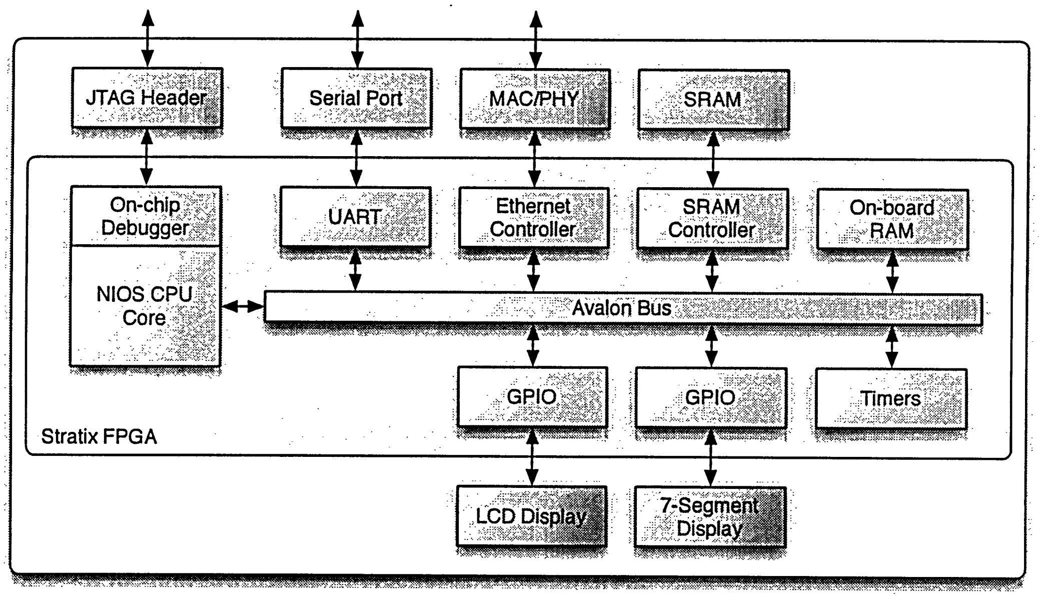


Figure 4.2: SoPC internal IP blocks and on-board components.

mass manufacturing brings costs down to reasonable levels. These boards do not need special care in terms of handling fast (megahertz or gigahertz) clocks and high speed RAM signalling requirements.

- Power Consumption

  We are using an SoPC with a softcore processor running at 50MHz. This is enough speed for the various program loops and requires minimal power. The support chips also have low power requirements. There are no fans or harddrives, which require a notable amount of power to start and stay spinning.

- Heat output

  Due to the low running frequency, the cooling system does not require any heatsink. High heat generation will have an effect on operational costs, and in some cases affect the reliability as well.

- Operational cost

  This is mainly affected by power consumption as we're not dealing with consumable items. Again with the slow clock and no moving parts we require minimal power, making this cheaper to run.

- Computational power

  We have much less computing power available as compared to a general purpose computer, however, specialities such as floating point processing, streaming instructions, or multimedia features aren't needed in a simple home control situation. With the advent of multimedia computing, which is arguably a "next-step" in smart home control systems, the features can be added via self contained plug-in modules. The modules can include high performance computing, or interface with a fully capable workstation. This translates into having a simpler system as a base for a comprehensive system.

- Custom Hardware

  One benefit of an FPGA based SoPC is the programmable area. If a small amount of custom hardware is required, or a system bug needs a hardware fix, it can be accommodated by re-flashing the SoPC. The hardware cost remains constant since the FPGA is already present. If any additional hardware module is required, it can also be added off-board and a provision is made to the system (i.e. memory map changes) by re-flashing the SoPC.

- Reliability

  The embedded system is inherently more reliable than a low-end PC because of low heat output and no moving parts. While flash memory has a finite number of writes, only a selected number of configurations will be saved to flash. Most or all of the running system variables will be stored in SRAM, which is faster, and won't wear out.

4.2 Altera Development Environment

Quartus II and SoPC Builder work together to facilitate the construction of an embedded system having a NIOS processor core and glue logic for support interfaces to memory, networking, etc. Using SoPC Builder, IP cores and memory mapping can be adjusted. Custom logic developed in VHDL or Verilog using Quartus II can be added into the system at any time, and the system can be re-created. Table 4.1 lists the IP cores used in the smart home controller design. Additionally, the NIOS processor core is configured as shown in Table 4.2.

Table 4.1: IP Cores used in Controller Prototype.

Device	**Address**	**Size**
ext_ram (SRAM)	0x00800000	0x00100000
onchip_ram_64_kbytes	0x00900000	0x00010000
lan91c111	0x00910000	0x00010000
boot_monitor_rom	0x00920000	0x00000800
cpu	0x00920800	0x000000FF
uart1	0x00920900	0x0000001F
timer1	0x00920940	0x0000001F
lcd_pio	0x00920970	0x0000000F
seven_seg_pio	0x00920990	0x0000000F
low_priority_timer2	0x009209E0	0x0000001F

The Avalon bus connects the on-board components together within the FPGA, such as the CPU, memory controllers, and any custom logic, into a system [25]. The bus was designed with SoPC design in mind, and has many features over a "standard bus" or "just a bunch of wires." The trade off is allowing simplification of the master and slave interfaces at the expense of a more complex bus. As an example, the bus allows for multiple master

Table 4.2: NIOS CPU configuration.

Name	**Device**	**Offset**	**Address**
Reset Location	boot_monitor_rom	0x00000000	0x00920000
Vector Table	ext_ram	0x000FFF00	0x008FFF00
Program memory	ext_ram		
Data memory	ext_ram		
Primary serial	uart1	0x00920900	

devices, but each master can act as a lone master. In our implementation, we have two masters due to the NIOS processor core. The NIOS processor uses a master each for data and instruction bus connections.

NIOS is a 32-bit RISC processor based on the Harvard architecture [26]. It has a 5-stage pipeline, 16-bit instruction set, and a customizable ALU (arithmetic logic unit). We are using an unmodified NIOS processor core running at 50MHz as a general purpose processor to execute our smart home controller code. 64kB of on-board memory and 1MB of SRAM is available to the processor as boot loaders, program monitor, instruction and data storage areas.

