

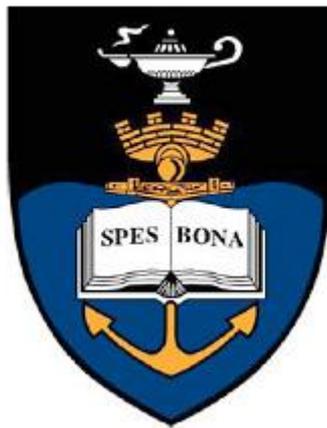
The copyright of this thesis vests in the author. No quotation from it or information derived from it is to be published without full acknowledgement of the source. The thesis is to be used for private study or non-commercial research purposes only.

Published by the University of Cape Town (UCT) in terms of the non-exclusive license granted to UCT by the author.

Design of a Remote Monitoring and Diagnostics Platform for Air Conditioning Installations

Prepared for
Department of Electrical Engineering
The University of Cape Town

Prepared by
Greg Cohen



Submitted to the Department of Electrical Engineering, University of Cape Town, in fulfillment of the requirements for the degree Master of Science in Electrical Engineering.

Cape Town, March 2008

Keywords

Remote Monitoring, Automated Diagnostics, Air Conditioning

Plagiarism Declaration

This report and the project on which it is based is entirely my own work. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the works of others has been attributed, cited and referenced. I have not allowed, nor will allow, any other student to copy my work with the intention of passing it off as their own.

I acknowledge that plagiarism is wrong, and declare that this report, and the project on which it is based, is entirely my own work.

X

Greg Cohen

University of Cape Town

Abstract

Faults and inefficiencies in air conditioning installations account for between 2% and 11% of all energy consumed by commercial buildings in the United States each year. Diagnostics systems have been proven to improve the performance of air conditioning plants but the high costs of purchasing, retrofitting and maintaining such a system results in limited market adoption of such systems.

This thesis discusses the design, implementation and results of low-cost remote monitoring and diagnostic platform for use in air conditioning installations. The design of the various hardware components is presented along with the structure of the framework developed for each device. The thesis also contains information regarding the selection, integration and installation of the various types of sensors required on the various installations.

A specially-designed protocol was also developed to handle communication between the hardware devices. Both the physical configuration and details of the protocol structure are presented in detail in this thesis. The mechanism through which the device uploads data to a server is also described in this thesis and includes details on both the hardware and the server technologies used in the upload process.

The system has been installed on two different sites in Cape Town, South Africa and has produced meaningful diagnostic information since November 2007.

University of Cape Town

Acknowledgements

Several individuals and institutions provided invaluable assistance to the author during the completion of this project. The author would like to particularly acknowledge the following individuals:

- Harry Schuurmans and Dr. Wilkinson for their guidance and suggestions throughout the entire thesis.
- David Karpul for assisting with almost every aspect of this project and for designing the analogue circuitry for the Sensor Interface Boards.
- Paul Schuurmans for performing all the physical installation on the chillers and for tirelessly putting up with the constant requests of the author.
- Richard Parry for his guidance on GSM units.
- Brandon Resnick, Anton Smit, Dean Marks, Greg Priday and Steven Cohen for allowing the author to ramble on about the project to them at great length.
- Audrey, Deirdre and Proto for their constant support throughout the project.

University of Cape Town

Table of Contents

Plagiarism Declaration	i
Abstract.....	ii
Acknowledgements.....	iii
List of Figures	vi
List of Tables	viii
1 Introduction	1
1.1 Subject of this Thesis.....	1
1.2 Objectives of this Thesis.....	1
1.3 Scope and Limitations of this Thesis	1
1.4 Plan of Development	1
1.5 Terminology	3
2 Introduction to Air Conditioning Systems.....	4
2.1 Overview of Air Conditioning Systems.....	4
2.2 Sources of Heat in a Building	4
2.3 Psychometrics	5
2.4 Components of an Air Conditioning System	6
3 Motivation for the Design of a Remote Monitoring Platform	13
3.1 Summary of the TIAX Report to the U.S. Department of Energy.....	13
3.2 Defining System Requirements to Match Shortcomings in Current Technologies.....	16
3.3 Benefits of a Remote Diagnostic Tool in a Southern African Context	17
3.4 Fundamental Design Choices Based on Market Requirements.....	19
3.5 Related Projects	20
3.6 The Proposed Monitoring and Diagnostics System	22
4 Sensor Selection and Installation.....	25
4.1 Pressure Sensors.....	25
4.2 Temperature Sensors.....	30
4.3 Current, Energy and Power Sensors	37
4.4 Wet-Bulb Temperature and Humidity Sensors.....	38
4.5 Running Time Monitoring	39
5 Design of the Sensor Interface Boards.....	40
5.1 Overview of the Sensor Interface Boards	40
5.2 Sensor Interface Board Configurations.....	44
6 Design of the Host Controller	48
6.1 The Central Role of the Host Controller.....	48
6.2 Design of the Original Host Controller.....	48
6.3 Design of the Revised Host Controller	56
7 Network and Bus Design	59

7.1	Design of the Physical Bus.....	59
7.2	The Communication Protocol	63
7.3	Addressing Modes.....	75
7.4	The Reading Acquisition Process	87
7.5	Network Simulation	92
8	Uploading Methodologies.....	96
8.1	Introduction	96
8.2	Requirements of the Uploading System	96
8.3	Comparison of Uploading Technologies	98
8.4	Justification for Uploading using an HTTP Server	117
9	Results.....	118
10	Conclusions	124
11	Bibliography	127
	Appendix A – List of Abbreviations	130
	Appendix B - Unit Conversions and Physical Constants.....	132
	Appendix C – Glossary.....	135
	Appendix D – Additional Graphs for April 2008.....	138

University of Cape Town

List of Figures

Figure 1-1: Flow Chart Outlining the Structure of this Thesis.....	2
Figure 1-2: Illustration of the terms "Installation", "System" and "Platform".....	3
Figure 2-1: Components of a Typical Air Conditioning Installation	7
Figure 2-2: An Air Conditioning System with a Water-Cooled Condenser	8
Figure 2-3: A Typical Air Conditioning System using Chilled Water.....	9
Figure 2-4: Diagram of a Simple Air Handling Unit	11
Figure 2-5: Diagrammatic and Sectional View of an AHU with Fresh Air Intake	11
Figure 3-1: Breakdown of Fault Losses by Fault Type.....	14
Figure 3-2: Structure of the Monitoring System	23
Figure 4-1: Method to attach a Pressure Sensor using a Piercing Valve	27
Figure 4-2: Two Different High Pressure Sensors in the Range of 0 - 300 PSI	28
Figure 4-3: Diagram showing Chilled Water and Condenser Water Temperature Points.....	30
Figure 4-4: Sensor Pockets Installed using a T-Piece (left) and Welded into a Pipe (right).....	31
Figure 4-5: Temperature Points in an Air Handling Unit with an Economy Cycle	32
Figure 4-6: Current Transformers Affixed with Cable Ties in a Chiller.....	38
Figure 5-1: Schematic of a Simple Power Jumper Board	41
Figure 5-2: The Communication Hardware on the Sensor Interface Boards.....	41
Figure 5-3: Lever Connectors and Mounting Holes on a Sensor Interface Board	42
Figure 5-4: The 4T1P Sensor Interface Board	44
Figure 5-5: The 3P1C Sensor Interface Board	45
Figure 5-6: The 3P1C4T Sensor Interface Board	46
Figure 5-7: The 8L8C Sensor Interface Board.....	47
Figure 6-1: A Photograph of the Original Host Controller	49
Figure 6-2: Components of the Original Host Controller Design	50
Figure 6-3: Hardware De-bouncing for the Host Controller Input Buttons	52
Figure 6-4: Circuit Diagram for the Atmel External Flash Memory	53
Figure 6-5: Steps in the Main Application Loop of the Host Controller Firmware	55
Figure 6-6: Components of the Revised Host Controller	57
Figure 6-7: PCB Layout for the Revised Host Controller	58
Figure 7-1: Examples of Star and Bus Topologies	60
Figure 7-2: Illustration of Pass-through Network Propagation	62
Figure 7-3: Illustration of the buffer-and-forward method of data propagation	63
Figure 7-4: Structure of a Bus Message	66
Figure 7-5: Illustration of Packet Numbering using the Flags Field	69
Figure 7-6: Illustration of the different methods to terminate a bus message.....	70
Figure 7-7: Structure of a Bus ACK.....	70
Figure 7-8: Structure of a Ping Message	73
Figure 7-9: Structure of a Set Address Message	74
Figure 7-10: Structure of a Bus Reset Message	74
Figure 7-11: Propagation of Addresses through the System	76
Figure 7-12: Illustration of the Network Discovery Process	78
Figure 7-13: Illustration of the Network Inventory Process	79
Figure 7-14: Example of how a Pressure Board packs data into the payload fields	81
Figure 7-15: The Network Perspectives viewed by each layer of the protocol	82
Figure 7-16: Example of the benefit to supporting multiple channels on the host controller.....	83
Figure 7-17: Channel-Merging in Multiple Channel Host Controllers	84
Figure 7-18: Illustration of a Network Discovery using Sensor-Based Addressing	85
Figure 7-19: Structure of a TIME ANNOUNCE message.....	86
Figure 7-20: Timing diagram for a temperature Sensor	88

Figure 7-21: Illustration of the 4-stage Reading Acquisition Process	90
Figure 7-22: Acquiring Current and Temperature readings with the 4-Phase Acquisition Process	91
Figure 7-23: Screenshot of the Protocol Upload Panel.....	93
Figure 8-1: Diagrammatic view of uploading data using a centralized email server (SMTP Server)	98
Figure 8-2: Diagrammatic view of uploading data using a FTP server.....	99
Figure 8-3: Diagrammatic view of uploading using a proprietary TCP server	100
Figure 8-4: Structure of Client Requests to the Diagnostics TCP Server.....	102
Figure 8-5: Verbose and Abbreviated TCP Server Responses.....	103
Figure 8-6: Example of Performing an Upload to the Proprietary TCP Server	103
Figure 8-7: Diagrammatic outline of the Uploading using HTTP Commands	105
Figure 8-8: Structure of an HTTP Request.....	107
Figure 8-9: The Use of the Hosts Header when accessing Virtual Web Servers.....	111
Figure 9-1: Layout of the Plant Room at the Installation Site.....	118
Figure 9-2: Network Configuration for "Daikin B" Chiller	119
Figure 9-3: Outside Air Temperature for the Month of November.....	119
Figure 9-4: Graph of the Chilled Water Temperature for the 5th of November	120
Figure 9-5: Compressor Currents for the 5th of November	121
Figure 9-6: Graph of Board Temperature and Compressor Current.....	121
Figure 9-7: Graph of Condenser Water Temperatures for the 5th of November	122

University of Cape Town

List of Tables

Table 3-1: Energy Impact of Faults in Commercial Buildings in the U.S.	14
Table 3-2: Sensor Platform Requirements.....	20
Table 4-1: Table showing the 1-Wire© ROM Commands	34
Table 4-2: Table showing the 1-Wire© Command Specific to the DS18S20	35
Table 5-1: Naming Convention for Board Configurations.....	44
Table 7-1: Comparison of the features of Daisy-Chain and Direct-Drop Busses	61
Table 7-2: Valid Message Preambles	67
Table 7-3: Summary of the Different Message Types.....	68
Table 7-4: Summary of the Bus ACK Flags	72
Table 8-1: Proprietary TCP Server Request Commands.....	102
Table 8-2: Proprietary TCP Server Responses.....	103
Table 8-3: HTTP Request Methods in HTTP Version 1.1	108
Table 8-4: HTTP Headers required in the Upload Process	110
Table 8-5: Proprietary AT Command Implemented on the GSM Unit.....	115

University of Cape Town

1 Introduction

1.1 Subject of this Thesis

This thesis presents a complete design of a remote monitoring and diagnostics platform for use in industrial applications. The system designed was implemented on large air conditioning installations and a discussion on the outputs of these systems is presented. The monitoring platform consists of specialized hardware, firmware and software components which allow the seamless and autonomous collection of readings from a wide range of different sensors. These readings are periodically uploaded to a server where diagnostics processes analyse the data to detect faults and developing problems.

