Proof of Concept: Home Automation Solution with Potential for Seamless Integration and Vast Expansion

by

Guy Sawyer

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Electronic Engineering in the Faculty of Engineering at Stellenbosch University

Department of Electrical and Electronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Supervisor: Dr. M.J. Booysen

December 2014

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: 04 November 2014 ....

Abstract

Proof of Concept: Home Automation Solution with Potential for Seamless Integration and Vast Expansion

G. Sawyer

Department of Electrical and Electronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MScEng (Electronic)

December 2014

The ever-increasing existence of electronic systems and devices within the residential environment, along with the human desire to simplify life and daily routine, is generating increased interest in the field of home automation (HA) and intelligent environments. A large variety of home automation solutions have been conceptualised or developed. However, many of these solutions are designed by experts and require professionals to install and/or operate them. Furthermore they lack the potential for seamless integration into an already functioning home environment. To bridge the gap between consumer and expert as well as allowing for integration into any existing home environment without physical alteration to the building, a modular home automation solution with seamless integration potential is proposed.

The implemented system uses open source software and hardware allowing for development to continue within the existing, large open source community. It can be installed and configured without professional skills or physical alteration of the environment itself and due to its modular design, the system also allows users to add and remove functional components to and from the system providing them with a seamlessly customisable home automation solution.

Conceptual design and practical implementations are covered in this document, along with recommendations for both continued research and potential avenues for expansion.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations:

- To my supervisor, **Dr. M.J. Booysen** for his expertise and guidance throughout the course of this project.
- To **MTN** for providing the funding required to make this possible.
- To everyone in the **DSP Lab** during 2013 and 2014 for creating a balanced working environment and providing the motivation to go on.
- To my parents, **David** and **Antoinette** and my brother **Matthew** for your love and support and the knowledge that you will always be there for me.
- To my **friends**, near and far - I am who I am because of you guys. To Christian, for being there through it all.
- To my dear **Ania** for your absolute love and support. Thank you for staying by my side the whole way.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

HA	Home Automation
$HVAC$	Heating, Ventilation and Air Conditioning
$M2M$	Machine to Machine
$WIMP$	Windows, Icons, Menus, Point-and-Click
$HTML$	HyperText Markup Language
$GPRS$	General Packet Radio Service
PDA	Personal Digital Assistant
ANN	Artificial Neural Network
BBB	Beaglebone Black
I/O	Input / Output
CPU	Central Processing Unit
URS	User Requirement Specification
SRS	System Requirement Specification
ADC	Analog to Digital Converter
PWM	Pulse Width Modulation
GPS	Global Satellite Positioning
PIR	Passive Infrared
$Wifi$	Wireless Fidelity
TCP	Transmission Control Protocol
IP	Internet Protocol
$DHCP$	Dynamic Host Configuration Protocol
$GPIO$	General Purpose Input Output
GUI	Graphical User Interface
API	Application Program Interface
AC	Alternating Current

Variables

V_{Supply}	Supply voltage	[V]

V_{AINmax}	Maximum analog input voltage	[V]
V_{out}	Output voltage	[V]
I_P	Primary current	[A]
I_{PN}	Primary nominal current	[A]
R_1	Voltage divider resistor 1	[Ω]
R_2	Voltage divider resistor 2	[Ω]
i_{in}	Input current	[A]

Symbols

C	Degrees Celsius
V	Volts
A	Amperes
Ω	Ohms

Subscripts

a	Adiabatic
a	Coordinate

Chapter 1

Introduction

1.1 Home Automation

Home Automation (HA) is the residential extension of building automation. It is automation of the home, housework or household and may include centralised control of lighting, heating, ventilation and air conditioning (HVAC), appliances, security locks, and multiple other systems found in the home environment. Interest in the field of HA as increased greatly in recent years due to advances in M2M communications networks and a much higher affordability and simplicity through the emergence of smartphones and tablets.

A HA system, by definition, is the integration of electrical devices (including actuation of non-electrical devices such as blinds) and the monitoring and control of the home environment through an intelligent system, or by a user either locally or remotely, over a communication medium, such as Bluetooth or WiFi, and coordinated by a computer. Within this connected environment, tasks can be made more intuitive and effective. The home network handles all communication and a user is given access to the system via a personal computer, smartphone or tablet from either within the home or from a remote location over the Internet. The integration of information technologies and the home environment allows systems and appliances to function with convenience, energy efficiency and safety benefits.

HA is applicable to more than just the in-home environment. An HA system also integrates the home environment with service providers in industry. HA allows energy generators to monitor power consumption with much greater accuracy and are therefor able to provide energy more efficiently and cost effectively. In the not-so-distant future, produce retailers may be able to monitor the contents of a fridge and despatch deliveries without the habitants being aware of a shortage.

The overall HA architecture therefore consists of the devices and appliances within the home connected to the home network, the user and the industry service providers. Fig. 1.1 explains this concept visually.

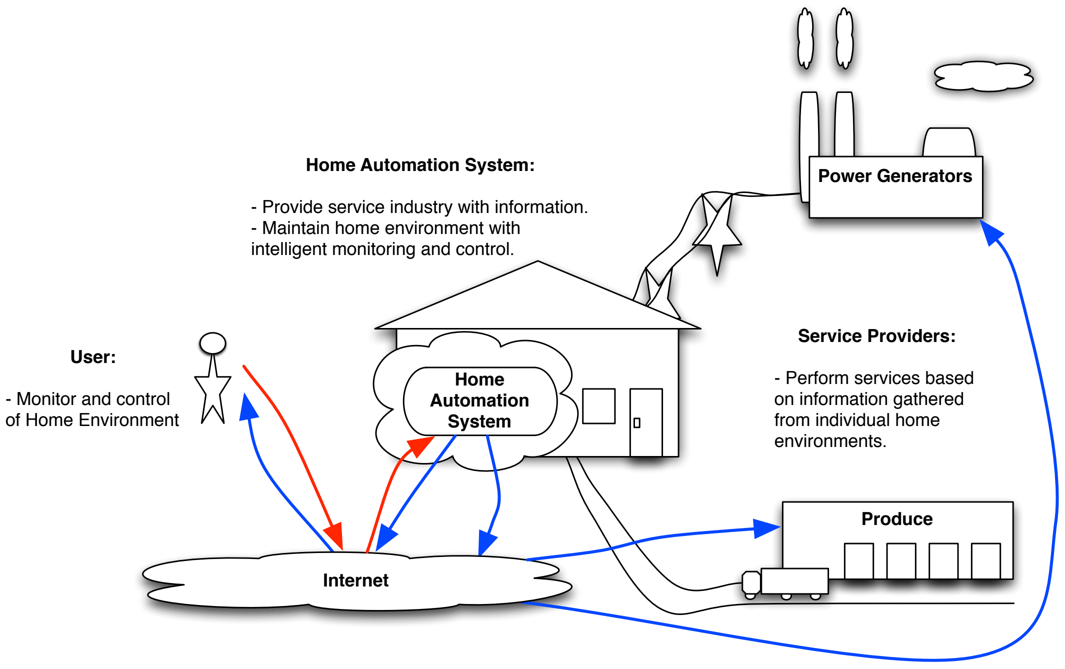

Power Generators
Home Automation System:
- Provide service industry with information.
- Maintain home environment with intelligent monitoring and control.
User:
- Monitor and control of Home Environment
Home Automation System
Service Providers:
- Perform services based on information gathered from individual home environments.
Produce
Internet

Figure 1.1: Conceptual HA Architecture

It should be noted that in reality not only power generation and produce sectors of the service industry would be connected. HA is also an advantage to courier services, domestic services, marketing companies and many other service or retail related businesses.

1.2 Research Problem

Automated homes have made common appearances in technology expositions and science fiction films. However complexity, retailer competition and the use of multiple incompatible standards as well as the resulting expense are all factors that have limited the penetration of HA systems to the wealthy and hobbyist consumers.

Existing HA systems are uniquely designed to suit specific consumers home environments and requirements. These systems are complex in design and are often embedded into the physical structures of the home. It is normal that most homes are in a state of constant change. Appliances and entertainment systems are swopped out for newer, better models as they become available. Living patterns change and the requirements of the consumer inevitably shift. This places the burden on the consumer to be satisfied with a system that satisfies outdated requirements and that cannot be reconfigured, expanded or maintained without incurring the high cost of a professional [1, 2]. The impact of smartphones and tablets has also made people more capable of managing technical tasks and they will want this to be the case with their HA systems.

In the past decade it has been a race to develop HA products capable of all sorts of functions. This competition has lead to the existence of many HA projects that run on incompatible communication and hardware standards. The definition of HA mentioned earlier states that it integrates electrical devices with each other. Without employing one standard for communication and hardware integration, it cannot be said that a particular system fits the definition of HA.

Adding to the problems with HA, is the fact that existing solutions are not open to third party development. With such a large variety of home environments, living patterns and human requirements, HA is a field that requires the input of the greater community. A small group of developers will not be capable of developing a solution that is applicable to every home, and every person, and at the same time is capable of reconfigurability and expansion without high cost. With the use of open source hardware and software packages, a HA solution is open to third party development allowing the technology to progress and expand as homes and people change over time.

After an extensive study of the existing research in HA and the existing projects in the field, a set of challenges facing HA were identified in Chapter 2. This set of challenges is the the focus of this section. To summarise - HA is currently faced with the following problems:

- Existing HA systems are **complicated and expensive** which prevents adoption by the average consumer and limits the potential for reconfiguration and expansion in order to satisfy the changing needs of the consumer.

- Developer competition has lead to the **lack of a standardisation** of hardware and software platforms used in HA preventing the realisation of a HA system capable of integrating with all home devices.

- The exclusion of open source technology and the current, competitor based, methodology that creates closed box products prevents the **growth and expansion** of HA that would otherwise be possible with the input of a larger developer community and an open ended design philosophy.

- **Intelligence** within HA systems has normally been concentrated in the central controller of the system. Distributing the intelligence among the slave and server side nodes opens up a lot of potential in so far as device integration and intuitive user interfaces go.

1.3 Proposed Solution

To solve the problems discussed earlier, a seamlessly integratable, open source, reconfigurable and expandable HA proof-of-concept is proposed. To serve as a platform upon which research can continue it should make use of open source hardware and software. It should also allow for multiple configurations of sensors and actuators and be easy to modify resulting in a modular design. This will allow the third party community to reconfigure the system such that it suits the requirements of each unique household.

From the concepts considered, (as discussed in Chapter 4) it was decided that the system would run on the Beaglebone Black development board [3] design by CircuitCo [4] using the Steward: a management software package developed by The Thing System [5]. The Beaglebone Black (BBB) is an open source development board with a mid-range processor, a variety of I/O capabilities, 3D graphics rendering capabilities and USB slave and host connections. Many different operating systems can be loaded onto the BBB, including Linux and Android (both of which are open source operating systems). The Steward is a HA development platform designed for the purpose of integrating "things" within the home into an HA system. The selection of both the BBB and the Steward are discussed in greater detail in Chapter 4.

The rest of the system is designed around the BBB and the Steward making use of a modular, slave/master system design. The master and all slave nodes are connected to and communicate via a home WiFi router (common in most homes). A user gains access to the system either locally over WiFi, or remotely over the Internet. Certain environmental sensors and electronic actuators are included with each slave in this proof-of-concept. The user interface is developed as a locally hosted web server allowing the system to be controlled from almost any device that is equipped with a browser. A simple diagrammatical representation of the system architecture can be seen in Fig. 1.2. The red arrows represent user side communication that is only transmitted between the master node and the user device, whereas the blue arrows represent slave side communication that is only transmitted between the master node and slave nodes. A detailed discussion of the hardware evaluations and choices that resulted in the aforementioned system is given in Chapter 4.

1.3.1 User Requirements Specification

The User Requirements Specification (URS) is required to develop the System Requirements Specification (SRS). They identify the top level, or user level, requirements of a project that would later be translated into system requirements (necessary for the conceptual design phase). These user requirements are formulated in order to address the challenges facing HA, that are identified

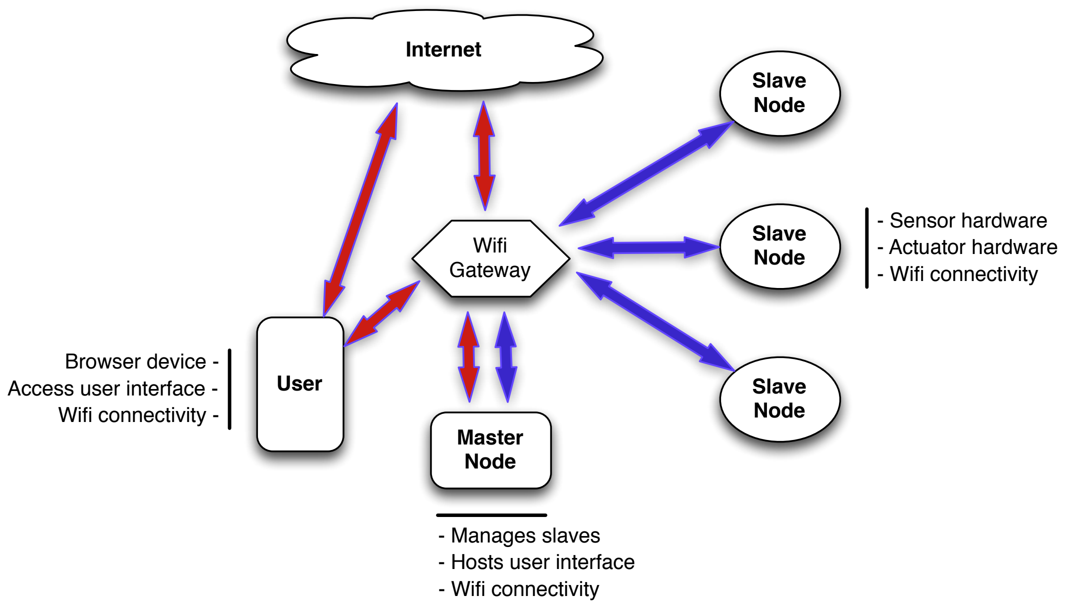


Figure 1.2: Diagram depicting entire system layout.

in Chapter 2. They are also intended to assist in producing a HA system that serves as a successful proof-of-concept.

The user requirements are therefore, to create a HA proof-of-concept:

1. that adheres to the given definition of a HA system (**URS1**).
2. with the potential to integrate with any home electrical device (**URS2**).
3. that can be installed into the home without having to break boundaries such as walls or ceilings (**URS3**).
4. that is simple and intuitive to install and operate (**URS4**).
5. that is open to development by a third party community to promote expansion of the platform (**URS5**).
6. that incorporates suitable sensors and actuators capable of demonstrating a successful proof-of-concept (**URS6**).
7. that implements a modular master/slave design to allow for seamless expansion and reconfiguration by the user (**URS7**).
8. with a simple and cost effective user interface that reaches the majority of the consumers (**URS8**).
9. with mid-range processing power to allow for integration with devices and sensors that may require labour intensive work (**URS9**).

10. that distributes system intelligence across the master and slave nodes as well as the user interface (**URS10**).

1.3.2 Contribution

The proposed solution is a proof-of-concept HA system with the potential to integrate all existing household devices into a smart system with large expansion potential. The design is uniquely based on open source hardware and software aiming future development at the open source community. This aids in preventing restrictions on the expansion of the concept, as well as enhancing the potential for growth and integration. This concept is also unique in it's distribution of intelligence and also the modular architecture it employs. The resulting contribution is a proof-of-concept with vast expansion and integration potential.

1.4 Document Layout

This document begins with a thorough study of the existing research literature on HA in Chapter 2. The challenges facing HA identified in the literature are directly related to the objectives and user requirements defined in this chapter, and from this, the system requirements specification is drawn up in Chapter 3. Thereafter a conceptual design is discussed in Chapter 4. In this chapter a conceptualised system is formulated that is based on the URS and SRS, hardware and software components are compared, and a discussion on the selected components is presented. The design of the HA proof-of-concept, using the components selected in the conceptual design, is then discussed in Chapter 5. This document concludes with an analysis on the performance of the system, a discussion of the achievements, proposal of future work and the final conclusion in Chapter 6.

Chapter 2

Literature on Home Automation

Home Automation is the integration of electrical and electronic devices within the residential household, via a centralised control system, which provides the habitant with control over the devices, either locally or remotely, as well as the ability to automate certain processes or functions within the home [6, 7, 8]. Home automation has received a lot of attention recently due to the ever increasing number of electronic and electrical devices within the home. The ability to host communications between a number of devices and a centralised intelligent control system, opens the door to a lot of potential for growth [9]. An area of HA which has received significant interest recently, has been that of augmenting HA networks with a multitude of sensors. An environment which hosts the integration of devices and sensors in an interconnected network with an intelligent control system, is known as an intelligent environment. An intelligent environment is characterised by having computing technology embedded within the environment, in such a way that it becomes virtually invisible [10]. With the addition of communications and an awareness of a humans presence and location, users could be empowered by a digital environment that is sensitive, responsive and adaptive to their behaviour and needs [11, 12].

Research in home automation can be categorised into five areas, namely in-home health monitoring, safety and security, smart grid and power consumption automation, device function automation, and adaptive control system software. In-home health monitoring is focused on caring for the frail who wish to live independently in a familiar environment [13, 14, 15]. Safety and security applications involve access control, detection of hazardous situations, and automated emergency services notification [16, 1, 6]. Smart grid and power management systems will monitor power consumption and enable the user to control electricity consumption [17, 18, 19]. Device function automation systems perform automated control of various devices based on a set of user defined rules [8, 7, 20]. Such devices include fans, ovens, smart television sets, ice machines etc. Lastly adaptive control systems monitor user behaviour, location and actions and then use this information to optimise its rule base [12, 10, 16, 7]. There are a few driving forces behind the development of home automation. In health care, the ability to monitor patients for 24 hours a day from within their own home, has obvious benefits to quality of life for the patient, and also the cost of caring for them [21]. It also provides the ability to set patient specific care plans and medical insurance as well as benefiting researchers by providing large amounts of medical data [15]. In the field of power management, HA can reduce costs in home power consumption, make personalised billing schemes possible, and also give producers the ability to predictively generate power.

2.1 General Home Automation Architecture and Application Areas

Home automation has attracted a lot of attention, over the past decade, due to the rapid increase in electronically controlled devices and the emergence of wireless machine (M2M) communications research. Research on home automation can be categorised broadly into five research areas. These areas are:

- M2M communications.
- Intelligent control systems.
- User interface design.
- Localised positioning.
- Hardware support.

Fig. 2.1 depicts the generic architecture of four separate home automation networks, i.e. four physically separated homes. Two network topologies can exist within the home network, namely: fully distributed and co-operative. In a fully distributed network, all nodes are connected as peers and can share information amongst themselves. Normally, one of the nodes acts as a super-peer that has the ability to connect through some gateway to the Internet, however in the case of HA, variability and adaptability are important requirements. Therefore all nodes are also connected directly to the intelligent controller and are capable of independently accessing information over the Internet. A Cooperative network is not, strictly speaking, a sub-network. Network nodes communicate with each other via a home controller, and no direct communication takes place.

The typical HA network is comprised of appliance devices, sensors, smart grid components, healthcare devices and security systems. Intelligent controllers manage the home networks' communication between nodes within the system, and between the system and the outside world via the Internet. They

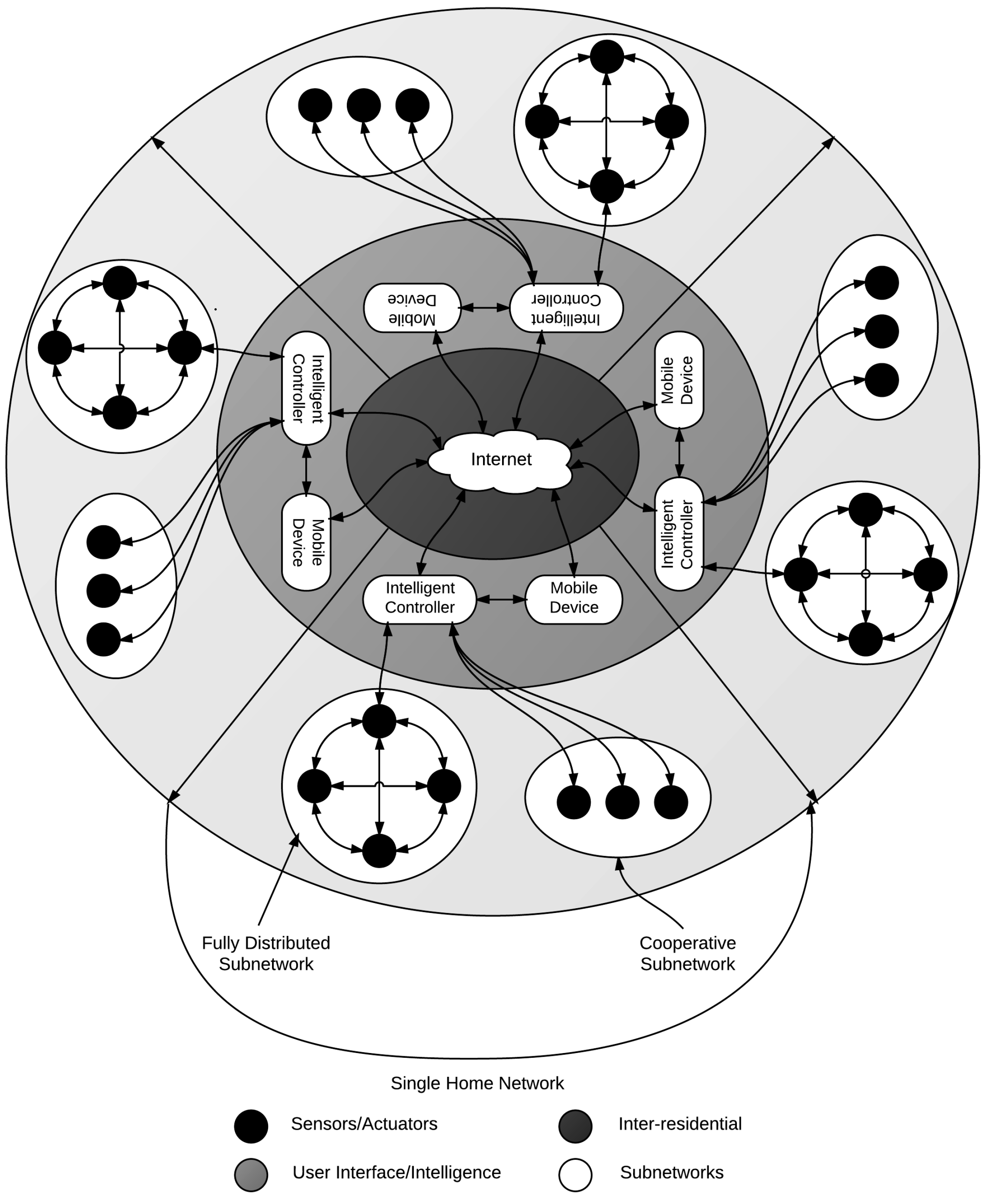


Figure 2.1: Generic Home Automation Architecture

also provide a user interface for the system that is controlled either directly or via a mobile device.

The goal of a home automation network is to improve quality of life of the user by performing tasks based on a set of rules, developed using information from a multitude of data collectors. These tasks can be anything from toggling lights to preparing the home for a long holiday when the inhabitants will be absent. The rules are defined by the user of the system and can be manipulated thereafter by either the user or by the system's built-in adaptive intelligence. The latter would base its adaption process on the user's behaviour. For instance, if the user switched a light on when the system assumed it should be off, the rules would be adjusted to account for this. Data is collected by the system using sensors placed in and around the home environment and by monitoring of the users behaviour or from other sources through an internet connection. This information would facilitate in the decision making and rule manipulation processes.

Fig. 2.2 shows how appliances and sensors could be interconnected using a number of communication standards in order to achieve the goal of HA. Below, we discuss how home automation is applicable to the selected areas shown, and also briefly describe how home automation incorporates intelligent environments and localised positioning to improve its function.

2.1.1 Healthcare

Health care home automation systems have received a lot of attention recently, due to the numerous benefits it introduces. As the global population grows and professional treatment becomes scarce, the ability to autonomously monitor patients has become more important. Many patients also wish to live in familiar environments and home automation makes this possible. In-home health care can be implemented in a number of ways. [13] and [15] present a system incorporating body sensors, acoustic sensors, and infrared location sensors to determine the state of health of an inhabitant in the environment. Rules embedded in the control system flag unusual events (such as high accelerometer readings during a fall), indicating a problem with the patient, and also monitor temperature, heart rate, and other vital signs for early warning. [21] and [14] present a similar system, with the exclusion of body sensors, that aims to detect safety and health critical situations. The system monitors certain appliances within the home to track occupant actions. The system raises a warning when it detects a break in routine. Body sensors are intentionally excluded due to unreliability, since a person must remember to wear it. Battery life and cost of such devices are further considerations motivating the design. Home automation in health care allows for patients with long term conditions or terminal illnesses to be monitored from their home. Patients and doctors can be informed when health risks are detected, and doctors have the ability

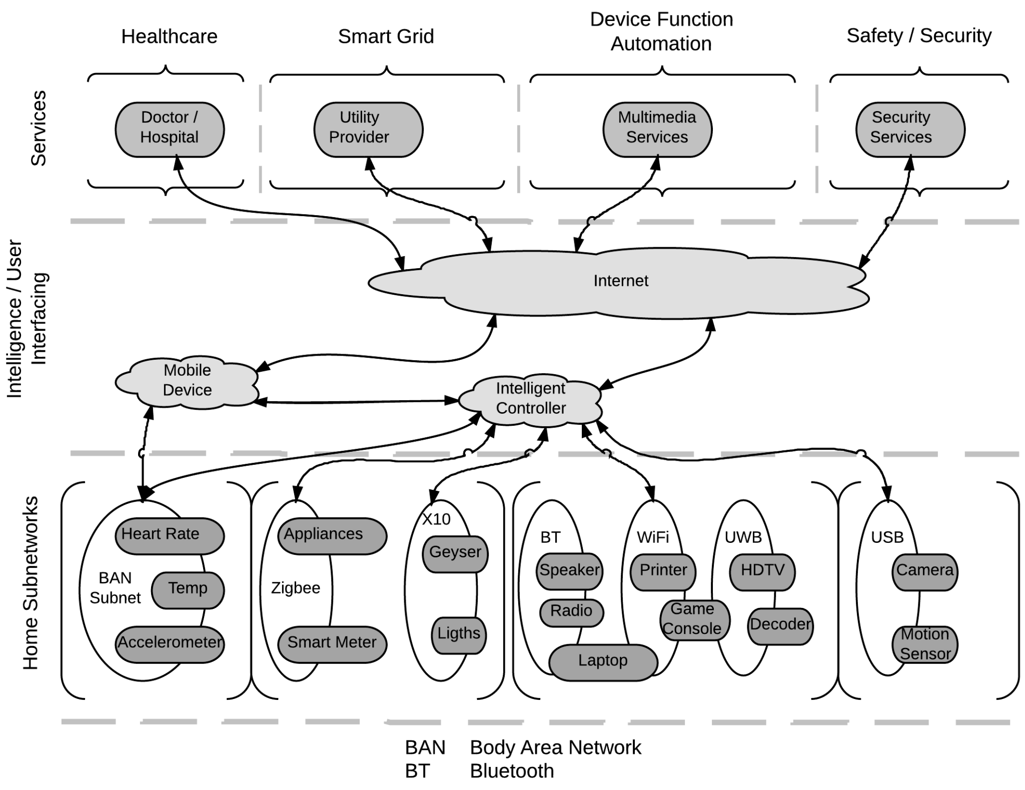


Figure 2.2: Home Automation Applied in Healthcare, Smart Grid, Device Automation and Safety and Security

to care for a larger group of patients, since physical space is less of a concern.

2.1.2 Smart Grid

A smart grid is a combination of electrical and information networks which provide utility generators and users with intelligent energy solutions [22]. The main objectives of smart grids are to increase the efficiency of power transmission, increase the quality of service to utility users and to reduce the economic and environmental cost of power generation and consumption. There are two sides to efficient operation of a smart grid: It is the responsibility of the utility generators, on the one side, to provide energy to those who need it when they need it. In order to do this in the most efficient way, the generator needs to know certain details about the consumers. This is where the other side of the smart grid comes in and home automation proves beneficial. With a control and monitoring system built into the home electrical network the user has the ability to monitor consumption of each device individually and also limit or prevent use of one or more devices. To truly realise the smart grid, utility generators would be given access to this network in order to monitor usage.