The smart home controller code is written in C, using standard libraries and Altera-supplied libraries for functions relating to the specific IP cores. LCD display routines and network initialization are two examples of such functions. The build and debugging tools are based on gcc and gdb, generating a standard ELF binary that runs on NIOS.

Serial UART and parallel general purpose IO (GPIO) are used for simple displays. The serial connection is used as standard output from the controller code, but it can also be used as input as well. GPIO are used for the LCD and 7-segment display, which display short messages and indicators, respectively.

An Ethernet connection is available and is the main communications medium for the smart home controller. It is a standard 10/100Mbps capable PHY (physical interface layer) but is currently limited to 10Mbps due to the clock speed. This is more than enough bandwidth for home control.

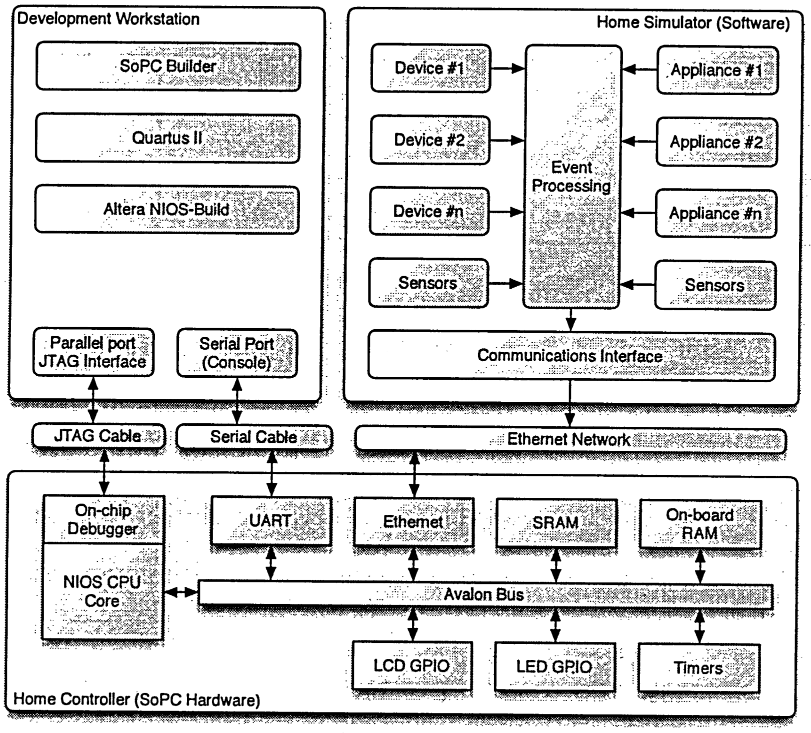


Figure 4.3: SoPC development workstation, board, and simulator connections.

During the development process, the SoPC board is connected to both the workstation hosting the development software as well as to the workstation hosting the home simulator program as shown in Figure 4.3. Programming of the SoPC is enabled via a JTAG cable to the workstation. A serial connection provides a console to display various output messages. The embedded home control system communicates to home simulator via Ethernet.

4.3 Topology

We are introducing a modified and hybrid centralized topology. Most devices and appliances are physically connected using star topology. The overall communications employs a centralized (master-slave) system. The system expects direct device to controller communication except where basic sensors are used. In this case, the sensor is connected to an area hub which covers all the sensors for that area. The hub acts as a device to the controller and doesn't need a special case for them as it behaves as if it were the sensor itself.

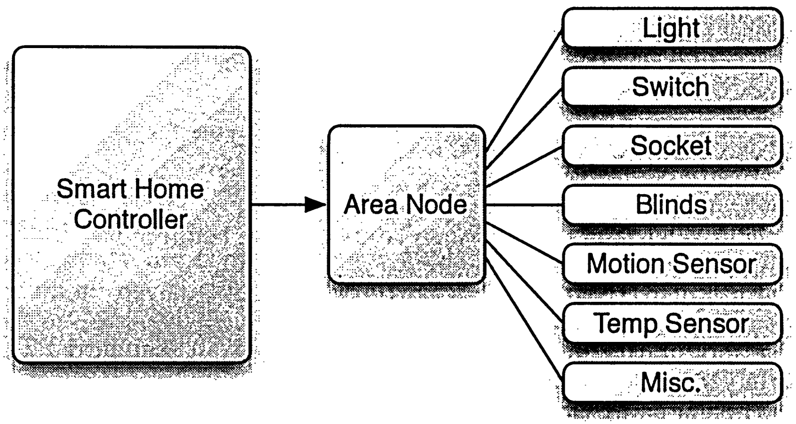


Figure 4.4: Area Hub and Sensor Topology.

4.3.1 Device Communication

The Altera SoPC development environment has working low-level software support, but higher level software functions must be built on top of it to increase its usability. We are layering a handshaking protocol in software for device-to-host communication. It is a simple "banter" protocol illustrated in Figure 4.5.

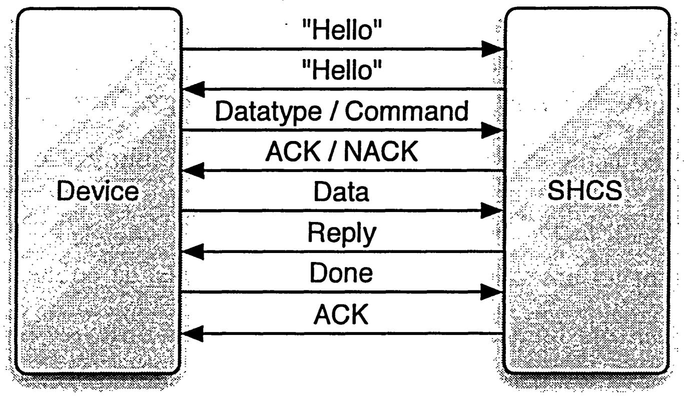


Figure 4.5: "Banter" handshake and data transmission protocol.

4.4 Smart Home Control Software

The smart home controller is essentially a software system running on the NIOS processor. The current structure of the software is a partially interrupt driven monitoring loop. It is written with some modularity in mind, in case of porting to a standard RTOS (Real-Time Operating System) such as uCOS-II.

Upon startup, the controller executes its initialization stage. This initialization includes interrupts for display, buffers, networking and communications, resource tables, and device and appliance status tables. After this, the controller enters its monitor loop.