1.2 Objectives of this Thesis

The objectives of the thesis mirror the requirements of the sensor platform developed. A brief summary of these requirements is given below:

- The monitoring system must be able to support distances of up to 500 meters between sensors.
- The system must be designed in a modular fashion to make it easily customizable to fit different sized installations.
- The system needs to be able to support installations with a large number of sensors.
- The cost of the system must be kept as low as possible.
- The installation of the hardware must be made as easy as possible and require no expertise in order to install a system (beyond basic wiring and mechanical installation skills).
- The hardware must be rugged and able to function in an industrial environment.
- The system must be entirely self-reliant and not require any existing infrastructure beyond mains power. This includes telephone and internet or network access.

A more detailed explanation of these requirements is presented in chapter 3.2.

1.3 Scope and Limitations of this Thesis

This thesis discusses the design, implementation and results of the monitoring platform. The design of the various hardware components is presented along with the structure of the framework developed for each device. The thesis also contains information regarding the selection, integration and installation of the various types of sensors required on the various installations.

A specially-designed protocol was also developed to handle communication between the hardware devices. Both the physical configuration and details of the protocol structure are presented in detail in this thesis. The mechanism through which the device uploads data to a server is also described in this thesis and includes details on both the hardware and the server technologies used in the upload process.

This thesis does not discuss the various diagnostic processes run on the acquired sensor information as it is highly specific to the installations in question and often relies on confidential information supplied by the equipment owner and manufacturer. A brief discussion on the technology behind the diagnostics system is presented at the end of this thesis.

1.4 Plan of Development

This thesis begins with a brief introduction to air conditioning in order to introduce the concepts and equipment mentioned throughout the rest of the document. The motivation for developing the system is presented in Chapter 3 which includes a description of other such systems and a complete

list of the system requirements. The rest of this thesis follows the flow of information through the system from the sensors through to the diagnostic engine.

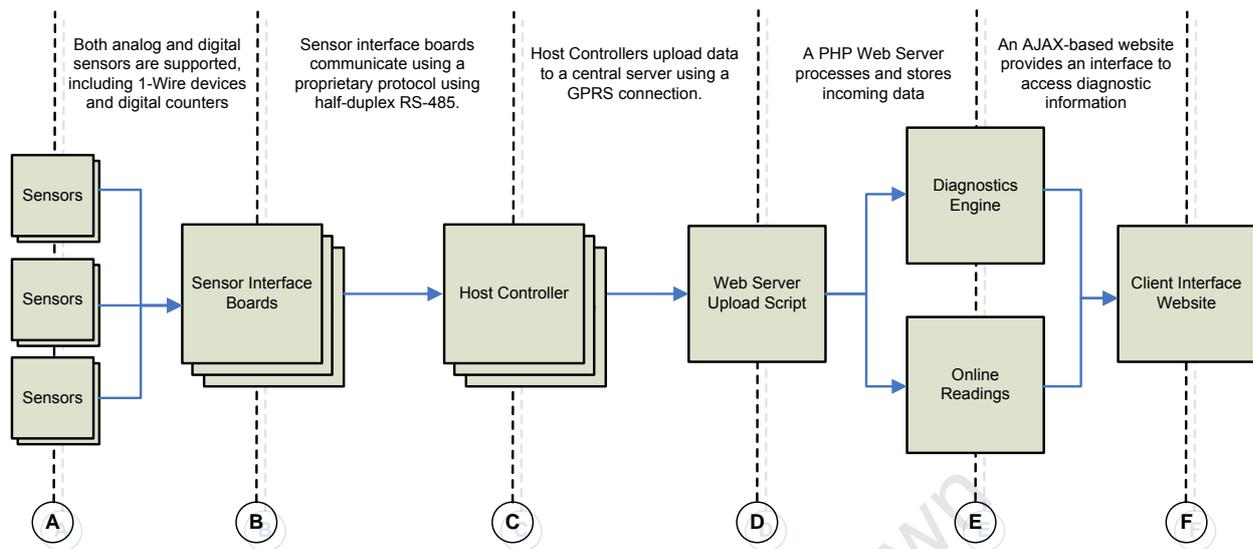


Figure 1-1: Flow Chart Outlining the Structure of this Thesis

The above figure presents an overview of the general structure of both the system and the rest of this document and illustrates the flow of data through the system. It attempts to show all the connections between the various components of the monitoring system. The stacked items (such as the host controller) indicate that there may be multiple items which communicate at that point in the system. Hence multiple sensors may be connected to multiple Sensor Interface Boards.

Chapter 4 outlines the various sensors used to physically collect data. The focus of this chapter is providing insight into the wide range of sensors that can be supported on this system and details on the means by which they were incorporated into the system.

Chapter 5 describes the design, structure and functioning of the Sensor Interface Boards. These boards provide the interface between the sensor technology and the network, and therefore contain a wide variety of analogue and digital circuitry. It provides details on the various board configurations that were developed and the motivation behind their creation.

Chapter 6 presents the structure and design of the host controller unit. The Host Controller is the most complicated component in the system as it coordinates actions on its network of Sensor Interface Boards and is responsible for communication with the upload server. Two versions of the Host Controller were designed and are discussed in this section.

Chapter 7 discusses the design of the physical network and the protocol written to provide communication between the Host Controller and a network of Sensor Interface Boards.

Chapter 8 discusses the various uploading methodologies that were considered and the methods by which the Host Controller uploads data to the diagnostic engine.

Chapter 9 details the results obtained from the initial site being monitored.

Chapter 10 states the conclusions of this thesis.

1.5 Terminology

There are a number of important distinctions that need to be made in order to remove any ambiguity regarding the terminology used in this document. These relate to the various views of the system and the express definition below should serve to make the distinction between them clearer.

The monitoring system designed is capable of supporting a number of independent remote sites. Each remote site has one Host Controller and any number of Sensor Interface Boards. All the Host controllers connect to a single server which houses the diagnostics engine and generates the client interface.

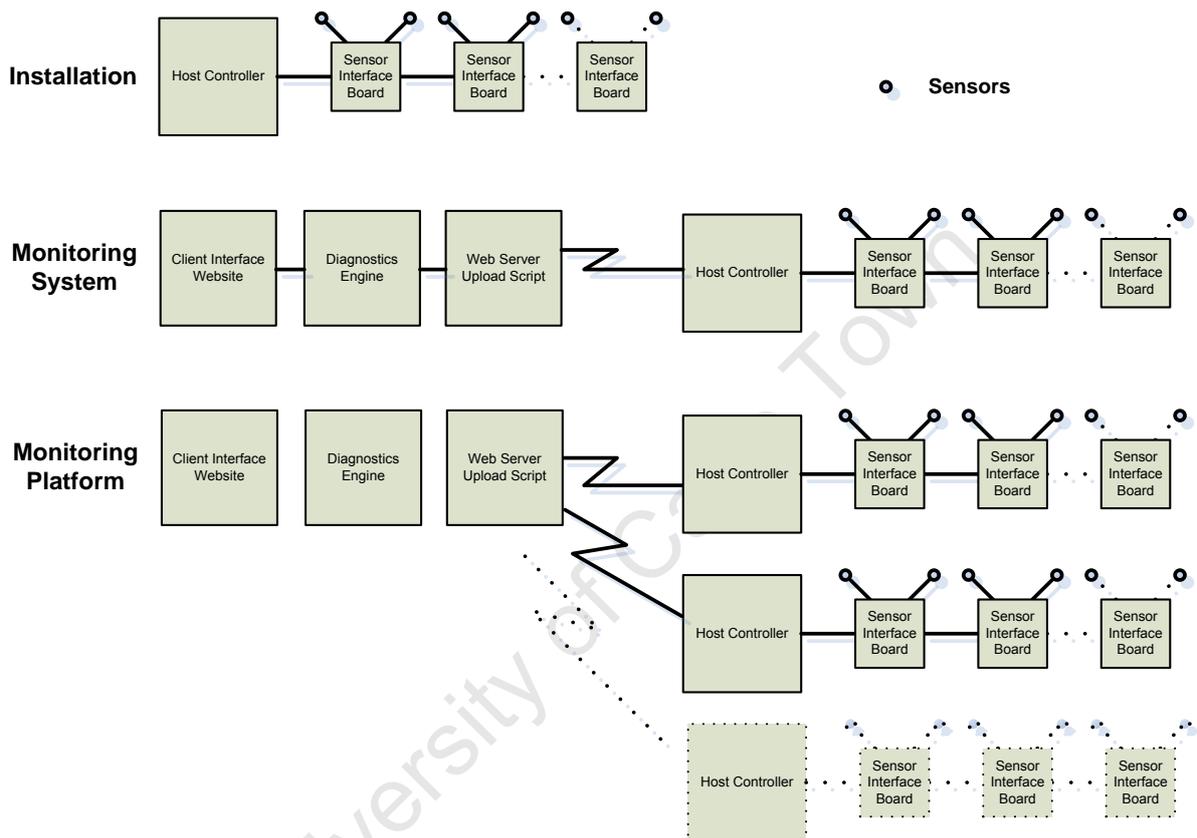


Figure 1-2: Illustration of the terms "Installation", "System" and "Platform"

Each Host Controller manages a local network of Sensor Interface Boards. This is referred to as an installation or a site. Each site acts autonomously and acts independently of other sites. As a result many discussions are limited to a single installation for the sake of clarity.

The term "monitoring platform" refers to the entire system consisting of all the active installations (Host Controllers and their network of Sensor Interface Boards), the upload server, diagnostics engine and client interface.

As each Host Controller acts independently of other installations, each interact with the upload server as if it were the only active Host Controller on the network. For this reason, when referring to the monitoring system, this is intended to include only the Host Controller in question and the monitoring components active on the server.

2 Introduction to Air Conditioning Systems

2.1 Overview of Air Conditioning Systems

Air Conditioning is the process of regulating and creating an atmospheric environment for either human comfort or the specific requirements of an industrial process [1]. Air conditioning encompasses more than just cooling or ventilation as it includes additional factors such as air quality, oxygen content, circulation, pressure and relative humidity [2]. The ASHREA definition of air conditioning is “the process of treating air so as to control simultaneously its temperature, humidity, cleanliness, and distribution to meet the requirements of the conditioned space” [3].

This definition does not take into account the many other factors that have recently become important issues in air conditioning. Factors such as sound and vibration reduction, water conservation and the phased-removal of CFC refrigerants have become key requirements in the field of air conditioning. As air conditioning systems constitute a considerable portion of the energy requirements for buildings, there is an ever increasing need to reduce energy consumption as much as possible without compromising the effectiveness of a system.

The definition of air conditioning can therefore be expanded to include these factors. A more complete definition of air conditioning would be the process of treating air so as to control simultaneously its temperature, humidity, cleanliness and distribution to meet the requirements of the conditioned space whilst ensuring that power consumption and the impact to the environment are minimized.

It is important to note the difference between ventilation and air conditioning. Ventilation is the process of providing fresh air to enclosed spaces for reasons such as attaining safe levels of oxygen and carbon dioxide, the dilution of odours and the pressurizing of spaces for fire and safety reasons. Air conditioning often includes ventilation systems for the same reasons but is also responsible for the regulation of the internal environment.

Most air conditioning systems therefore have some form of refrigeration or heating plant. This is usually the biggest consumer of energy in the air conditioning system and is what distinguishes air conditioned spaces from ventilated ones. It is also important to note that the definition of air conditioning requires that the humidity also be controlled, although this is often not explicitly regulated. An air conditioning system that does not regulate humidity can be considered to be a partial air conditioning system.

2.2 Sources of Heat in a Building

In order to estimate the heat load of a building, it is important to understand the factors that contribute to the generation of heat in a room. Air conditioning is often thought to be the production of a comfortable environment for occupants of an enclosed space. Although this is certainly one of the most predominant uses of air conditioning systems, air conditioning is also required in many industrial installations and processes.

Infiltration and solar heat are two primary sources of heat in an enclosed space. Infiltration is the heat entering a space through the walls, floor and ceiling. This heat (or loss) is a result of the temperature difference between the enclosed space and the surrounding area. The material and the construction of the walls play a big factor in determining the amount of infiltration.