This would provide them with a detailed, house by house breakdown of energy consumption giving them the power to predictively produce energy when it is required, and save energy when it is not [19]. It is then also possible for utility generators to provide users with real time billing scales and even personalised billing schemes that are dependent on their behaviour.

2.1.3 Device Automation

One of the more obvious applications of Home Automation is that of devices and functions within the home. There are many advantages to lights that switch on and off as you enter or exit a room, air conditioning that manages and optimises its operation and the ability to operate devices and monitor your home's condition remotely over the Internet [23]. Although there is no industry standard system that can be bought to implement this concept easily, there is a lot of research into achieving such a system [24, 2, 9, 23]. One of the largest challenges facing standardisation in home automation is the communication protocol. Many concepts have focused on developing software capable of handling the broad variety of data when dealing with home automation networks [2, 20]. Only a few have considered using existing communication technologies and implementing hardware to handle device specific communication and data [23, 25].

Wide-spread adoption of home automation is also a challenge to progress. Manufacturers and consumers are reluctant to get into the market without there being a simple-to-use and effective solution to home automation. For this to be possible, certain design considerations must be met [8]. A viable Home Automation system must be scalable. Users will always want to see progress in their home, therefore they must be able to add or remove devices from the system with ease. Such a system must also be reconfigurable. Two different users will not necessarily require the same operation from a particular device. They must therefore have the option to configure the control of their device in their own way. Lastly, the system must be Internet based. This is necessary to ensure expansion and growth of the home automation system.

2.1.4 Safety and Security

Safety and security is an important consideration in the eyes of the consumer. Not only the network security of the system, but more importantly the level of security the system can provide for the home and its occupants. Existing security systems only provide simple functionality, such as, electric fences and entry point break beams, which results in a lot more potential for application of an a HA system. When it comes to security, there are many methods available to deter and prevent burglars and home intrusions.

Unfortunately, there is no guarantee that protective services will arrive in time or with the appropriate resources to deal with the problem. Home automation systems can be of great benefit to the safety of people and the effectiveness of protective services. HA systems can monitor the boundaries of the house and impose a lock down of sorts in the event of an intrusion. The home automation network can notify authorities when an intrusion occurs, and it can also provide them with detailed information on the situation (such as number of intruders, position of occupants/intruders within the home and even weapons in use).

Safety within the home is also an important consideration and home automation systems can be effective in preventing unexpected accidents. Certain hazardous situations are easy to detect, but within the home environment people are not always conscious of potential dangers [16]. One such example is a gas leak. Unless the occupant is aware of the leak itself, he won't be concerned with the rising gas level in the house. An HA system could monitor gas level's and provide the user with a warning when levels are high. Furthermore it could prevent use of the gas stove and fireplace should the users be unaware of the warning.

2.1.5 Intelligent Environments

An intelligent environment is defined as one having computing technology embedded within the environment in such a way that it becomes virtually invisible. Through the use of M2M communications and intelligent control systems, as well as an awareness of a human's presence and location, the aim is to empower people with a digital environment that is sensitive, adaptive and responsive to their behaviour and needs. Most modern day automation systems are designed to carry out a predetermined process [6, 25], triggered by an event; such as time, user action, sensor reading or the result of a previous process (by process we mean actuation of a device in the home such as switching lights or changing the temperature of the air conditioner). These processes can be complicated with rules and dependencies on a number of variables but once set, the rule remains unchanged until the user physically changes it. The efficiency and performance of these types of systems can be improved by using fuzzy logic control concepts, but adaptability is still required.

The aim of developing an intelligent environment is to eliminate user interference as much as possible and increase the benefits to the user living within the environment. Once a rule has been entered by the user, it is expected that the system will monitor future behaviour of the user and optimise or modify this rule, and create more rules in accordance with their reactions. Also, the system is capable of accessing information from multiple sources, including the Internet, in order to adapt real time decisions [26].

2.1.6 Localised Positioning

In order to realise the full potential of intelligent environments, along with the capabilities already mentioned, the system must be able to detect occupancy and position of users within the environment. All rules within the system are dependent on input variables. The addition of the position of the user is an important tool in the adaption of the rule base. Whether or not the user is in a certain position when they carry out a specific action can have a large effect on how their behaviour is interpreted. Up till recently, existing localisation techniques have not been able to provide the accuracy required in HA. The authors in [27] investigate a localisation technique passive RFID and trilateration. This method has proven to provide better accuracy at a lower cost than most of the solutions that currently exist. The authors in [28] investigated a system using ZigBee wireless communication and also achieved accuracies greater than existing systems. Both of these systems use signal strength between stationary nodes, and a node on the occupant to find his position, but with different complex algorithms to calculate the result.

2.2 Research to Support Development in the Field of Home Automation

The preceding sections demonstrate that home automation has a wide variety of application areas covering a broad range of research fields. While a lot of time is spent researching and developing these areas individually, there is not a lot of focus on the home automation system that incorporates everything. It must be understood that in order to develop a fully capable home automation system, there must be concern for all research fields as well as the seamless integration thereof.

The system must integrate with all types of hardware within the home, and allow for addition and removal of devices to and from the network. That is to include devices with logic and communications, such as a smart television, and also devices without, like a toaster or lights. It must be designed with a user friendly interface allowing one to exploit its full potential, and the software must establish a standard for communications as well as adopt a high level of security to protect against network related threats. Lastly the control software must include logic to enforce safety and security within the home and also to lower the requirement for human intervention as much as possible.

In Table 2.1 we list the goals of recent research efforts to support development in the related fields of home automation. We go on to discuss the related fields in detail.

Goal	**Ref**
Doctor et al. applied ubiquitous computing intelligence to an inhabited environment. The goal of the research is to realise the vision of ambient intelligence.	[10]
Kastner et al. discusses using artificial intelligence in smart homes. The goal is to optimise energy efficiency and user comfort at the same time.	[29]
Al-Ali et al. developed a Java based home device automation system, using an embedded system board, capable of providing local and remote access to appliances.	[1]
Alkar et al. investigated a low cost, flexible and internet based solution to home device automation capable of integrating with most appliances.	[23]
Bjelica et al. developed a solution to home device automation using hardware that already exists in most homes, a set-top box.	[24]
Borodulkin et al. developed a 3D virtual reality user interface to make home automation as simple, intuitive and attractive as possible.	[30]
Yeoh et al. develop a lightweight home automation system for operation in areas with limited connectivity.	[31]
Spinsante et al. developed a user interface using head tracking for applications of assisted living.	[32]
Piyare et al. proposed a home automation system using bluetooth and an Arduino board to realise a low cost yet flexible and secure system.	[25]
Guizani discusses the challenges facing M2M communications in home networks and develops a standard that satisfies quality of service requirements.	[33]
Niyato discusses the application of M2M communications in the smart grid and proposes a traffic concentration optimisation scheme to reduce the cost thereof.	[22]
Guillemin et al. developed algorithms to identify appliances and their parameters using their consumption signal.	[34]
Rossello-Busquet et al. developed a home energy management system to reduce consumption by implementing rules in an intelligent control system.	[19]
Lee et al. described a system for home safety. The goal was to develop a system that could automatically detect emergency events.	[16]

Table 2.1: Existing Literature in Support of Home Automation

2.2.1 Hardware Interfacing

The most common application of home automation is that of the appliances and devices within the home. The switching of lights based on occupancy or preparing the kitchen for the morning routine are both attractive capabilities of home automation however, no two homes use the same appliances, nor do they use them in the same manner. The challenge is therefore to develop a home automation system capable of integrating seamlessly into any home no matter the appliance type or function.

A simple system was proposed by [25] that allowed the user to control the lights in their home via their mobile phone. The system hardware comprises of an Arduino board with Bluetooth capability, relays connected to the digital outputs for switching, and a Bluetooth capable phone running any Symbian operating system. The hardware is easy to use and implement, and is also readily available.

[23] proposed a slightly more complicated system whereby a PC, a purpose-built master node, and purpose-built slave nodes are used to integrate with appliances within the home. The PC serves as the database, interface and web server. It controls the master node which in turn controls the slave nodes. Each slave node is connected to one or more appliances of similar type and will carry out the control of those appliances based on commands from the master node. Communication between the master and slaves is wireless using radio frequency and the master uses RS232 to communicate with the PC.

[1] described a similar system, except the hardware was embedded within the computer on a PC Card. All appliances are connected to the input/output ports of the embedded PC Card and their statuses are recorded and altered this way. The server is built with Java and the user interface with interactive C. The most significant difference in this system is that all appliances must be hardwired to the PC Card.

A different approach was taken by [24] where no new hardware was developed, except that of an extension for X10 and Zigbee communications, but instead the authors focused on using hardware that already existed within most homes. They developed software that runs on a home's set-top box, a common device found in many homes and typically houses a tuner that connects to a television When the set-top box is connected to the home network, the software automatically detects all appliances and devices on the network and provides the user with control capabilities from the box. Additional hardware is unfortunately required before the software can detect the appliances. To toggle household appliances, the AM12 with impulse relays is required and for lighting, LW12 actuator modules are used. Furthermore, to support X10 communications, the Marmitek CM11 is required.

2.2.2 User Interfacing

Home automation infrastructures are generally quite complex due to the variety of devices, sensors and actuators involved in the network. One of the challenges of designing an interface between the user and the system is doing it in such a way that it is made easy for the user to associate the physical devices within their household to the applicable elements within the user interface.

Another challenge faced by smart user interface design is the limitation on bandwidth in certain countries. Less developed countries may only have GPRS available. The authors of [31] present a user interface system capable of functioning in environments with limited connectivity. The software runs on any mobile device with an email client and communicates with the home controller via email using a set of short hand commands. Due to the small amount of information required to control the HA system, this solution is ideal for situations where only GPRS connectivity is available.

In [32] the authors propose a user interface suitable for disabled or handicapped people. The system uses computer vision to track the head of the user in order to provide control of the system. Typically the user would be able to control lighting, air conditioning, etc., and is most applicable to users who, due to their disability, would find it complicated to complete these tasks. Tracking is achieved using the common OpenCV library for computer vision. Some of the algorithms provided within the library have been adapted to suit this application. A few other libraries are also included to help with identification of objects and their shapes, classification of geometric shapes, and motion detection and object tracking. A camera provides the system with a video feed of the user's head, and simple head movements correlate to certain control commands.

The authors of [30] proposed a virtual reality design where a 3D virtual representation of the home is given with all the sensors, devices and actuators visually represented within the virtual home. This concept aims to move away from normal windows, icons, menu's, point-and-click (WIMP) structure in order to simplify user interaction with the system. Three methods for interaction with the home appliances have been provided. Push buttons are available to toggle the state of appliances, and text commands can also be sent to the server. Lastly the 3D virtual reality method displays the home in 3D with transparent walls. Then by clicking on the screen, a line is projected through the home highlighting all appliances that lie on the line. The user can then make their selection after which possible actions are given. This system can only be operated by direct interaction with the computer.

2.2.3 Finding Communications Standards

The largest challenge facing progress in home automation is the lack of a communication standard [33, 35]. With such a wide variety of devices within the home, producing so many different forms of data, a communication standard that efficiently and effectively hosts the transfer of multiple forms of data between a multitude of various devices has proven to be elusive [36]

The author in [22] discusses the role that smart grids play in the advance of M2M communications, and also discusses a method to optimise traffic concentration so as to reduce the cost of communication. This is done by finding the optimal cluster formation resulting in the lowest cost (clusters being groups of homes with smart meters in a service area). Each cluster is assigned a concentrator which gathers smart meter consumption information and relays it to the control centre. The optimal cluster formation is found using an optimisation algorithm whereby the cluster size is the controlled variable and cost of communication and cost of concentrators are uncontrolled. The cluster size is then continuously reduced while the cost reduces until the optimal cluster size is found.

In [33], the author proposes a home gateway system capable of managing communications between a variety of typical home networks. Some of the common communication standards are Bluetooth and WiFi, but many are included. Instead of focusing on finding one communication protocol that can handle data from any and all devices, the author suggests gateways could be used to connect with all available networks, manage communications between them and ensure quality of service requirements are met. The Home Gateway (HGW) forms the backbone network and is the central machine. It manages all communications within the network and connects the home network to the outside world. Network related functionalities are implemented in the HGW such as application and network management as well as network interconnection (2G/3G, WLAN, WPAN, and wired modes). Access control, security management, QoS management, and multimedia conversion are also functions of the HGW. The rest of the home network is made up of sub-networks, each with a Sub-Gateway (SGW) at its endpoint. The SGW connects the sub-network to the HGW.

2.2.4 Smart Grid Implementation in the Home

Smart grids are a large driving force behind research into home automation. With the increasing awareness on energy, efficiency within the home, and not only the industrial sector, is becoming more and more important [35]. Smart grid technology can only be beneficial when utilised in full scale throughout the home. However, the lack of a standardised home automation system is making this difficult achieve. More importantly, for a smart grid to be successful, it needs to be employed in almost every home. With a widely adopted home automation system already in place, this task is made simple since the hardware and network within the home already exists.

The authors in [34] discuss a system to automatically identify electrical appliances and their parameters by monitoring their consumption signals. Different appliances exhibit unique properties, such as on/off frequency, time, duration, and power usage, allowing automatic identification through power monitoring. If transmitted to a central handler, this knowledge would give utility providers the ability to far better schedule their energy production and distribution. The system comprises of an electronic box connected to outlet power and then to an appliance. The box is coded in java and xml files are used to input and output data.

In [37] the authors present a proof-of-concept system to monitor and control household water heating on a large scale, using a web based interface. The overall system is composed of three sections. The sensing and actuating section include the hot water cylinders within the home as well as the necessary sensors and actuators. The network section includes a cellular modem and service provider network while the visualisation and control section include a remote server, database, and user interface. The system gives a user the ability to monitor and alter the HWC temperature and also safely empty the HWC in the case of failure or maintenance. The pressure of the HWC is also monitored to detect failures.

In [19] the authors describe a more complicated approach where all appliances on the home network are monitored using a home gateway. The home gateway also gives the users the ability to optimise consumption of their appliances by creating and modifying rules for the management system. This approach not only provides identification of the appliances on the network but also implements some intelligence to reduce consumption.

2.2.5 Intelligent Environments

One of the challenges facing home automation is its acceptance among consumers. Intelligent environments is one research field that will help to alleviate this problem by providing the consumers with increased comfort and quality of life [35], creating a desire for the technology. This is achieved with a system that monitors a multitude of inputs from sensors and user actions. The data gathered helps the system to build a profile on the users preferences in order to control their environment such that they don't have to.

The authors in [29] proposed a system that optimises energy efficiency and user comfort at the same time. A use case described by the authors uses the system to predictively manage the heating of a building. To do this accurately, the system requires knowledge of the desired temperature, occupancy schedule and detailed information on the buildings thermal inertia. In most realistic cases, not all of this information is available, therefore the described system makes use of current indoor and outdoor temperature, desired temperature and an occupancy schedule. An ANN is used to constantly evaluate the environment in order to adapt its control scheme. The type of ANN used in this setup is a multi-layer perceptron, or competitive network. It is characterised by a feed forward architecture where information only moves forward from the input nodes to the output nodes. There are no loops within the network. In order to sensibly use an ANN, it must be trained. This is done by applying multiple sets of input and output values that represent the desired relationship between input and output, thereafter adapting link weights and error values. The purpose of the ANN is to convert the limited number of inputs into a time (The time it will take to reach the desired temperature for comfort).

An intelligent environment solution using fuzzy logic controls rather than neural networks is described by [10]. The authors describe a purpose built environment with embedded actuators (computer music player, window blinds, desk lamp, etc.) and sensors (pressure sensors in bed, and desk chair, etc.) capable of monitoring a user's behaviour and learning their preferences. The goal of the system is to perform the routine, repetitive tasks that would instead be performed by the user, such as, closing the blinds and switching off the lights when going to sleep. After associating certain user actions with sensor monitored situations, the system waits until a similar sensor monitored situation arises, and then performs the associated action so that the user does not have to. This is achieved using their adaptive online fuzzy inference system. The system monitors the users interactions and records input/output data associated with their actions. From here it extracts fuzzy membership functions and fuzzy rules from the recorded data, followed by using these rules to control the users environment along with continued user monitoring. Also, mobile control and monitoring of the home network is provided via a mobile device or personal computer.

2.3 Challenges Facing Future Progress of Home Automation

Home automation still faces a number of challenges preventing it from progressing beyond its current state. In this section, we present and discuss some of the challenges that still need to be addressed to enable the widespread standardisation and adoption of home automation systems.

2.3.1 Broad Acceptance

The most significant challenge facing the progression of HA is acquiring a broad acceptance of the idea. In the eyes of the consumer, automating the household is an expensive and unnecessary endeavour. Currently, HA solutions are small and fragmented units with dedicated functions and no possibility for expansion. Also, the HA systems available today are purpose-built to spec for each home and therefore require professional intervention to install and setup. This inflates the cost of Home Automation and also creates the illusion that it is intended as a luxury, ie. used with media and entertainment alone. This illusion along with the high cost of these systems drives consumers away and this has resulted in a lack of positive end-user generated data (or consumer sentiment) to support the financial advantages, and other benefits of Home Automation. Functionality is a priority, which means the automation system must be interconnected with all devices within the home and be capable of a multitude of functions [35]. What the average user must see is a simple and innovative solution to avoid cumbersome tasks within the home, as well as improving energy efficiency, financial advantages and increasing in-home safety.

2.3.2 Cost

With no solid standard for HA, current home automation systems provide limited features at a high cost. Each system must be designed from the start according to the users requirements. Repetition of the design phases as well as the cost of once-off hardware largely increases the over-all cost. In order to reduce the cost of HA, researches need to develop hardware that can be applied in all HA situations. Software can then be implemented that allows users to decide how the system performs in their environment. Standard hardware and software, produced on a large scale, will drastically reduce the cost of implementing Home Automation.

2.3.3 Simplicity

The challenge to develop HA systems that are simple and intuitive has a large impact on the broad acceptance of HA. For Home Automation to progress, large manufacturers need to buy in to the concept. For this to happen, HA needs to be desirable to the average user. Consumers decisions are largely affected by the usability of a system [24]. The challenge is to develop a system that is simple and intuitive within an environment that is highly complex, due to the many different types of devices and information. This means that the hardware must be easy to understand and implement. The user must be capable of reconfiguration of the hardware network without requiring expert help, at added cost. Also, the user interface must be understandable and intuitive. It is the most important aspect of the design and must allow the user to easily configure and control the system. Some of the current user interfaces are innovative but can be better with the implementation of gesture control.

Current HA systems are specifically designed for the users environment, installed and configured by experts, and then left to run without further intervention. Unfortunately, due to the complexity of any home network, configuration of the system is never fully completed, and since the users have limited knowledge of the system, they cannot reconfigure the home without the cost of an expert to assist.

2.3.4 Distributed Intelligence

Home Automation systems require more than just remote control of home appliances, they require intelligence and decision making. The location of the intelligence within the system is an important consideration and poses a challenge to its design. HA systems typically consist of a server, a master node, and slave nodes. Often, the server is hosted online while the master and slave nodes are positioned within the home.

The most obvious solution would be to place all the intelligence (intelligence meaning logic and processing portion of the system) on the master node. The master node then commands the slave nodes, and when a user makes a change to the system, this is done on the web server from which the master node is updated. However, one can foresee the required control of complicated devices on the slave node end. In this case it would be necessary for the slave node to carry a portion of the intelligence to enable communication with these devices. Furthermore, to increase usability of the system it might be beneficial to include some intelligence in the web server. This can be used to improve the interaction experience with the user in ways such as preventing the user from making changes that would logically not make sense.

2.3.5 System Expansion

Technology within the home is an ever changing scenario pressing the need for easy expansion of any HA system. The ease with which a system can be grown and expanded will rest on the coding language of the software, the operating system and the hardware being utilised. This will also define how much room there is for third party development and growth. All of these factors are important challenges to the development of Home Automation.

Some viable operating system solutions are open source software such as Android and Linux. Android allows for very easy update via application addons and is also open to third party development. Linux is more suited to third party development and has a large community supporting the development of projects based on Linux.

Chapter 3

System Requirements Specification

The System Requirements Specification (SRS) is required to translate the top-level objectives into technical system requirements. These are the technical specifications and guidelines to be used during the conceptual and detailed design phases.

3.1 A Home Automation System

The most top level requirement is that this proof-of-concept must be a Home Automation system, **URS1**. It must therefore contain all the components that would make it a Home Automation system, according to the definition given in Chapter 1: integration of electrical devices within the home, intelligent monitoring of the home environment and control of devices within the home, accessible and controllable by a user either locally or remotely, communication over a medium and coordination by an electronic system. Figure 3.1 is a reconstruction of Figure 1.2 which will be used as an aid in outlining the system requirements that will satisfy the definition of a Home Automation system.

3.1.1 Coordination

Coordination is a crucial component of a Home Automation system. Without a coordinating subsystem, the most any HA oriented system could amount to is a complicated universal remote. Referring to number 1 in Figure 3.1, the system must contain a Master Node that is to serve as the coordinating component. The Master Node will be responsible for providing a user with access to and control of the Home Automation system. It will also manage the Slave Nodes: detect when a node is removed (or has gone offline) or when a new node is added, relay information such as sensor status or control commands between the user and the Slave nodes, and maintain control loops (i.e

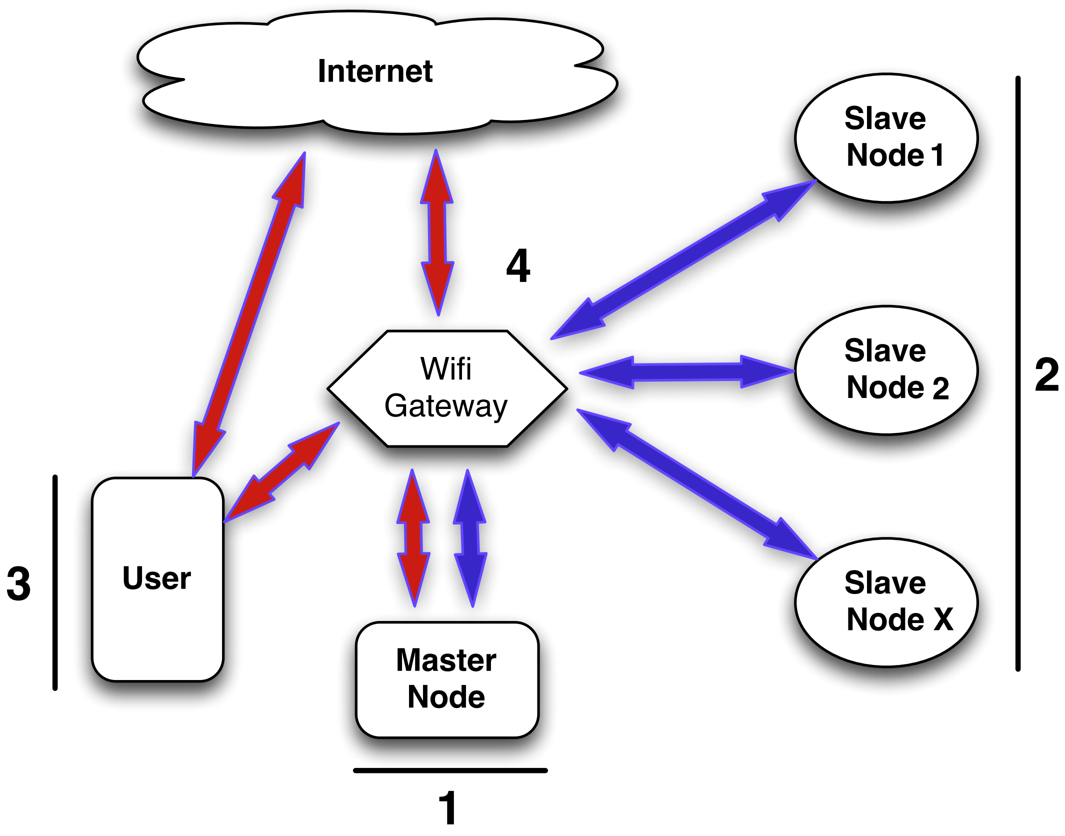


Figure 3.1: Home Automation proof-of-concept system architecture.

the intelligence, which will be discussed later in this manuscript). To satisfy the definition mentioned earlier, this component needs to take the form of an electronic system such as a computer or micro-controller. Potential systems are analysed and one is selected in the conceptual design phase covered in Chapter 4.

3.1.2 Integration

To satisfy the definition of a Home Automation system, it must be able to integrate the electrical devices within the home. In other words, the devices must be linked through the HA in such a way that the actions of one can be dependant on the actions or status of another device, based on information shared over the HA network. Referring to number 2 in Figure 3.1, it can be seen that the network setup resembles that of a star topology, with one important difference. The majority of household electrical devices are dumb devices, i.e. they lack micro controllers and the ability to communicate, and are therefore not connected directly to the network. Instead, an intelligent slave node is used that will integrate dumb devices into the Home Automation network.

As time goes on, the home environment changes and devices are added and removed, only the relevant slave nodes will require upgrading rather than the whole HA system. At the same time the HA system can continue to operate while those upgrades are completed.

3.1.3 User Interface

Without a user interface, a system can still behave like a Home Automation system but not without a cost to user comfort. No system can predict the desires of the user and for this reason, it is required that a user be able to access the system to monitor their home environment and make changes as they wish. Referring to number 3 in Figure 3.1, a user must be able to access the HA system, monitor and control their home devices either locally or remotely over the internet.

3.1.4 Communication

It is required that a medium of communication be employed for a system to be considered a Home Automation system. Referring to number 5 in Figure 3.1, the system must make use of a communication medium to transfer status information from the slave nodes to the user, and control information in the opposite direction. There are a number of suitable communication technologies such as Bluetooth, ZigBee and WiFi. These, and other, communication solutions are evaluated and one is selected in Chapter 4.

3.1.5 Intelligence

One of the main goals of Home Automation is to simplify life and increase comfort experienced by the user within their home environment. Without an intelligent component within the HA system, a user would have to manually monitor and control the home environment as if the HA system never existed in the first place. It is the duty of the intelligence to remove the requirement for human intervention with simple, closed loop control processes. This Home Automation proof-of-concept must therefore contain an intelligent component capable of automating functioning based on rules and environmental conditions. Referring to number 1 again in Figure 3.1, the master node will carry the majority of the intelligence, but it will also be distributed among the user interface and slave nodes.

3.2 Subsystem Specification

The remaining System Requirements are discussed in this section. These System Requirements are derived from the User Requirements Specification and are intended to achieve two goals. The first goal is to address the key problems in Home Automation which are summarised in Chapter 1 (after having been identified in the extensive literature survey detailed in Chapter 2).

The second goal is to produce a system that serves as a satisfactory Home Automation proof-of-concept. This is to create a system that demonstrates the functions and capabilities that were mentioned in the proposed solution of Chapter 1, as well as contributing to the research field through the development of a unique system.

This section translates the remaining User Requirements into System Requirements and clearly identifies the problems that each System Requirement addresses.

3.2.1 Total Integration

URS2 specifies the requirement to integrate with any home electrical device, and it directly addresses the problem of a **lack of standardisation**. However, it is a certainty that there will be cases in which it is not possible to integrate with a certain device, most of the time this can be overcome by using a different, compatible, device that performs the same function. These devices can range from HVAC units and televisions down to simple devices like a kettle. In order for this to be possible, the Home Automation system must be equipped with, or be capable of the necessary interfacing technologies. The Slave Nodes are responsible for delivering these capabilities, as it is there that interfacing with devices takes place.

For ease of explanation, device types are separated into three categories, namely: Intelligent devices, simple devices and standalone devices. Intelligent devices are those that contain their own micro-controllers and are capable of some form of communication, be it network connectivity or infrared control etc. (which would imply they can be controlled through communication on this medium). Simple devices are those without such technologies, such as lights or kettles. These devices are commonly characterised by an On/Off control nature, but slightly more complex devices do exist such as heaters with a temperature setting. The final category, standalone devices, are those that cannot easily be integrated into the system. This is due to their "closed box" design, capable of complex functions, without any communication capability to be taken advantage of. These devices (such as automated coffee machines and microwaves) are the exception to a Home Automation system. However, the design in this study has made provision for the integration of such devices in the futures.