The monitor loop contains various functions that are represented in Figure 4.6. Essentially they are "scheduled" in a round-robin style. This loop is slowed by a sleep call so that various function calls (checks and maintenance) are done approximately once every second, and will each run to completion. To show the operation of the controller, an LED blinks as a heartbeat. The sleep loop allows for easily stopping the process when debugging by monitoring for keypresses, specifically when the Escape key is detected. Similarly, a push-button routine is available for debugging networking, which resets the handshake protocol and communication states. In an RTOS scenario, the calls would be scheduled according to

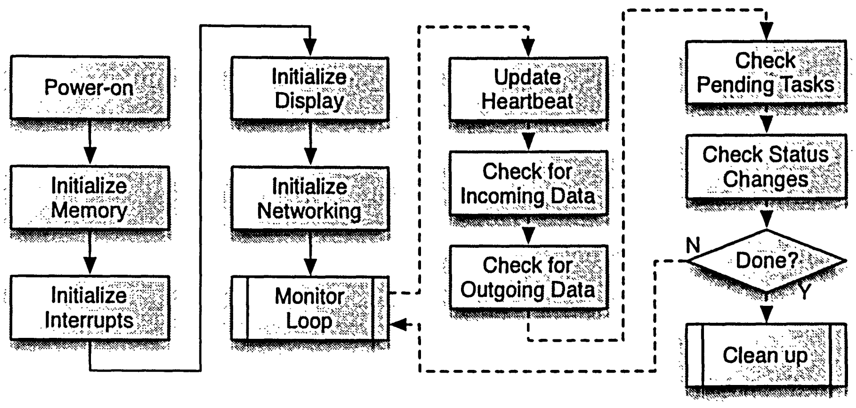


Figure 4.6: Smart home controller monitor loop.

the scheduler and must be made safe for pre-emption by careful use of any global variables and protection thereof.

The smart home controller is event driven. Aside from timer interrupts, the event is data arriving on the network. This means that a device is trying to communicate. This is taken care of by interrupt service routines which process the data and place partially processed data into buffers. These buffers are accessed by the monitor loop as maintenance items. It sends out any data that needs to be sent (in accordance with the banter protocol), and processes any maintenance that needs to be done on the controller's status and resource tables.

4.4.1 Simulation and Verification

During the design implementation of the smart home controller, a home simulator has been developed to provide the messages that are needed to adequately test the system. This simulator mimics all the devices and appliances in the home and provides feedback to the user and the controller.

It is programmed and executed on a Macintosh running OS X 10.3.9, using Objective-C and the Cocoa framework. This allows for an adequately fast production of both the status interface and the programming interface for the controller and simulator. The simulator block diagram is illustrated in Figure 4.7. In the simulation, since there are no physical sensors in place, all sensor reads and data acquisition are numerical calculations based on data stored in tables. There currently is no physical sensor read in the program loop.

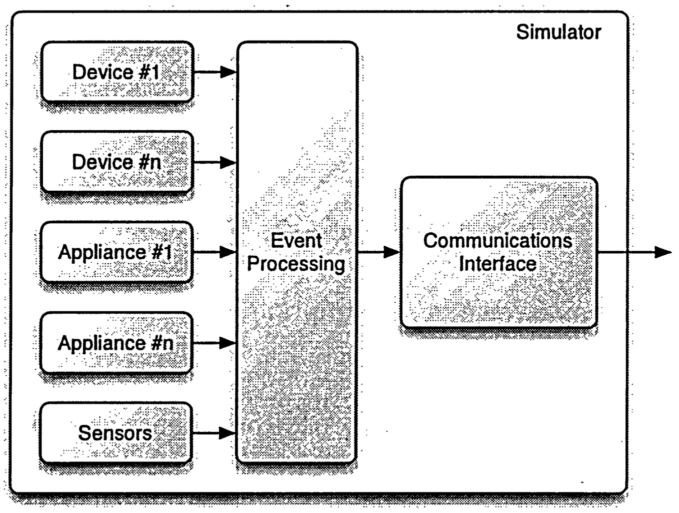


Figure 4.7: Simulator block diagram.

The display of the current status and the control of the devices would ideally be a dynamically created interface. This would automatically match the system that we are testing - this is especially true for the status page. Since we are dealing with a data system, not hardwired signals, this would be an asset. Unfortunately being a prototype, the interface design is not created a style that fits all the goals of being appealing, functional, and easy to use.

The simulator software has two primary windows. These windows are the status and control windows. The status window is akin to the Smart Display idea, as it shows the current state of the controller, and can be used to bring any situation to the attention of the user.

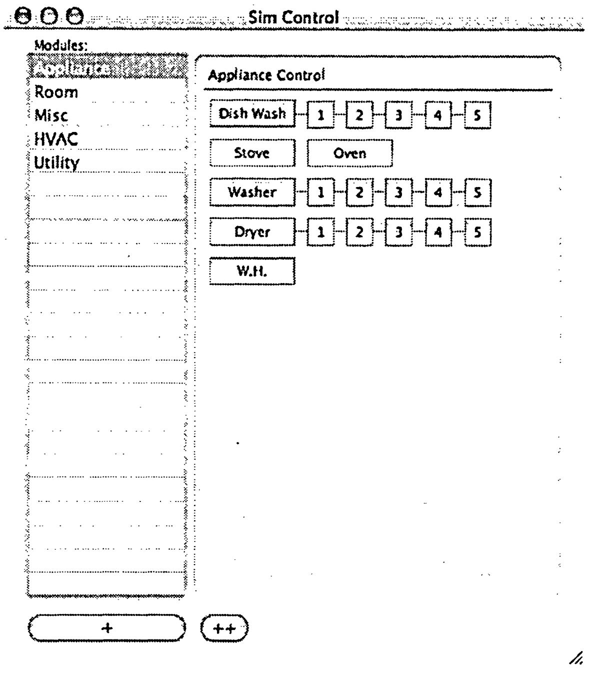
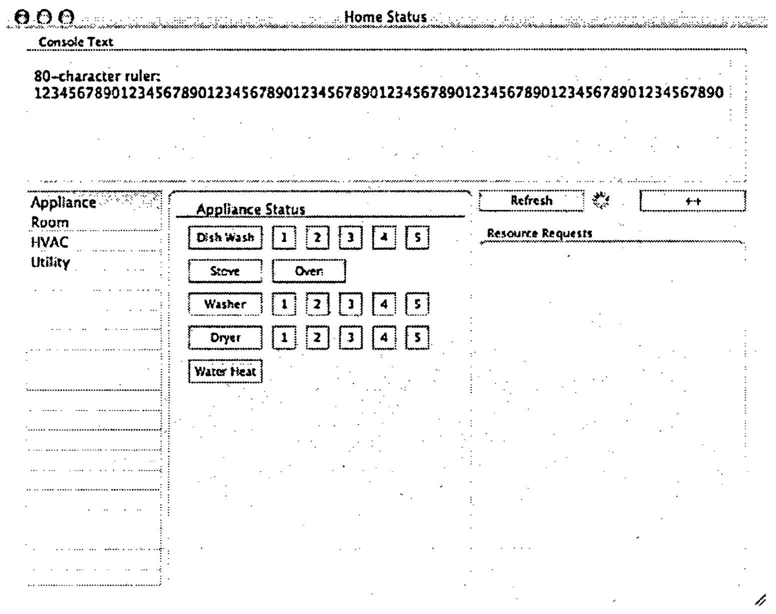