Solar heat occurs from solar warming of the sides of the enclosed space. It is often modelled as simply increasing the temperature difference between the two spaces and plays a significant role when considering windows and roofs. Solar heat is also not evenly distributed around the sides of a

building as it varies with the direction the side is facing and the time of year. As a result, it is possible for a corner office to become hotter in winter than it would be in summer.

Another primary source of heat in a room arises from the need to supply fresh air. Fresh air needs to be drawn from the outside and will therefore be at the outside temperature. As this air is injected directly into an air conditioned space, it can be considered as a source of heat.

Other sources of heat include lighting, electrical machinery in the room and any heat arising from industrial processes in the room (such as steam). Heat can also be introduced through objects at a higher temperature being placed in the room to cool down. One final source of heat is the occupants of the room themselves which is dependent on their level of activity. People generate heat and release moisture into the surround air which increases both the heat and the relative humidity of the air.

These factors are mentioned as many of them can be measured and used to construct accurate heat load estimations for a building. For example, temperature and humidity sensors can be used to calculate the temperature of outside air, the orientation of the building and the date can be used to estimate solar heating and energy consumption can be used to determine the heat generated by overhead lighting. Accurate models of a building can therefore be generated which can be compared to actual parameters to provide insight into the efficiency of an air conditioning system.

2.3 Psychometrics

Psychometrics is the field of science encompassing the determination of the physical and thermodynamic properties of gas-vapour mixtures [4]. Psychometrics is extremely important in air conditioning as it provides a means of modelling the interactions between water vapour and air. The moisture content of the air determines factors such as the level of humidity and directly affects the heat content and is therefore of paramount concern in the design of air conditioning systems.

In terms of air conditioning, psychometrics can be considered to be “the science involving thermodynamic properties of moist air and the effect of atmospheric moisture on materials and human comfort” [5].

There are five thermophysical characteristics which are commonly used in air conditioning which are often used in characterizing air conditioning systems. These are Dry-bulb temperature, Wet-Bulb temperature, Dew Point, Relative Humidity and Specific Enthalpy. Using a psychometric chart, it is possible to calculate any of these values from only two others (assuming a known pressure).

As these terms are widely used in this document, a brief definition of each term is given below.

2.3.1 Dry-Bulb Temperature

Dry-Bulb temperature is the most commonly used psychometric property and the easiest property to measure. It is the temperature reading from a standard thermometer when shielded from radiation and moisture and is one of the most important climatic variables when considering human comfort and building energy efficiency [6].

2.3.2 Wet-Bulb Temperature

Wet-bulb temperature is defined as the temperature of air if it were to be cooled by water evaporating into it, until the air is saturated. If the air is saturated with water, the wet-bulb value will equal that of the dry-bulb temperature. When the air is dry, the wet-bulb temperature will be lower than the dry-bulb.

In practice, wet-bulb temperature is measured using a temperature sensor encased in a moist wick. When surrounded by dry air, the moisture from the wick will evaporate and thereby cool the temperature sensor, yielding a lower wet-bulb temperature. When the air is saturated, no moisture

will evaporate off the wick and the wet-bulb temperature will be equal to the dry-bulb temperature [7].

Air speed must be taken into consideration when measuring the wet bulb temperature in this fashion. Wet-bulb temperatures should be taken at high air speeds (approximately 80km/h) but this proves to be difficult and often dangerous to implement. Wet-bulb temperatures taken at low air speeds (1 – 2 m/s) are referred to as screen temperatures, whereas wet-bulb temperatures taken at higher speeds (above 3.5 m/s) are known as sling temperatures.

2.3.3 Dew Point Temperature

The dew point temperature of air is the temperature at which the moisture vapor in an air sample would begin to condense assuming the pressure remains constant. The dew point temperature represents the temperature at which the air would be saturated with water vapor and is therefore always lower than, or equal to the dry-bulb temperature.

Dew point temperature can be measured using specialized equipment, but is more easily calculated using any of the other two psychrometric properties. Although it is a useful property in refrigeration, it is not commonly used.

2.3.4 Relative Humidity

The relative humidity can be thought of as the ratio of water vapor to moist air in a given sample of air. It is a dimensionless quantity and is often represented as a percentage. Relative humidity is defined as the ratio of the partial pressure of water vapor in the air to the saturated vapor pressure of water at a given temperature. When dealing with substances other than air and water, this ratio is known as relative saturation.

Relative humidity is an important property in the design of an air conditioning system. It is a property that is directly related to human comfort and is therefore often specified in the requirements for a building. The ASHRAE standards consider a relative humidity in the range of 40% - 60% to be healthy and comfortable [1].

Relative humidity is also pressure dependent and can therefore vary with altitude. It is therefore critical to know the atmospheric pressure when calculating the relative humidity and a number of companies supply capacitive-based devices for measuring relative humidity.

2.3.5 Specific Enthalpy

The specific enthalpy is defined as the heat content per unit mass. In air conditioning, the specific enthalpy of air is commonly used as it is the sum of the heat content of both the water vapor and the moist air. Specific enthalpy is usually specified in terms of kilojoules of energy per kilogram (kJ/kg) or in BTU per pound (BTU/lb).

Specific enthalpy is almost always calculated from the dry bulb temperature and either the wet-bulb temperature or the relative humidity. It is extremely useful when modeling the heat load of a building as it provides a means to assess the true heat content of air.

2.4 Components of an Air Conditioning System

There are four main elements in a typical air conditioning installation. These are the compressor, the condenser, the evaporator and the expansion device. An air conditioning plant can use a refrigerant-based system to provide either heating or cooling using the same equipment. It is important to keep in mind that a compressor-based refrigerant system does not create heat or cooling directly. It simply transfers energy between locations. In this manner, a 50 kW air conditioning system is capable of cooling a load of 200 kW.

For the sake of this discussion only the case of refrigerant-based cooling will be considered. In order to provide heating with the same equipment, the process is simply run in reverse. Devices heating in this manner are often referred to as heat pumps.

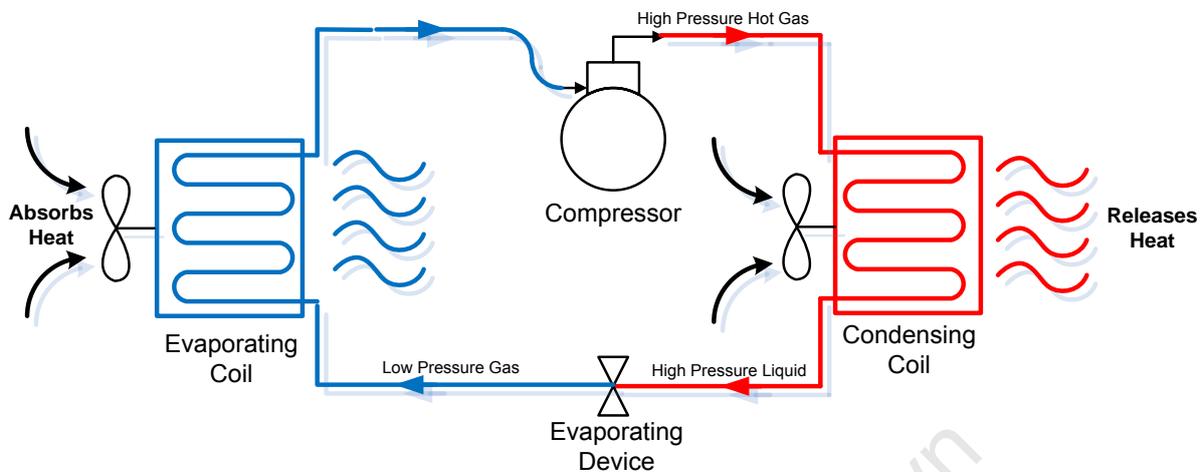


Figure 2-1: Components of a Typical Air Conditioning Installation

A typical air conditioning system consists of a closed loop of a refrigerant gas which is pumped through a series of coils by a compressor. A compressor is used to mechanically compress the refrigerant into a high-pressure gas. This gas then passes through a coil known as a condenser.

Condenser coils cool the refrigerant from a hot gas into a liquid which is then forced through an expansion device. The expansion device allows the refrigerant to expand back into a gas, which then passes through another coil known as the evaporator coil, absorbing heat in the process. The refrigerant, which leaves the evaporator as a cool, low-pressure gas is then fed back into the suction of the compressor.

There are a number of other components that are used in the air conditioning process to assist with the process described above. In many large air conditioning systems, it is impractical to pipe hot refrigerant gas around a building¹. In order to prevent this, water is used as a medium to transfer heat. This can be used on both the condensing coils or the evaporating coils and provides additional benefits. These systems are discussed below under their relevant sections.

2.4.1 The Compressor

The function of a compressor in an air conditioning system is to increase the pressure of the refrigerant before it passes through the condensing coil. There are two different means by which this can be achieved; using positive-displacement compressors or dynamic compressors.

A positive-displacement compressor works by reducing the volume of a gas through mechanical means, thereby increasing the pressure. Dynamic compressors work in a slightly more complicated manner by using angular momentum to create a pressure increase. Although the effect of these two mechanisms is the same, the means by which they need to be monitored varies significantly.

Compressors are often fitted with devices called unloaders, which essentially decrease the effective compressing power and therefore lower power consumption. These devices are often used to

¹ It is possible to circulate the refrigerant gas around a building using a technology known as VRV. In these systems variable-speed compressors are used to dynamically adjust the power consumption to meet the requirements of the building. Most suppliers of air conditioning chillers provide VRV devices but such installations are not common.

provide multiple stages of cooling in order to allow the air conditioning system to adjust to a lower building load. An unloader may be as simple as a mechanism that disables a piston in a small positive-displacement chiller or may involve special pressurization techniques on larger devices.

Different types of compressors also have different operating pressure ranges. Certain dynamic compression devices such reciprocating compressors work from -1 Bar to 2 Bar whilst screw compressors (a positive-displacement device) operate at much higher pressure and can never operate at a vacuum. Other factors, such as oil pressure range differ greatly depending on the type of compressor as certain compressor designs require a dedicated oil pump.

The parameters on a compressor that require monitoring include the suction and discharge pressures, the oil pressure (if an oil pump is present), the motor current, the oil temperature, the refrigerant temperature and in some instances, the bearing temperature if accessible.

2.4.2 The Condenser

The condenser is essentially a heat exchanger through which heat is removed from the system. It relies on a cooler medium to reduce the temperature of the hot gas pumped in by the compressor. The condenser can be cooled by using either air or water.

Air cooled condensers simply cool the gas by using a fan to push air over a set of coils. The coils are arranged in rows with the hot gas entering on the side closest to the fan to allow the maximum temperature differential to be maintained across the coil. The coils often have small metal fins attached to them to further promote the heat exchange with the air.

In many situations, air-cooled condensers are impractical due to either capacity or size restraints. Often plant rooms are located in the basement of a building and it becomes prohibitively expensive to place the condenser coils on the roof. In these situations a water-cooled condenser provides a better solution.

Water-based condenser coils use water instead of air to cool the condenser coil. The water is pumped over the coil and then to a device called a cooling tower, which lowers the temperature of the water. A cooling tower generally uses outside air to cool the condenser water and therefore cannot cool the water below the atmospheric wet bulb temperature. As an example, water may leave the condenser coil at 35° C and return from the cooling towers at 30° C.