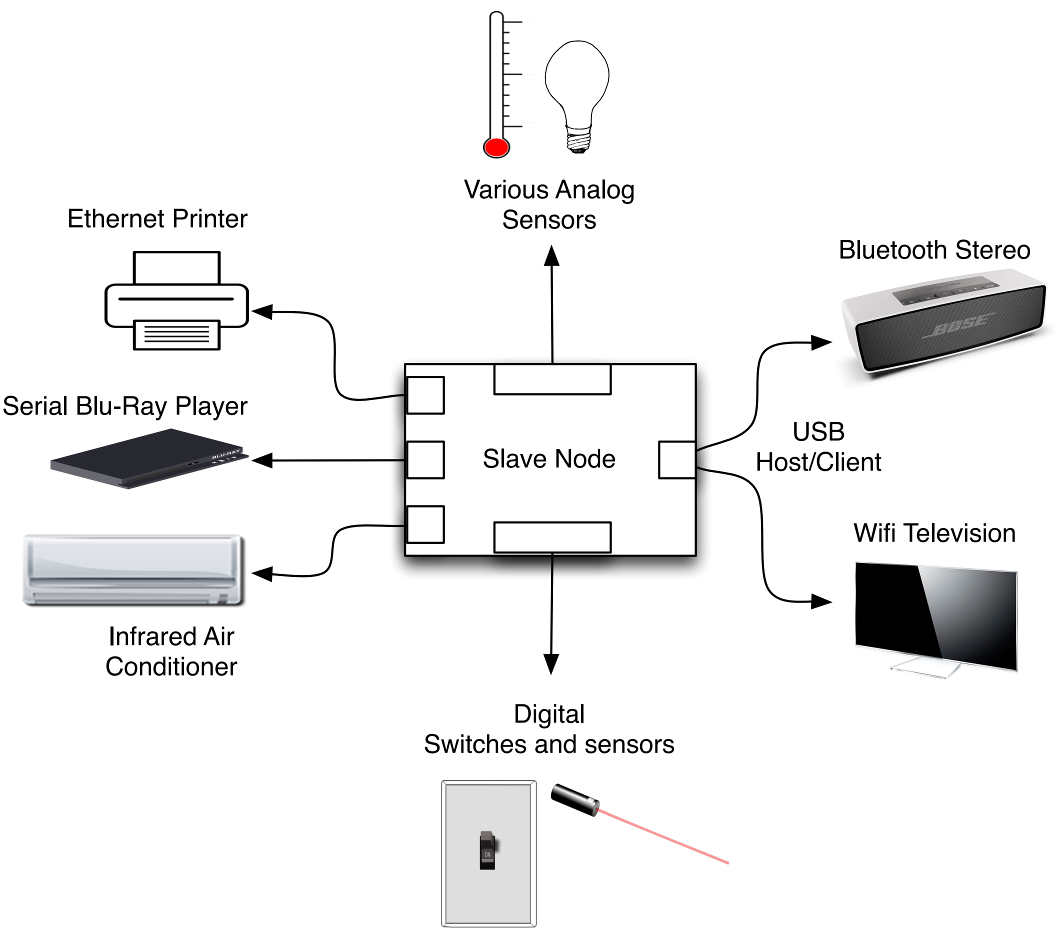


Figure 3.2: Example of a slave node capable of interfacing with various home devices (not all possible home devices are depicted in this figure).

The Home Automation proof-of-concept must therefore be capable of simple On/Off control, and allow for complex communication based control. On top of that, the system must leave room for potential integration with the aforementioned "standalone" devices should it become possible in the future.

The system architecture presented earlier in Figure 3.1 depicts the slave nodes as the components responsible for the integration of devices into the Home Automation system. A graphical representation of the slave node satisfying this system requirement is shown in Figure 3.2. It is important to note that the Slave Node is only capable of control of such devices and does not cater for the transmission of media data.

3.2.2 Non-Intrusive Installation

URS3 states that the Home Automation proof-of-concept should not interfere with the physical structure of the home. This is intended to ensure that the resultant system is simple and inexpensive to install and remove, directly addressing the **complicated and expensive** problem facing Home Automation.

The main consideration is the communication connections between master and slave, and the slave and its associated devices. Conceptually, Slave and Master might be located large distances apart, or in separate rooms. To facilitate a non-intrusive design, a wireless communication technology should be adopted for this connection.

In terms of the connection between the Slave and its associated devices however, a non-intrusive design is complicated by the lack of communication capabilities on common household devices. The simplest, while most costly solution is to provide a Slave Node for each device, but this is poor design practice and will also drive the cost of such a system unnecessarily high. To keep to good engineering practice, as well as minimising cost, more than one device will be associated which each individual Slave Node. To achieve a system that can be installed with minimum intrusion to the physical environment, a wireless technology (that does not depend on communication with the device itself) should be incorporated for switching of "simple" devices.

The next consideration is the effective placement of each slave within the home environment keeping in mind that they also require power. In some cases, placement of the slave will not be ideal in terms of the position of power points within the home. Unfortunately, this will result in a wire stretched from a power point to the slave node. The slaves' power supply and wire should therefore be kept small, and flexible to assist in ease of placement.

The System Requirement is therefore to incorporate a wireless technology for communication between Slave and Master Nodes, as well as the switching of "simple" devices from a slave node. Also the power supply and wiring should be kept as small and convenient as possible.

3.2.3 Intuitive and Simple

URS4 specifies the requirement of an intuitive and simple to install Home Automation system, also addressing the **complicated and expensive** problem. This implies that the functions of each component of the system should be clear and unambiguous, while installation should require little more than plug-and-play installation.

The system should therefore require no more intervention by a user than to enter their wireless gateway information for initial setup. Thereafter, the user should only need to ensure that all their devices are physically connected and that the system is on and running. More advanced intervention should only be necessary when modifying the functions of a slave node, and altering the master node and user interface to accommodate these modifications.

3.2.4 Open Source

URS5 states that the system should be open to development by a third party in order to promote expansion of the platform. This User Requirement directly addresses the problem of **growth and expansion** by forcing the project to involve the third party development community, a community that promises to be the best way to expand a system that must function in such a variety of home environments.

Developing a system that involves a third party community would require making use of open source hardware and software packages. There are a number of open source packages, some of which are high-end packages with small communities and others are good packages with very large communities working all the time to develop and progress the technology of today.

A detailed discussion on the available options, their advantages and disadvantages is presented in Chapter 4, but for now this System Requirement is to make use of open source software and hardware.

3.2.5 Successful Proof-of-Concept

To satisfy **URS6**, a successful proof-of-concept must demonstrate home automation that exhibits sensing properties that are relevant to the available actuation possibilities. There are a number of devices that are persistent throughout most homes, and environmental properties that are managed by an occupant on a frequent basis. Control and monitoring of these devices and environmental properties would allow for a good demonstration of the proof-of-concept.

The System Requirement is therefore to include control and sensing functionality of the most common household devices and environmental factors. This may include devices such as light switches and heaters, and sensors such as motion sensors and light sensors.

3.2.6 Modular Architecture

The length of the useful life cycle of a Home Automation system depends on its potential for expansion as time goes forward and the home environment evolves. A key factor in developing an expansible Home Automation proof-of-concept is to allow for addition and removal of devices from the system without the need for reconfiguration of the system. This can be achieved by making use of a modular design that allows a user to connect and disconnect slave nodes to and from the system while the core function continues to run.

Two factors will play a role in the design of a modular system, the first of which is the selection of the slave hardware to be used. The hardware must be able to function on its own without a connection to the master node required for boot or runtime. A connection should only be necessary for transfer of status information or control commands. The second factor is the software to be used. The software must not depend on any one slave node to be connected in order to function. Slave nodes should not be identified by hard code, but rather treated as clients each with a unique identifier provided by the slave node itself.

To summarise, the System Requirement, translated from **URS7**, is to utilise slave hardware that can function alone, apart from the system and to design the software to treat slave nodes as clients that provide their own credentials rather than hard coding, which would result in a inexpansible system. This System Requirement directly addresses the problem of **growth and expansion**.

3.2.7 All-Access User Interface

Not one of the largest factors adding to the **complicated and expensive** problem, but certainly an important consideration is the design of a user interface that is accessible by many users. There are many sleek and intuitive user interface design packages, but few of them are available to the consumer without added costs or upgrades. Some examples are to develop an Android application, or to develop the software from the ground up. However, only a small percentage of the consumers own an Android based computer/smartphone and developing from the ground up will greatly increase cost to the consumer.

URS8 translated to a System Requirement is to design a user interface that is simple (not require additional products or software packages) and accessible to the majority of consumers (excludes as few users as possible due to incompatibility).

3.2.8 Processing Power

Certain Home Automation applications such as media and entertainment require fair amounts of processing power. High resolution sensors require an analog interface and must be capable of fast floating point calculations. Although this proof-of-concept won't be demonstrating such capabilities, it must be considered in the design phase to adequately address the **growth and expansion** problem in Home Automation.

To translate **URS9** into a System Requirement, the Home Automation proof-of-concept must be designed to make use of a microprocessor with a fair amount of processing power, including an ADC (Analog to Digital Converter) chip and decent floating point operation capacity.

3.2.9 Distributed Intelligence

(URS10) aims to directly address the problem of **intelligence** in Home Automation, by distributing the intelligence, across the Home Automation system, to the Master Node, Slave Nodes and the User Interface. Commonly, Home Automation systems have focussed on integrating sensors and devices within the home by using a central control node, or intelligent controller. The goal of these systems is to automate tasks and routines within the home through the use of learning algorithms. This has resulted in the development of HA systems with highly intelligent central controllers that lack the integration and expansion capabilities of a system that distributes its intelligence. Figure 3.3 shows how the architecture of a system with its intelligence focussed at a central control node might look. Such a system would be capable of powerful automation, but it would excluded many devices and sensors that are necessary for rich and beneficial Home Automation. Also, a user interface implemented with this architecture is not as flexible as it could be.

Distribution of the intelligence in Home Automation is advantageous to a few of the User Requirements already addressed. A Slave Node with communications, and processing abilities supports **URS2** to **URS5** , **URS6**, and **URS9**. The addition of intelligence to the user interface will aid in achieving **URS4** , **URS5** , and **URS8**.

The Master Node will of course still hold a majority steak in the intelligence share as it must manage the entire system as well as maintain the home environment using control loops. However, sharing computational power among the rest of the system is a necessary feature that will aid in the growth and expansion of this Home Automation proof-of-concept that is well prepared for future development.

The architecture shown earlier in Figure 3.3 is altered in order to integrate the excluded devices and sensors, as well as implement a smart user interface. Figure 3.4 shows how the intelligence can be distributed and what the new system architecture might look like.

With this architecture, simple and complex devices can be integrated into the system as well as small, standalone sensors that have no built-in integration technology. Also, a smart user interface can be accessed by a variety of cross-platform devices as well as retain user information and personalised system configurations. It can also be possible to allow the user to generate intelligence (such as logic rules) that govern how the system automates the home.

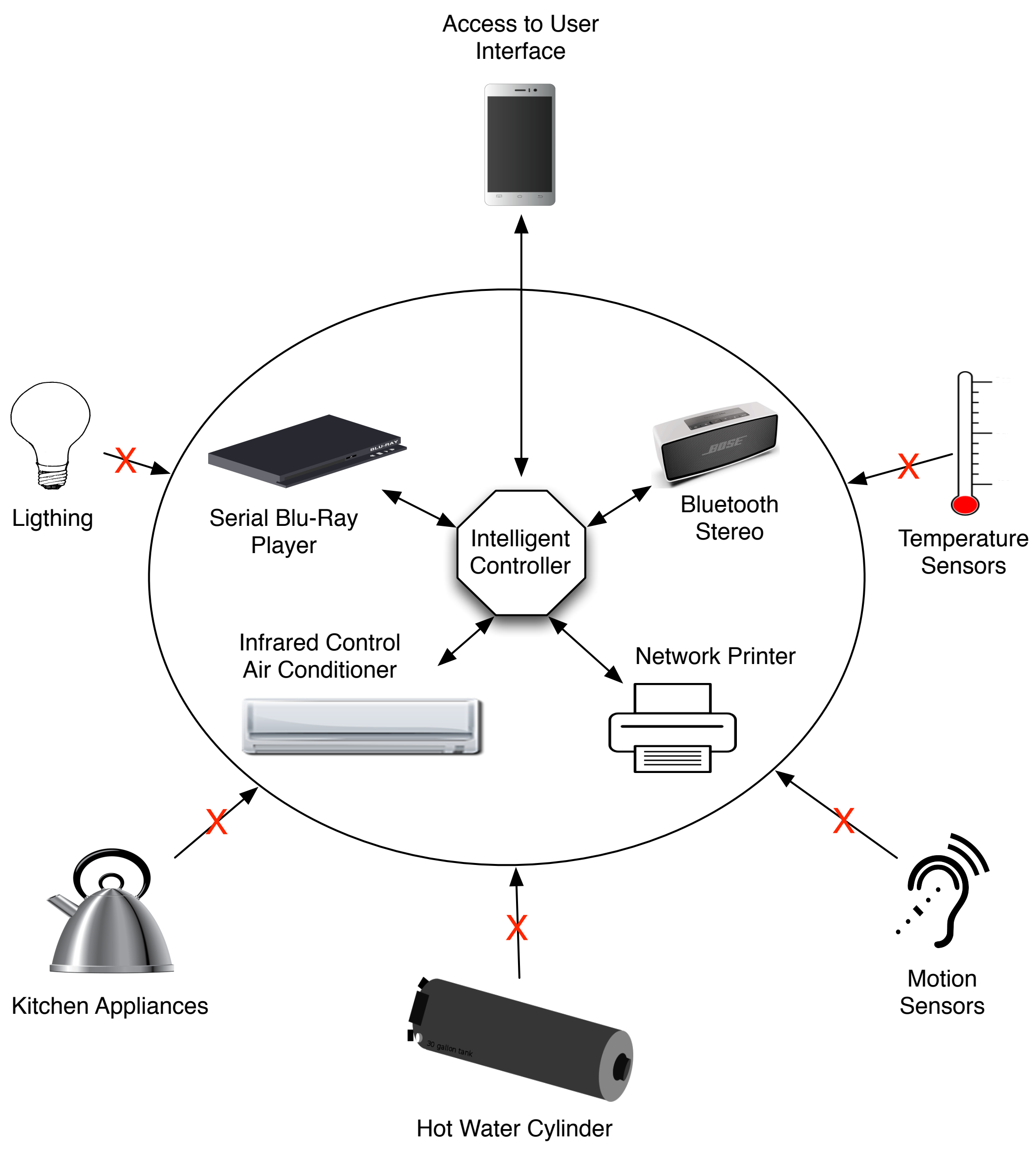


Figure 3.3: System with intelligence focussed at the central control node preventing the integration of simple devices that lack communication capabilities.

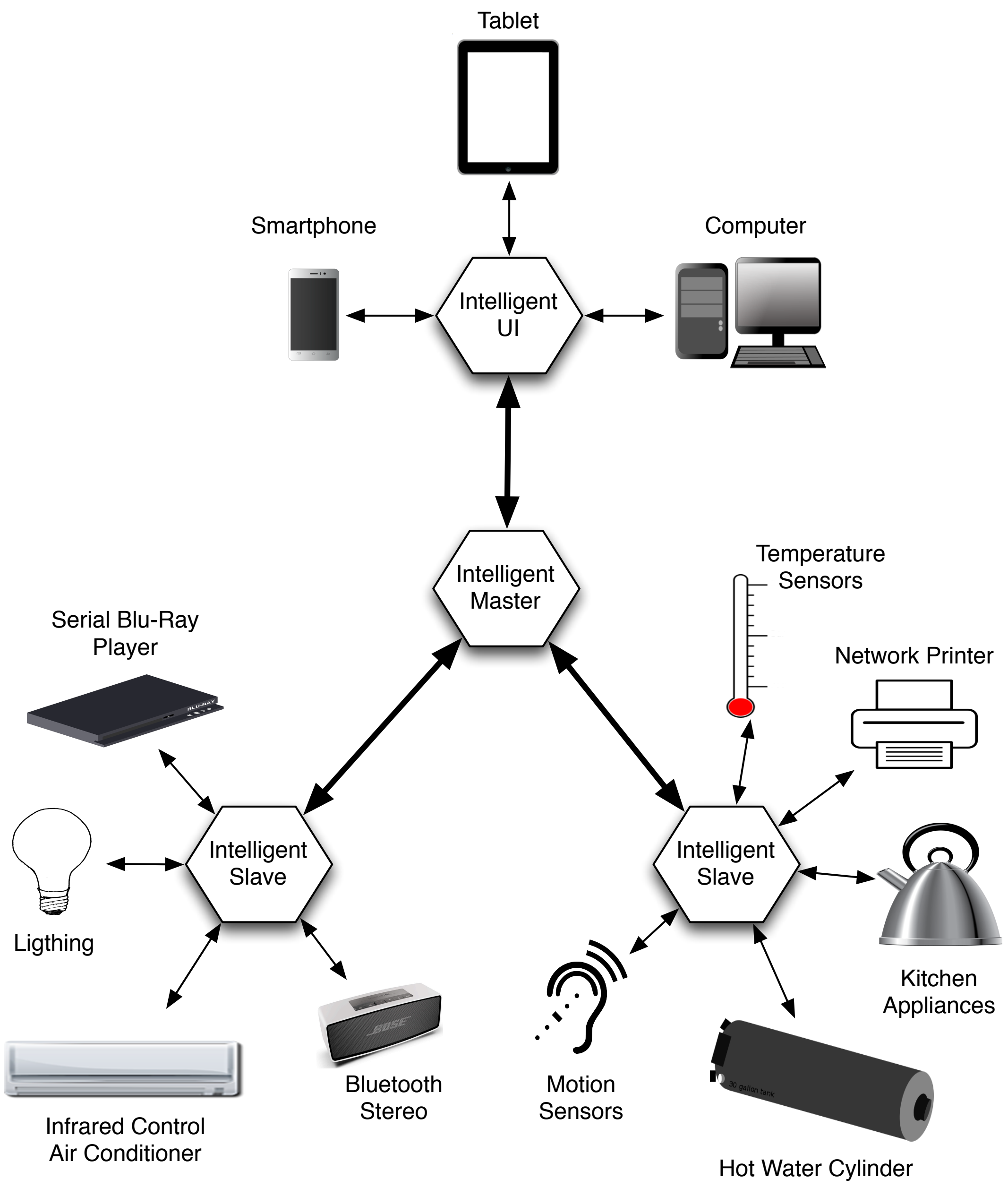


Figure 3.4: System with intelligence distributed across all nodes of the system.

This System Requirement is therefore to distribute, or make it possible to implement, intelligence across the Master and Slave Nodes as well as the User Interface.

3.3 Further Considerations

During the conceptual design phase, consideration also needs to be given to the following aspects.

Power Consumption - The system will most likely be connected directly to the home power grid and not operate on batteries. However the micro-controllers will only be capable of providing a certain amount of current to the actuators and sensors connected to it. Unless power is provided to these components from another source, the selection of these components must take into consideration the available power.

Physical Size - To prevent the Home Automation system from taking up space in the home and intruding on the users lifestyle, the selected components must be as small as possible. This also adds to the capabilities of the system when integration with a device in a small space is required. With small hardware the user can conceal parts of the system and mount nodes on the walls or roof increasing the effectiveness of certain sensors and removing the system from the living space.

Component Availability and Support - This is a factor that should be taken into consideration in any design project. It helps to prevent unnecessary complications during the hardware design process, and it will also benefit the expansive nature of the system. Therefore, components that are readily available and that have good support should be selected.

Cost - Prototype development is usually a costly endeavour but as this is a research project, the cost of the selected components should be kept as low as possible.

3.4 Traceability of System Requirements

		System Requirements Specification													
		System Components					Hardware				Software				
		Coordintating Master Node	Device Integration	User Interface	Communication Medium	Intelligent Control	Slave node with digital I/O and communication interfaces	Wireless relay control	Include most common devices and sensors in proof-of-concept	Slave nodes with intelligent potential	Micro-controller with ADC and high sampling potential	Self configuring system	Cross-platform user interface	Slave node and user interface with intelligence potential	Open Source
User Requirements Specification	A Home Automation System	X	X	X	X	X									
	Integrate with all devices		X				X			X					
	Non-Intrusive installation				X			X		X					
	Simple and intuitive		X	X						X		X		X	X
	Open to third party development		X							X			X		X
	Successful demonstration		X				X	X	X			X	X		
	Modular master/slave design		X							X		X		X	X
	Accessible user interface			X									X		
	Mid-range proccessor										X				
	Distributed intelligence		X		X	X	X			X				X	

Table 3.1: Matrix tracing System Requirements back to the User Requirements Specification.

Chapter 4

Conceptual Design

Following the establishment of a System Requirements Specification, the design phase of the project can begin. The first step in the design phase is to conceptualise a high-level implementation of the system. This entails outlining the system architecture and characterising the functions of each component in the system. Emphasis is placed on determining where each component is located in the system hierarchy and the paths of information flow. The second step, Concept Selection, is to analyse available hardware to determine which hardware best satisfies the System Requirements identified earlier. An overall system diagram can be seen in Figure 4.19 and can be folded out for easy reference while commencing with this chapter.

4.1 Conceptualisation Based On User Requirements Specification

The most important consideration in the conceptualisation of a system implementation is the fulfilment of the User Requirements identified in the URS, as these are intended as guidelines to ensure a successful design. This is achieved by developing a concept that satisfies the System Requirements. The most important requirements are listed below and a conceptual design is reached for each individually. These components are later combined to form the overall system conceptualisation.

A Home Automation System (URS1):

In Chapter 3, an overall system architecture is partly discussed. The simplest Home Automation system gathers information from sensors situated around the home, and carries out intelligent control of devices within the home. It also allows a user to view the status of the home, and directly control the de-

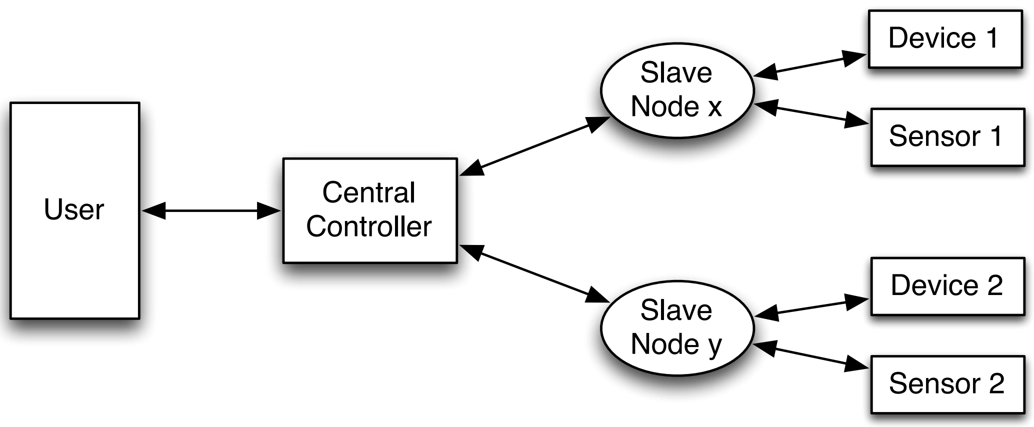


Figure 4.1: Simplest Home Automation system architecture.

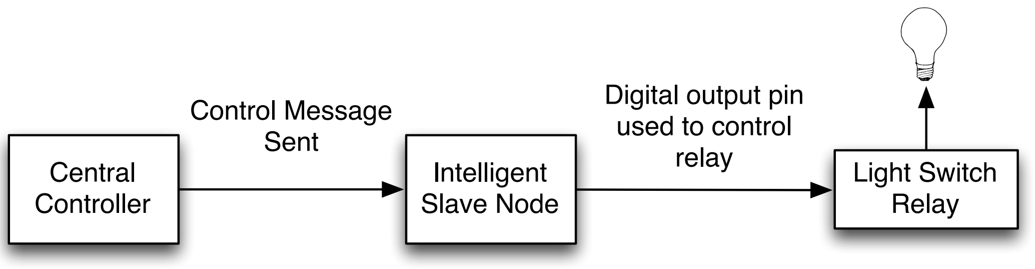


Figure 4.2: Conceptual slave design.

vices connected to it, all through a central intelligent controller. The simplest Home Automation system architecture is depicted in Figure 4.1.

Complete Integration (URS2) and Distributed Intelligence (URS10):

As has been emphasised, integrating with all household devices (or the capability to do so without modifications to the system at its core) is a very important aspect of this Home Automation proof-of-concept. Conceptually, this is made possible by distributing the intelligence of the system. Without an intelligent gateway to interface with the simple household devices, only the devices capable of some form of control through a communication medium would be able to integrate with the system. This way, simple devices are integrated into the system by using an intelligent Slave Node that interprets information from the central controller and handles control of the device based on this information. A conceptual Slave Node design is shown in Figure 4.2.

Furthermore, distributed intelligence allows us to empower the user interface by providing the user with an intelligent control interface. Capabilities of such a system include allowing the user to customise the naming convention of their devices as well as retain this information in the event of a blackout. The user would also be able to create rules that would govern the automation of

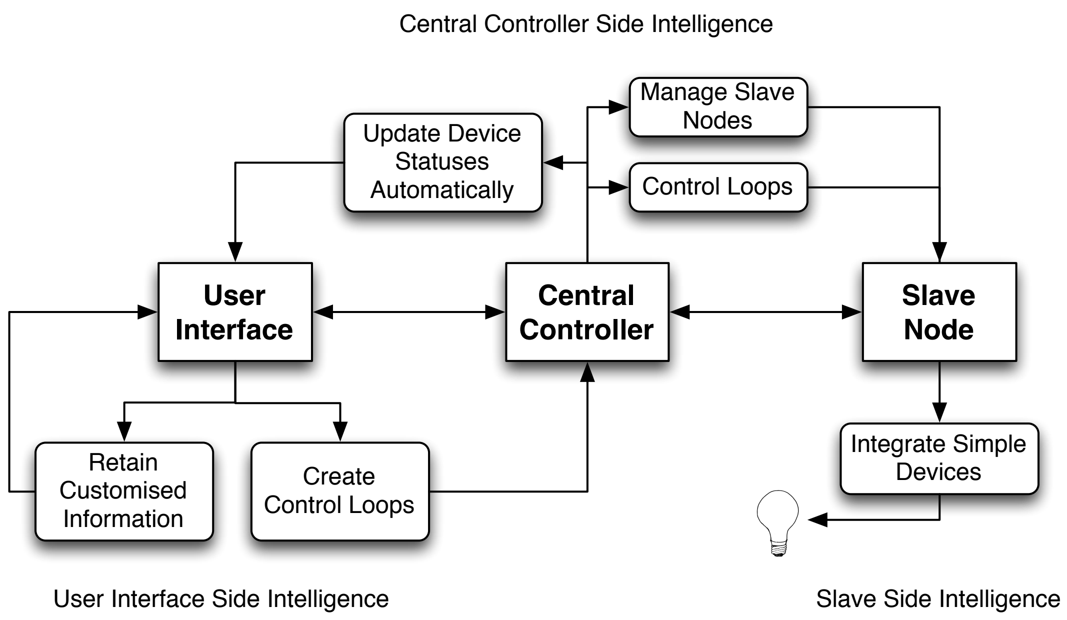


Figure 4.3: Conceptual intelligence distribution.

their home by selecting trigger events and control actions to form intelligent control loops. Figure 4.3 shows how the intelligence is conceptually distributed across the system and some of the functions this might aim to achieve.

Non-Intrusive Installation (URS3):

One of the User Requirements is to design a system that can be installed without intruding on the physical structure of the home. The System Requirement Specification contains two items that will ensure this is possible. First is the use of an intelligent micro-controller as the Slave Node. This makes it possible to achieve the second, which is the use of wireless communication technologies firstly between the Master and the Slaves and secondly between the Slaves and the devices they control. This conceptual design caters for the placement of Slave Nodes throughout the home without running cables back to the Master as well as wireless control of devices. Figure 4.4 clearly shows how this concept aids in preventing intrusion on the physical boundaries and space of a home.

Modular Master/Slave Design (URS7):

A modular design is one of the important aspects of the User Requirements Specification. The ability of the system to expand and grow in the future, with the input of third party development, is highly dependent on the Master/Slave relationship of the system. Often, Home Automation systems are designed as a closed box, with certain integration capabilities and automation functions.