Figure 4.8: Simulator control window.

The control window is equivalent to the house that the smart home controller is installed in. It is the simulator. The control window generates the data that the smart home controller will interact with. As stated above, the status and control window are currently not dynamically generated and contain a fixed number of devices to be simulated.



```
80-character ruler:
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

Figure 4.9: Simulator status window.

On the development workstation, several utilities are available to build a system (the Altera development tools). One such utility, the NIOS Console, is shown in Figure 4.11, which is used to download code to SRAM and execute it on the SoPC hardware. It is possible to issue debugging commands from this console, such as memory dumps, as well as simply running the code.

The simulation run shown in Figure 4.12 is the output of the controller console after power-up. The software configures input and output ports (GPIO), interrupt service routines, global variables and networking. The current configuration has the controller listening on port 8738, with an IP address of 192.168.0.51. After initialization, the short message "We

```
[Session started at 2006-09-01 13:45:58 -0400.]
2006-09-01 13:45:59.912 SHCS[6438] SocketControll init
2006-09-01 13:46:08.063 SHCS[6438] Show mWindow: 0x3ca0b0
2006-09-01 13:46:08.075 SHCS[6438] simControl mWindow: 0x3ca0b0
2006-09-01 13:47:47.110 SHCS[6438] SHCS: Connecting to 192.168.0.51 on port 8738
2006-09-01 13:47:47.110 SHCS[6438] SocketController: disconnect
2006-09-01 13:47:47.137 SHCS[6438] SHCS: Connected
2006-09-01 13:47:56.121 SHCS[6438] [SHCSApp:Delegate:sendtocontrol]:Data:
2006-09-01 13:47:56.121 SHCS[6438] POWER
2006-09-01 13:47:56.121 SHCS[6438] Banter[S1]
2006-09-01 13:47:56.122 SHCS[6438] SHCS NetSocket: Data sent
2006-09-01 13:47:56.594 SHCS[6438] SHCS NetSocket: Data available (268)
2006-09-01 13:47:56.595 SHCS[6438] SHCS NetSocket: Data:    1  16HELLO
2006-09-01 13:47:56.595 SHCS[6438] SHCS NetSocket: decode: 1, 16
2006-09-01 13:47:56.595 SHCS[6438] Banter[R1]
2006-09-01 13:47:56.595 SHCS[6438] talk: [HELLO
2006-09-01 13:47:56.595 SHCS[6438] Banter[S2]
2006-09-01 13:47:56.595 SHCS[6438] SHCS NetSocket: Data sent
2006-09-01 13:47:58.651 SHCS[6438] SHCS NetSocket: Data available (268)
2006-09-01 13:47:58.651 SHCS[6438] SHCS NetSocket: Data:    1  16ACK
2006-09-01 13:47:58.651 SHCS[6438] SHCS NetSocket: decode: 1, 16
2006-09-01 13:47:58.651 SHCS[6438] Banter[R2]
2006-09-01 13:47:58.651 SHCS[6438] talk: [ACK
2006-09-01 13:47:58.651 SHCS[6438] Banter[S3]
2006-09-01 13:47:58.652 SHCS[6438] SHCS NetSocket: Data sent
2006-09-01 13:48:00.857 SHCS[6438] SHCS NetSocket: Data available (268)
2006-09-01 13:48:00.857 SHCS[6438] SHCS NetSocket: Data:    1  16Custom Reply
2006-09-01 13:48:00.857 SHCS[6438] SHCS NetSocket: decode: 1, 16
2006-09-01 13:48:00.857 SHCS[6438] Banter[R3]
2006-09-01 13:48:00.857 SHCS[6438] Banter[S4]
2006-09-01 13:48:00.858 SHCS[6438] SHCS NetSocket: Data sent
2006-09-01 13:48:02.766 SHCS[6438] SHCS NetSocket: Data available (268)
2006-09-01 13:48:02.767 SHCS[6438] SHCS NetSocket: Data:    1  16ACK
2006-09-01 13:48:02.767 SHCS[6438] SHCS NetSocket: decode: 1, 16
2006-09-01 13:48:02.767 SHCS[6438] Banter[R4]
2006-09-01 13:48:02.767 SHCS[6438] Banter: Finished
2006-09-01 13:49:22.251 SHCS[6438] SocketController: disconnect
```

Figure 4.10: Simulation run log - Device communication.

```
System Analyzer for Nios® Processor
Serial number Altera
Version 1.7.5 build 2
Copyright (C) 1998-2003 First Silicon Solutions, Inc.
Limited Version
User reset.
00920004  9802  pfx %hi(0x40)
00920006  6D20  movhi %g0,0x9
Loaded jchan.srec
pc 0x008036F8
o7 0x00490000 (reset address 0x00920000)
1>
```

Figure 4.11: NIOS Console.