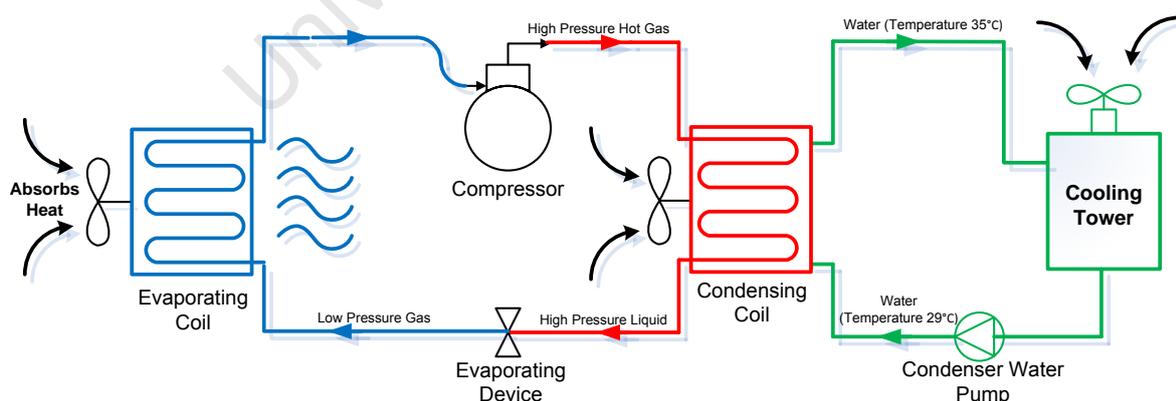


Figure 2-2: An Air Conditioning System with a Water-Cooled Condenser

Water-based condensers are generally more expensive than an air-cooled condenser as it requires a pump, piping and a cooling tower. The piping between the cooling tower and the condenser does not need to be insulated as the water is usually above atmospheric temperature and therefore any losses can be considered to be minimal.

The important parameters to monitor on the condenser are the temperatures of either the air or the water entering and leaving the condenser. Water flow across the condenser may also prove to be useful in certain circumstances.

2.4.3 The Evaporator and the Expansion Valve

The evaporator coil is also a heat exchanger and is the point which absorbs heat into the system. It has many parallels with the condenser and can be used to cool either air or water. When the evaporator is used to cool air directly it is referred to as a direct-expansion system. These are commonly found in smaller air conditioning units and packaged unit.

A large air conditioning plant needs to provide cooling for an entire building and therefore needs to distribute the cooling around the building. This could be accomplished with a direct-expansion unit which feeds the entire building through a series of ducts but the size and the airflow required through such ducts would make them impractically large. In order to overcome this, a chilled water based system can be used.

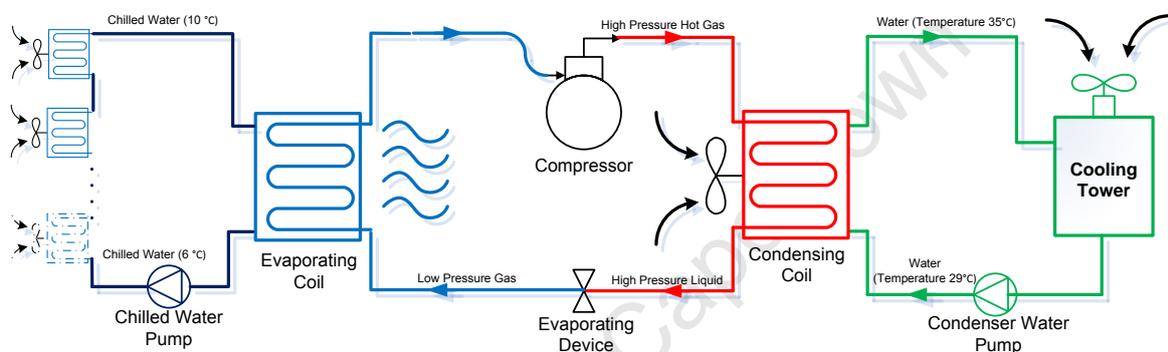


Figure 2-3: A Typical Air Conditioning System using Chilled Water

In a chilled water system, the air conditioning plant uses the refrigeration cycle to produce cold water instead of cold air. This cold water can then be pumped around a building and used by air handling units to produce cold air where required. The piping still needs to be insulated but leaks are not as serious as they would be if the refrigerant gas were piped around the building.

The chilled water temperature is usually kept in the range of 6° C. If the temperature becomes too low, the evaporator coils may freeze up and become ineffective. The temperatures of the leaving and returning chilled water is extremely important in determining the load and functioning of the air conditioning system and are therefore the two most important parameters to be monitored.

2.4.4 Chillers and Packaged Units

A chiller is complete air conditioning component which produces chilled water. Chillers usually contain compressors, an evaporator and a condenser with all the required control and safety elements built in. Both air-cooled and water-cooled chillers are available and are available in a wide variety of capacities. Chillers often contain multiple compressors connected internally in order provide the required capacity and implement all the necessary control needed to run them efficiently.

Chillers are popular as they provide a compact solution which benefits from the lowered costs of mass production. They are also standardized, which allows for easier maintenance. As a chiller is simply a pre-built system consisting of the elements described above, all the points mentioned for compressors, evaporators and condensers need to be measured on a chiller.

Packaged units take the concept of modularity one step further by incorporating the role of the air handling unit as well. These units essentially produce cooling directly and are intended to be placed

on the roof of a building to provide a one-step air conditioning solution. Generally they use a direct-expansion evaporator and can use either an air-cooled or water-cooled condenser.

Packaged units are a complete air conditioning system in a single package and therefore require all the parameters mentioned above to be measured.

2.4.5 Cooling Towers

Cooling towers are used as a heat exchanger for water-cooled condenser coils. These towers cool the water down to atmospheric temperature using a combination of heat and mass transfer. The incoming water is broken up to expose maximum surface area to the air, which is often forced through using a fan.

There are two types of cooling towers; open cooling towers and closed-circuit cooling towers. An open cooling tower (or evaporative cooling tower) exposes the condenser water to air directly by spraying it into the tower. As the water is exposed to the air it is cooled rapidly but acts as an air scrubber and can contaminate the water.

Indirect cooling towers do not expose the condenser water to the outside air and treats the condenser water as a closed system. These systems are less efficient but do not run the risk of contaminating the condenser water. This is particularly important on a system where the condenser coil cannot be cleaned.

Open cooling towers are both cheaper and more efficient than the indirect cooling towers but require some form of cleaning or water treatment. Cooling towers also run the risk of becoming infected with Legionnaires' disease if not properly treated [8]. For this reason, a comprehensive water treatment system needs to be implemented and should distribute both biocide and anti-scaling chemicals.

Cooling towers present a number of different challenges in order to properly monitor its health and efficiency. The pH and Total Dissolved Solids (TDS) of the condenser water can be used to ascertain the quality of the condenser water. A method to detect the presence of harmful bacteria in a tower would also be extremely useful. Finally, the water pressure supplied to the tower is also of importance as each tower is designed to operate optimally at a certain pressure.

2.4.6 Air Handling Units

In order to effectively air condition a large building, a chilled water system required to distribute the cooling effect across the entire building. This chilled water is produced in a central plant room and supplied to the building through a closed set of piping. Air Handling Units are devices that utilize the chilled water in order to provide cooling for a room.

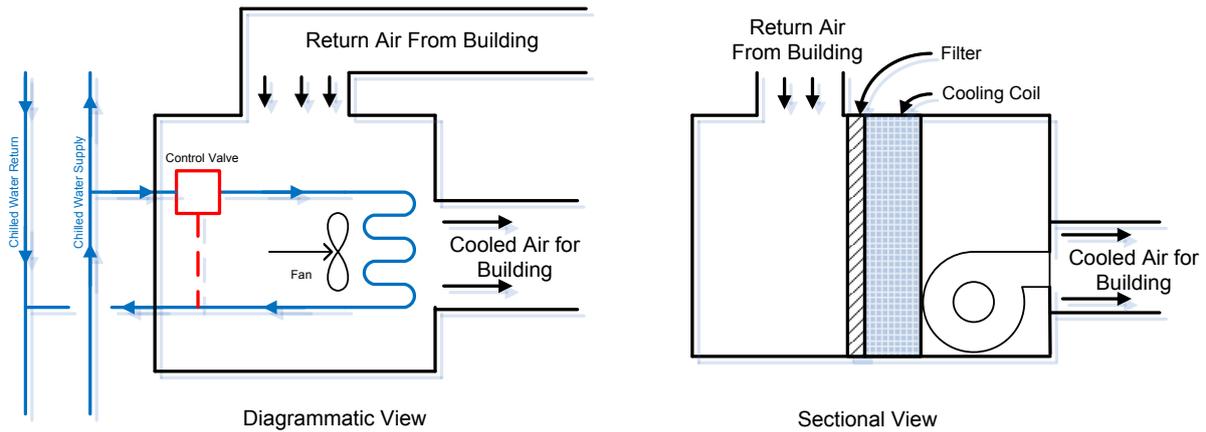


Figure 2-4: Diagram of a Simple Air Handling Unit

There are three basic components to an Air Handling Unit; a coil, a fan and a control valve. Every air handling unit contains a coil through which chilled water is passed. The air handling unit works by driving air over the coil in order to transfer the heat into the chilled water and thereby cool the air. A control valve of some sort is used to restrict the flow of chilled water through the coil and thereby control the temperature of the air exiting the unit.

A diagram of a basic air handling unit is shown in Figure 2-4. It shows both a diagrammatical representation and sectional view of a typical air handling unit. The sectional view shows an additional air filter which is not present on the diagram as these are only included to prevent dirt collecting on the fins of the coil.

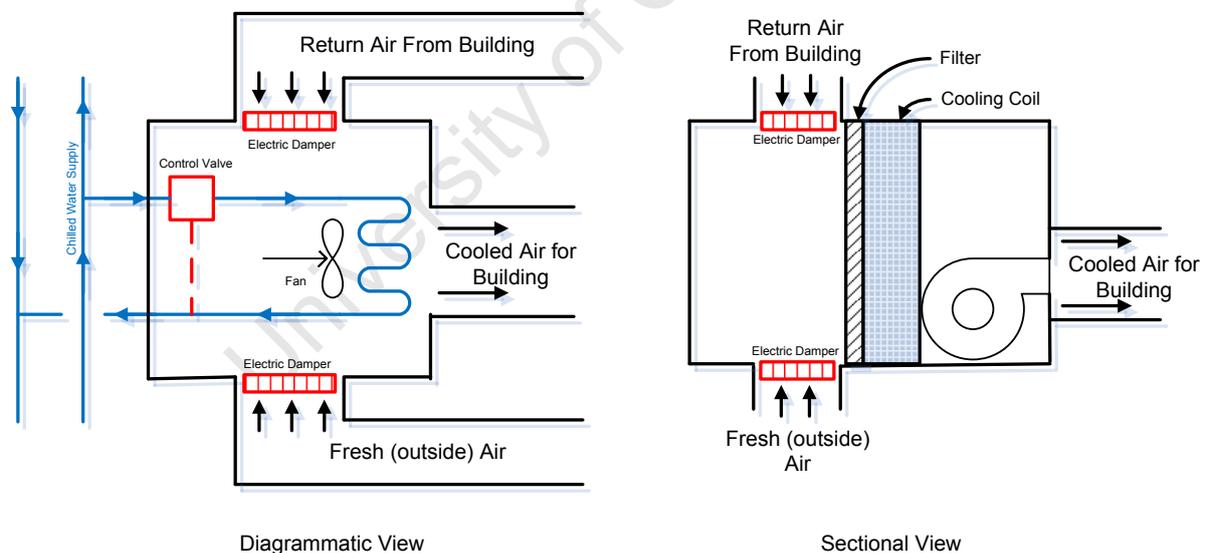


Figure 2-5: Diagrammatic and Sectional View of an AHU with Fresh Air Intake

Two different types of control valves can be used on the air handling unit. The first is a simple throttling valve, which limits the flow through the coil. The second method involves the use of a three-way valve, which enables a shunt (shown as a dotted line in Figure 2-4) to allow chilled water to bypass the coil. The choice of valve depends on the pump used to circulate chilled water around the building. If a constant speed pump is used, then three-way valves are required to maintain a constant flow. A variable speed pump is capable of handling the opening and closing of valves and therefore can handle the cheaper throttling valve.

Many air handling units are also responsible for introducing fresh air into an air conditioning system. Air handling units generally draw air from the building itself and therefore need a separate duct (or opening) in order to draw fresh air. Many air handling units have dampers on the outside air supply in order to control the amount of fresh air entering the unit (see Figure 2-5). This is done to allow for an energy-saving process known as an economy cycle.

The economy cycle comes into play when the outside temperature is cooler than the returning air from the building. This could occur on a cold day when the internal areas of a building will still require cooling. In such a case, it is far more energy efficient to use the cooler fresh air than to cool down the returning air from the building and the air handling unit would fully open the fresh air damper. On a hot day, when the outside air temperature is higher than the returning air, the damper may close to allow only the required amount of fresh air to enter the unit.