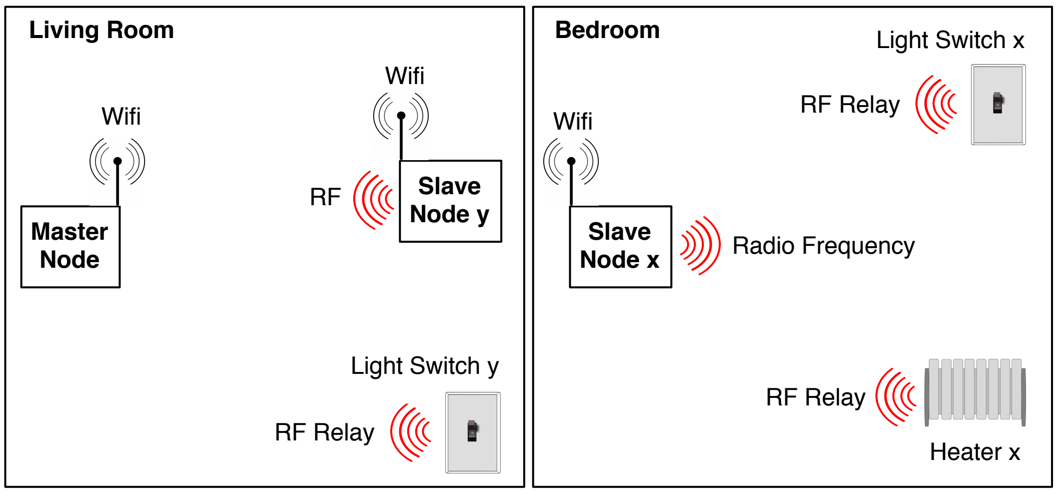


Figure 4.4: Conceptual use of wireless technologies to reduce physical intrusion during installation.

The use of a modular system allows Slave Nodes to be added and removed without altering the Home Automation system at its core. Also, Slave Nodes can be designed with specific purposes suiting the needs of individual users.

Conceptually this requires a system architecture that treats the slaves as clients rather than as functions. After starting up, a Slave Node would sign in to the Home Automation system and broadcast its connected devices and functions. The Home Automation system would in turn handle each Slave Node as an independent client, but still allow integration between Slave Nodes. Figure 4.5 shows this relationship between the Master Node and one Slave Node. Depicted is the transfer of information from the starting of a Slave Node up to normal function.

4.2 High Level System Overview

A system conceptualisation has been formed based on the important aspects of the User Requirements Specification considered individually. These concepts are now combined to form an overall system conceptualisation that stems from the URS.

The conceptual system is centred around a modular Master/Slave design that distributes the intelligence to enhance expansion and growth capabilities. The wide variety of homes that exist along with the constant development of new household technologies is accounted for with the flexible nature of such a system. This system concept is also powerful in its ability to integrate not only intelligent devices, but the simpler devices that are very seldom integrated into Home Automation systems. The overall conceptual system architecture is shown in Figure 4.5.

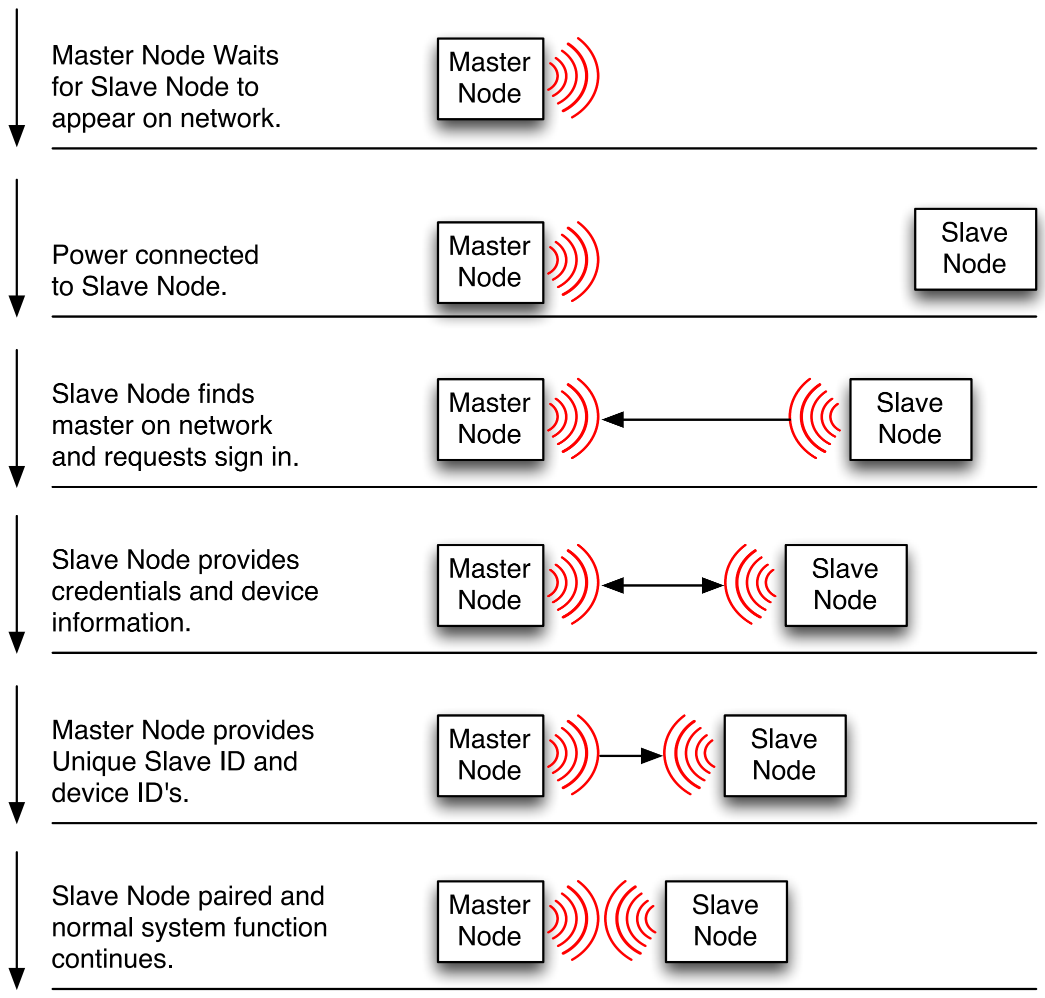

Master Node Waits for Slave Node to appear on network.
Master Node
Power connected to Slave Node.
Master Node
Slave Node
Slave Node finds master on network and requests sign in.
Master Node
Slave Node
Slave Node provides credentials and device information.
Master Node
Slave Node
Master Node provides Unique Slave ID and device ID's.
Master Node
Slave Node
Slave Node paired and normal system function continues.
Master Node
Slave Node

Figure 4.5: Conceptual functioning of a client based Slave Node connecting to the system.

From the figure, it can be seen that certain levels of abstraction exist. The user is removed from the system core functionality, and the system core is removed from the device integration component. This way the user is only provided with relevant, high level information and given control of an intuitive user interface. Also, the system core only communicates relevant information to and from the Slave Node and does not have to manage device integration as well. It should be noted that this figure is a representation of only one Slave Node, whereas the full system will integrate with many Slave Nodes and allow Slave Nodes to integrate with one-another.

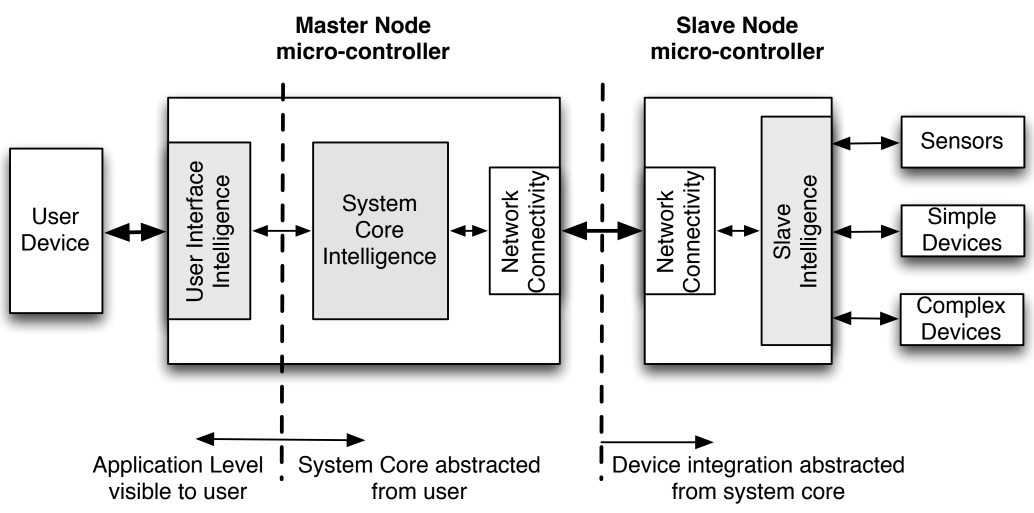


Figure 4.6: Conceptual architecture of the system as a whole.

4.3 Conceptualisation Based on System Requirements Specification

Having conceptualised a top level system architecture with regard to the URS, a low level system architecture can be conceived. The two drivers behind this process are, firstly, the system conceptualisation discussed in Section 4.1 derived from the URS and secondly, an analysis of the System Requirements Specification.

4.3.1 Hardware

Master Node - The SRS requires that a central controller, or Master Node, manage the coordination of the system along with implementing intelligent control. Also, it is a requirement that the user interface be accessible across multiple platforms.

From the system architecture given earlier, it is clear that the user interface will be situated on the Master Node and that the Master Node must be capable of some form of network access. The Master Node must therefore be a micro-controller capable of functioning as a standalone unit as well as running some form of operating system. Running an operating system, as opposed to a dedicated micro-controller, provides greater potential for expansion and development that is not hindered by the limitations of a dedicated micro-controller. To comply with the open source requirement of the SRS and to aid in developing a cross-platform user interface, this operating system must be open source. Selection of the micro controller, network connectivity and the operating system, as well as the user interface software is discussed further in

Sections 4.4 and 4.5.

Slave Node - An important component of the system is the Slave Node, as it is responsible for integrating with the multitude of household devices. It will also determine the extent to which expansion of the system is possible and whether or not third party development can take place. From the SRS, the Slave Node must firstly be a micro-controller capable of functioning on it's own, with network connectivity and an open source operating system, like the Master Node. To add to that, it must have on board digital input and output pins for integration with simple devices as well as some form of serial communication interface for integration with more complex devices. Also it must have on board analog inputs in order to integrate with the relevant sensors. Lastly, as is mentioned in the URS conceptualisation, it must be capable of some form of wireless/remote relay control.

4.3.2 Software

User Interface - The SRS states that the system should include an intelligent user interface that is accessible across platforms. These platforms include personal computer running a variety of operating system, and mobile devices such as smartphones and tablets running operating system like Android, iOS and Blackberry.

An intelligent user interface would conceptually be capable of retaining user defined information after a shutdown. This information can vary from customised device names, making it more intuitive, to user defined control loops. The cross platform nature of the user interface will be determined by the selected software upon which the interface is developed. An analysis of available options and the selection process is discussed further in Section 4.5.

Master - The SRS states that the system software should be self configuring. This means that it should not require user intervention during boot up and normal operation (unless it is booting for the first time, in which case the user will be required to enter network information), or when a new slave is connected. The master software should be capable of retaining pertinent network information, and of integrating new Slave Nodes into the system autonomously.

Slave - Finally, the SRS states that the Slave Node should have intelligent capabilities. This is mostly covered in the hardware discussion of Slave Nodes, however the important consideration is that the slave software is capable of being configured to integrate with almost any home device. It was mentioned earlier that this proof-of-concept is small in comparison to its potential, but it is import that there is potential. This is determined by the Slave Nodes capability to run intelligent, configurable software.

4.4 Hardware Component Selection

Hardware to be used is an important consideration in the component selection process. This will determine the extent to which the system is capable of expansion from a hardware and software perspective and must therefore take place before the software selection process. The components to be selected are the micro-controllers to be used for the Master and Slave Nodes (a summary of the analysed micro-controllers is given in Table 4.1), the sensors that will be included in the proof-of-concept as well as the actuators that will be included. The available components are analysed and compared and those that best suit the requirements are selected. This selection process needs to closely consider the conceptualised design reached earlier as well as User Requirements **URS2 - URS6** and **URS9**.

4.4.1 Microcontroller

Arduino:

The first micro-controller to be considered is the Arduino. Arduino is an open source electronics platform intended for interactive projects [38]. There is a variety of Arduino boards, the most popular of which is the Arduino Uno. This board is based on the ATmega328 micro-controller with a 16 MHz clock. It also has a variety of digital I/O pins, PWM pins and analog inputs. The Arduino Uno is very easy to program with its own development environment, and is small in size. Unfortunately it does not run an operating system, nor does it have any network connectivity although it does have a USB connection and serial communication pins.

The Arduino Yun on the other hand is a newer model from Arduino, boasting an Ethernet port, Wifi chip and an embedded Linux machine. The Yun also has more digital I/O pins and analog inputs. Otherwise it is quite similar to the Arduino Uno. Figure 4.7 is an image of the Arduino Uno and Figure 4.8 shows the Arduino Yun.

Raspberry Pi:

Similar to the Arduino range of micro-controllers, but slightly more powerful is the Raspberry Pi. This is a very common micro-controller powered by an embedded Linux operating system. It is a credit-card sized computer that can be connected to a television and keyboard allowing a user to do many of the things standard computers are used for [39].

The Raspberry Pi is equipped with network capabilities as well as an open source operating system, however it falls short with analog inputs and PWM outputs. In comparison with the other options, as shown in Table 4.1, It sells for a reasonable price and has a good processor that will allow good software

Parameter	Arduino Uno	Arduino Yun	Raspberry Pi	Android Mini PC	Beaglbone Black
Clock Speed	16 Mhz	16 MHz	700 MHz	1.6 GHz Quad	1 GHz
Operating System	No	Linux	Linux	Android 4.2	Unique IDE, Debian, Android, Ubuntu, and more.
Open Source	Yes	Yes	Yes	Yes	Yes
On-board Network Connectivity	No	Ethernet and Wifi	Ethernet	Wifi and Bluetooth	Ethernet
USB Ports	1 Client	1 Client	2 Host	2 Host	1 Host 2 Client
Digital I/O Pins	16	20	See GPIO Pins	0	See. GPIO Pins
PWM Output Pins	6	7	1	0	8
Analog Input Pins	6	12	0	0	7
ADC	10 bit Resolution at 9600 Hz	10 bit Resolution at 9600 Hz	NA	NA	12 bit Resolution at 200 KHz
GPIO Pins (General Purpose I/O)	0	0	26	0	69
Serial Communication	SPI	SPI	SPI, I2C, I2S, UART	0	4xUART, 2xSPI, 2xI2C, 2xCAN Bus
RAM	No	64 MB DDR2	512 MB	2 GB DDR3	512 MB DDR3
Memory	32 KB	16 MB		8 GB	4 GB
Memory Expansion	No	Micro SD Card	SD Card	SD Card	Micro SD Card
Operating Voltage	5 V	5 V	5 V	5 V	5 V
Qualitative Measure of Hardware Complexity	Low	Low to Medium	Low to Medium	High	Medium
Cost	R320.00	R1025.00	R446.00	R1000.00	R491.00

Table 4.1: Micro-controller Comparison.

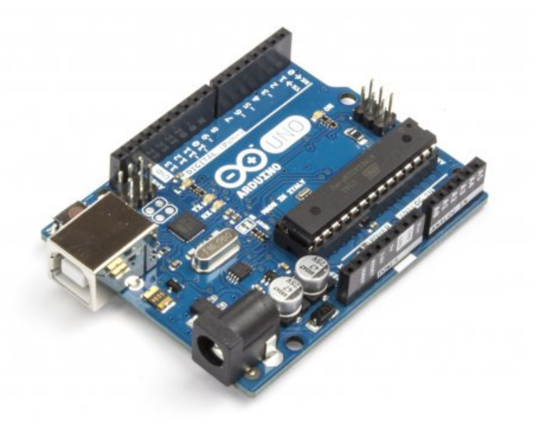


Figure 4.7: Image of an Arduino Uno [38].

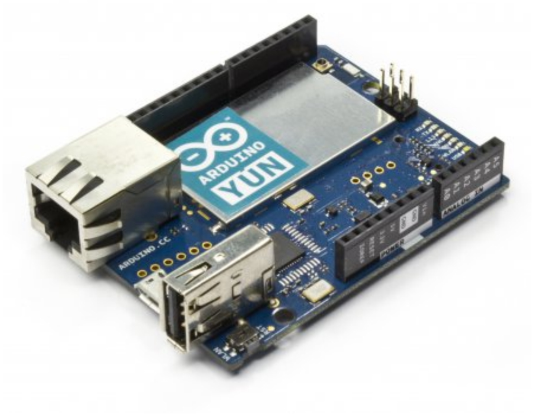


Figure 4.8: Image of an Arduino Yun [38].

expansion. A Raspberry Pi is shown in Figure 4.9 below.

Android Mini PC:

The Android mini PC [40] was considered along side the Arduino as suitable Master Node hardware. It has a very powerful CPU and large Memory for its compact size, and also runs an open source Android operating system. There are however a large variety of these devices in terms of size and specification. Also there is no official distributor, which may be the cause of the large variety, nor are there any distributors in South Africa (at the time of this consideration). Although this device could offer a lot of potential as a Master Node in the system, it is not a suitable Slave Node because it does not have the necessary peripheral interfaces. The Android Mini PC is shown in Figure 4.10 below.

Figure 4.9: Image of a Raspberry Pi [39].

Figure 4.10: Image of an Android Mini PC [40].

Beaglebone Black:

The Beaglebone Black is quite similar to the Raspberry Pi, but again, it has a faster CPU and many more I/O pins as shown in Table 4.1. It is not as common as the Raspberry Pi and Arduino, but it was designed to be compatible with both. Third party developers develop products for both the Raspberry and Arduino, and this includes developing hardware to expand the capabilities of these boards. This hardware is referred to as a cape or shield, and is capable of adding functions such as network connectivity or GPS. The Beaglebone Black was designed to be compatible with this technology which means development does not need to start over, but can just be applied to the Beaglebone. Figure 4.11 shows an example of an Arduino shield that can be used to add ZigBee (wireless communication technology similar to Bluetooth) functionality to the Beaglebone Black. The Beaglebone Black also boasts a 1 GHz processor, multiple operating system compatibility, and a host of other capabilities. An example of the Beaglebone Black micro-controller board is given in Figure 4.12.

Figure 4.11: Image of an Xbee Shield for the Arduino [38].

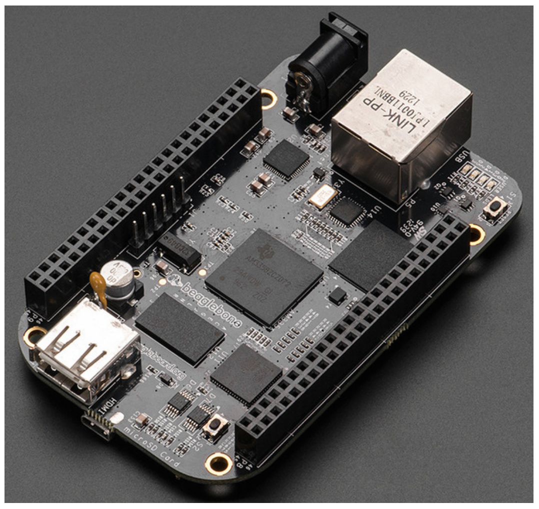


Figure 4.12: Image of a Beaglebone Black [3].

Each one of the mentioned micro-controllers are suitable components for either the Master or the Slave or for both nodes of the system. To keep cost and complexity of the system low, it is ideal to select a micro-controller that can be used for both. Table 4.1 provides a detailed comparison of each of the micro-controllers side-by-side. When analysing the table, a few aspects were kept in mind and they a drawn directly from the User Requirements Specification and the System Requirements Specification. Cost must be kept as low as possible, while the selected component must still be capable of meeting the system requirements.

Figure 4.13 is a graphical representation of the selection process detailing aspects of the URS and SRS that aided in the selection.

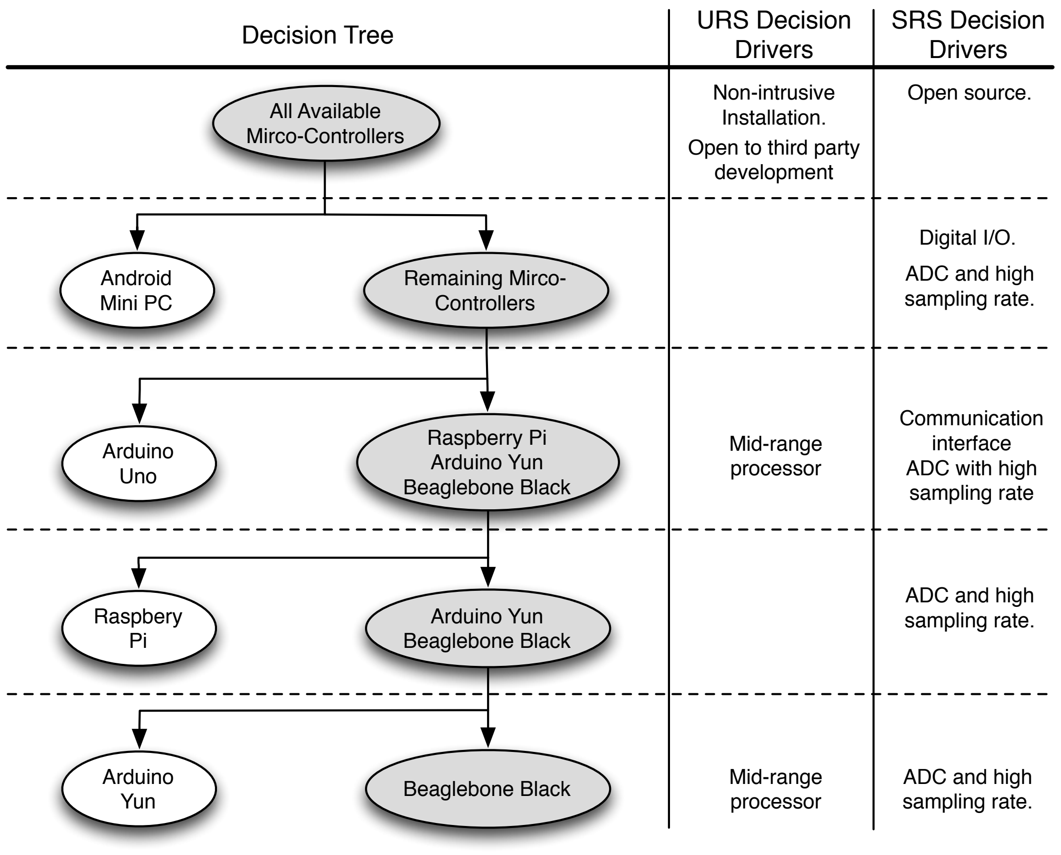

Decision Tree
URS Decision Drivers
SRS Decision Drivers
All Available Mirco-Controllers
Non-intrusive Installation.
Open to third party development
Open source.
Android Mini PC
Remaining Mirco-Controllers
Digital I/O.
ADC and high sampling rate.
Arduino Uno
Raspberry Pi Arduino Yun Beaglebone Black
Mid-range processor
Communication interface ADC with high sampling rate
Raspbery Pi
Arduino Yun Beaglebone Black
ADC and high sampling rate.
Arduino Yun
Beaglebone Black
Mid-range processor
ADC and high sampling rate.

Figure 4.13: Decision tree detailing the selection process of the micro-controller.

4.4.2 Sensors

It was not feasible to include all possible sensors in this proof-of-concept and for that reason, only sensors that apply to the most common homes and environmental factors were selected. Some of the common factors that a user would most often be concerned with are thermal comfort, light and dark, and the energy usage of their home. It was therefore decided to include 4 specific sensors in this proof-of-concept, namely: temperature, light, motion and AC current.

Temperature:

The applications of Home Automation in the thermal comfort of a home are extensive. Many factors can affect the way in which heating or cooling is utilised. Occupancy of a room, open windows or doors and the external air temperature are all applicable variables in air conditioning decisions. Thermal comfort is also the environmental factor that any user would notice first about a home. Adding temperature sensors was therefore a large consideration, and a few different implementations of temperature sensors were analysed.

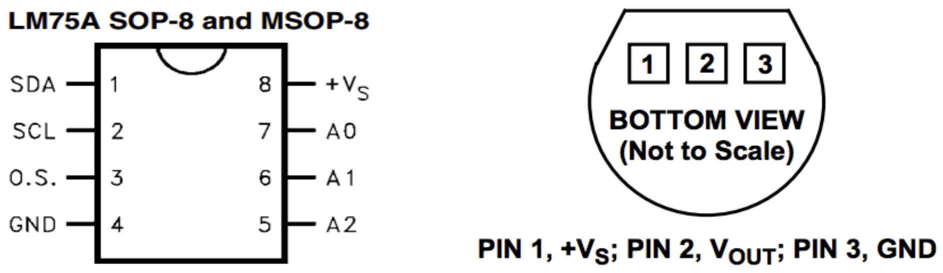


Figure 4.14: The hardware pinouts of the **TMP36** and the **LM75A** side-by-side.

LM75A - This is a popular 2-wire, 9 bit temperature sensor capable of accuracy to within 2 degrees Celsius within the range -25°C to 100°C. Supplied with between 2.7 to 5.5 Volts, it communicates temperature readings via serial-I2C.

TMP36 - Capable of the same temperature accuracy as the **LM75A**, this sensor works slightly differently. Also supplied with between 2.7 and 5.5 Volts, instead of transferring a temperature reading over serial communication, it outputs a voltage that is proportional to the supply voltage and the measured temperature. In this case, the output is scaled by 10 mV per degree celsius and the output is a constant 750 mV at 25°C.

Figure 4.14 shows the pinouts of the two sensors. The **TMP36** requires fewer connections and is simpler to implement. This will aid in setup and debugging of the sensors, and for these reasons it is the chosen temperature sensor in this proof-of-concept.

Light:

Home Automation systems can do more with the use of light sensors than switching lights on when it goes dark. Dimming blinds when sunlight is direct or switching a geyser from solar to electric heating are both functions that require a knowledge of light level. Along with temperature, light is one of the largest environmental factors to play a role in the lives of people.

The **TSL257** was considered for its functional similarity to the selected temperature sensor. It is a light-to-voltage converter that runs on a 2.7 to 5.5 Volt supply. Using a photodiode and an operational amplifier the sensor is able to convert light intensity to an output voltage. An image of the light sensor pinout is shown in Figure 4.15.

Motion:

A good complimentary sensor to temperature and light, is local positing.

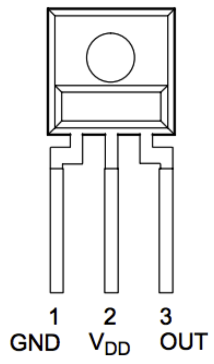


Figure 4.15: Description of the TSL257 light sensor pin connections.

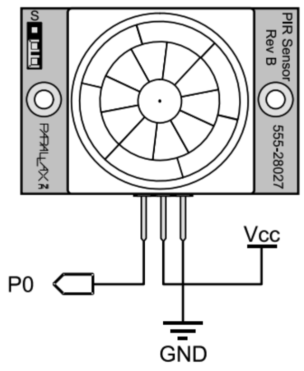


Figure 4.16: Description of the Parallax PIR #555-28027 motion sensor pin connections.

A users location within a home, even specific position within a room, is often governed by their needs at the time. A lot of research has been done on technology capable of tracking the whereabouts of a person within the home, but most of them require the person to carry part of the tracking system.

Although this technology would be advantageous to Home Automation, it is out of the scope of this proof-of-concept. The knowledge of a persons location on a room by room basis is sufficient for the successful implementation of this system. Motion sensors are typically made using Passive Infrared sensor modules. The selected motion sensor, the Parallax PIR Sensor #555-28027, runs on a 5 Volt supply and is capable of detecting motion up to 30 feet away. The sensitivity can be lowered, using a external jumper on the unit, to a range of 15 feet. Figure 4.16 shows the motion sensor pinout.

Current:

Lastly, a predominant focus of Home Automation is that of power consumption management. Concern is always placed on the power usage of devices within the home, and how routines can be changed to save energy. Especially

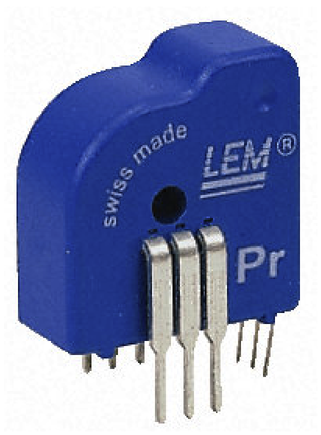


Figure 4.17: figure showing the selected LTS 25-NP current transducer.

in South Africa with a current energy shortage, this is not only a concern to the cost of living but also the stability of the power grid.