```
----------------------------------------
Press any of SW0-SW2 to show time and button.
Press SW3 to escape banter lock.
[main(init)]: SW_ PIO .. LCD .. rsrc .. Global Init
Device table init
Resource table init
Comm table init
Packet Queue table init
plugs ..  [lan91c111] nr_lan91c111_reset: chip id = SMC91C11xFD
 [lan91c111] r_lan91c111_detect_phy: found lan83C183 (lan91C111 internal)
 [lan91c111] r_lan91c111_init_phy: 100bt
 [lan91c111] r_lan91c111_init_phy: full duplex
   [plugs] +-----------------------
   [plugs] | initialized adapter lan91c111 at 0x00910000, 192.168.0.51
IP address  = 192.168.0.51
Gateway IP  = 192.168.0.1
MAC address = 00:07:ed:0c:04:b0

Listening on port 8738 at t =     1759

We are alive.
```

Figure 4.12: Simulation run log - Controller initialization.

are alive." is displayed and the controller enters its control monitor loop.

The simulation run shown in Figure 4.13 is output from the home controller console during the handshaking and data transmission routines. It contains debugging information such as the banter protocol states for each device, when it enters each phase of the protocol, and the data contained in the packets.

4.5 Smart Home Resource Control

Resource control is an interesting data service. The sensor network provides information that is available for the software layers to interpret. Resource control is one of these software layers. While most day to day activities are self- or user-regulated, it may still be useful for the uncommon situations that arise and can cause a heavy shift in resource usage, such as doing extra loads of laundry during the evening after an extended trip away. An automatic lawn sprinkler may use a lot of water as well as the laundry. In this case the system may want to put a hold on the automated sprinkler to give priority to user activities.

Resource Management

The initial implementation of the resource software layer results in providing limits to the

```
[tcp_listen_proc]: Accepted connection from 192.168.0.101  port 60502 at t =
70461
[tcp_proc2]: data comes in, length 268
pqAdd()
pqAdd - Add link
pqPeek: Packet found
[cFD(): checkForData found PQ_ONE data.
cFD(): Packet decoded, going to banter.
<Banter>
Banter state for tag 16 not found, adding
Banter[c1]
[decode]: sendr:raw: [  16]
[decode]: recvr:raw: [   1]
[decode]: data: [HELLO


                                ]
pqAdd()
pqAdd - Add link
Device 16 found, removing...
</Banter>
[tcp_proc2]: data comes in, length 268
pqAdd()
pqAdd - Add link
pqPeek: Packet found
[cFD(): checkForData found PQ_ONE data.
cFD(): Packet decoded, going to banter.
<Banter>
Banter state for tag 16 is 2
Banter[c2]
[decode]: sendr:raw: [  16]
[decode]: recvr:raw: [   1]
[decode]: data: [TheDataType


                                ]
pqAdd()
pqAdd - Add link
Device 16 found, removing...
</Banter>
[tcp_proc2]: data comes in, length 268
pqAdd()
pqAdd - Add link
pqPeek: Packet found
[cFD(): checkForData found PQ_ONE data.
cFD(): Packet decoded, going to banter.
<Banter>
Banter state for tag 16 is 3
Banter[c3]
[decode]: sendr:raw: [  16]
[decode]: recvr:raw: [   1]
[decode]: data: [POWER


                                ]
pqAdd()
pqAdd - Add link
Device 16 found, removing...
</Banter>
[tcp_proc2]: data comes in, length 268
pqAdd()
pqAdd - Add link
pqPeek: Packet found
[cFD(): checkForData found PQ_ONE data.
cFD(): Packet decoded, going to banter.
<Banter>
Banter state for tag 16 is 4
Banter[c4]
[decode]: sendr:raw: [  16]
[decode]: recvr:raw: [   1]
[decode]: data: [DONE


                                ]
pqAdd()
pqAdd - Add link
Device 16 found, removing...
</Banter>
```

Figure 4.13: Simulation run log - Communications debugging.

use of resources. Devices and appliances that know that they will use a significant amount of resources will request to sign out resources from the system before using them, and check-in the resources when they are done with them. Figure 4.14 illustrates what a typical appliance control loop looks like. The appliance knows what tasks it is working on and will request and return resources as necessary until it is done, when all resources are returned for other devices and appliances to use.

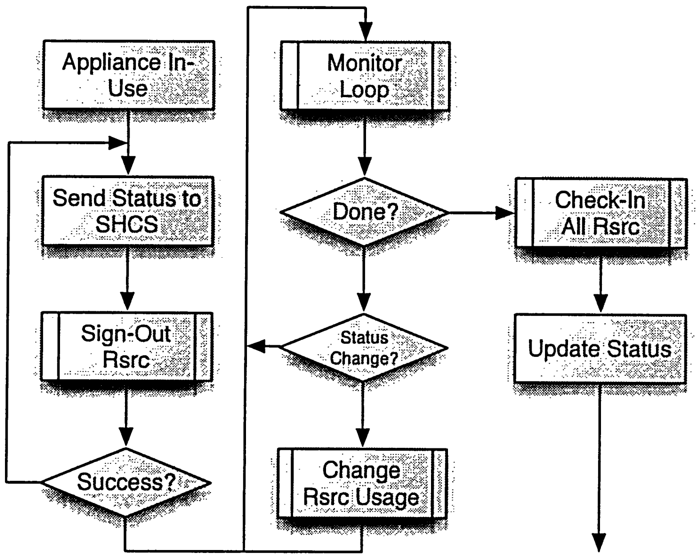


Figure 4.14: Appliance control loop.

Risk Resource Management

The risk and attention resources are managed in identical fashion to the three basic resources, with regards to the devices. Any change in the operation of a device can cause it to request changes in risk and attention resources.

Management Issues

All the resources are initialized and tracked by the controller. The limits are either set by

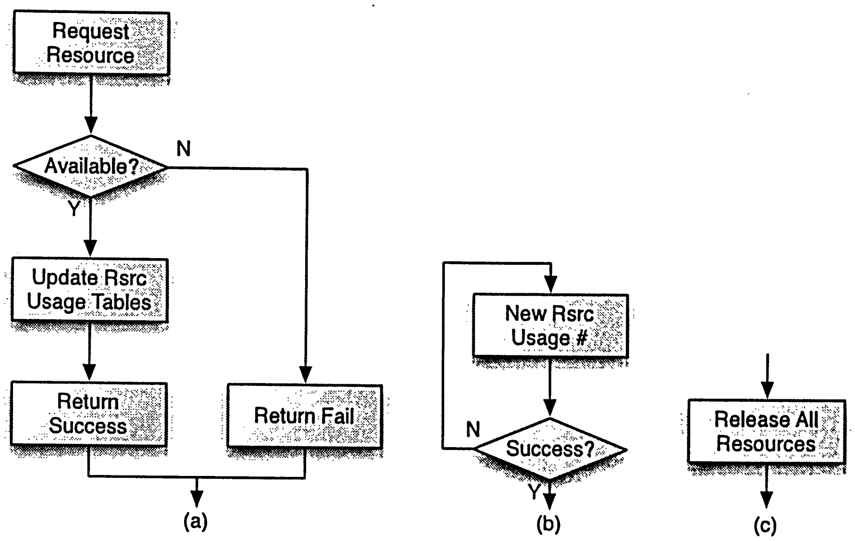


Figure 4.15: Resource (a) sign-out, (b) change, and (c) check-in procedures.

the user in the simulator control window, or set by appliances. Most resources start at a finite level and decrease as device and appliances use them, however, it is possible that one starts at zero, and increases as a device or appliance supplies or creates that resource. One such resource is hot water created by the water heater.

When the water heater is activated, it tells the controller to create a hot water resource. This resource can then be used by other devices or appliances, such as washing machines, dishwashers, showers, etc. The initial value of hot water flow and capacity are sent by the water heater. As hot water is used, the hot water flow and elapsed time are monitored. Using this data, its remaining capacity can be calculated. The water heater itself can track this information and send updates to the home controller when conditions change.

The resource control rules can be illustrated with a scenario. A modern water-saving shower head can flow between 450-1140 litres per hour. Let a dishwasher and washing machine each use 500 litres per hour when operating. The home controller is set to allow up to 1500 litres of water per hour. If a person is taking a shower, using about 800 litres per

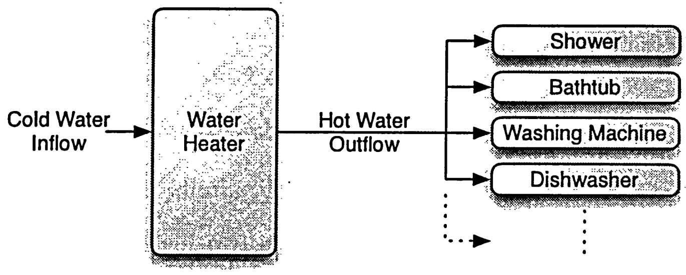


Figure 4.16: Hot water supply.

hour of water, and the washing machine is filling, we have a total of 1300 litres per hour. A partial run log of this situation is shown in Figure 4.17. This is now only 200 below the limit. At this point, if the dishwasher was activated, the controller sees that the request would place it above the limit and denies the request. At this point, the dishwasher can wait and try again later. Since it only takes a few minutes for the washing machine to fill before its wash cycle, the dishwasher will have its request granted after the washing machine has stopped filling.