Air handling units have so many points to monitor that a special board was developed to handle them. The temperatures of the supply air, fresh air and returned air are all important parameters. The motor current on the fan is also useful in determining the action of the fan. The static pressure in the duct immediately after the fan is also important to monitor as is the differential pressure across the filter.

University of Cape Town

3 Motivation for the Design of a Remote Monitoring Platform

The motivation for the design of a Remote Monitoring and Diagnostics system is based on a report provided to the U.S. Department of energy regarding the potential energy impact resulting from building faults. This section provides a short summary of the report and uses the results to draw up a list of requirements for a monitoring system that addresses the shortcomings of current technologies.

The section then discusses the additional benefits such a technology could provide in a Southern African context and then provides a summary of similar and related projects that have been carried out around the world.

A number of fundamental decisions are drawn based on the requirements of industry and a short outline of the proposed system is then presented.

3.1 Summary of the TIAX Report to the U.S. Department of Energy

The U.S. Department of Energy commissioned TIAX LLC to produce a report on the energy impact of improper control and diagnostics systems in large buildings. The report, entitled “Energy Impact of Commercial Building Controls and Performance Diagnostics: Market Characterization, Energy Impact of Building Faults and Energy Savings Potential” [9], forms the basis for the motivation behind this thesis.

The report, which was presented to the Department of Energy in November 2005, attempts to quantify the potential energy savings resulting from improved building controls and diagnostics. In addition, the report also provides insights into the reasons behind the limited market penetration of such technologies. These topics are discussed over four sections which collectively provide a thorough assessment of the entire controls and diagnostic market.

The first section of the report discusses the energy implications of various faults that can occur in a commercial building and provides an estimate of the resulting energy loss. The report then analyses the energy saving potential of various control and diagnostics systems providing insight into both the practical and economic implications of such systems. The third section of the report deals with the barriers that prevent the widespread adoption of control and diagnostic systems. The final section of the report seeks to present the driving factors that influence the purchase of such systems.

3.1.1 The Energy Impact of Faults

TIAX LLC conducted a thorough literature review in order to determine the various faults that could result in wasted energy in a typical building. A fault is defined as a deviation from the designed condition of a system or piece of equipment which results in a drop in operation efficiency. TIAX LCC compiled a list of over 100 faults that commonly affect large building and, based on industry feedback, refined the list down to thirteen items.

The report calculated the energy impact on a national scale, producing the energy impact in terms of quads. A quad is a unit of energy used by the Department of Energy in the U.S. when discussing national or international energy matters. A quad is defined as a quadrillion (10^{15}) BTUs or 1.055 exajoules (1.055×10^{18} joules)². For purposes of perspective, the total energy produced globally in 2004 was equal to 443 quads [10].

² 1 quad is also equivalent to 2.931×10^{11} kWh (approximately 0.3 Trillion kWh) of primary energy. This is roughly equivalent to the energy contained in 172 million barrels of oil. Source: US Energy Information Administration (<http://www.eia.doe.gov/>).

The report estimated the total energy cost of each fault by taking into account the relevant energy loss resulting from the fault, the frequency of occurrences and average percentage increase in energy consumption resulting from the fault. The results are reproduced in the table below:

Fault	Fault Type	Energy Consumption (quads)
Duct Leakage	Air Distribution	0.30
HVAC Left on When Space Unoccupied	HVAC	0.20
Lights Left on When Space Unoccupied	Lighting	0.18
Airflow Not Balanced	Air Distribution	0.070
Improper Refrigerant Charge	Refrigeration Circuits	0.070
Dampers not Working Properly	Air Distribution	0.055
Insufficient Evaporator Airflow	Air Distribution	0.035
Improper Controls Setup / Commissioning	Controls	0.023
Control Component Failure or Degradation	Controls	0.023
Software Programming Errors	Controls	0.012
Improper Controls Hardware Installation	Controls	0.010
Air-Cooled Condenser Fouling	Refrigeration Circuits	0.008
Valve Leakage	Waterside Issues	0.007
Total		0.993 quads

Table 3-1: Energy Impact of Faults in Commercial Buildings in the U.S.

Commercial buildings in the U.S. consume approximately 17.1 quads of primary energy annually during the years preceding the publication of this report [11]. As the estimated likelihood of these faults occurring is broadly distributed, the actual energy implications range from approximately 2% to 11% of all energy consumed by commercial buildings in the United States.

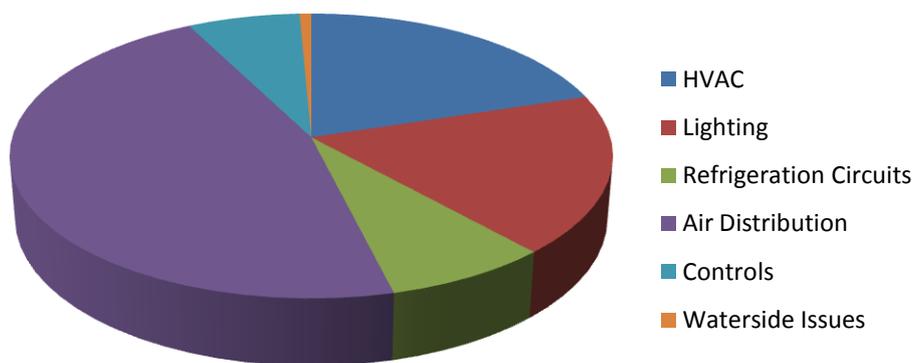


Figure 3-1: Breakdown of Fault Losses by Fault Type

The chart presented in Figure 3-1 above shows the faults listed by the type of fault. The chart clearly shows that the majority of the fault losses occur from faults relating to Air Distribution, HVAC and lighting. The report suggests that this data should be used to help direct and prioritize diagnostic development efforts.

3.1.2 The Energy Saving Potential of Control Diagnostic Approaches

The report makes an important distinction between the benefits gained through diagnostics and those gained through more sophisticated control systems. Diagnostics processes are used in order to reduce operating inefficiencies that cause the system to operate below the designed level of performance. The report describes this as the gap between the “sub-par performance” and the “as-intended” performance.

The addition of sophisticated control systems seeks to improve performance of a plant beyond the designed performance of the system through dynamic and reactive processes. This results in an additional saving as the plant operates above the “as-intended” performance.

The report discusses various control and diagnostic processes in great detail and calculates an estimate of the potential energy savings that can be obtained with each technology. The report breaks the technologies down into three main categories; diagnostics, control and enabling technologies.

The results of the report indicate that diagnostics methods, such as commissioning and whole-building diagnostics are the most efficient methods to reduce energy-related losses. Commissioning is an in-depth process involving systematic testing and improvements to ensure the efficiency of a system. It is often only provided by a consultant at an additional fee and is therefore not always performed. Retro-commissioning is the process of commissioning a plant after it has been constructed and is often more effective as it takes into account the degrading performance of components. The report states that commissioning can reduce all fault-related energy consumption, with the exception of duct leakage.

Whole-building diagnostics involves the use of specialized equipment that continually monitors a plant against a performance baseline to ensure that it is running at optimal efficiency. Whole-building diagnostics can be considered to be a form of continual commissioning and can therefore also dramatically reduce the fault-related losses of a system.

Control technologies do not produce the same levels of improvement as they are intended to improve performance beyond the designed level. Enabling technologies include items such as sensors which do not provide a direct energy-saving but are a required cost in order to improve the efficiency of diagnostic and control systems.

It is important to note that diagnostic procedures do not directly save energy but prevent the wastage of energy when equipment underperforms. Another important point to note is that the presence of a diagnostics system is only effective when an infrastructure exists capable of rectifying the problem. Many building managers lack the time to confirm diagnostic problems and therefore negate the benefit of the diagnostics system [12].

3.1.3 Barriers to Entry in the Control and Diagnostics Market

The third section of the report analyses the barriers that prevent the widespread adoption of control and diagnostic technologies. This section is particularly relevant to this thesis as it forms the basis for the requirements of the designed system.

The report points out that energy costs constitute only a fraction of the total annual costs of a building. As a result, there is no impetus to improve performance from a cost perspective. Further complicating the matter is that tenants often pay for the utilities in a rented office space, thus allowing the building owner to forgo costs related to energy efficiency and pass the costs directly to the tenants.

The report highlights the following common barriers facing control and diagnostic approaches:

- Cost / Payback Uncertainty
- Difficulty of Implementation
- Initial Costs (up-front investment)
- Lack of Industry Awareness
- Reliability Concerns
- Unproven Performance

The cost of a control or diagnostic system is of particular concern to a building manager or designer. Control systems are often one of the last aspects of a building design and suffer from parallelized nature of modern building design and construction. As a result, the decision regarding controls and diagnostic equipment is often made at a time when the budget is already stretched thin. Building designers and managers are therefore hesitant to commit to new technologies, unproven systems or technologies with long-term returns.

Building designers and managers also tend to regard any new technology with a certain degree of scepticism and tend to assume that new technologies are inherently unreliable. Most people involved in the HVAC industry are predominantly mechanical engineers and therefore prefer the tried and tested methods of control [13]. Another complication arising from the implementation of sophisticated controls is the need for qualified and capable personnel to implement and maintain such systems. The current industry structure does not allow for the hiring of such individuals [14].

The report also mentions other factors, such as the lack of interoperability between systems and conflicts with regulations such as building code. Additionally, due to the low market penetration and lack of skilled operation at sites where sophisticated controls are installed, there is a negative perception surrounding the performance of such systems.

3.1.4 Key Opportunities in the Controls and Diagnostics Market

The final section of the report delves into the opportunities that currently exist in the controls and diagnostics arena. The report clearly states that “in all cases, building controls and diagnostics can greatly increase their value by enhancing the core business of the building – be it employee, productivity or sales of goods and services. The increase in value comes from increased productivity resulting from an optimal working environment.

The report suggests that, despite the small total cost of energy in a building, it is still a good angle in which to push advanced controls and diagnostic systems. Energy costs are an increasingly large part of a building's operational costs and rising energy prices are making any opportunity for conserving power more attractive.

Building maintenance accounts for over 20% of the operational costs of a building and therefore any diagnostic or control mechanisms to reduce these costs can prove attractive to building owners. Energy Management and Control Systems were initially marketed as a tool for improving performance and reducing expenses related to maintenance.

The report mentions one final opportunity that is particularly relevant to this thesis. It states that technologies that can be easily installed on existing systems without incurring overwhelming expenses would be extremely desirable. Currently, the cost of retrofitting an existing installation with a sophisticated control or diagnostics system is often much more expensive than incorporating one during the design and construction phases.

3.2 Defining System Requirements to Match Shortcomings in Current Technologies

The TIA report lists a number of barriers and opportunities that exist in the Controls and Diagnostics market. The overarching goal of this thesis is to design a system to address all the issues

affecting current diagnostic tools whilst taking advantage of the numerous opportunities present in the market.

The barriers to entry were used to identify three critical areas in which the system designed must exceed the offerings of other companies in the market. These areas are the total system cost, the ease of installation and the maintainability of the system.

The total system cost includes both the initial costs of installing such a system and the operational expenses involved in maintaining and running it. One issue that limits the adoption of current technologies is the cost and difficulty associated with retrofitting a system on an existing installation. The high initial cost of all these systems also limits their applicability to large installations. Medium-sized buildings or buildings with decentralized utilities (such as power and air conditioning services) are simply not able to justify the cost of the existing systems.

The benefit of lowering the initial cost of the system has the knock-on effect of reducing concerns over the financial return on investment. Current Building Management Systems are faced with the task of having to justify a substantial up-front investment with a long-term remuneration plan. A less expensive system would have to prove more realistic return on investment in terms of energy and maintenance savings.

The ease of installation and operation are two critically important criteria for the building manager. Complicated equipment often requires experienced and skilled technicians to install and configure the system. These technicians are often only responsible for the software or firmware configuration and still require additional people to perform the physical installation.

A system that is both easy to install and requires no maintenance on behalf of the building owner is extremely desirable and therefore forms one of the core requirements of the design. There is a noticeable shortage in the number of personnel capable of operating and maintaining modern building management technologies and therefore an autonomous system would potentially remove the need for such a person in an organization.