A current transducer measures the level of current running through a particular wire. Simple current sensors work by converting the electromagnetic field, generated by current traveling through a wire, into a secondary current loop that can be easily measured.

The **AC1015** was considered first due to its simple hardware design. However due to the fact that power is an alternating signal, this current sensor would require complicated rectifying circuits and calculations to determine the power being consumed.

The **LTS 25-NP** on the other hand is simpler to operate. It requires a supply of 5 Volts, and outputs a relative analog voltage based on the measured current, between 0 and +- 80 Amps. A image of the LTS 25-NP is shown in Figure 4.17.

4.4.3 Actuators

Low Power Relay Switches:

Simple on/off switching is made possible using relay switches. They are simple to implement and control, but unfortunately require a physical connection which does not satisfy the System Requirements Specification. Implementing a wireless communication technology to switch the relays circumvents this fact, but it adds to the complexity of the switching. The most obvious wireless technology for this application is radio frequency (RF) communication.

While researching possible wireless relay solutions, a South African company called Qwikswitch [41] was found. Qwikswitch specialises in wireless lighting control and manufactures a host of wireless lighting products, including RF relay and dimmer receivers. These units are compact in size, and simple to operate with that ability to switch currents up to $5A$. They replace the physical light switch, and allow a user to remotely switch and/or dim their lighting.

The transmitter units are available in a variety of forms including key rings, stick-on light switches and remote controls. The PCB's in the key ring transmitters allow them to be integrated into and controlled by other circuitry.

The combination of compact, easy to use receivers and the PCB of the key ring transmitters make these products ideal for integration into this proof-of-concept, also preventing a reinvention of the wheel through redesigning wireless RF switches. The receiver units are also capable of dimming using their own on board circuitry making these products more attractive as wireless relay solutions.

High Power Relay Switches:

The Qwikswitch relays selected are not capable of high power switching. Devices such as the kettle and hot water cylinder are capable of drawing currents in excess of $5A$ and will therefore require higher power relays.

Without being able to predict the maximum current that any one user may want to switch, it is necessary to select a relay that is capable of high current, but also cost effective and compact in size. At the same time, if the user wishes to switch a higher current than is capable, the selected relay can be used to switch a higher power relay provided by the user. A compact 10 Amp relay is therefore selected for high power switching. It will however not be capable of wireless control due to the fact that wireless relays are already available.

4.5 Software Component Selection

Selection of the software to be used is partly dependent on the selected hardware. Certain software components will not be compatible with the selected hardware, these components are not mentioned due to their irrelevance. The components to be selected here are the language in which to program the user interface as well as the Master and Slave Nodes. Most importantly however, selection needs to take place on how to design the system control software. The selection of the software components must closely consider the conceptualised design reached earlier as well as User Requirements **URS1, URS7, URS8** and **URS10**.

4.5.1 Management

While holding to the distributed intelligence requirement, the system will still require a central controller running some form of management software. It is important that there exist a managing component to allow for intelligent control without human intervention, but it is also necessary to make the modular design concept possible. The management software is thus the most important component of the systems software design.

REST:

In the research phase of this project, a number of possible design avenues were identified. REST, or Representational State Transfer is an architecture style consisting of a coordinated set of constraints applied to components, connectors and data elements. It is basically an abstraction of the architecture of the World Wide Web, and when combined with a web server, and a communication protocol, it can be designed to serve as a Home Automation management package. The communication protocol investigated for use with this setup was JSON (Javascript Object Notation). It is a lightweight data-interchange format that is easy for humans to read and write, and for machines to parse and generate. JSON is a text format that is completely language independent, but it uses conventions that are familiar to a host of other languages, making it an ideal data-interchange format.

Shortly after research into developing a management software package based on REST had started, an open source third party project was discovered. This project is called The Thing System [5]. The Thing System project is focussed on implementing an Internet of Things type architecture, but on a larger scale. The Internet of Things refers to the interconnection of uniquely identifiable embedded computing-like devices within the existing internet structure. The Thing System aims to avoid developing one large universal remote that depends on Internet access, by enabling things to watch for sensor events, resulting in control actions on those things. The project has developed a software system called the Steward [42] responsible for making this possible.

Steward:

The Steward is a software system implemented in Node.js [43], a server side javascript platform that runs on almost everything including the Beaglebone Black. In the old days, a "steward" was a trusted servant who ran the household on behalf of the owner. The home-owner sets the rules, and it is the Steward's duty to ensure that they are correctly implemented [5].

The Steward software system is an intelligent management system that automatically waits for a household device/thing to enter the network. Once a thing has been detected, and if the Steward knows how to interface with that particular thing, it is added to the list of things integrated into the Steward system. These things, such as Phillips Hue Lightbulbs or Nest Thermostat, are then available for the user to monitor and control from the Stewards user interface. The list of things that the Steward can interface with [44] is quite extensive, and increases everyday. However, the Steward is not capable of integrating with any household device that is not capable of communication.

With the Steward in existence, writing a management software package is not necessary. The Steward will therefore be used at the core of the system to handle management of the home devices and access by a user, but the user-side and device-side system will still need to be designed and programmed such that they can interface with the Steward. This means that the Slave Nodes will need to be designed such that they can integrate household devices with the Steward, and a custom Master Node User Interface will need to be designed so that the user has access to the intelligence of the Steward and these devices.

4.5.2 Master - User Interface

Android:

Initially, it was considered that the user interface should be written as an Android app. Android is a stable, open source platform with a lot of support. It would also allow for the design of a very intuitive and user friendly interface. However, this design choice would immediately limit the acceptance of this Home Automation proof-of-concept to Android users. The same would be the case if iOS or Blackberry were used. Also, because the Steward is built using different software, the user interface would have to be maintained on a separate machine, adding to the cost and complexity of the system.

Web Server:

A web server based user interface was then analysed and it was discovered that a web server can be designed quite easily using Node.js, just like the Steward software, and run on a Linux machine such as the Beaglebone Black. A web server based user interface would allow any user with a smartphone or a network connected computer to access the system. The technology is also fairly simple to use resulting in a wider adoption among the open source community promoting expansion and development. Figure 4.18 shows the spread of mobile phone and computer technology across the United States. The image shows that designing a user interface from the ground up on unique software would result in the largest accessibility potential, however it has already been mentioned that this would drive the cost to the consumer up, for numerous reasons, and result in lower adoption of the system.

4.5.3 Slave - Control Software

The Slave Node is responsible for two things namely: integrating all possible devices into the Home Automation system, and interfacing with the Steward

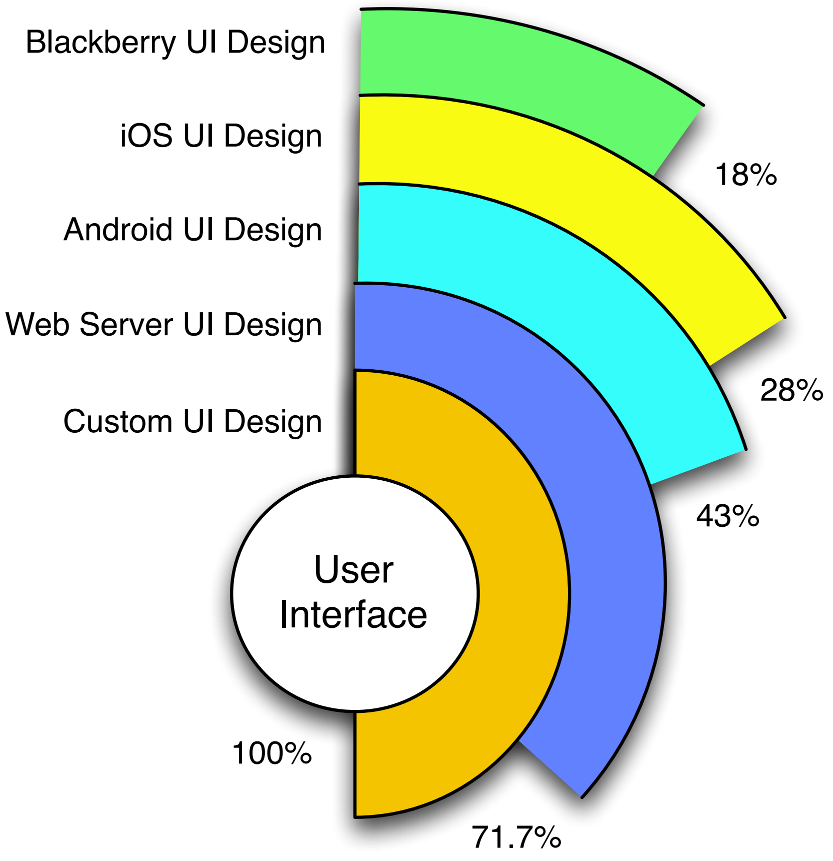


Figure 4.18: Graphical representation of the percentage of US citizens that use various smartphone and computer technologies (data gathered from [45] and [46]).

management software. These two aspects governed the slave software decision. To begin with, the Beaglebone Black is shipped with an Angström distribution of Linux, and its own IDE, called BoneScript, for interfacing with the onboard peripherals. Soon after development began with the Angström Distribution, it was decided that Angström had far too many issues that needed solving on the manufacturer side before it could be successfully implemented into this system.

Due to compatibility between the Steward software and the Debian Wheezy distribution of Linux, Debian Wheezy was chosen as the operating system to move forward with. With a functioning operating system, Node.js was considered as the development environment for the Slave Nodes due to the fact that it it was being used in the Steward and it had been chosen for the Master Node.

Further research exposed a community of open source packages that made many functions and applications possible on the Beaglebone Black using Node.js. For this reason, Node.js was selected as the development environment for the Slave Node.

4.6 Selection Summary

The selection process has resulted in a conceptual, component-focused, system design. To clarify the result, Table 4.2 presents a summary of the selected components in the hardware and software subsystems.

			Master Node	**Slave Node**
Software	**Management**		Steward	
	OS		Debian Linux	Debian Linux
	Programming Language		Node.js	Node.js
	User Interface		Web Server	
Hardware	**Micro-controller**		Beaglebone Black	Beaglebone Black
	Sensors	**Temperature**		TMP36
		Light		TSL257
		Motion		PIR
		Current		LTS 25-NP
	Actuators	**Low Power Relay**		Quickswitch
		High Power Relay		Finder 10A
		Dimmer		Quickswitch
		Infrared		X

Table 4.2: Summary of the selected components from the conceptual design.

4.7 Chapter Summary

In this chapter the conceptual design of the system was discussed followed by the selection of the hardware and software components that would realise such a system. A conceptualisation of the entire system based on the User Requirements Specification was reached in Section 4.1. This process took into account the URS to ensure that the top level requirements were satisfied. A high level system overview is reached in Section 4.2 and it is shown that the system architecture will be based on a Master/Slave design. An entire system overview representing the modular design including the selected micro-controllers can be seen in Figure 4.19.

In Section 4.3 the conceptual design is expanded on from a hardware and software perspective by looking more closely at the System Requirements Specification. This section explains in more detail the conceptual design at a system level.

Finally Sections 4.4 and 4.5 discussed the hardware and software selection process. Each component selection is discussed in detail and related back to the URS or conceptual design. The next chapter discusses the detailed design of the system using the selected hardware and software components. It aims to make clear how the system functions and what it is capable of from a lower system level.

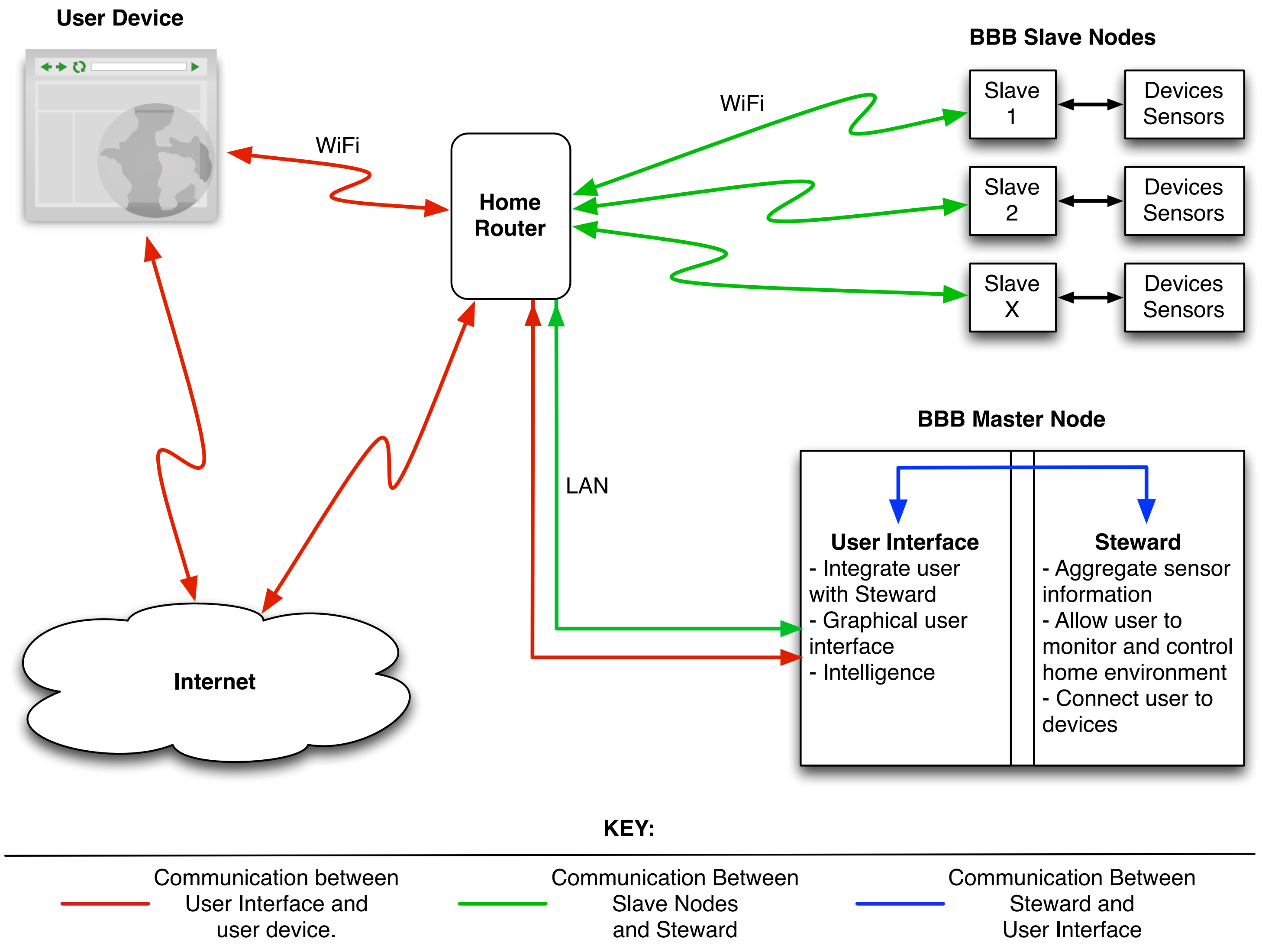

User Device
WiFi
Home Router
WiFi
BBB Slave Nodes
Slave 1
Devices Sensors
Slave 2
Devices Sensors
Slave X
Devices Sensors
LAN
BBB Master Node
User Interface
- Integrate user with Steward
- Graphical user interface
- Intelligence
Steward
- Aggregate sensor information
- Allow user to monitor and control home environment
- Connect user to devices
Internet
KEY:
Communication between User Interface and user device.
Communication Between Slave Nodes and Steward
Communication Between Steward and User Interface

Figure 4.19: Setup of the communication network as well as location of hardware and software components within the network.

Chapter 5

Detailed Design

With a conceptualised design the selected components are now combined to realise the conceived system. The detailed design is separated into two phases.

The hardware design phase involves setting up the Beaglebone Black master and slave nodes to ensure that they are capable of meeting the hardware and software requirements, i.e. capable of running the selected software packages and interfacing with the required device hardware. Then the sensor and actuator components are integrated with the Beaglebone Black slave nodes using a custom designed printed circuit board (PCB), ensuring that the circuitry and the inputs and outputs function correctly.

The software design phase involves writing the slave node and user interface software. The slave node software must manage the connected hardware as well as integrate into the Steward management software package. The user interface must perform all the required functions, and also integrate into the Steward management software package.

5.1 Software Design

5.1.1 Steward

The Steward is responsible for aggregating Slave Nodes and sensor information, providing a path for device control and allowing a user to interact with the home environment. The Steward is located on the Master node along side the user interface software, as a separate package. The user interface communicates with the Steward locally to receive sensor information and send control commands. Control commands all pass through the Steward where it gets routed out to the correct Slave Node, and finally the correct device. Sensor information is transmitted from all the Slave Nodes back to the Steward where it is packaged and transmitted to the user interface as sensible data.

At the top most level, the Steward aggregates device information of all the connected sensors and actuators, and facilitates communication between them.

To make this possible, the Steward must have prior knowledge of the API used in the device it wishes to integrate into the system. The API information of sensor and actuator devices is added to the Steward by the creators on an on-going basis. Devices must be capable of communication and contain their own open API in order to integrate with the Steward in this way.

On the next level down, the Steward aggregates status information from the connected devices. The status information of a device refers to the state of its actuation, or the measurements of its sensors. Also, the Steward is able to affect control of the connected devices through the use of the device API (if control is initially possible through the API). NEST, a smart thermostat is a good example of a device, with an open API, that has already been integrated into the Steward framework. The Steward is capable of reading status information from the NEST thermostat, and also of controlling the set temperature in real time. Lastly, the Steward can be instructed to wait for a certain event to occur (an event is the arising of a particular device state i.e. actuator state, or sensor reading) and to perform a particular action should that event occur.

This brings us to the lowest level of the Steward's framework where advantage is taken of the Steward's completely open API. At this level, clients (devices or software packages that integrate with the Steward on a management level) that produce unique sets of data or that perform unique functions can be developed and integrated in the Steward framework. Client access to the Steward is intended for one of two main applications. Either, the client is used to monitor and control the behaviour of the Steward (and the connected devices), or a client is developed to produce unique data sets or perform unique control functions (in the event that no device already exists that is capable of achieving these functions). It is this capability of the Steward that is exploited to achieve a system that has the potential to integrate all devices (not only those with communication ability and an open API) into a Home Automation system.

5.1.2 Slave

The software on the Slave Node is responsible for integrating the connected devices into the Home Automation system, through the Steward. It is also responsible for interpreting the sensor data correctly and transmitting that to the Steward where all the sensor data is collected, as well as making control of the connected devices possible by receiving and interpreting commands from the Steward. As discussed earlier, the slave must integrate with the Steward as a client in order to make this possible.

The Slave software is discussed in three components: Steward integration, status management and control management. The first component of the Slave software is responsible for integration and communication, the status component for producing and transmitting sensor data and the control component for interpreting and executing control commands.

Integration With Steward

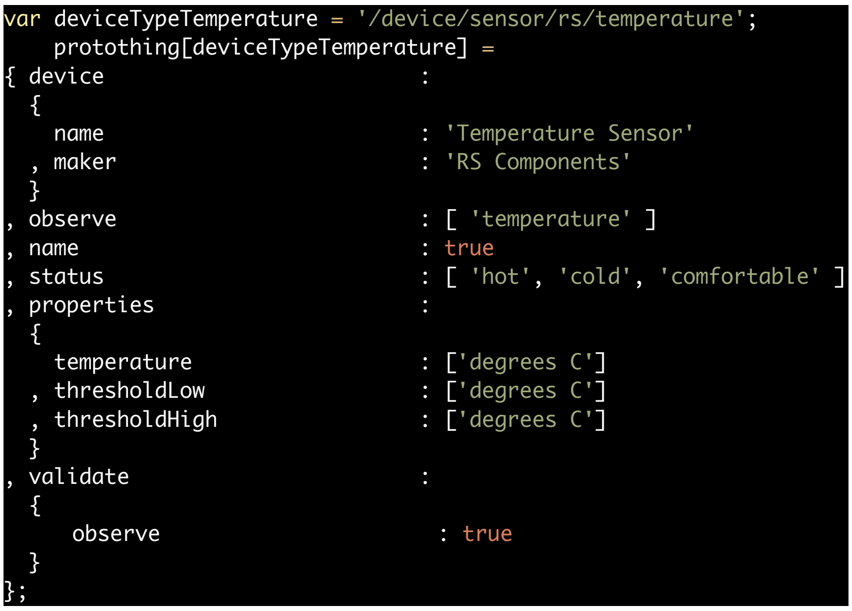
The integration component of the program manages the integration of the Slave Node, and all it's connected devices, into the Steward framework. This involves the registration of the Slave with the Steward and the defining and identification of the connected devices.

Registration of the Slave Node with the Steward involves two steps.. The first step is pairing in which the Slave Node provides a unique name and the server certificate (generated by the Steward for authentication). Once paired, the Steward responds with a block of information in JSON. This block contains the Slave Nodes unique ID and a unique authentication key. In the second step the Slave Node provides its unique ID and the unique authentication key for registration. In future, the pairing step is bypassed as long as the unique ID and authentication key have not been lost.

Once registration of a Slave Node has completed, the Steward must be made aware of the devices connected to that particular Slave Node. This is necessary for the Steward to know which information is relevant to which device, and also which devices are located on which Slaves. The Steward is informed of the connected devices by using a JSON code block identified as a device prototype, followed by a device definition that conforms to that prototype. A device prototype identifies a device as either a sensor or actuator, as well as defines the information it provides or the functions it performs. The device definition can be thought of the creation of a device that conforms to a certain prototype. An example of the JSON device prototype for a temperature sensor is shown in Figure 5.1.

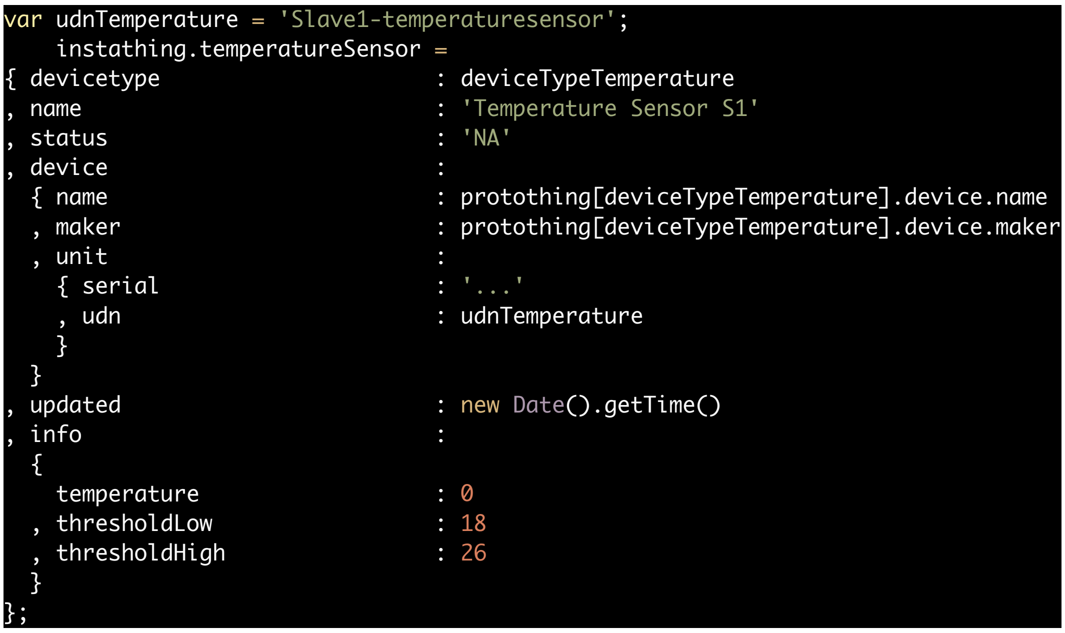
From the JSON code block it can be seen that the prototype contains information such as device name, elements it observers, status definitions and information properties. The prototype JSON code block for an actuator is quite similar, but along with an observe parameter it would include a perform parameter.

Once the device prototype has been provided to the Steward, a device conforming to that prototype can be registered with the Steward. To register a device, a unique device name must be given (that is unique across all devices across all Slave Nodes) as well as the relevant prototype. After transmitting the registration information, the Steward responds with a unique device ID. Device ID's are only valid for as long as they are connected to the Steward, and must be reregistered after disconnection (only Slave Nodes retain their unique ID's after disconnection). This unique device ID will be used for the remainder of the session to identify each device, understand which information is applicable to that device, and which control commands are intended for that device. An example of the JSON code block to register a temperature sensor is shown in Figure 5.2. Note that the "devicetype" parameter corresponds to



```
var deviceTypeTemperature = '/device/sensor/rs/temperature';
    protothing[deviceTypeTemperature] =
{ device                          :
  {
    name                          : 'Temperature Sensor'
  , maker                         : 'RS Components'
  }
, observe                         : [ 'temperature' ]
, name                            : true
, status                          : [ 'hot', 'cold', 'comfortable' ]
, properties                      :
  {
    temperature                   : ['degrees C']
  , thresholdLow                  : ['degrees C']
  , thresholdHigh                 : ['degrees C']
  }
, validate                        :
  {
      observe                       : true
  }
};
```

Figure 5.1: Example JSON code block of a temperature sensor prototype.



```
var udnTemperature = 'Slave1-temperaturesensor';
    instathing.temperatureSensor =
{ devicetype                      : deviceTypeTemperature
, name                            : 'Temperature Sensor S1'
, status                          : 'NA'
, device                          :
  { name                          : protothing[deviceTypeTemperature].device.name
  , maker                         : protothing[deviceTypeTemperature].device.maker
  , unit                          :
    { serial                      : '...'
    , udn                         : udnTemperature
    }
  }
, updated                         : new Date().getTime()
, info                            :
  {
    temperature                   : 0
  , thresholdLow                  : 18
  , thresholdHigh                 : 26
  }
};
```

Figure 5.2: Example JSON code block of a temperature sensor device definition.

the temperature sensor prototype shown earlier and the "name" parameter is a unique value, "Temperature Sensor S1" indicating that it is the temperature sensor connected to Slave Node 1.

This program component is also responsible for catching errors and handling the closing of connections and the program.

Status Management

This component is responsible for handling the continuous reading of sensor information and actuator states and the continuous communication of states back to the steward. It is at this component of the program where the analog signals from the light, temperature and current sensors are analysed, and converted into sensible data that makes sense to humans. The motion sensor digital output is also read, interpreted and communicated at this level and.

Light Sensor - As will be discussed in Section 5.2.3, the light sensor output ranges between $0V$ and $1.8V$. The BBB analog input pins read a maximum of $1.8V$, and the voltage on the pin is read as a decimal percentage of this maximum. i.e. if $0.9V$ is placed on the analog input, the value read in would be 0.5. The only calculation necessary for the light sensor is to multiply the analog value by 100 to give a light reading that resembles a percentage of direct sunlight.

Temperature Sensor - The temperature and current sensors, on the other hand, require a little more manipulation to arrive at a value that makes sense. For the temperature sensor, the relationship between temperature and the sensor output is discussed in Section 5.2.3. This relationship is used to convert the sensor output voltage into a logical value representing the measured temperature. First, the analog read value, which is a decimal percentage of 1.8, is converted to a voltage by multiplying by 1.8. An if statement then determines whether or not this voltage value is above or below the temperature sensors output voltage, $750mV$, at the standard temperature of 25°C. If the value is above, then the equation to find the measured temperature is as follows:

$$T_{measured} = 25 + \frac{V_{measured} - 750}{10} C \tag{5.1.1}$$

If the value is below, the equation is as follows:

$$T_{measured} = 25 - \frac{V_{measured} - 750}{10} C \tag{5.1.2}$$

Current Sensor - The current sensor requires a more complicated process due to the fact that the analog signal is oscillating when alternating current is measured. The input signal must be sampled for a certain time period, and at a frequency at least twice that of the input signal (in order to avoid aliasing which results in a distortion of the input signal). The common alternating current frequency is $50Hz$ but some countries also use $60Hz$. As mentioned in Section 4.4 the BBB has a maximum sample rate of $200KHz$, making it more than capable of avoiding aliasing when sampling a $50Hz$ signal. The signal is sampled for approximately $1s$ and the following equation, which relates the measured current to output voltage, is used to calculate the measured current I_P.

$$V_{out} = 2.5 \pm (0.625 * I_P/I_{PN})V \tag{5.1.3}$$

Within the sampling time, a number of point measurements are made, n. These measurements are squared and summed together and the result is divided by n to calculate a mean. The square root of the result is the Root Mean Square of the measured current signal which can be used to calculate power consumption in $Watts$.