```
Hot water: 250 / 1000
Cold water: 250 / 1000
Total: 1300 / 1500
pqAdd()
pqAdd - Add link
Device 16 found, removing...
</Banter>
[tcp_proc2]: data comes in, length 268
pqAdd()
pqAdd - Add link
pqPeek: Packet found
[cFD(): checkForData found PQ_ONE data.
cFD(): Packet decoded, going to banter.
<Banter>
Banter state for tag 16 is 4
Banter[c4]
[decode]: sendr:raw: [  16]
[decode]: recvr:raw: [   1]
[decode]: data: [DONE


                                   ]
pqAdd()
pqAdd - Add link
Device 16 found, removing...
</Banter>
```

Figure 4.17: Simulation run log - Water resource is approaching the usage limit.

Chapter 5

Conclusions and Future Work

The smart home controller is a standalone system that enables devices and appliances to communicate over a standard Ethernet network. This communication allows for many features to be implemented such as resource management in terms of usage tracking and control. These features allow for greater quality of service and increased safety of the home occupants. The smart home controller is implemented using an Altera SoPC development environment. This provides the required hardware, such as the NIOS processor core, communications hardware, and glue logic. The firmware is written using Altera's NIOS build tools. It's an adequate platform for prototyping and demonstrates what SoPC can do for a final implementation.

Simulation of the home is provided by a software system running on Mac OS X, which communicates with the controller over an Ethernet network. Several appliances have been implemented, communicating with the controller for resource management. We are working without a model home environment and the simulator provides required inputs needed from the devices and appliances in the home. This is actually quite flexible and can be used before finalizing the hardware devices and modules. Protocol implementation can be changed easily at this point in development together with the hardware as they both evolve. Ethernet is a very good choice because it allows easier design and testing. Common two-wire or proprietary bus communications would require custom hardware interfaces to be built before software testing could be done. Being able to use common readily available equipment reduces anxiety associated with being locked-in to a specific vendor's proprietary hardware.

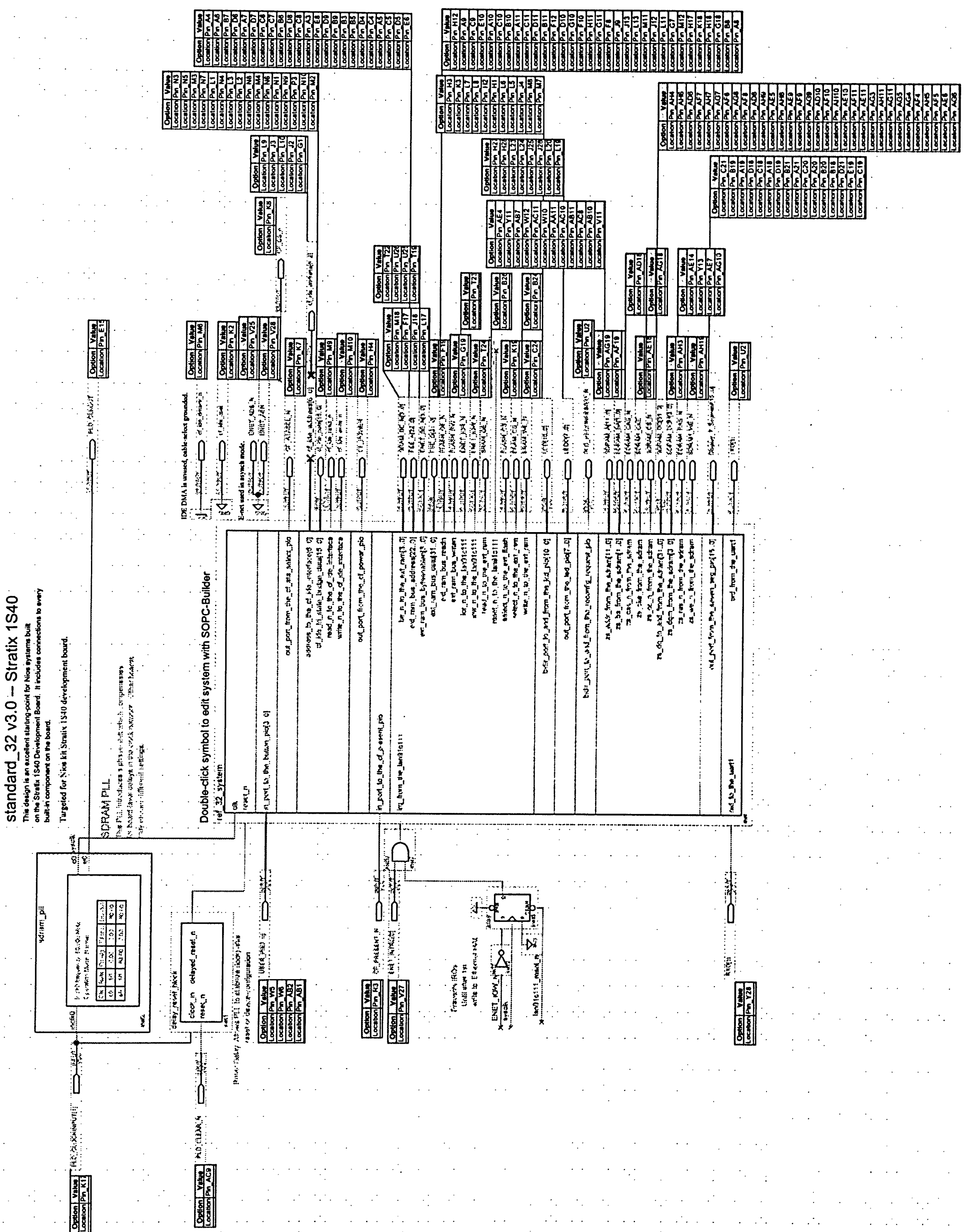
The network connection allows resource management to be implemented, which enables the controller to track resource usage, providing quality of service in the smart home. Water usage and its problems can be easily observed in terms of water pressure and flow rate. Resource management implemented in this project, via usage tracking and defined rules, allows some degree of control to prevent problems from occurring by limiting concurrent use of devices and appliances that heavily use one particular resource. Usability is good but only with some defined rules. Smart home occupants have a system that behaves in a consistent manner, allowing an easier transition from traditional home to a smart home. While these aspects of the smart home controller meet our goals, there are many improvements still to be made.

Aside from more real-world applications to be developed, the base system will also need updating over time. While the current design is flexible and allows for different standards to be applied, there may be situations where it needs to be updated to accommodate newer technologies and facilities. One issue is that the controller is inflexibly written in C. This is due partially to the facilities of the existing development environment as well as our key considerations for the development of the prototype. This has the consequence that any change to the controller, either because of new features or other changes, requires that the code be re-compiled. This raises the barrier of entry for customizing and working with the controller in general. The change that is required is to write the main controller logic using a interpreted or scripting language, which would then run by an interpreter or virtual machine on the controller. This has its own drawbacks, as the controller's processor may need to be upgraded for adequate performance and the memory requirements may grow substantially, both of which we wanted to avoid in our initial considerations. The benefits with the new approach may outweigh the drawbacks. It allows easier customization and adjustments to the home controller's main logic and allows more flexibility from the devices and appliances. New program logic can be transferred to the controller as a supplement to assist interaction with the controller. Resource management can be made more comprehensive in the future. There are many other research areas still available to improve the controller, such as the multi-touch interfaces and human-computer interaction studies in general. The interface is very important but difficult to develop because of the differences in the people that interact with the interface.