3.3 Benefits of a Remote Diagnostic Tool in a Southern African Context

There are a number of additional benefits to providing a remote diagnostic tool which are of significant importance in a South African context. These problems are not as common in more developed countries in Europe and the Americas and often do not receive the same focus. Many of these factors were taken into account when designing the system and are therefore discussed here.

3.3.1 Providing Expertise to Remote Locations

One of the primary benefits of a remote monitoring and diagnostic tool is the ability to provide expertise to remote installations. Air conditioning installations in remote areas are often run to failure as a result of the high costs of bringing in skilled engineers and technicians to diagnose and repair problems. A remote monitoring system would allow a remote installation to be monitored in the same fashion as any other building, and therefore allow planned preventative maintenance to be implemented.

The proposed system uses GPRS technology which is widely deployed throughout Africa. Cellular support often exceeds that of traditional land-line communications due the relative ease with which it can be deployed. This allows the proposed monitoring system to be installed in places which do not have fixed infrastructure, such as broadband internet, or even telephone lines.

An example of where such a system would be useful is in Nigeria, which has an established cellular infrastructure and a fast-growing business sector [15]. Due to the exceedingly hot temperatures, with mean average temperatures exceeding 35° C in summer [16], air conditioning systems are often operated at close to their limits and can suffer as a result. There is also a shortage of skilled air

conditioning technicians and engineers, which makes maintenance difficult. The proposed monitoring system would provide a method of solving many of these problems.

Additionally, the monitoring hardware has been designed to be easily installed and not require a specialized technicians or tools. This would allow the system to be distributed to remote areas and installed with minimum cost.

3.3.2 Improving the Lifespan of Air Conditioning Installations

Proactive maintenance directly increases the lifespan of a plant by reducing unnecessary wear on equipment. The proposed monitoring and diagnostic system would be capable of detecting problems before they result in damage to equipment or machines and therefore extend the lifespan of existing equipment.

Current building management technologies are capable of providing such diagnostic information but the high cost of both installing such systems and retrofitting a system on an existing air conditioning installation limit the applicability of such a solution. The low cost of the proposed monitoring and diagnostics system coupled with the ability to install it on existing installations make it far more suitable to the types of air conditioning systems installed in South Africa.

3.3.3 Improving the Efficiency of Air Conditioning Installations

Energy consumption is becoming an increasingly important concern in South Africa due to potential electrical supply shortfalls and the rising cost of energy. As a result, there is increased pressure on business and industry to reduce energy consumption. This translates into increasing pressure on building managers to ensure that large systems such as air conditioning plants function optimally.

The energy consumption of an air conditioning installation can be reduced by ensuring that the system performs at the designed specifications (i.e. 100% efficiency). Poor maintenance, unexpected wear and improper controls are just a few of the factors that can cause a drop in plant efficiency. All faults need to be continually checked and repaired in order to maintain such high levels of efficiency.

A second method for reducing energy consumption is to ensure the output of the air conditioning plant matches the demands of the building. Air conditioning systems are often not correctly calibrated and therefore operate at capacities far greater than demanded by the building. Through monitoring of the equipment, the building and the outside conditions, the proposed diagnostics system can accurately determine if the air conditioning plant is providing the correct amount of cooling or heating.

Certain buildings employ an economy cycle in their air handling units, which allow the use of outside air instead of refrigerated air when conditions outside are cool enough. This method is very effective in reducing energy consumption but currently there is no method to determine if such economy cycles are actually functioning as intended. These systems rely on automated dampers which are often never checked and therefore may not open when required. Special hardware was developed to ensure that such systems can be adequately monitored.

3.3.4 Health and Safety Benefits

The monitoring system designed is also capable of accommodating other sensors such as pH and Total Dissolved Solids (TDS) which allow the system to track other factors in the air conditioning system.

Cooling tower water can be monitored to detect the level of scaling and biological build-up. If left unchecked, cooling water tower can develop bacteria that can be dangerous to public health [17]. Although sensors to detect the presence of bacteria such as Legionella do not currently exist, the monitoring of factors such as TDS can be used to regulate a water treatment protocol which dramatically reduces the risks posed by such bacteria.

Sensors to detect the concentration of CO₂ and O₂ in the air are also supported and would allow the system to monitor the quality of air provided. This would allow the monitoring system to determine the adequacy of ventilation and the overall quality of the air provided. This is especially important in older buildings as ventilation systems often rely on inline fans which are rarely ever checked.

The system also supports various refrigerant gas detectors which are traditionally used to detect leaks in the refrigerant lines. These sensors often provide additional outputs, such as an output proportional to the concentration of the refrigerant, which can be incorporated into the monitoring system to ensure the safety in the plant room.

3.4 Fundamental Design Choices Based on Market Requirements

A number of important design decisions were made prior to the design of the system. The first decision was to initially target HVAC systems, as they account for the majority of the energy losses in a commercial building. The system was designed to support additional equipment at a later stage, but the focus of the design centred on air conditioning equipment.

The second decision was to provide only a diagnostics service and not to implement any control functionality. This decision was made to keep the cost of the system low and to allow full operational autonomy without impacting existing systems. The primary focus for the system is therefore to reduce the energy losses associated with sub-par performance and to improve equipment lifespan through continuous monitoring.

Divesting the control functionality from the system allows it to be more easily installed in existing buildings. The lower cost of the system also makes it more attractive to small and medium-sized buildings, where existing control systems can be left intact.

The third decision was to perform the diagnostics at a central location. Each system would transmit data to a centralized server where diagnostic information would be generated and processed. The goal would be to provide the diagnostics as a service and allow building managers to rely on external expertise to interpret the result. Essentially, the system would allow buildings access to skilled engineering expertise and experience for both the generation and interpretation of their diagnostic information.

Finally, it was decided that proprietary hardware needed to be developed to prevent the system from becoming reliant on external technologies. The system should act as a transparent layer placed upon an installation and not interact with the controls or existing systems. Interoperability with existing systems (primarily to access their sensor resources) would be explored during development of the initial system.

The following table outlines the system requirements grouped by category. The three broad categories mentioned in the previous section are included along with modularity. As a result of the required scalability, it was decided that the system needed to be extremely modular to allow it to be cost-effective in all situations. This also allows buildings to deploy the technology in stages, allowing for more flexible budgeting for such a system.

Requirement	Implications
Low Cost	<ul style="list-style-type: none"> Designed to be as inexpensive as possible. Must be cost-effective to retrofit on an existing installation. Must be cost-effective, even on small installations. Must be modular to allow for phased deployments. Must have low operational costs. Minimize cable length required to save cost.
Modularity	Flexible enough to cover an entire building with a single system.

	<p>Rugged enough to handle an industrial environment such as a plant room.</p> <p>Support for alterations and extensions to an existing system.</p>
Ease of Installation	<p>Installation should require only basic wiring ability (apart from the mechanical installation).</p> <p>Hardware designed in a foolproof and logical manner.</p> <p>No configuration must be required on-site.</p> <p>The system must provide power to entire system from a single point.</p> <p>Must support long cable lengths to reduce any limitations on the placement of sensors.</p>
Maintainability	<p>System should operate autonomously.</p> <p>Must not require any configuration after complete power down.</p> <p>Must not require any periodic maintenance.</p> <p>Must provide some level of self-monitoring and diagnosis.</p> <p>Must be independent of the existing installation.</p>

Table 3-2: Sensor Platform Requirements

3.5 Related Projects

3.5.1 The University of California's Remote Monitoring and Control Project

In July 1996, the University of California published a paper on a prototype system capable of performing remote monitoring and control of air conditioning systems [18]. The aim of the project was split into two phases with the first phase being the implementation of a monitoring system for multiple sites from a single location. The second phase of the project was to implement remote control of these installations.

The system was designed to utilize an existing Energy Monitoring and Control Systems (EMCS) present in the target buildings and to develop a number of technologies to extract, encapsulate and transmit that data to a central site, referred to as a Remote Building Monitoring and Control Centre (RBMCC). This centre would then analyse the data, compare the data to simulations and produce diagnostic data and visualizations.

The data would be transmitted over the internet using standard TCP/IP functionality. A gateway device was required to marshal the data from the EMCS system onto the internet. Common Object Request Broker Architecture (CORBA) protocols were used along with a relational database in order to present a unified interface for the remote controlling device.

In a subsequent paper, entitled "Object Lessons Learned from a Distributed System for Remote Building Monitoring and Operation", the researchers discuss the outcomes and experiences with the system [19]. The system was operational for over one year on two different buildings and gathered useful data which was used to model chiller behaviour [20].

The paper discussed a number of concerns that arose during the implementation of the system. Concerns relating to security and privacy were raised as the researchers were unable to successfully implement security and authentication services in their system. This is particularly important when implementing remote control of plants as unauthorized access would result in potential damage to equipment and services.

The researchers also experienced performance problems with CORBA which resulted in increased complexity in the design. This introduced concerns over the scalability of such a system. The team also encountered a number of problems relating to time zone differences and unit conversions. The researchers conclude that prototype was successful in proving the viability of their system.

The system proposed in the papers mentioned above has similar goals to that of this thesis. Both systems revolve around the concept of transmitting sensor information to a central location to allow for monitoring and diagnostics operations to be performed. The differences appear when considering the method in which the data is gathered and transmitted. Unlike the University of California's system, which required an existing EMCS, the system outlined in this thesis includes all the equipment required to acquire the sensor data.

Another difference is in the method by which the systems upload data. The system details in this thesis uses a specially developed GSM link in order to allow remote deployment in almost any situation.

3.5.2 Monitoring and Analysis of an Absorption Chiller at a University in Spain

A team of researchers at the University Rovira I Virgili (URV) in Tarragona, Spain developed a system to monitor an absorption chiller³ located in one of the university buildings [21]. The team built a monitoring and analysis tool capable of gathering data from the plant and storing it in a database.

A software application, referred to as the "local supervisor" was developed to act as an interface to the building control systems. It provided a visual tool to inspect and control various aspects of the air conditioning plant. Using a Java Client/Server application, the "local supervisor" transmits energy data to a remote server where it is placed in a MySQL database.

The team was able to extract valuable information from the database in order to draw up a list of conclusions regarding the efficiency and suitability of the air conditioning installation. Data was collected during the summer of 2002 and used to assess the energy balance and operation efficiency of the installation.

The paper makes a number of interesting points regarding the need for monitoring systems. It mentions the effectiveness of energy simulation tools but points out that a lack of real operational data hinders their usefulness [22].

3.5.3 GSM-based Monitoring and Control of Photovoltaic Power Generation

A team of researchers at the University of Rome developed a GSM-based system for the remote control and monitoring of stand-alone photovoltaic installations [23]. Photovoltaic panels are becoming an increasingly popular method of supplementing existing power networks and to provide power for devices that cannot be easily connected to the power supply grid. These devices include traffic lights, street signs and remote radio installations.

As a result, these locations are often hard to access and therefore the ability to remotely monitor and exert some form of control would increase their lifespan and improve their suitability in remote areas. The researchers chose to use GSM due to its widespread coverage and relative low costs. Specifically, the researchers chose to implement the monitoring and control using the short messaging service (SMS) technology.

The team successfully implemented their system in both a laboratory setting and in a real-world situation. The team concluded that the system was successful in demonstrating both the applicability and the benefits of introducing remote control and monitoring in such systems.

The system designed in this thesis also implements communication using GSM technology. Unlike the photovoltaic system, the system detailed in this document does not use SMS technology but

³ The method by which an absorption chiller produces a cooling effect differs from the definition of a chiller given previously. Adsorption chillers utilize other energy sources, such as heat or solar energy, in order to produce a cooling effect. These devices do not operate with a compressor but rather rely on other techniques, such as evaporative cooling in order to produce a cooling effect.

rather the General Packet Radio Service (GPRS). SMS technology is effective for transmitting data but is limited to 160 characters per SMS [24] and can result in unpredictable delays between transmission and reception.

Currently, the costs of sending an SMS are much higher than the cost of transmitting the same quantity of data using GPRS. This is a general trend in the international telecommunication industry.