With all the sensor inputs converted to logical values, the information can be communicated to the Steward. This occurs once every 10 seconds, or when the Steward directly requests an update on a specific device status. The program also sends a heartbeat message once every minute to inform the Steward that it is still active and connected.

Control Management

This component receives and executes control commands sent by the Steward, as a result of a user interaction with the system or an intelligent control loop triggered by a sensor event. The command message is analysed to determine which device it is meant for, based on the provided device ID, and the appropriate function is carried out. Currently, the possible control commands are "Toggle" and "Set", and the possible devices are the high power relay, one of three RF relays and an RF dimmer. The "Toggle" command is used to switch the state of any of the relays, and the "Set" command is used to set the dimmer level to either Off, 30%, 70% or 100%.

The function to control the high power relay is simple. When the relay is set to off, the digital output pin connected the the transistor is set to LOW. When the relay is set to on, the output pin will apply a constant $3.3V$ to the transistor, allowing $5V$ to be applied to the relay coil and subsequently switching the relay.

Controlling the RF relays and dimmer is slightly more complicated. In order to simulate a button press, the outputs connected to the remote need to be activated for only a short period. In the case of the RF relays, this period is $300mS$. Setting the dimmer works slightly differently as a button press switches the dimmer between Off, and 100% On and holding the button slowly varies the dim level between 0% and 100% until the button is released, at which point the dim level is set. Simulating this action is complicated by the lack of feedback from the dimmer module. In practice, a system such as this is dependant on human feedback, but this system aims to eliminate human intervention. A solution was found when it was discovered that the dimmer consistently takes approximately 1s to vary its dim level from 0% to 100%. The program could then be set to simulate a button press for a percentage of a second resulting in a relative dim level. The dim level options were limited to 30% and 70% for simplicity and to eliminate redundancy.

5.1.3 User Interface

The user interface is designed as a local web server running on the Master Node, along side the Steward. When the Master Node is accessed over the network, via IP address and port number, by the browser of a remote device, the web server immediately provides the user interface web page. Scripting code on the web page allows the sensor and device information to be streamed directly from the Steward, where it is aggregated, to the user in real time. Changes in the home are reflected quickly on the web page, and control commands are carried out seamlessly. The user interface also allows customisation of device names.

Integration With Steward

This component of the user interface program manages the integration of the user into the Steward framework. This makes it possible for the user to monitor and control their home environment in real time.

Connection to the Steward is achieved in much the same way as with the Slave Nodes. The user interface registers with the Steward, and then uses a unique ID and authentication key for future connections.

Information is received from the Steward over a Websocket connection in real time. The Steward is designed to transmit all device status information at a set interval, but it also transmits status information of any particular devices that has changed state as and when it happens. This information is captured by the integration component of the user interface program and subsequently translated to the web page that is viewable by a user. At the same time, control commands are translated from the web page into JSON commands that are subsequently transmitted to the Steward for processing. Lastly, the parameters defined by a user for the creation of an intelligent rule (which is discussed later under Intelligence) are combined into a JSON instruction which is also transmitted to the Steward.

Code Language

The web server is created using a Node.js package called express. Express is simple and compact allowing small web servers to be created without worrying about the server management in the background. The web page scripting is written in Javascript, a widely used code language fully compatible with

Node.js and the express web server. Communication between the web page and the web server is achieved using Websocket communication.

Graphical User Interface Design

The GUI component of the user interface is the most important component with regard to the user. The GUI must be simple to use and understand, while also providing all the necessary information and control capability. It must also be accessible and viewable across all devices from desktop computers to smartphones. To make all this possible, the GUI is designed using a responsive layout and grouping sensor information and control functions by Slave Node.

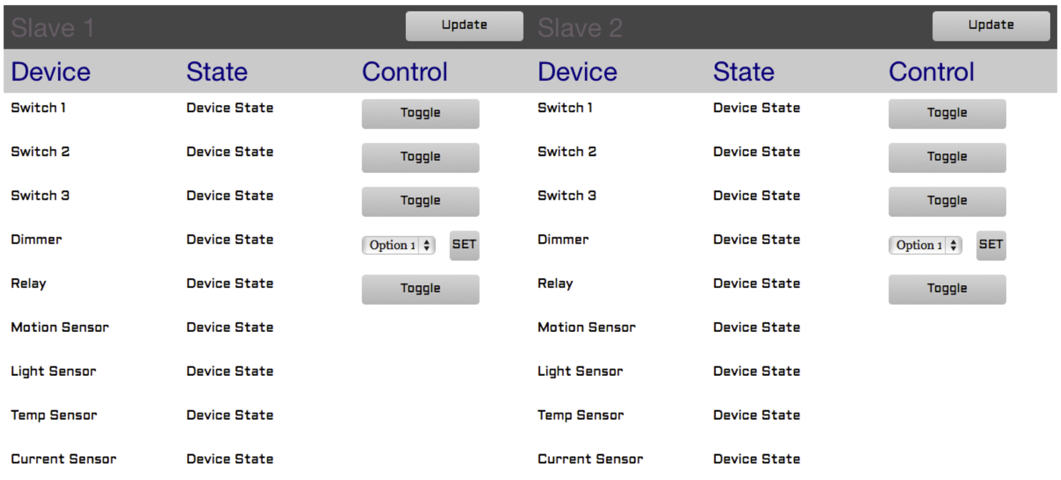
A responsive website layout uses CSS scripting to adjust the layout and appearance of the information on the page such that it best fits the screen of the device being used to view it. This makes it possible to optimise the webpage for use across all devices. The position and size of all the components on the page are carefully selected for each variation of device screen size resulting in a truly cross-platform graphical user interface.

By grouping sensor information and control commands by Slave Node, it is easier to understand the relationship between the information on the webpage and the physical home environment. It is most likely that one Slave Node is responsible for a certain area in the home, for example a bedroom. Grouping all the information on the web page that relates to that specific Slave Node makes it easier to understand what the information is related to. Also by including control components in the same place as the sensor information, users can watch sensor data reflect their control decisions better than if the control components were placed on a separate page. Figure 5.3 is a capture of the front page of the user interface showing the grouping of sensor information and control components by Slave Node.

Intelligence

Intelligence is an important component of Home Automation, and how this would be implemented was carefully considered. This proof-of-concept allows users to create intelligent control loops using sensor events to trigger control of their connected devices. This is implemented using a drop down box method allowing the user to create a rule by selecting the specific sensor event and device action in a "If this Then that" type fashion. Figure 5.3 is a capture of the rule creation page showing more clearly how this is implemented.

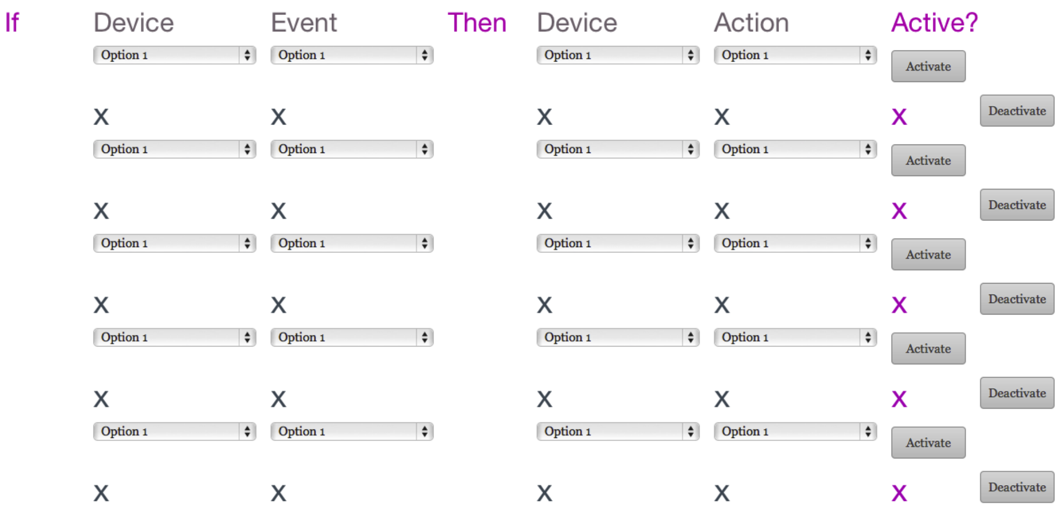
Drop down boxes are populated with all the available sensors and connected actuators. The first two drop down boxes in a line are for the selection of a sensor, and the sensor event to act as a trigger. for example the temperature sensor might be selected, and the event options would be hot or cold. Then the two drop down boxes on the right are for the selection of a device and a resulting action should the selected event come about. So in response to



Slave 1 Update Slave 2 Update

Device	State	Control	Device	State	Control
Switch 1	Device State	Toggle	Switch 1	Device State	Toggle
Switch 2	Device State	Toggle	Switch 2	Device State	Toggle
Switch 3	Device State	Toggle	Switch 3	Device State	Toggle
Dimmer	Device State	Option 1 SET	Dimmer	Device State	Option 1 SET
Relay	Device State	Toggle	Relay	Device State	Toggle
Motion Sensor	Device State		Motion Sensor	Device State	
Light Sensor	Device State		Light Sensor	Device State	
Temp Sensor	Device State		Temp Sensor	Device State	
Current Sensor	Device State		Current Sensor	Device State	

Figure 5.3: Two Slave Nodes with sensor information and control components grouped separately.



If	Device	Event	Then	Device	Action	Active?
	Option 1	Option 1		Option 1	Option 1	Activate
	X	X		X	X	X Deactivate
	Option 1	Option 1		Option 1	Option 1	Activate
	X	X		X	X	X Deactivate
	Option 1	Option 1		Option 1	Option 1	Activate
	X	X		X	X	X Deactivate
	Option 1	Option 1		Option 1	Option 1	Activate
	X	X		X	X	X Deactivate
	Option 1	Option 1		Option 1	Option 1	Activate
	X	X		X	X	X Deactivate

Figure 5.4: Rule creation by selecting sensor event and device action in an "If this Then that" fashion.

a hot event on the temperature sensor, a fan connected to one of the relays could be switched on. Logic rules such as OR and AND functions can also be implemented by combining two rules in a certain way. An OR condition can be created by creating two rules with the same actuator output, but different sensor events and an AND condition can be created with a logic manipulation using De Morgan's Theorem.

This proof-of-concept accommodates the creation of 5 separate rules that can be activated or deactivated at any time. Activating a rule communicates the rule parameters to the Steward. From then on, the Steward waits for the event to occur, and carries out the corresponding action. Deactivating a rule communicates to the Steward to delete the corresponding rule parameters and stop waiting for the event.

The last intelligent component designed into the user interface allows a user to customise device names. It can be seen in Figure 5.3 that standard device names, such as "Switch 1", are set. By allowing these device names to be customised, a user can more easily connect the virtual space of the user interface to the real environment of their home. If switch 1 is connected to the bedroom lamp, the name can be customised to read "Bedroom Lamp", forming a direct link. This user customised information is shared through to the rule creation component and is also retained, even in the event of a blackout.

5.2 Hardware Design

5.2.1 Beaglebone Black Setup

Operating System

The Beaglebone Black is shipped with the Angström distribution of Linux as well as its unique Cloud9 IDE, preinstalled on Node.js, including the BoneScript library. Angström is tailored for embedded devices and offers a smooth operating system with very little overhead. The Beaglebone also comes with a preinstalled web server that is accessed via a USB cable, and used as a "Getting Started" base. The Cloud9 IDE, a web server based IDE, is accessed in the same way and used to program the Beaglebone to interact with its many input/output pins. The IDE makes use of the BoneScript library, the link between the hardware and software, to make this possible.

Initial setup of the Beaglebone Black begins with updating and upgrading the preloaded software to the latest versions. In order to perform the update, the Beaglebone requires an Internet connection and an accurate system time, which complicates the process. Internet access is obtained by connecting the Beaglebone to a router with an Internet connection, but the system time will not be up to date due to the lack of a real time clock onboard the Beaglebone.

Network Time Protocol (NTP) is therefore used to synchronise the onboard system time over an internet connection which allows the Beaglebone to be updated. Although this works, the NTP update proves to be unreliable and unstable due to issues with the Angström distribution of Linux. For this reason, and motivation provided below and in Section 5.1 (where the main motivation is provided) later, Debian Wheezy is chosen as the Beaglebone Black Operating System.

The use of Debian Wheezy on both the Beaglebone Black Master and Slave components (or Nodes) provides a stable operating system allowing reliable NTP time synchronisation, internet connection and software installation and updating.

Network Connectivity

With a fully functioning Operating System capable of meeting the requirements of this project, the next phase of the Beaglebone setup is network connectivity and eliminating the need for hard wired USB connections. Since placement of the Master Node within the home is not important, it can be situated close to the home gateway (router) allowing it to be connected to the network via ethernet. Placement of the Slave Nodes however is very important, and so they will require wireless connectivity over WiFi. Ethernet connectivity on the Beaglebone is preconfigured to function without any changes. The Beaglebone is setup to receive an IP address via DHCP server, but can be setup to use a static IP.

WiFi on the Beaglebone requires a more complicated setup for two reasons. The Beaglebone does not come with an onboard WiFi chip, and as a result is not preconfigured to work correctly. Therefore a WiFi USB dongle is required for each Beaglebone slave, and each slave must be configured to control the WiFi connection. Also, it was discovered later that there are not many WiFi adapters that are compatible with the BBB.

The D-Link DWA-131 Wireless N Nano USB Adapter was acquired, and setup began. At this point in time, adding WiFi connectivity to a BBB had not been done much, and it eventually became clear that the list of compatible WiFi Adapter chipsets was quite limited. The D-Link DWA-131 was unfortunately not on this list, regardless of how many claims there were to writing a compatible driver. One of the compatible chipsets on the list is the Realtek RTL8192CU. The Edimax EW-7811Un Wireless Nano USB Adapter is based on this chipset, and therefore compatible with the Beaglebone Black.

Once the WiFi Adapter is installed and configured to connect to the home gateway, the next step is to ensure that the connection occurs automatically, and is stable. Unfortunately the Angström distribution is not stable in managing WiFi connections. It often fails to automatically connect to WiFi networks (due to the boot process) and thereafter refuses to reattempt any connection. These reasons lead to the use of Debian Wheezy as motivated above, and in

Section 5.1 later.

The use of the Edimax EW-7811Un in combination with Debian Wheezy on the Beaglebone Black provides a stable WiFi platform that automatically connects to the predefined network.

Input/Output

Interacting with the input and output functionality of the Beaglebone requires either the manipulation of system files using read and write commands, or the use of a library, such as BoneScript, which simplifies this process and allows the creation of an executable program. BoneScript is a Node.js library, based on Javascript, that makes it easy to control the Beaglebone I/O's using a language that is common in many development applications. However, the Beaglebone was released with BoneScript compiled and functioning on the Angström Linux distribution. The selection of Debian Wheezy as the Beaglebone operating system added complications as this OS is not fully compatible with BoneScript.

Without the ability to use BoneScript, I/O functionality on the beaglebone would need to be achieved another way (without resorting to file manipulation). Research began on a solution and it was soon found that a Node.js library called "onoff" [47] had been developed within the open source community. This library provides GPIO access on both the Beaglebone and Raspberry Pi, and because it is a Node.js library, is very easy to install and utilise. However, "onoff" is not capable of handling analog inputs, which is required when using analog sensors, and so another library must be found capable of performing this function. After some time, Debian Wheezy had undergone some changes and the latest update made it compatible with the BoneScript library. A better, more stable, version of BoneScript had also been created called OctalBoneScript [48], and so the combination of the latest Debian Wheezy release as well as OctalBoneScript allowed access to the analog pins on the Beaglebone.

Input/Output on the Beaglebone was made fully functional through the use of the latest Debian Wheezy release, "onoff" Node.js library for digital I/O access, and the OctalBoneScript Node.js library for analog I/O access.

5.2.2 Network Setup

The network setup for this proof-of-concept is fairly simple, but dependant on the pre-existence of a home WiFi router. Slave Nodes are equipped with WiFi capability allowing them to connect to the home WiFi router. The Master Node is connected to the WiFi gateway through its Ethernet port allowing Master and Slaves to communicate.

The Steward and user interface components are both located on the Master Node, but run concurrently as two separate programs. Communication between the two occurs locally. The Slave software runs on each individual

Slave Node, and communicates with the Steward via the identified network setup. Figure 4.19 in Chapter 4 clearly shows the location of the Steward, user interface, and Slave software on either the Master or Slave Node, and how communication takes place including what information is shared between which components. The green arrows in the figure represent communication between the Slave Nodes and the Steward. This information is not relevant to the user interface. The arrows in red represent information transmitted between the user interface, and the user's device. This information consists of the web page viewed by the user, all the status information seen by the user and the control commands sent by the user. The blue arrow represents communication between the user interface and the Steward. This communication occurs locally on the BBB Master Node and is not relevant to the Slave Nodes or the Users themselves. This information consists of control commands or status information that is relevant to the Steward alone.

5.2.3 Sensors

This section discusses the detailed design of the sensor circuitry used in this proof-of-concept. This involves the design of the circuitry to provide the correct supply voltage to each of the sensors, and ensure that the output voltages entering the input pins on the beaglebone are within the capable limits of the Beaglebone. There are two important voltage levels that must be considered at all times when designing the sensor circuits. The digital I/O pins are capable of a maximum voltage of $3V3$ while they register a logic HIGH at voltage levels above $2.6V$. The analog pins are capable of a maximum input of $1.8V$.

Finally, many adjustments are made to the sensor output voltages (in order to comply with the BBB maximum pin voltages), and so care must also be taken to ensure that the voltages that eventually enter the Beaglebone inputs are sensible values that relate correctly to the sensor information in terms of the data sheets and sensor operation.

Light Sensor

The light sensor used is the TSL257 High-Sensitivity Light-to-Voltage converter. This sensor used a photodiode and an operational amplifier to convert irradiant light into a voltage output representing the intensity of the irradiant light. The TSL257 requires an input voltage of between $2.7V$ and $5.5V$, and according to the relevant section of the data sheet, provided in Appendix A.2, has a maximum output voltage swing of $4.2V$ when supplied with $4.5V$ and connected to a $10k\Omega$ resistive load.

The output of the light sensor will be connected to an analog input pin on the Beaglebone (due to the analog nature of light level readings). According to the system reference manual for the Beaglebone Black, the maximum input voltage on on any of the analog pins is $1.8V$. The light sensor output voltage

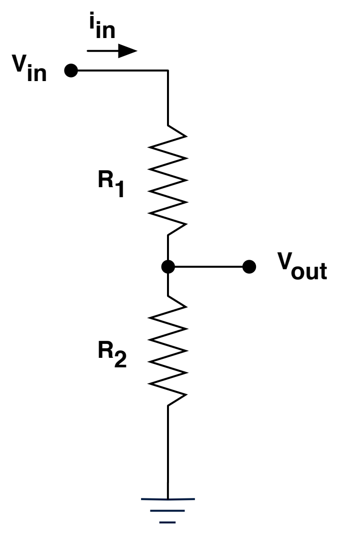


Figure 5.5: Common voltage divider circuit.

will therefore need to be stepped down such that no damage is caused to the Beaglebone. Also, the Beaglebone Supply voltage is $5V$ and the sensors will be connected to the same power supply. The supply voltage on all sensors is therefore also $5V$.

Important values are therefore:

$$V_{Supply} = 5\ V \tag{5.2.1}$$

$$V_{AINmax} = 1.8\ V \tag{5.2.2}$$

Based on the assumption that the light sensor output voltage swing will be proportional, with a supply voltage of $5V$ and a load resistance of $10k\Omega$, to the voltage swing stated in the data sheet, the maximum output voltage swing is calculated as:

$$V_{OS5V} = V_{OM}/V_{DD} * V_{Supply} \tag{5.2.3}$$

This calculation gives us the new maximum output swing of the light sensor when supplied by $5V$ as:

$$V_{OS5V} = 4.66\ V \tag{5.2.4}$$

Since this value is much larger than the maximum input allowed by the Beaglebone's analog pins, a voltage divider circuit must be used. This voltage divider circuit will drop the maximum output from the light sensor to an acceptable value by selecting correct R_1 and R_2 values. The circuit is depicted in Figure 5.5.

The maximum voltage swing is dependent on a load resistance of $10k\Omega$. We therefore know that R_1 and R_2 must add up to $10k\Omega$. Using the maximum

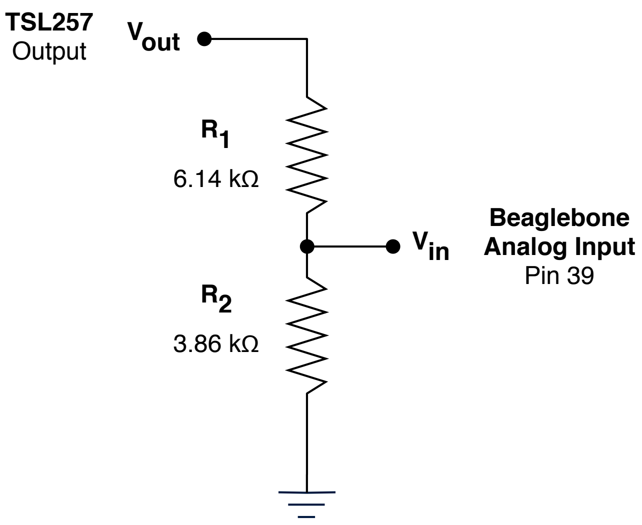


Figure 5.6: Circuit diagram depicting light sensor circuit showing resistor values, sensor output and Beaglebone input and pin number.

output swing and the load resistance value we can calculate the current flowing out of the light sensor output, given as I_in in the figure, as:

$$I_{in} = 4.66/10K = 0.466\ mA \tag{5.2.5}$$

Using this current, the maximum input voltage of the analog pins - $1.8V$, and Ohm's law Eq. 5.2.6, we can calculate the value of R_2 and thereafter the value of R_1.

$$V = I * R \tag{5.2.6}$$

$$R_2 = 1.8/0.466\ m = 3.86\ k\Omega \tag{5.2.7}$$

$$R_1 = 6.14\ k\Omega \tag{5.2.8}$$

In maximum direct sunlight, this light sensor circuit will output a value of $1.8V$, the maximum allowed input value of the analog pins on the Beaglebone Black. These calculations have been rounded to the second decimal place, but if accurate measurement of light is required, care should be taken to complete the calculations to further decimal positions and also to use low tolerance resistors. In this project however, accurate light readings are not necessary hence the calculation rounding and the use of 5 percent tolerance resistors. The final light sensor circuit is shown in Figure 5.6, depicting the connection of the sensor output, resistor values and connection to the Beaglebone input and pin number. A table detailing the pin numbers and functions of the relevant Beaglebone expansion header is provided in Appendix A.1.

Current Sensor

This proof-of-concept makes use of a LEM LTS 25-NP Current Transducer capable of measuring DC and AC currents of up to 80 Amps. This current transducer converts the measured current into an output voltage that oscillates around $2.5V$, between $0.5V$ and $4.5V$. A graph of the relationship between output voltage and primary current is given in Appendix A.4. The equation below describes this relationship in mathematical terms and is extracted from the relevant section of the LTS 25-NP data sheet which is also provided in Appendix A.4.

$$V_{out} = 2.5 \pm (0.625 * \frac{I_P}{I_{PN}})V \tag{5.2.9}$$

The current sensor requires a $5V$ input, which has already been accounted for, however its output voltage exceeds the limits of the BBB's analog input pins (as was the case with the light sensor). Another voltage divider circuit is therefore required. Also, due to the nature of current measurement, the resistors in this circuit will need to be of a high tolerance in order to calculate accurate current readings. Resistors with a 1 percent tolerance are used, and the resistance values of R_1 and R_2 are given below.

$$R_1 = 1.2k\Omega \tag{5.2.10}$$

$$R_2 = 820\Omega \tag{5.2.11}$$

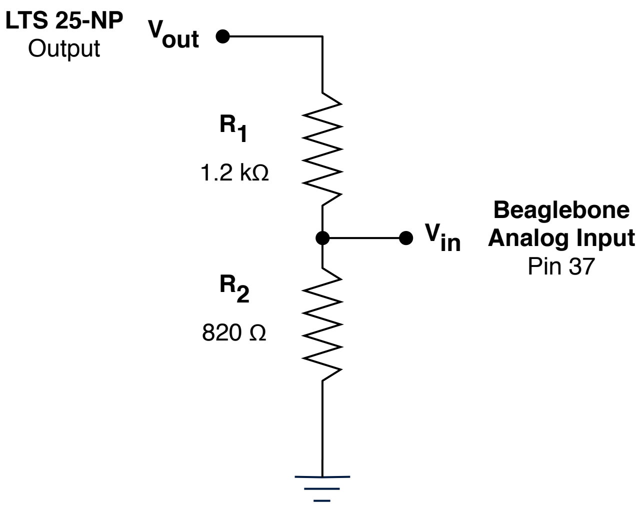


Figure 5.7: Circuit diagram depicting current sensor circuit showing resistor values, sensor output and Beaglebone input and pin number.

This setup will provide the Beaglebone with an input voltage that swings between $0V$ and $1.8V$ when an alternating current is being measured. In order to calculate power consumption, Equation 5.2.9 is used to calculate the measured current, which is in turn used to calculate power consumption. This is discussed in more detail in Section 5.1.

The final current sensor circuit is shown in Figure 5.7 depicting the connection of the sensor output to the Beaglebone analog input as well as the calculated voltage divider values.

Temperature Sensor

The TMP36 temperature sensor outputs a voltage based on measured temperature, and exhibits a linear relationship between temperature and output voltage in the range between $-40C$ and $125C$. As shown in the relevant section of the data sheet, provided in Appendix A.5, the TMP36 has a scale factor of $10mV/C$ and delivers an output voltage of $750mV$ at 25°C (room temperature). This sensor requires a supply voltage of between $2.7V$ and $5.5V$. The supply has already been set at $5V$, and so no adjustment of the supply voltage is required.

Due to the varying nature of temperature measurement, the TMP36 sensor output will be connected to an analog input pin on the Beaglebone Black. It has already been identified that the maximum permitted voltage on these pins is $1.8V$, however this particular temperature sensor has a maximum output voltage of $1.75V$ at the upper limit of it linear range. No consideration will be made for cases in which the measured temperature is above 125°C as this case is highly unlikely in the normal home environment.

Discussion of the interpretation of the TMP36 output voltage and calculations is discussed in the software design phase later in Appendix 5.1. No circuit diagram is provided due to simplicity, but the temperature sensor output is connected to analog pin 40 on the Beaglebone for which information can be found in Appendix A.1.

Motion Sensor

The motion sensor used in this proof-of-concept is a Parallax [49] passive infrared sensor. Passive infrared sensors detect changes in an environment by measuring the infrared light radiating from objects within its field of view. This particular PIR is capable of detecting movement up to 9 meters away within a 90 degree field of view. It also has an external jumper allowing the sensitivity of the sensor to be set lower, approximately 4 meter detection range, in cases of limited current supply.

The Parallax PIR sensor has three pins for supply, ground and output. It requires a supply voltage of between $3V$ and $6V$, and the output is a binary value of LOW or HIGH. A logic LOW is represented by a $0V$ output and in the case of a logic HIGH, the supply voltage is placed on the output - the BBB digital input pins will be used to read the PIR output value. The relevant section of the data sheet and a simple circuit is provided in Appendix A.3.

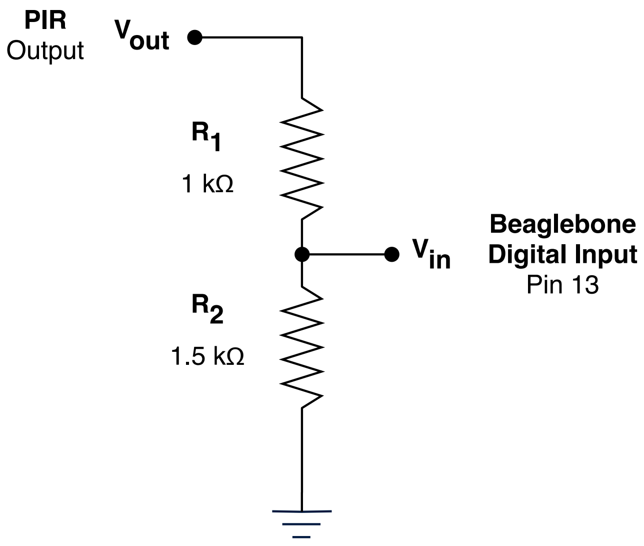


Figure 5.8: Circuit diagram depicting motion sensor circuit showing resistor values, sensor output and Beaglebone input and pin number.