Products will remain a niche market for many years. Although appliances themselves are getting more complex, with more electronic control. Since these appliances may already contain adequate computing power, smart home appliances may become a reality sooner rather than later. Unfortunately, building smart home appliances largely depends on the device and appliance manufacturers' whims and marketing forces. Fortunately, home builders are recognizing the added value and need for providing basic infrastructure to be installed during the home building process. More new housing is being built with whole-house networking installed, and often at least provisions for home theatre and network cabling.

Appendix A

Smart Home Embedded Computer SoPC Block Diagram


standard_32 v3.0 -- Stratix 1S40
This design is an excellent starting-point for Nios systems built on the Stratix 1S40 Development Board. It includes connections to every built-in component on the board.
Targeted for Nios kit Stratix 1S40 development board.
SDRAM PLL
sdram_pll
delay_reset_block
Double-click symbol to edit system with SOPC-Builder
ref_32_system
IDE DMA is unused, cable-select grounded.
E-net used in asynch mode.
Option Value
Location Pin_K17
Location Pin_AC9
Location Pin_E15
Location Pin_M8
Location Pin_K2
Location Pin_V25
Location Pin_V28
Location Pin_U2
Location Pin_Y28

Appendix B

Abbreviations

ACK	- Acknowledge
ALU	- Arithmetic Logic Unit
API	- Application Programming Interface
ARM	- Advanced RISC Machines, creator of the ARM processor
ASIC	- Application Specific IC
CEA	- Consumer Electronics Association
CEBus	- Consumer Electronics Bus
CMC	- Canadian Microelectronics Corporation
CPU	- Central Processing Unit
EIA	- Electronic Industries Association
EIB	- European Installation Bus
FM	- Frequency Modulation
FPGA	- Field Programmable Gate Array
GPIO	- General Purpose Input Output
HCI	- Human Computer Interaction
I/O	- Input / Output
IP	- Internet Protocol, also Intellectual Property ("IP Core")
JTAG	- Joint Test Action Group
LAN	- Local Area Network
LCD	- Liquid Crystal Display
LED	- Light Emitting Diode
NACK	- Negative Acknowledge

PC	- Personal Computer
PCB	- Printed Circuit Board
PDA	- Personal Digital Assistant
PHY	- Physical Interface layer of OSI (Open System Interconnection) reference network model
RAM	- Random Access Memory
RF	- Radio Frequency
RISC	- Reduced Instruction Set Computer
RTOS	- Realtime Operating System
SHCS	- Smart Home Control System
SOA	- Service Oriented Architecture
SoPC	- System on Programmable Chip
SoC	- System on Chip
SRAM	- Static RAM (Random Access Memory)
TCP	- Transmission Control Protocol
TCP/IP	- Transmission Control Protocol over Internet Protocol
UART	- Universal Asynchronous Receiver Transmitter
UI	- User Interface
USB	- Universal Serial Bus
VHDL	- VHSIC (Very High Speed Integrated Circuit) Hardware Description Language
X-10	- Legacy home control standard using power-line signalling

Bibliography

[1] ARM, Inc. "ARM Milestones," http://arm.com/aboutarm/milestones.html.

[2] B. Lewis, I. Bolsens, R. Lauwereins, C. Wheddon, B. Gupta, and T. Tanurhan, "Reconfigurable SoC - What Will it Look Like?" *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 660-662, 2002.

[3] C. Berthet, "Going Mobile: The Next Horizon for Multi-million Gate Designs in the Semi-Conductor Industry," *Proceedings of the 39th conference on Design Automation*, pp. 375-378, 2002.

[4] K. Wacks, "The Success and Failures of Standardization in Home Systems," *2nd IEEE Conference on Standardization and Innovation in Information Technology*, pp. 77-88, 2001.

[5] W. Greene, D. Gyi, R. Kalawsky, and D. Atkins, "Capturing user requirements for an integrated home environment," *Proceedings of the third Nordic conference on Human-computer Interaction*, vol. 82, pp. 255-258, Oct. 2004.