3.5.4 Development of a Data Acquisition System for Remote Monitoring of Renewable Energy Systems

Researchers at the University of Crete in Greece developed a remote monitoring system for renewable energy plants [25]. The motivation behind the development of such a system was to provide the ability to monitor plants in inaccessible areas. The researchers developed a client/server application that was able to gather readings and make them available to users via the internet.

The system described had three main components; A collector, a server and an interface applet. The collector consisted of a data acquisition system running on a PC. The collector would gather data from wireless RF sensors and transmit these readings to a common server. Multiple collectors could communicate with a single server. Clients would then access the server using a Java applet and view the readings.

The entire system was implemented in Java in order to be platform independent. Each component however requires a PC in order to operate. The system is very similar, in functionality, to the system proposed in this thesis. Both systems allow independent sites and provide a mechanism to remotely gather and interpret data.

The system proposed in this thesis differs in the fact that it removes the requirement for a PC and utilizes GPRS technology to provide a data connection. The benefit is that the system is autonomous and does not require any additional infrastructure on site. A potential downfall is that it may not be as suitable in remote areas where cell phone reception is unavailable.

3.6 The Proposed Monitoring and Diagnostics System

In order to meet the varied requirements described in chapter 3.2, a complete monitoring system was designed. It was decided that a system which encompasses the entire monitoring and diagnostic process would provide a better integrated and reliable solution. For that reason, the design of the system starts with the physical hardware responsible for acquiring sensor information and then proceeds to discuss all the hardware, software and firmware responsible for ultimately producing a final diagnostic report.

3.6.1 Structure of the Monitoring System

In order to accommodate the limitless variety in the design of air conditioning installations, a modular system was proposed. Such a system would be expandable to suit any installation by simply adding more modules as required. It was decided that each site would require one device that would be responsible for uploading data to the diagnostics server. This device would be called a Host Controller.

A second device was then needed to acquire the sensor readings. This device would need to be placed physically close to the quantity requiring measurement. These devices became known as Sensor Interface Boards.

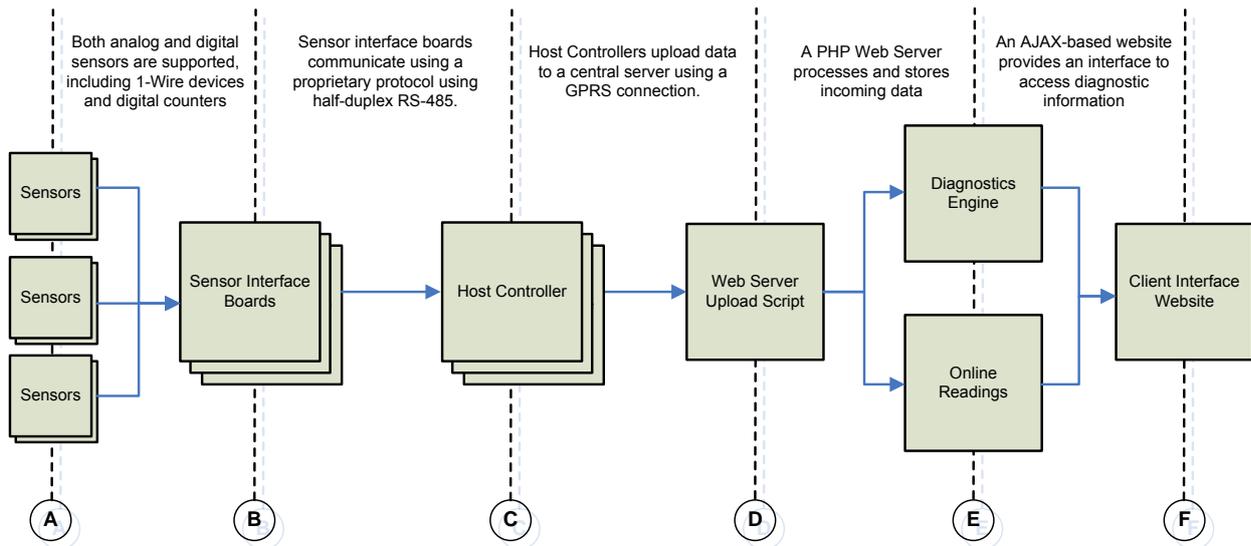


Figure 3-2: Structure of the Monitoring System

The structure of the proposed monitoring system is shown in the figure above. Each Host controller can support multiple Sensor Interface Boards, each of which can support multiple sensors. All the Host Controllers upload data to a single Upload server, which makes the data available to the diagnostic engine. The details of the Host Controller and Sensor Interface Boards are discussed in more detail below:

3.6.2 The Centre of an Installation: The Host Controller

The Host Controller serves as the centre of each installation. One is therefore required on every site requiring monitoring. The Host Controller manages a network of Sensor Interface Boards from which it will acquire readings and then upload them periodically to the diagnostics server. The Host Controller needs to operate autonomously and should not require any input from an operator in order to perform its function.

In order to simplify the installation, the Host Controller also should not require any configuration on site. The configuration must be done either prior to the device being dispatched or remotely via SMS or GPRS. In order to accomplish this, a special protocol needs to be developed in which the Host Controller can dynamically discover all the attached Sensor Interface Boards and configure them without any configuration or operator input.

The Host Controller should also incorporate a GSM module in order to upload data via GPRS. This would allow the upload process to be tightly integrated into the functioning of the Host Controller. The Host Controller should also use the upload process to calibrate its Real Time Clock, further reducing the amount of configuration required.

3.6.3 The Branches of an Installation: The Sensor Interface Boards

The Sensor Interface Boards are responsible for actually acquiring the sensor data. The concept is to develop a number of these boards to monitor specific pieces of equipment and allow an infinite number of devices to be attached to each Host Controller. In this manner, the monitoring system would be expandable to handle any size plant by the addition of Sensor Interface Boards.

Unfortunately, air conditioning equipment is often located at distant points around a building and therefore the Sensor Interface Boards need to support being placed far apart. Different configurations of the board will be required to monitor different pieces of equipment and as a result,

a framework consisting of hardware and firmware must be developed to aid in the development of different Sensor Interface Board configurations.

3.6.4 The Proprietary Network Protocol

There exist a number of protocols which are used in the building controls industry to communicate between devices in an industrial environment. One of the most relevant protocols is BACnet™ which is a communications protocol designed by ASHRAE to provide a standard communication infrastructure between integrated controls systems. The ultimate goal of BACnet™ is to provide a common communications interface between the various building management technologies provided by different vendors [26].

It would seem that BACnet™ should be logical choice for the communication protocol on a monitoring system. BACnet™ is an object-oriented system and is capable of communicating on four different physical interfaces. Unfortunately, BACnet™ is a very high-level protocol and requires that all the basic features must be implemented on every device. It is believed that this is one of the main contributors to the slow uptake of the technology [27]. BACnet™ is also a proprietary technology and is not an open standard. Whilst open-source projects do exist, the viability of such a solution is not clear at present.

It was felt that a proprietary protocol should be developed to suit the exact requirements of the monitoring system. As no control would be performed, the protocol could be optimized for performing monitoring functions. For these reasons, a complete protocol, encompassing a physical and network layer needs to be designed and tailored to providing a reliable and configuration-free network.

The system should also be designed in such a way that a BACnet™ gateway could be designed to provide some level of interoperability between the two protocols.

3.6.5 The Online Diagnostics Server

The Online Diagnostics server would be responsible for analysing incoming readings and producing meaningful reports and statistics based on the data. The purpose of the diagnostics system is to detect problems and provide possible solutions.

An air conditioning diagnostics system had already been developed prior to the work done in this thesis and could easily be adapted to utilize the diagnostic information being received from the Host Controllers. This data could then be displayed to the client via a webpage or through paper reports.

4 Sensor Selection and Installation

4.1 Pressure Sensors

Pressure readings often provide the most insight into the functioning of an air conditioning system and there are generally two different ranges that need to be measured. The first range is a relatively high pressure required when measuring the pressures in the gas lines of a compressor. The exact range varies with the type of machine being used and the gasses involved and may even involve a vacuum. The second pressure range is extremely low and is used to measure the static pressure in ducts.

Although both these devices measure the same physical quantity, they are so fundamentally different that they are treated as separate sensors. The unique nature of both the sensors and the means by which they are installed is discussed in the following sections. It is important to note that some of these sensors may need to measure pressures below atmospheric pressure, extending down to a vacuum. The means by which these values are obtained can differ greatly from measuring pressure above atmospheric [28].

One important characteristic common to both sensors is the various means by which a pressure can be measured. The common terms are absolute, gauge and differential pressure. Differential pressure is the simplest of the three and involves a device with two pressure inputs. The value returned is simply difference in the pressure values between the two inputs. Differential pressures are useful for measuring pressure drops over such items as filters.

Gauge pressure is the pressure relative to atmospheric pressure. It is the value one would expect to read on a mechanical pressure gauge and can be considered to be a differential pressure with one port exposed to the ambient air pressure. It is important to note that this pressure therefore fluctuates with altitude.

Absolute pressure is simply the pressure relative to zero pressure. It is not as commonly available as gauge and differential pressure and is used primarily in measuring barometric pressure [29]. The choice of sensors does not impact its functioning in the system but it is important to note which value is used as the same pressure can be represented with three different values. Most of the pressure sensors ordered were of the gauge variety to allow easy cross-checking with the existing mechanical gauges.

Another confusing issue surrounding pressure sensors is the wide variety of units used to express the pressure range. These consist of both imperial and metric units and include Pounds per Square Inch (PSI), Pascals (Pa), Atmospheres (Bar), Millimeters of water (mmH₂O) and Millimeters of Mercury (mmHg). The notation most convenient for the specified application will be used in the rest of the document and will be clearly specified. A conversion table of commonly used factors is presented in the appendix.

4.1.1 Pressure Sensors for High Pressures

High-pressure sensors measure one of the most important physical quantities in an air conditioning installation, the suction and discharge pressure of a compressor. These values, coupled with the oil pressure and the current, provide a deep insight into the functioning, efficiency and state of a compressor and can be used to diagnose both current and future problems.

There are a number of requirements placed on the high-pressure sensor. Firstly, it must be of a robust and sturdy construction as it is one of the few direct interfaces between the monitoring system and the installation equipment. If the device were to fail, it could potentially release refrigerant gas, which would then damage the compressors and may even pose a serious health risk

as some gasses are toxic or corrosive. For this reason, the burst pressure of the sensor must be carefully considered.

The second factor that needs to be taken into account is that the sensor is exposed to a refrigerant gas containing oil vapour. Certain refrigerant gasses are corrosive and may either corrode the sensor or become contaminated by the presence of the sensor. For this reason, it is critical to only utilize sensors that can withstand the various gasses currently in use.

These two factors have the effect of dramatically raising the price of the pressure sensor to the point where it is the most expensive single item in the system. As a result, care was taken to ensure that the cost of the sensor is kept to a minimum without sacrificing safety or reliability.

4.1.1.1 Application of High-Pressure Sensors in Air Conditioning

As mentioned previously, high-pressure sensors are most commonly installed to measure the pressure on the gas lines in a compressor. Oil pressure in compressors is also monitored where available and appropriate⁴. Chillers often contain multiple compressors, which can be connected in a variety of different configurations and therefore may have common suction or discharge pressures. In these devices, only a single pressure sensor can be used to monitor the common line.

Another area in which high-pressure sensors can be employed is to measure water pressure in various points in the system. An example of where this may be useful is in the piping between a water-cooled chiller and its cooling tower. The pressure entering the cooling tower is an important factor that directly determines the efficiency of the cooling tower as the spray nozzles inside operate optimally at a specified pressure.

4.1.1.2 Sensor Technology used in High-Pressure Sensors

The pressure sensors considered for use employed either resistive or capacitive methods of quantifying the pressure. The resistive sensors use the mechanical strain on a set of four resistors configured in a full Wheatstone bridge in order to obtain the pressure value. The Capacitive sensors use a similar principle but base the value on the capacitive change rather than the resistive change. Resistive sensors have a longer lifespan and better ageing characteristics than the capacitive sensors but are generally more expensive. It is important to note that the raw output of the sensor is usually not linear and is subject to changes in temperature. Most sensors include circuitry to internally correct these errors.