As stated earlier, a supply voltage of $5V$ is used across all sensors. The output voltage on the PIR, in the case of a logic HIGH output, is equal to the supply voltage which is $5V$.. Therefore a voltage divider circuit, such as the circuit used in Figure 5.5, must again be used to step the PIR output voltage down to below the maximum BBB input voltage of $3V3$ but above the voltage required to register a logic HIGH, as mentioned in the beginning of the section. To include a margin of safety, the maximum output voltage is stepped down to 3V using R_1 and R_2 resistors with the following values:

$$R_1 = 1k\Omega \tag{5.2.12}$$

$$R_2 = 1.5k\Omega \tag{5.2.13}$$

Figure 5.8 depicts the circuit connecting the output of the PIR to the input on the Beaglebone, as well as the resistor values. Refer to Appendix A.1 for details on the Beaglebone pin used in this circuit.

5.2.4 Actuators

This section discusses the detailed design of the actuator circuitry used in this proof-of-concept. This design phase involves the correct connection of the Quikswitch remote PCB and the onboard high-power relay. The high power relay is directly driven by the BBB output pins whereas the Quikswitch remote is a separate PCB with external pins that are connected to the BBB for control.

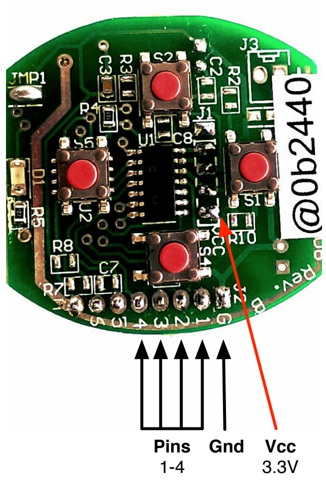


Figure 5.9: Photograph of the Quikswitch remote PCB showing the external pins.

Quikswitch Remote

The Quikswitch remote is a 4 button keyring remote that uses RF communication to control the switch receivers. The remote PCB allows for external connections to be made to the buttons to allow for electronic control. The remote is powered by a $3V$ battery, which is close enough to the $3.3V$ regulator on the Beaglebone allowing the remote to be powered, through the V_{cc} pin, by the Beaglebone. The remote requires a maximum current of $2.544mA$ (which occurs during a press of button S5 on the remote) and the BBB is capable of providing a maximum of $4mA$ from the $3V3$ regulator (including the digital output pins).

When a button is pressed, the associated pin on the remote micro-controller is grounded. This initiates the transmission of an RF command that will activate or deactivate a switch on the receiver. A receiver and remote button are linked by pressing a button on the receiver module so that its LED blinks, and thereafter pressing the button on the remote. Up to four independent receivers can be linked with one remote, as there are only four buttons, however additional receivers can be linked to one button but independent control will not be possible. A link is erased by holding the button on the receiver module for 10 seconds.

Figure 5.9 is a photograph of the remote PCB showing the external pins that can be used to control the remote with a micro-controller. Pins 1 to 4 are connected to each of the 4 buttons. While $3.3V$, from the BBB output pin, is applied to a remote pin the button is technically in the unpressed state. A button press is emulated by grounding the associated remote pin resulting in a short circuit across the remote button straight to ground (an identical result to physically pressing the button).

Importantly, the pins on the remote cannot be grounded by simply drop-

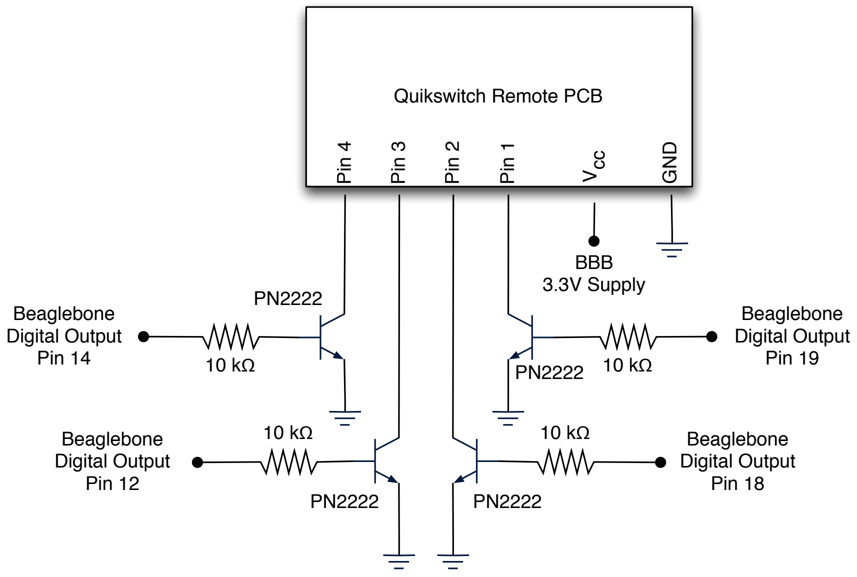


Figure 5.10: Circuit diagram of the Quikswitch remote connected to the Beaglebone using transistors for switching.

ping the BBB output pin voltage to $0V$. The remote pin must actually be short circuited to ground (both the remote and the BBB are connected to a common ground). To do this, transistors are used which and controlled by the BBB digital output pins. When $3V3$ is applied to the transistor, the connection is closed (collector and emitter are connected), and a pin on the remote is connected directly to the common ground. While there is $0V$ applied to the transistor, the connection between a remote pin and ground is broken. The circuit diagram with all four buttons, the transistors and Beaglebone connections is shown in Figure 5.10. The ground represented in the figure is the common ground shared by both the BBB and the remote module.

PN2222 transistors are used as they are standard and readily available. $10k\Omega$ resistors are used also as it is a common resistor to use on the base of a transistor but it also limits the current drawn from the Beaglebone's digital output pins to $0.27mA$. The maximum output current on the BBB GPIO's is 4mA. If the BBB digital output is $3.3V$, and the minimum saturation voltage (minimum saturation voltage is used in order to account for the worst case scenario) of the PN2222 transistor is $0.6V$, as seen in the relevant section of the data sheet given in Appendix A.6, then the current drawn from the BBB is calculated as follows.

$$I_{out} = \frac{V_{out} - V_{BE(sat)}}{R} \tag{5.2.14}$$

$$I_{out} = \frac{3.3 - 0.6}{10000} \tag{5.2.15}$$

$$I_{out} = 0.27mA \tag{5.2.16}$$

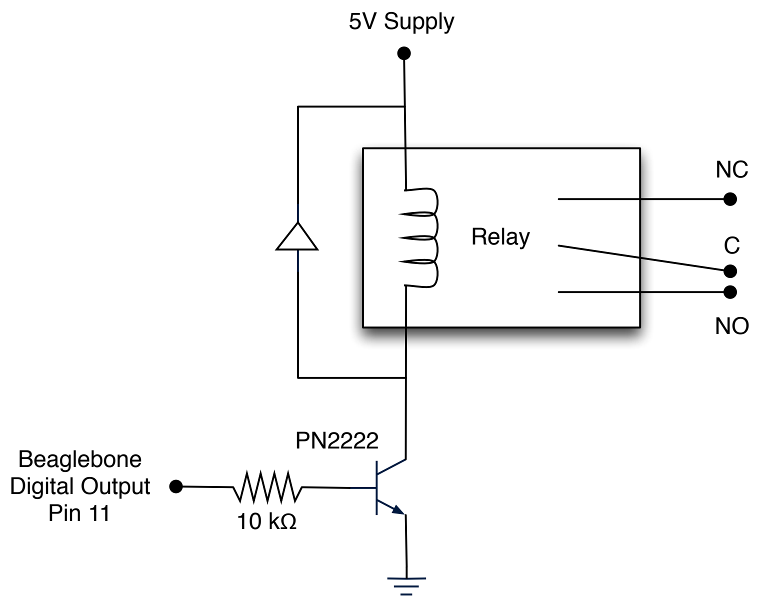


Figure 5.11: Circuit diagram of the high power relay and connection to the BBB digital output pin.

With this circuit, the BBB digital output pins can be used to control the Quikswitch remote and therefore gain control over wireless RF relays. Operation of a standard switching relay and a dimmer relay differs slightly, and is handled by the software which is discussed later in Section 5.1.

High Power Relay

This proof-of-concept includes one $10A$ relay that switches on $5VDC$. Relays work by energising a magnetic coil which switches a physical connection within the unit. This relay is included so that devices that draw large currents can be integrated into the HA system.

As already mentioned, there is a $5V$ supply voltage to all sensors and actuators, so no supply voltage manipulation is required. The relay coil is connected to the supply voltage on one side, and to ground on the other again using a transistor with a $10k\Omega$ resistor to perform the switching. Figure 5.11 shows the circuit diagram that will connect the relay to a BBB digital output pin.

5.2.5 Printed Circuit Board

Printed circuit boards are used to mechanically support and connect electrical components using conductive tracks etched from copper sheets and laminated onto a non-conductive substrate. PCB's vary from single sided (one copper

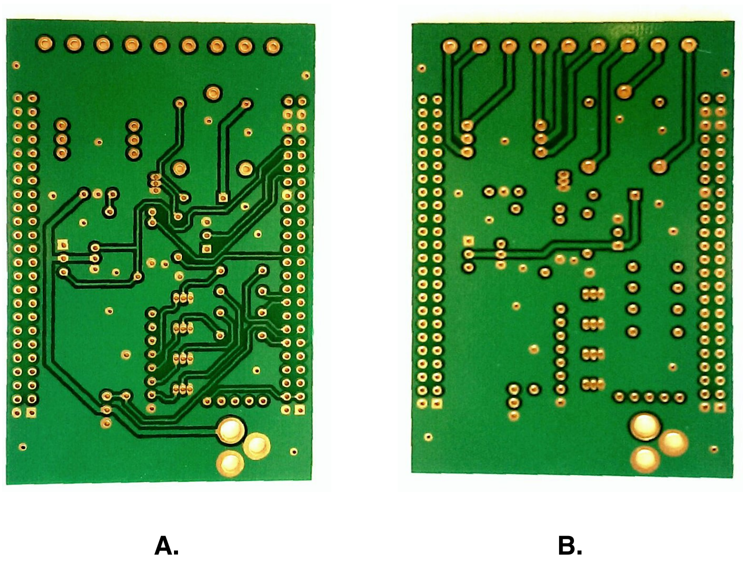


Figure 5.12: A. Top side of the printed circuit board. B. Bottom side of the printed circuit board.

layer), to double sided (a copper layer on both sides) and multi-layer, where copper layers are also embedded within the substrate.

A PCB is custom designed for use in this proof-of-concept. The PCB aids in keeping the final design compact, positioning sensors on a stable platform and providing easy to use connectors for the current sensor and high power relay. A double layer PCB design is selected as it finds the right balance between complexity and size. It allows the components to be placed close together without problems arising in track routing, while keeping cost of manufacturing low. The manufactured PCB is shown in Figure 5.12 and Figure 5.13 shows the PCB with the components attached and identified.

The components numbered in Figure 5.13 are as follows:

1. High power relay.
2. Temperature sensor.
3. 5V power connector.
4. PIR motion sensor.
5. Light sensor.
6. Current sensor.
7. Current sensor and high power relay connectors.
8. RF relay remote circuit board.

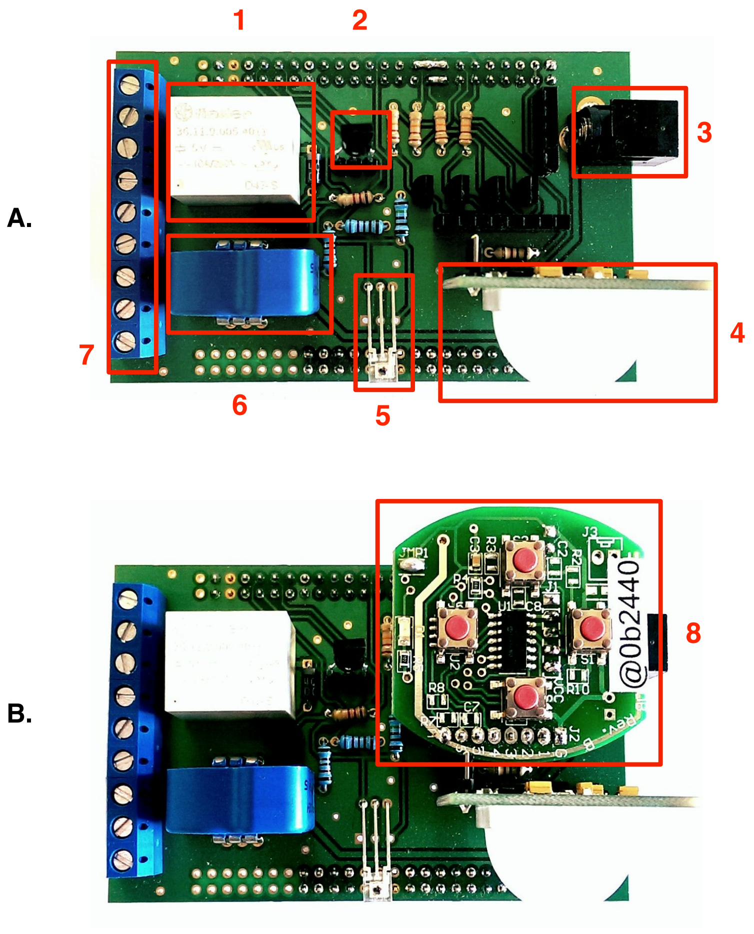


Figure 5.13: A. PCB with components attached. B. PCB with components attached as well as RF relay remote connected.

Chapter 6

Conclusion

6.1 Proof-of-Concept Analysis

The proof-of-concept is analysed and its performance is quantified. Satisfaction of the User Requirements Specification is an important factor and also the degree to which the proof-of-concept addresses the challenges summarised in Section 1.2 will determine how successful the design of this proof-of-concept has been. A few other factors are also analysed in order to grade the success of this proof-of-concept namely: security, network usage and response time.

6.1.1 URS Satisfaction

A summary of the system components that satisfy the User Requirements Specification is given in Table 6.1.

6.1.2 HA Challenges

Following the satisfaction of the User Requirements Specification, an analysis can be done on the degree to which the proof-of-concept addresses the challenges facing Home Automation. To begin with, existing Home Automation systems are complicated and expensive. These systems require complex design processes, professional installation and continued professional maintenance. There is no room for expansion or reconfiguration without requiring a large amount of work to alter the system. The proof-of-concept that has been designed addresses this challenge with a low cost design, open source platform and seamless capability for expansion and reconfiguration the the use of a modular architecture. If in the future there are cases where installation and configuration of this system is be challenging to the user, the cost of professional help will be low due to the simplicity of the system.

Standardisation is one of the greatest challenges to Home Automation that has caused the development of a multitude of HA systems that are incompatible with one another. This proof-of-concept addresses this challenge through

URS1	**Integration**	Slave Nodes carry out device integration.
	Coordination	Master Node is a capable coordinating component.
	Intelligence	Intelligent rule creation made possible.
	User Interface	User interface accessible across all platforms.
	Communication	Radio frequency, WiFi and local network communication is utilised.
URS2 - Total Integration		Slave Nodes are capable of integrating all household devices.
URS3 - Non-Intrusive Installation		Use of WiFi and Radio Frequency allows for seamless installation.
URS4 - Intuitive and Simple		Plug-and-Play design with minimal configuration requirements.
URS5 - Open Source		Use of Beaglebone Black, Linux and Node.js ensures fully open source system.
URS6 - Successful Proof-of-Concept		Proof-of-concept is capable of demonstrating all the features that were designed for.
URS7 - Modular Architecture		Use of the Steward, and intelligent Slave Nodes resulted in a modular architecture.
URS8 - All-Access User Interface.		User interface designed as a web-server which is accessible across all platforms.
URS9 - Processing Power		The Beaglebone Black has a more powerfull CPU than the majority of micro-controllers available.
URS10 - Distributed Intelligence		Intelligence has been distributed across the user interface as well as the Master and Slave Nodes.

Table 6.1: Summary of the elements that satisfy the User Requirements Specification.

the use of open source, linux-based hardware and open source software. Instead of creating a new standard and attempting to force conformity, the use of an already widely available and open platform helps to promote a standard in Home Automation. The use of open source platforms also promotes the growth and development of HA within the open source community. This open ended design philosophy will result in faster growth and expansion of the field than if a competitor-based design philosophy were adopted.

Lastly, this proof-of-concept distributes the system intelligence across all components of the system. This addresses the intelligence challenge facing HA which is limiting the capabilities of HA systems. With distributed intelligence, this proof-of-concept is capable of integrating household devices that have seldom been included in HA systems as well as providing the user with an intuitive and user friendly experience.

6.1.3 Security

Remote access to the system is achieved by forwarding a port on the WiFi router to direct traffic from the web server to the external incoming connection. The routers external IP address is typed into a browser with Internet connectivity and the router handles the rest. Security is achieved by adding user authentication to the web server. The user is prompted to proved a username and password in order to gain access to the system. Without user authentication, the system would be accessible by anyone connected to the internet so long as they have prior knowledge of the WiFi routers external IP address.

This proof-of-concept design has not implemented user authentication in order to simplify the testing environment. The addition of authentication is simple, and is recommended as future work for the system.

6.1.4 Network Usage

Communication between the Master and Slave Nodes is the largest consumer of network bandwidth. Sensor status information, or control commands are transmitted about once every 10 seconds. It is done this way to ensure that all system information is relevant in real time, and that control commands initiated by the user are carried out immediately.

When a user accesses the system via a browser on their mobile device, a TCP/IP communication channel is opened and information is continually transmitted between the Master Node (where the user interface web server is located) and the user's browser. While the user accesses the system locally (over the local WiFi network), this will have no effect on the Internet usage and cost. However, if the user accesses the system from a remote location, this traffic would be transmitted over their Internet connection and be applicable to data usage costs. In this scenario, the system consumes $0.033MB$ of data per second. This equates to 1.98MB per minute. These values were captured while the system was running with two Slave Nodes connected, and with no user input (i.e. no control commands were initiated). A second test was run with the addition of a moderate amount of user input (a control command was sent roughly once every 30 seconds). This resulted in data transfer of roughly 3.5MB per minute, and increase of more than 1.5MB per minute. With the addition of more Slave Nodes generating more sensor information, this value will rise to above 7MB per minute.

Internet data in South Africa is quite expensive, costing anything between R6.33 and R190.00 per MB [50]. If a user purchases the cheapest option at R6.33 per MB with a 3GB data cap, they might deplete their available data by remotely accessing their system for just 15 minutes per day.

6.1.5 Response Time

A Home Automation system that is slow to communicate status information to the user, or slow to carry out control commands sent by the user will be more of a nuisance than an improvement on the home environment. This proof-of-concept is therefore designed to transmit status information directly to the user interface in real time. The delay between an event such as movement, and this information becoming available on the user interface is no more than 5 seconds (at its worst). Also, transmission of control commands occurs immediately after a user takes action and the actuation effects are witnessed no more than 2 seconds after the action. Closed loop performance of the system also occurs in real time with action being take quickly after sensor events are triggered.

6.2 Future Work

In this section we analyse the flaws in the current proof-of-concept design, and propose future work to rectify the flaws.

6.2.1 Flaws In Proof-Of-Concept System

Stability - The system relies on a few aspects that are crucial to continual functioning of the system. The WiFi router must remain on and available to host the system communications and the system requires constant power. So in the event the WiFi router fails, or power to the system is cut a reboot will be necessary.

Most importantly however, continuous functioning of the Slave Nodes requires an uninterrupted connection to the Master Node. In the event this connection is broken, the Slave software program halts and it is not capable of reconnecting to the Master autonomously. The program, or the Slave Node must be restarted. Also, in the event that the program runs into an error (this can occur during the reading of analog inputs, or when memory becomes overloaded), it is not capable of restarting itself and will require user intervention.

Adaptive User Interface - The user interface provides all the functions necessary for the system in current state. Development of new Slave Nodes with different capabilities can be integrated into the system and will function, however, while the user interface is only a proof-of-concept, it is not equipped to receive information or send commands to Slave Nodes other than those within this proof-of-concept.

Moving forward into the development of Slave Nodes that integrate many other devices into added to the system will also required the development of a far more intelligent and adaptive user interface.

Integration - The Slave Nodes are responsible for the integration of devices into the HA system. Currently, a limited number of devices can be integrated into the system. While this serves as a satisfactory proof-of-concept, it is difficult to quantify real world performance of the system within integration of more complex devices such as those that require communication using Infrared.

Network Usage and Security - While a user is connected to the system over a local connection, data usage is not of a large concern. However, due to a continuous connection and transmission of data, the system inherently creates large amounts of network traffic. This can result in very high Internet costs to the user if the system is accessed remotely for long periods of time.

As mentioned earlier, the proof-of-concept has not implemented user authentication. This is recommended as future work and can be achieved through the use of a Node.js package called Passport [51].

6.2.2 Proposed Future Work

Stability - Additional work on the Slave software program is required to cater for errors or breaks in connection. Bugs need to be worked out of the code to allow for autonomous reconnection to the Master Node or restart of the program. An unstable Home Automation system will most quickly lead to users uninstalling the system.

Adaptive User interface - A user interface that is capable of adapting to the types of devices and sensors that are connecting to the system must be developed. This means that no matter how many Slave Nodes are connected, or the devices they are connected to, the user interface must be capable of adapting the way in which that information is conveyed to the user.

The user interface in this proof-of-concept is coded to know what information to expect sand what control commands it must provide. In the future, the user interface should adapt its layout according to the types of devices connected and types of information being communicated.

Integration - The addition of Infrared control will make it possible to integrate a host of devices into the system. This will provide a much better understanding of the performance of the proof-of-concept and allow it to be better tested in real world scenarios. Infrared control should be capable of transmitting commands as well as learning commands, preventing the need to add all possible commands to the system in an attempt to predict the devices a user might want to control.

6.3 Conclusion

This manuscript discussed the design of a Home Automation solution with the potential for seamless integration and vast expansion potential. The manuscript followed a standard design process by starting with a problem definition and proposing a solution. The proposed solution makes use of open source hardware and software and adopts a modular design architecture. It also employs the Steward, a software package designed to connect devices and allow communication between them. Lastly, the system employs a variety of sensors and actuators integrated into the system and served to the user via a cross platform web-server.

An extensive study of the related literature revealed the existing work in the field and also the challenges facing Home Automation. Some of the main challenges included the prevention of the adoption of HA by the consumer due to cost and complexity. The lack of a hardware and software standard added to this problem, and also limited the growth and expansion capabilities of existing HA systems. Finally, distribution of system intelligence had not been extensively explored and was resulting in the exclusion of many household devices from HA system. An analysis of the challenges facing HA lead to the definition of a User Requirements Specification. The URS is intended to guide the development of the proof-of-concept in a direction that would satisfy the defined goals in order to address the challenges and problems facing HA.

In Chapter 3 the URS is translated into system requirements. The SRS identifies design requirements on a system level to address each of the user requirements. The SRS is used during the development of a concept system to aid in the selection of hardware and software components.

Thereafter, the concept development phase was entered. This involved the selection of hardware and software components capable of satisfying the SRS and therefore the URS. Main components to be considered were the microcontroller, actuator and sensor components. On the software side, the main considerations were that of the Master and Slave Node operating systems, and code language of the programs.

Finally the detailed design in Chapter 5 discussed the electronic circuitry used to integrate the actuators and sensors, the software used to manage and control the devices, and the software responsible for communicating all the relevant info to the user. One of the largest design task was the integration with the Steward software package. Device information and control commands all passes through the Steward, and all information and control capability is provided to the user through the Steward. The design phase included the development of an intuitive and intelligent user interface.

The resulting proof-of-concept system is a Home Automation system with the potential to integrate all household devices, and the ability to autonomously control devices based on closed "event/trigger" loops created by a user. The design is also capable of seamless integration into any home, and vast expansion due to its open source, modular design. Installation and reconfiguration of the system is simple and cost effective.