[6] X10.com. "Interface Communication Protocol," Aug. 2001, ftp://ftp.x10.com/pub/manuals/cm11a_protocol.txt.

[7] S.W. Kim, S.H. Park, J.B. Lee, Y.K. Jin, H. Park, A. Chung, S.E. Choi, and W.S. Choi, "Sensible appliances: applying context-awareness to appliance design," *Personal and Ubiquitous Computing*, vol. 8, pp. 184-191, 2004.

[8] S.H. Park, S.H. Won, J.B. Lee, and S.W. Kim, "Smart Home - digitally engineered domestic life," *Personal and Ubiquitous Computing*, vol. 7, pp. 189-196, 2003.

[9] K. Hara, T. Omori, and R. Ueno, "Detection of unusual human behavior in intelligent house," *Proceedings of the 2002 12th IEEE Workshop on Neural Networks for Signal Processing*, pp. 697-706, Sep. 2002.

[10] D. Spinellis, "The information furnace: consolidated home control," *Personal and Ubiquitous Computing*, vol. 7, pp. 53-69, 2003.

[11] A. Yates, O. Etzioni, and D. Weld, "A Reliable Natural Language Interface to Household Appliances," *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pp. 189-196, Jan. 2003.

[12] M. Nakamura, "Implementing Integrated Services of Networking Home Appliance Using Service Oriented Architecture," *Proceedings of the 2nd International conference on Service oriented computing*, pp. 269-278, 2004.

[13] P. Darbee, "Insteon The Details," *Smartlabs Inc.*, Irvine CA. 2005.

[14] CEBus Industry Council. "CEBus Specification," http://www.cebus.org/.

[15] EIA Consumer Electronics Group. "EIA-600.10 Introduction to the CEBus Standard," 1995.

[16] K. Davidson, "CEBus Update - How is the health of EIA's Baby?" *Circuit Cellar*, No. 15. pp. S2-S10, Jun./Jul. 1990.

[17] Echelon Corporation. http://www.echelon.com/.

[18] Siemens. "DELTA switches and sockets," http://www.automation.siemens.com/et/delta/index_76.htm.

[19] Siemens. "GAMMA building management system," http://www.automation.siemens.com/et/gamma/index_76.htm.

[20] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper, "Implementing a Sentient Computing System," *Computer*, vol. 34, No. 8, pp. 50-56, 2001.

[21] T. Koskela, and K. Väänänen-Vaainio-Mattila, "Evolution towards smart home environments: empirical evaluation of three user interfaces," *Personal and Ubiquitous Computing*, vol. 8, pp. 234-240, 2004.

[22] M. Kolberg, E. Magill, M. Wilson, P. Burtwistle, and O. Ohlstenius, "Controlling Appliances with Pen and Paper," *Proceedings of IEEE Consumer Communications and Networking Conference*, Las Vegas, Jan. 2005.

[23] A. Venkatesh, "Computers and Other Interactive Technologies for the Home," *Communications of the ACM*, vol. 39, No. 12, pp. 47-54, 1996.

[24] J.Y. Han, "Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection," *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, pp. 115-118, Oct. 2005.

[25] Altera Corporation, "Avalon Bus Specification Reference Manual," Jul. 2003.

[26] Altera Corporation, "Nios 3.0 CPU," Mar. 2003.

[27] M. Abramovici, C. Stroud, and M. Emmert, "Using Embedded FPGAs for SoC Yield Improvement," *Proceedings of the 39th conference on Design Automation*, pp. 713-724, 2002.

[28] S. Bhattecharya, "Intelligent monitoring systems: smart room for patient's suffering from somnambulism," *2nd Annual International IEEE-EMB Special Topic Conference on Microtechnologies in Medicine & Biology*, pp. 326-331, May 2002.

[29] N. Kohtake, J. Rekimoto, and Y. Anzai, "InfoPoint: A Device that Provides a Uniform User Interface to Allow Appliances to Work Together over a Network," *Personal and Ubiquitous Computing*, vol. 5, No. 4, pp. 264-274, 2001.

[30] B. Levesque, M. Lavoie, and J. Joly, "Residential water heater temperature: 49 or 60 degrees Celsius?" *The Canadian Journal of Infectious Diseases & Medical Microbiology*, vol. 15, No. 1, Jan./Feb. 2004.

[31] M. Rahman, C. Akinlar, and I. Kamel, "On Secured End-to-End Appliance Control Using SIP," *Proceedings of the 5th International Workshop on Networked Appliances*, Liverpool, pp. 24-28, 2002.

[32] J.A. Rode, E.F. Toye, and A.F. Blackwell, "The fuzzy felt ethnography–understanding the programming patterns of domestic appliances," *Personal and Ubiquitous Computing*, vol. 8, pp. 161-176, 2004.

[33] H. Takada, and K. Sakamura, "Compact, low-cost, but real-time distributed computing for computer augmented environments," *Proceedings of the Fifth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, pp. 56-63, Aug. 1995.

[34] Veris Industries. http://www.veris.com/.