Appendices

Appendix A

Data Sheets

A.1 Beaglebone P9 Header Pin Descriptions

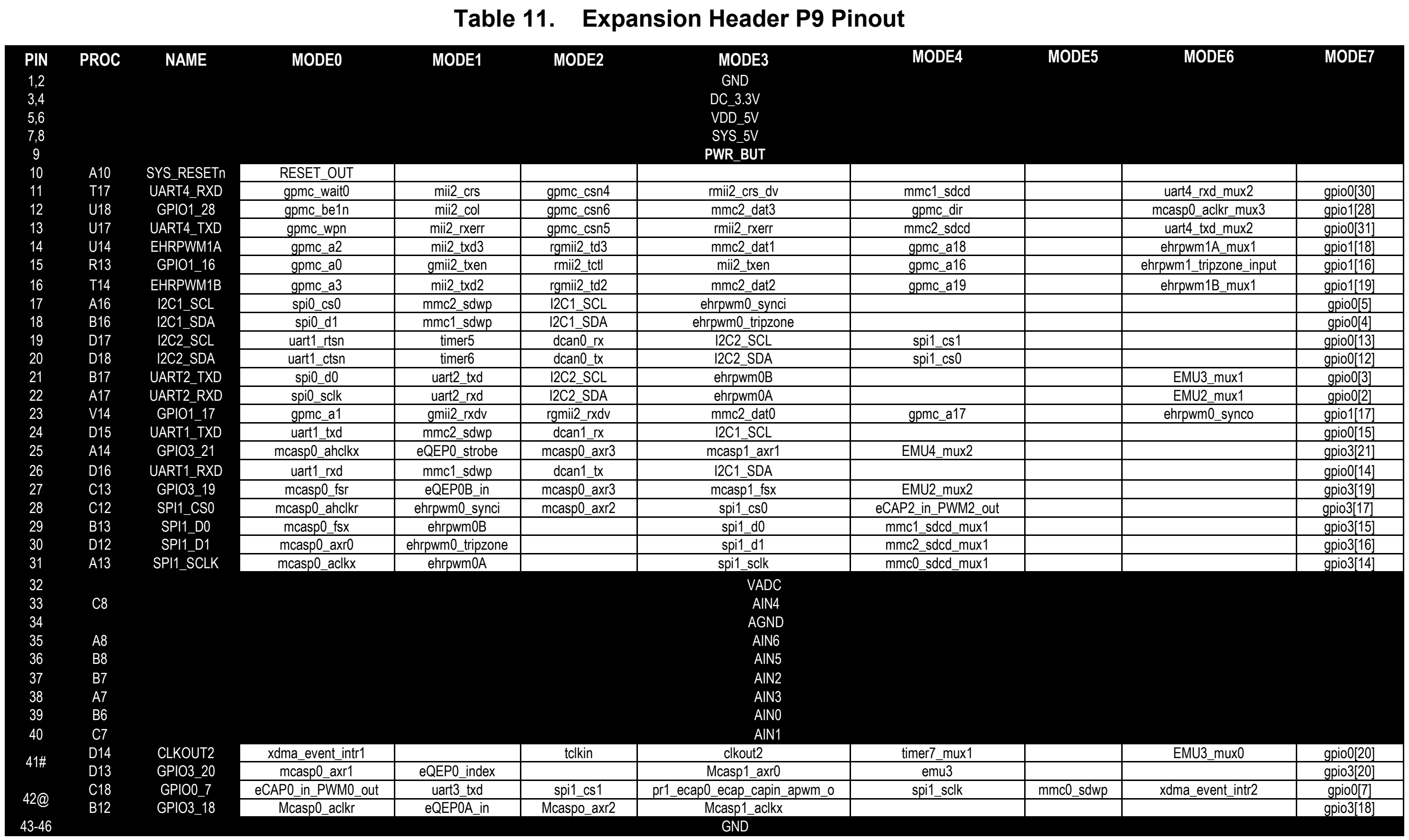


Table 11. Expansion Header P9 Pinout

PIN	PROC	NAME	MODE0	MODE1	MODE2	MODE3	MODE4	MODE5	MODE6	MODE7
1,2						GND				
3,4						DC_3.3V				
5,6						VDD_5V				
7,8						SYS_5V				
9						**PWR_BUT**				
10	A10	SYS_RESETn	RESET_OUT							
11	T17	UART4_RXD	gpmc_wait0	mii2_crs	gpmc_csn4	rmii2_crs_dv	mmc1_sdcd		uart4_rxd_mux2	gpio0[30]
12	U18	GPIO1_28	gpmc_be1n	mii2_col	gpmc_csn6	mmc2_dat3	gpmc_dir		mcasp0_aclkr_mux3	gpio1[28]
13	U17	UART4_TXD	gpmc_wpn	mii2_rxerr	gpmc_csn5	rmii2_rxerr	mmc2_sdcd		uart4_txd_mux2	gpio0[31]
14	U14	EHRPWM1A	gpmc_a2	mii2_txd3	rgmii2_td3	mmc2_dat1	gpmc_a18		ehrpwm1A_mux1	gpio1[18]
15	R13	GPIO1_16	gpmc_a0	gmii2_txen	rmii2_tctl	mii2_txen	gpmc_a16		ehrpwm1_tripzone_input	gpio1[16]
16	T14	EHRPWM1B	gpmc_a3	mii2_txd2	rgmii2_td2	mmc2_dat2	gpmc_a19		ehrpwm1B_mux1	gpio1[19]
17	A16	I2C1_SCL	spi0_cs0	mmc2_sdwp	I2C1_SCL	ehrpwm0_synci				gpio0[5]
18	B16	I2C1_SDA	spi0_d1	mmc1_sdwp	I2C1_SDA	ehrpwm0_tripzone				gpio0[4]
19	D17	I2C2_SCL	uart1_rtsn	timer5	dcan0_rx	I2C2_SCL	spi1_cs1			gpio0[13]
20	D18	I2C2_SDA	uart1_ctsn	timer6	dcan0_tx	I2C2_SDA	spi1_cs0			gpio0[12]
21	B17	UART2_TXD	spi0_d0	uart2_txd	I2C2_SCL	ehrpwm0B			EMU3_mux1	gpio0[3]
22	A17	UART2_RXD	spi0_sclk	uart2_rxd	I2C2_SDA	ehrpwm0A			EMU2_mux1	gpio0[2]
23	V14	GPIO1_17	gpmc_a1	gmii2_rxdv	rgmii2_rxdv	mmc2_dat0	gpmc_a17		ehrpwm0_synco	gpio1[17]
24	D15	UART1_TXD	uart1_txd	mmc2_sdwp	dcan1_rx	I2C1_SCL				gpio0[15]
25	A14	GPIO3_21	mcasp0_ahclkx	eQEP0_strobe	mcasp0_axr3	mcasp1_axr1	EMU4_mux2			gpio3[21]
26	D16	UART1_RXD	uart1_rxd	mmc1_sdwp	dcan1_tx	I2C1_SDA				gpio0[14]
27	C13	GPIO3_19	mcasp0_fsr	eQEP0B_in	mcasp0_axr3	mcasp1_fsx	EMU2_mux2			gpio3[19]
28	C12	SPI1_CS0	mcasp0_ahclkr	ehrpwm0_synci	mcasp0_axr2	spi1_cs0	eCAP2_in_PWM2_out			gpio3[17]
29	B13	SPI1_D0	mcasp0_fsx	ehrpwm0B		spi1_d0	mmc1_sdcd_mux1			gpio3[15]
30	D12	SPI1_D1	mcasp0_axr0	ehrpwm0_tripzone		spi1_d1	mmc2_sdcd_mux1			gpio3[16]
31	A13	SPI1_SCLK	mcasp0_aclkx	ehrpwm0A		spi1_sclk	mmc0_sdcd_mux1			gpio3[14]
32						VADC				
33	C8					AIN4				
34						AGND				
35	A8					AIN6				
36	B8					AIN5				
37	B7					AIN2				
38	A7					AIN3				
39	B6					AIN0				
40	C7					AIN1				
41#	D14	CLKOUT2	xdma_event_intr1		tclkin	clkout2	timer7_mux1		EMU3_mux0	gpio0[20]
	D13	GPIO3_20	mcasp0_axr1	eQEP0_index		Mcasp1_axr0	emu3			gpio3[20]
42@	C18	GPIO0_7	eCAP0_in_PWM0_out	uart3_txd	spi1_cs1	pr1_ecap0_ecap_capin_apwm_o	spi1_sclk	mmc0_sdwp	xdma_event_intr2	gpio0[7]
	B12	GPIO3_18	Mcasp0_aclkr	eQEP0A_in	Mcaspo_axr2	Mcasp1_aclkx				gpio3[18]
43-46						GND				

Table A.1: Beaglebone P9 expansion header pin descriptions and numbers.

A.2 TSL257 Light Sensor

Electrical Characteristics at V_{DD} = 5 V, T_A = 25°C, λ_p = 470 nm, R_L = 10 kΩ (unless otherwise noted) (see Notes 2 and 3)

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_D	Dark voltage	$E_e = 0$	0		15	mV
V_{OM}	Maximum output voltage swing	V_{DD} = 4.5 V, No Load		4.49		V
		V_{DD} = 4.5 V, R_L = 10 kΩ	4	4.2		V
V_O	Output voltage	E_e = 1.54 µW/cm^2, λ_p = 470 nm, Note 5	1.6	2	2.4	V
α_{VD}	Temperature coefficient of dark voltage (V_D)	T_A = 0°C to 70°C		–15		µV/°C
N_e	Irradiance responsivity	λ_p = 428 nm, see Notes 4 and 8		1.18		V/(µW/cm^2)
		λ_p = 470 nm, see Notes 5 and 8		1.30		V/(µW/cm^2)
		λ_p = 565 nm, see Notes 6 and 8		1.58		V/(µW/cm^2)
		λ_p = 645 nm, see Notes 7 and 8		1.68		V/(µW/cm^2)
PSRR	Power supply rejection ratio	f_{ac} = 100 Hz, see Note 9		55		dB
		f_{ac} = 1 kHz, see Note 9		35		dB
I_{DD}	Supply current	E_e = 1.54 µW/cm^2, λ_p = 470 nm, Note 5		1.9	3.5	mA

NOTES:
2. Measured with R_L = 10 kΩ between output and ground.
3. Optical measurements are made using small-angle incident radiation from a light-emitting diode (LED) optical source.
4. The input irradiance is supplied by a GaN/SiC light-emitting diode with the following characteristics: peak wavelength λ_p = 428 nm, spectral halfwidth $\Delta\lambda\frac{1}{2}$ = 65 nm.
5. The input irradiance is supplied by an InGaN light-emitting diode with the following characteristics: peak wavelength λ_p = 470 nm, spectral halfwidth $\Delta\lambda\frac{1}{2}$ = 35 nm.
6. The input irradiance is supplied by a GaP light-emitting diode with the following characteristics: peak wavelength λ_p = 565 nm, spectral halfwidth $\Delta\lambda\frac{1}{2}$ = 28 nm.
7. The input irradiance is supplied by an AlGaAs light-emitting diode with the following characteristics: peak wavelength λ_p = 645 nm, spectral halfwidth $\Delta\lambda\frac{1}{2}$ = 25 nm.
8. Irradiance responsivity is characterized over the range V_O = 0.1 V to 4.5 V. The best-fit straight line of Output Voltage V_O versus Irradiance E_e over this range will typically have a positive extrapolated V_O value for E_e = 0.
9. Power supply rejection ratio PSRR is defined as 20 log $(\Delta V_{DD}(f)/\Delta V_O(f))$ with $V_{DD}(f = 0)$ = 5 V and $V_O(f = 0)$ = 2 V.

Table A.2: Electrical characteristics of the TSL257 Light-to-Voltage converter.

A.3 Parallax PIR Motion Detector

Pin Definitions and Ratings

Pin	Name	Type	Function
1	GND	G	Ground: 0 V
2	Vcc	P	Supply Voltage: 3 to 6 VDC
3	OUT	O	PIR signaling; HIGH = movement/LOW = no movement

Pin Type: P = Power, G = Ground, I = Input, O = Output

Jumper Settings

Symbol	Description
S	Reduced sensitivity mode, for a shorter range, about 15 feet maximum
L	Normal operation, for a longer range, about 30 feet maximum

Table A.3: Pin definitions and ratings of the Parallax PIR motion detector.

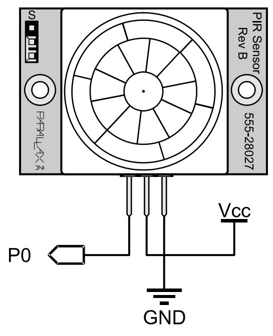


Figure A.1: PIR simple circuit diagram.

A.4 LEM LTS 25-NP Current Transducer

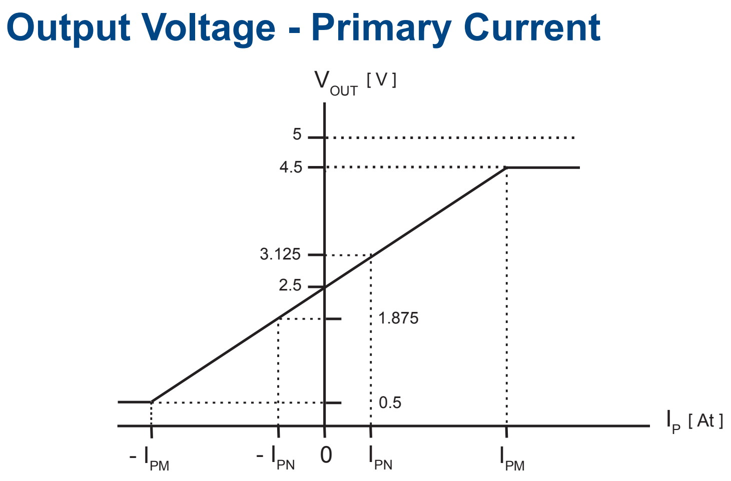


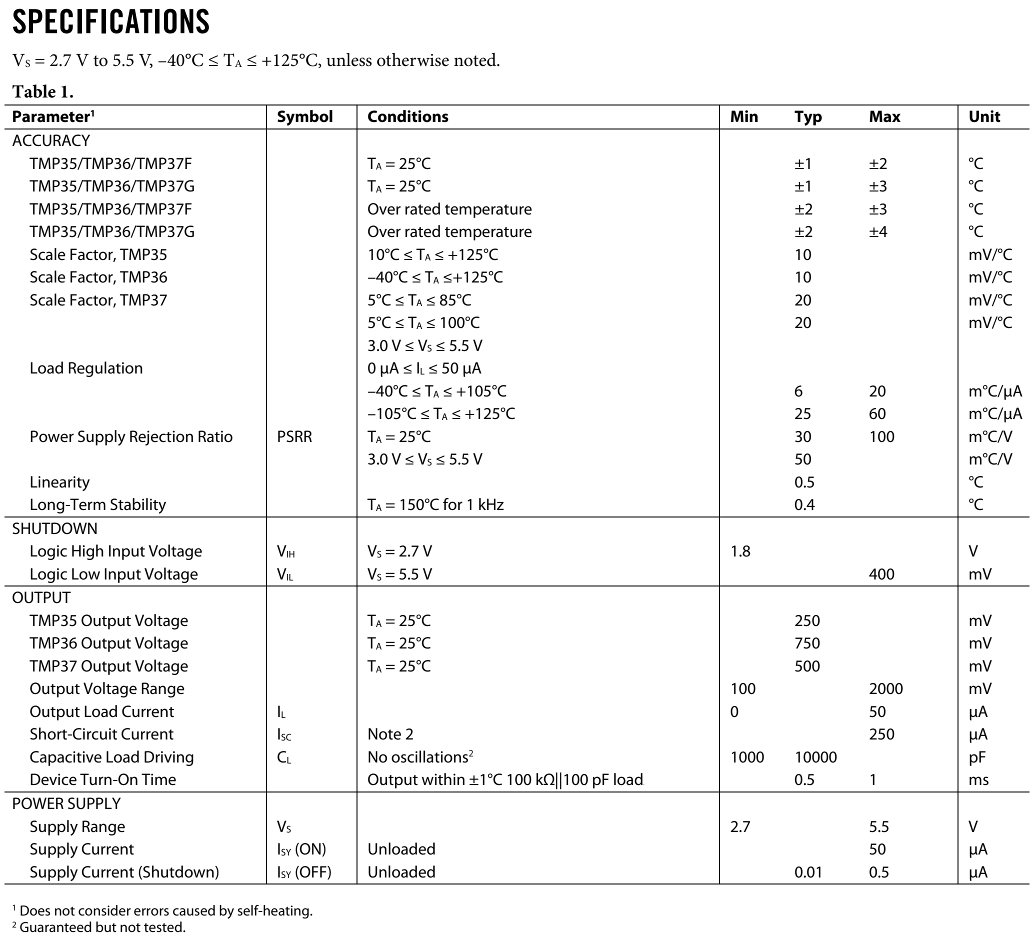
Figure A.2: Relationship between primary current and output voltage of the LEM LTS 25-NP Current Transducer.

Electrical data

$\mathbf{I}_{PN}$	Primary nominal current rms		25	At
$\mathbf{I}_{PM}$	Primary current, measuring range		0 .. ± 80	At
$\mathbf{V}_{OUT}$	Output voltage (Analog) @ $\mathbf{I}_P$		$2.5 \pm (0.625 \cdot \mathbf{I}_P/\mathbf{I}_{PN})$	V
	$\mathbf{I}_P = 0$		2.5 [1)]	V
G	Sensitivity		25	mV/A
$\mathbf{N}_S$	Number of secondary turns (± 0.1 %)		2000	
$\mathbf{R}_L$	Load resistance		≥ 2	kΩ
$\mathbf{R}_{IM}$	Internal measuring resistance (± 0.5 %)		50	Ω
$\mathbf{TCR}_{IM}$	Temperature coefficient of $\mathbf{R}_{IM}$		< 50	ppm/K
$\mathbf{V}_C$	Supply voltage (± 5 %)		5	V
$\mathbf{I}_C$	Current consumption @ $\mathbf{V}_C$ = 5 V	Typ	$28+\mathbf{I}_S$[2)]$+(\mathbf{V}_{OUT}/\mathbf{R}_L)$	mA

Table A.4: LEM LTS 25-NP current sensor electrical data.

A.5 TMP36 Temperature Sensor



SPECIFICATIONS

V_S = 2.7 V to 5.5 V, −40°C ≤ T_A ≤ +125°C, unless otherwise noted.

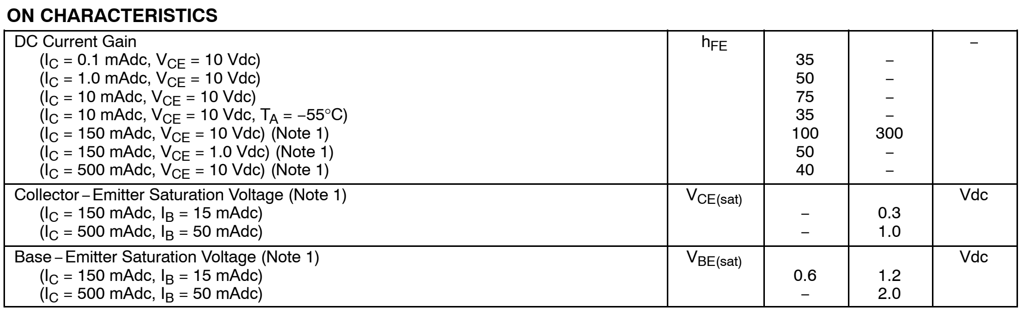
Table 1.

Parameter[1]	Symbol	Conditions	Min	Typ	Max	Unit
ACCURACY						
TMP35/TMP36/TMP37F		T_A = 25°C		±1	±2	°C
TMP35/TMP36/TMP37G		T_A = 25°C		±1	±3	°C
TMP35/TMP36/TMP37F		Over rated temperature		±2	±3	°C
TMP35/TMP36/TMP37G		Over rated temperature		±2	±4	°C
Scale Factor, TMP35		10°C ≤ T_A ≤ +125°C		10		mV/°C
Scale Factor, TMP36		−40°C ≤ T_A ≤+125°C		10		mV/°C
Scale Factor, TMP37		5°C ≤ T_A ≤ 85°C		20		mV/°C
		5°C ≤ T_A ≤ 100°C		20		mV/°C
		3.0 V ≤ V_S ≤ 5.5 V				
Load Regulation		0 μA ≤ I_L ≤ 50 μA				
		−40°C ≤ T_A ≤ +105°C		6	20	m°C/μA
		−105°C ≤ T_A ≤ +125°C		25	60	m°C/μA
Power Supply Rejection Ratio	PSRR	T_A = 25°C		30	100	m°C/V
		3.0 V ≤ V_S ≤ 5.5 V		50		m°C/V
Linearity				0.5		°C
Long-Term Stability		T_A = 150°C for 1 kHz		0.4		°C
SHUTDOWN						
Logic High Input Voltage	V_{IH}	V_S = 2.7 V	1.8			V
Logic Low Input Voltage	V_{IL}	V_S = 5.5 V			400	mV
OUTPUT						
TMP35 Output Voltage		T_A = 25°C		250		mV
TMP36 Output Voltage		T_A = 25°C		750		mV
TMP37 Output Voltage		T_A = 25°C		500		mV
Output Voltage Range			100		2000	mV
Output Load Current	I_L		0		50	μA
Short-Circuit Current	I_{SC}	Note 2			250	μA
Capacitive Load Driving	C_L	No oscillations[2]	1000	10000		pF
Device Turn-On Time		Output within ±1°C 100 kΩ‖100 pF load		0.5	1	ms
POWER SUPPLY						
Supply Range	V_S		2.7		5.5	V
Supply Current	I_{SY} (ON)	Unloaded			50	μA
Supply Current (Shutdown)	I_{SY} (OFF)	Unloaded		0.01	0.5	μA

[1] Does not consider errors caused by self-heating.
[2] Guaranteed but not tested.

Table A.5: TMP36 temperature sensor electrical specifications.

A.6 PN2222 Transistor



ON CHARACTERISTICS

DC Current Gain (I_C = 0.1 mAdc, V_{CE} = 10 Vdc) (I_C = 1.0 mAdc, V_{CE} = 10 Vdc) (I_C = 10 mAdc, V_{CE} = 10 Vdc) (I_C = 10 mAdc, V_{CE} = 10 Vdc, T_A = −55°C) (I_C = 150 mAdc, V_{CE} = 10 Vdc) (Note 1) (I_C = 150 mAdc, V_{CE} = 1.0 Vdc) (Note 1) (I_C = 500 mAdc, V_{CE} = 10 Vdc) (Note 1)	h_{FE}	 35 50 75 35 100 50 40	 – – – – 300 – –	–
Collector−Emitter Saturation Voltage (Note 1) (I_C = 150 mAdc, I_B = 15 mAdc) (I_C = 500 mAdc, I_B = 50 mAdc)	$V_{CE(sat)}$	 – –	 0.3 1.0	Vdc
Base−Emitter Saturation Voltage (Note 1) (I_C = 150 mAdc, I_B = 15 mAdc) (I_C = 500 mAdc, I_B = 50 mAdc)	$V_{BE(sat)}$	 0.6 –	 1.2 2.0	Vdc

Table A.6: PN2222 transistor on characteristics.

List of References

[1] A. Al-Ali and M. AL-Rousan, “Java-based home automation system,” *IEEE Transactions on Consumer Electronics*, vol. 50, pp. 498–504, May 2004.

[2] M. Z. Bjelica, I. Papp, N. Teslic, and J.-M. Coulon, “Set-top box-based home controller,” in *IEEE International Symposium on Consumer Electronics (ISCE 2010)*, pp. 1–6, IEEE, June 2010.

[3] “Beagleboard.” [Online]. http://beagleboard.org/; accessed 10/08/2014.

[4] “Circuitco.” [Online]. http://circuitco.com/; accessed 10/08/2014.

[5] “The thing system.” [Online]. http://thethingsystem.com/;accessed 10/08/2014.

[6] S. Folea, D. Bordencea, C. Hotea, and H. Valean, “Smart home automation system using Wi-Fi low power devices,” in *Proceedings of 2012 IEEE International Conference on Automation, Quality and Testing, Robotics*, pp. 569–574, IEEE, May 2012.

[7] F. Hamoui, C. Urtado, S. Vauttier, and M. Huchard, “SAASHA: A Self-Adaptable Agent System for Home Automation,” in *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 227–230, IEEE, Sept. 2010.

[8] N. Liang, L. Fu, and C. Wu, “An integrated, flexible, and Internet-based control architecture for home automation system in the Internet era,” in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 2, pp. 1101–1106, IEEE, 2002.

[9] C. Felix and I. Jacob Raglend, “Home automation using GSM,” in *2011 International Conference on Signal Processing, Communication, Computing and Networking Technologies*, pp. 15–19, IEEE, July 2011.

[10] F. Doctor, H. Hagras, and V. Callaghan, “A Fuzzy Embedded Agent-Based Approach for Realizing Ambient Intelligence in Intelligent Inhabited Environments,” *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 35, pp. 55–65, Jan. 2005.

[11] U. Bischoff, V. Sundramoorthy, and G. Kortuem, "Programming the smart home," in *3rd IET International Conference on Intelligent Environments IE 07 (2007)*, pp. 544–551, 2007.

[12] K. Wang, W. Abdulla, and Z. Salcic, "Multi-agent fuzzy inference control system for intelligent environments using JADE," no. 2, pp. 285–294, 2006.

[13] H. Medjahed, D. Istrate, J. Boudy, J.-L. Baldinger, and B. Dorizzi, "A pervasive multi-sensor data fusion for smart home healthcare monitoring," in *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*, pp. 1466–1473, IEEE, June 2011.

[14] J. Nazabal, "Home automation based sensor system for monitoring elderly people safety," *... (ICST), 2012 Sixth ...*, pp. 142–145, 2012.

[15] M. Hussain, M. Afzal, W. Khan, and S. Lee, "Clinical Decision Support Service for Elderly People in Smart Home Environment," *uclab.khu.ac.kr*, vol. 2012, no. December, pp. 5–7, 2012.

[16] H. Lee and S. Liao, "An Intelligent Agent of Emergency Process in Home Network Control," *Machine Learning and Cybernetics, 2007 ...*, no. August, pp. 19–22, 2007.

[17] S. Veleva, D. Davcev, and M. Kacarska, "Wireless smart platform for Home Energy Management System," in *2011 2nd IEEE PES International Conference and Exhibition on Innovative Smart Grid Technologies*, pp. 1–8, IEEE, Dec. 2011.

[18] S. Inoue, J. Dai, M. Shiba, S. Aoki, and H. Tsuji, "Knowledge management approach for saving home energy consumption," in *International Conference on Fuzzy Systems*, pp. 1–6, IEEE, July 2010.

[19] A. Rossello-Busquet, J. Soler, and L. Dittmann, "A Novel Home Energy Management System Architecture," in *2011 UkSim 13th International Conference on Computer Modelling and Simulation*, pp. 387–392, IEEE, Mar. 2011.

[20] A. Alvi and D. Greaves, "A logical approach to home automation," *Intelligent Environments, 2006. IE 06. 2nd ...*, pp. 45–50, 2006.

[21] L. Litz and M. Gross, "Covering Assisted Living Key Areas based on Home Automation Sensors," in *2007 IEEE International Conference on Networking, Sensing and Control*, no. April, pp. 639–643, IEEE, 2007.

[22] D. Niyato, "Machine-to-machine communications for home energy management system in smart grid," *IEEE Communications Magazine*, vol. 49, pp. 53–59, Apr. 2011.

[23] A. Alkar and U. Buhur, "An internet based wireless home automation system for multifunctional devices," *IEEE Transactions on Consumer Electronics*, vol. 51, pp. 1169–1174, Nov. 2005.

[24] M. Bjelica and N. Teslic, "A concept and implementation of the Embeddable Home Controller," *MIPRO, 2010 Proceedings of the 33rd . . .* , pp. 686–690, 2010.

[25] R. Piyare and M. Tazil, "Bluetooth based home automation system using cell phone," in *2011 IEEE 15th International Symposium on Consumer Electronics (ISCE)*, pp. 192–195, IEEE, June 2011.

[26] L. Zhang, H. Leung, and K. Chan, "Information fusion based smart home control system and its application," *IEEE Transactions on Consumer Electronics*, vol. 54, pp. 1157–1165, Aug. 2008.

[27] D. Fortin-Simard, K. Bouchard, S. Gaboury, B. Bouchard, and A. Bouzouane, "Accurate passive RFID localization system for smart homes," in *2012 IEEE 3rd International Conference on Networked Embedded Systems for Every Application (NESEA)*, pp. 1–8, IEEE, Dec. 2012.

[28] A. K. Nabih, H. Osman, M. Gomaa, and G. M. Aly, "New fuzzy-based indoor positioning scheme using ZigBee wireless protocol," in *2012 Seventh International Conference on Computer Engineering & Systems (ICCES)*, pp. 16–20, IEEE, Nov. 2012.

[29] W. Kastner, M. J. Kofler, and C. Reinisch, "Using AI to realize energy efficient yet comfortable smart homes," in *2010 IEEE International Workshop on Factory Communication Systems Proceedings*, pp. 169–172, IEEE, May 2010.

[30] L. Borodulkin, H. Ruser, and H.-R. Trankler, "3D virtual "smart home" user interface," in *2002 IEEE International Symposium on Virtual and Intelligent Measurement Systems (IEEE Cat. No.02EX545)*, no. May, pp. 111–115, IEEE, 2002.

[31] C.-M. Yeoh, H.-Y. Tan, C.-K. Kok, H.-J. Lee, and H. Lim, "e2Home: A Lightweight Smart Home Management System," in *2008 Third International Conference on Convergence and Hybrid Information Technology*, pp. 82–87, IEEE, Nov. 2008.

[32] S. Spinsante and E. Gambi, "Home automation systems control by head tracking in AAL applications," in *2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)*, pp. 1–6, IEEE, Oct. 2012.

[33] M. Guizani, "Home M2M networks: Architectures, standards, and QoS improvement," *IEEE Communications Magazine*, vol. 49, pp. 44–52, Apr. 2011.

[34] S. Guillemin and D. L. Ha, "Fuzzy and parametric method for self-configuration configuration of home energy manager system," in *2012 3rd IEEE PES Innovative Smart Grid Technologies Europe (ISGT Europe)*, pp. 1–5, IEEE, Oct. 2012.

[35] Z. L. Wu and N. Saito, "The Smart Home [Scanning the Issue]," *Proceedings of the IEEE*, vol. 101, pp. 2319–2321, Nov. 2013.

[36] Y. Chen and W. Wang, "Machine-to-Machine Communication in LTE-A," in *2010 IEEE 72nd Vehicular Technology Conference - Fall*, pp. 1–4, IEEE, Sept. 2010.

[37] M. J. Booysen, J. A. A. Engelbrecht, and A. Molinaro, "Proof of Concept : Large-Scale Monitor and Control of Household Water Heating in Near Real-Time," pp. 1–8, 2013.

[38] "Arduino." [Online]. http://arduino.cc; accessed 10/09/2014.

[39] "Raspberry pi." [Online]. http://www.raspberrypi.org/; accessed 10/09/2014.

[40] "Android mini pc." [Online]. http://www.timingpower.com/android-mini-pc-rk3188; accessed 10/09/2014.

[41] "Qwikswitch." [Online]. http://www.qwikswitch.co.za/; accessed 15/09/2014.

[42] "Steward." [Online]. https://github.com/TheThingSystem/steward; accessed 10/08/2014.

[43] "Nodejs." [Online]. http://nodejs.org/; accessed 10/08/2014.

[44] "Steward supported things." [Online]. http://thethingsystem.com/dev/supported-things.html; accessed 11/09/2014.

[45] "Computer and internet use in the united states." [Online]. http://www.census.gov/prod/2013pubs/p20-569.pdf; accessed 12/09/2014.

[46] "Android vs ios vs blackberry os." [Online]. http://www.tdktech.com/tech-talks/android-vs-ios-vs-blackberry-os; accessed 12/09/2014.

[47] "onoff." [Online]. https://www.npmjs.org/package/onoff; accessed 17/09/2014.

[48] "Octalbonescript." [Online]. https://www.npmjs.org/package/octalbonescript; accessed 17/09/2014.

[49] "Parallax." [Online]. http://www.parallax.com/; accessed 18/09/2014.

[50] "Hellcom." [Online]. http://www.hellkom.co.za/isp-prices/3gb-adsl/; accessed 29/10/2014.

[51] "Passport." [Online]. http://passportjs.org/; accessed 29/10/2014.