As with most sensors, the pressure sensors could be ordered with a number of different output options. These include the standard 4 – 20mA current outputs and the 0 – 10V DC output. Both of the configurations are used extensively by the many small controllers present in an air conditioning installation and are easily available.

The most pressing concern arises from the fact that the sensor needs to be in direct contact with the gas and therefore needs to be installed on the gas lines. A qualified pipe fitter is required in order to weld the correct fittings into the pipe. This incurs both the cost of labour and any cost associated with downtime as the system needs to be switched off and the gas pumped out in order for the sensors to be installed.

Due to the design of the sensor interface boards, most sensors ordered use a 5V DC supply with an amplified output in the range 0.5 to 4.5 V. Although other configurations involving both current and

⁴ The need to monitor oil pressure depends on the type and configuration of the compressor. Centrifugal compressors have an oil pump, which pumps oil onto the bearing. The oil pressure is therefore always a value between the suction and discharge pressure. Screw-Chillers, on the other hand, do not have an oil pump and therefore the oil pressure is not needed.

voltage can be implemented (with slight modification to the hardware), the above output characteristics were preferred.

4.1.1.3 Installation of High-Pressure Sensors

One predominant factor which was overlooked during the design of the system was the effort required to install such sensors on a pre-existing system. On a new installation, it is relatively easy to have extra T-Junctions installed where appropriate, but installing the system on a pre-existing installation poses a number of problems.

A possible solution to this problem is to install the sensor on a section of the system which can be isolated through a shut-off valve. Almost all systems with a compressor have mechanical gauges to allow the operator to check the state of the equipment. These gauges occasionally need to be replaced and are usually installed with an isolating valve so that they can be removed without pumping down the compressors. A small amount of refrigerant caught between the gauge and the valve is lost when the valve is disconnected, but this is negligible.

Sensors have been installed on this segment between the gauge and the isolating valve. This approach makes the installation much easier on certain machines but a skilled pipe-fitter is still required to cut and install a split in the gas line. The unit must still be turned off whilst the sensor is being installed but the compressor need not be pumped down. Unfortunately the space surrounding the gauges is very limited on many chillers and may often be very difficult to access. It therefore becomes extremely difficult to install the sensors or the appropriate fittings. This technique is very effective when it can be successfully applied.



Figure 4-1: Method to attach a Pressure Sensor using a Piercing Valve

The second solution proposed was to use a special valve known as a piercing valve. These valves are specifically designed for use in existing installations and consist of a bracket that fits around a pipe, a special pin that pierces into the pipe and a one-way valve on which a sensor can be attached. The valve is fastened around the target pipe and tightened. This forces the pin through into the pipe which allows gas to fill the chamber behind the one-way valve. The sensor can then be fitted onto the connection using a special fitting which depresses and opens the one-way valve when connected.

The benefit of installing the sensor using a piercing valve (as shown in Figure 4-1) is that it does not necessarily require a pipe fitter. Piercing valves are not generally liked as they have a tendency to

leak. A variation of the valve exists that is welded to the pipe to reduce the likelihood of it leaking but this somewhat defeats the point as a pipe fitter is then required. One interesting point to note is that if the valve were to leak, the diagnostics system would be able to detect it.

4.1.1.4 Electrical and Process Connection on the High-Pressure Sensors

Despite the choice of an analogue output, the distance between the sensor and the sensor interface board must be kept as short as possible. The original sensors were manufactured with a 2m cable directly connected to the back of the sensor during manufacture. This resulted in the need to install the sensor prior to wiring the system as the cable could not be detached from the sensor.

A second supplier of sensors was found and provided the sensors with a Packard connector. These connectors are extremely rugged and allowed the cable and the sensor to be supplied separately. Although it has no impact on the design of the system, the small change resulted in a simpler and less problematic installation of the sensors.

The process connection usually refers to the nature of the connection between the pressure sensor and its target application. A ¼" MPT (male pipe thread) connection was preferred due to the abundance of small fittings readily available.



Figure 4-2: Two Different High Pressure Sensors in the Range of 0 - 300 PSI

The above figure shows the two different pressure sensors that were used in the system. Both sensors have identical characteristics and have a ¼" MPT thread connection and a three-wire electrical connection. The sensor at the bottom of the image was the first sensor used in the system and was specially manufactured with the long cord. The sensor at the top of the image was sourced from a different manufacturer and supplied with a detachable (and durable) connector.

4.1.1.5 Acquiring Readings from High-Pressure Sensors

As the sensor outputs a voltage proportional to the pressure, the ADC on the microcontroller was used to digitize the value. Conditioning circuitry was included to properly condition the signal for the ADC. The specifics of this circuitry (which depend on the various configurations and versions of the Sensor Interface Board) are discussed in greater detail in section 5.

The pressure sensors can draw up to 10mA each when powered as they provide a constant output. Sensor Interface Boards can host up to four pressure sensors and therefore the pressure sensors are

only powered when required. This is performed using a set of external transistors and a control line from the microcontroller.

In order to take a reading, the host controller averages the readings from the ADC over a period equivalent to 50 Hz (or 60 Hz, depending on the country) in order to remove any noise from the electricity supply. It was felt that this was especially important as air conditioning installations involve high currents, large inductive loads and unpredictable electrical interference.

4.1.2 Pressure Sensors for Low Pressures

The low pressure sensors proved to be difficult to source and obtain. These devices are required to measure pressures in a much lower range than present in the gas lines and are often larger mechanical devices. A number of companies offer appropriate sensors which are intended to be used with specialized equipment such as building management systems or isolated control systems.

Low pressure sensors are required to measure the static pressure in ducts and across devices such as filters or fans. In variable-volume systems, it is very important to measure the static pressure in each duct as it gives a good indication of the state of the diffusers on that segment of duct. Differential sensors can be installed to measure the pressure drop over filters which is a good means of determining whether or not the filter needs to be cleaned (most filters specify a maximum pressure drop across them to indicate that they need cleaning).

The majority of the low pressure sensors found operate in a very similar manner to the high-pressure devices. All the sensors output a proportional value whilst power is applied, allowing the same acquisition techniques can be used to obtain readings. Ideally, the sensor would output a voltage in the range of 0.5 – 4.5 V DC but models with an output in the range of 0 – 10 V were accommodated through the use of additional interface circuitry but had the unwanted effect of introducing a higher degree of error into the system.

The installation of the low pressure sensors is often much easier than the high pressure devices. As the pressure is negligibly low and the medium is only air, drilling into ducts directly can be performed without minimal concern. Often a small plastic tube can be used to connect the sensor to a location such as a duct or space after a filter.

4.2 Temperature Sensors

Temperature is one of the most important characteristics in air conditioning. Temperature measurements are taken in many different points throughout the air conditioning process and of course, in the areas being air conditioned. The requirements for the temperature sensor vary wildly from one application to another and therefore a rugged and versatile solution was required.

After much consideration, it was decided to use a 1-Wire[®] digital temperature sensor from Dallas Semiconductors. The motivation for this choice is discussed in the application detail below. These sensors are low cost, easily available and can be implemented with very little external circuitry.

4.2.1 Applications of Temperature Sensors in Air Conditioning

A large air conditioning plant has numerous separate sub-systems that each requires a number of temperatures to be monitored. These sub-systems are fundamentally different from each other and require specialized consideration when developing a means to obtain accurate and meaningful temperature information.

As there are a number of different points at which temperature needs to be measured, each of which presents a different set of requirements, a discussion of the more important applications is presented below.

4.2.1.1 Monitoring Chilled Water Temperature and Cooling Tower Water Temperature

Many large air conditioning systems use chilled water to distribute cooling around a building. This is accomplished through a closed set of pipes running through the building. The water in this system is cooled by chillers in a central plant room, often located at the top or bottom of a building. The temperature difference between the leaving and returning chilled water is extremely important as it gives insight into both the functioning of the chiller and the load placed upon the building. When combined with other factors such as the outside air temperature and humidity, it characterizes both the conditions in the building and the demands placed on the cooling plant.

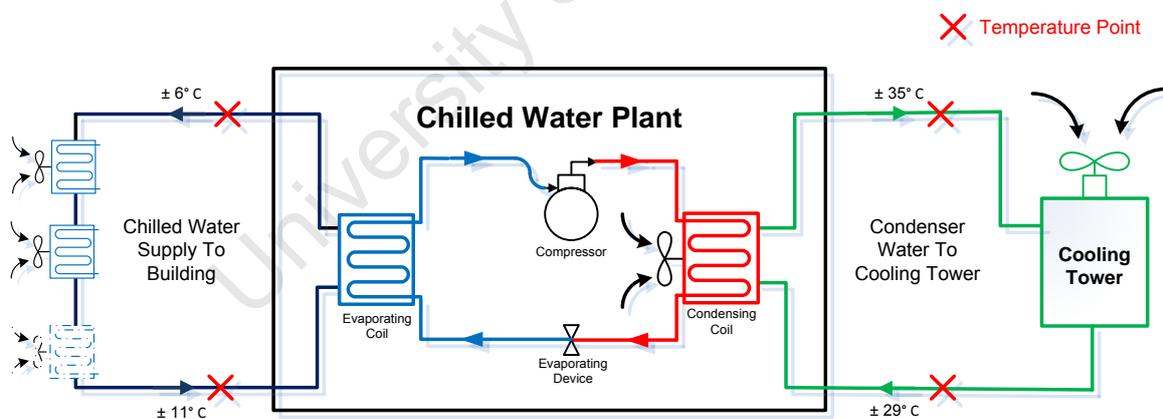


Figure 4-3: Diagram showing Chilled Water and Condenser Water Temperature Points

Another area where it is desirable to measure water temperature is found on water-cooled chillers as shown in the figure above. These machines require water to be circulated through cooling towers which lower the temperature of water through evaporation. The temperature of the water in these pipes is another indication as to the functioning of the chiller and also the efficiency of the cooling towers.

Chilled water varies from around 6°C to about 10°C . The water in condensing water pipes can vary quite substantially depending on the outside condition but is often in the range of 12° - 35°C . In both cases, the water temperature cannot be measured on the surface of the pipe but must be measured using a sensor pocket mounted in the water flow. These pockets need to be installed by a

pipe fitter but most existing installations will include numerous additional pockets for mounting thermometers and pressure gauges.

There are a number of concerns relating to installing sensors in such environments. The first and most pressing concern is due to the fact that the pipes often contain water which is much colder than the surrounding air. This can cause condensation to form on the pipe or sensor pocket, which may corrode the sensor. The degree of condensation may not be visible on the surface of the pipe as all chilled water piping is generally insulated, preventing condensation forming on the visible surface.

The sensor pockets are hollow cylinders, sealed at one end and attached to an industrial pipe connection on the other. The sealed end of the pocket is placed into the water stream through either a hole in the pipe or directly into one of the branches of a T-piece. The sensor is placed in the hollow cylinder and sealed so that changes in the water temperature directly affect the sensor.



Figure 4-4: Sensor Pockets Installed using a T-Piece (left) and Welded into a Pipe (right).

In order for this to occur, a suitable potting compound is required to properly seal the pocket. The compound needs to be thermally conductive but not electrically conductive. The concern is that moisture could become trapped around the sensor and could potentially corrode it. In practice though, this does not seem to be the case, even in sensors without any potting compound.

The number of water temperatures required to characterize a system can vary greatly depending on the configuration of the plant. Often multiple chillers will share a common cooling water system and therefore lessen the number of condenser water sensors required. Most buildings have only a single chilled water system and therefore a common chilled water supply and return for the entire building. Unfortunately, most chillers are only connected to this common pipe by a pump and therefore still require their own chilled water sensor.

4.2.1.2 Measuring Ambient Air Temperature

Ambient air temperature is the easiest temperature to measure as it is always located in air conditioned spaces where both the humidity and temperature ranges are known. These sensors are used to monitor temperatures inside the building and can be scattered randomly around the building or located at specific points of interest. A common example of the need for such a sensor is

in a server or network room in a large building, where the temperature should be continuously monitored.

These sensors need only to be exposed to the ambient air and therefore can be mounted as needed.

4.2.1.3 Measuring Air Temperatures in Air Handling Units

In certain air conditioning systems, it is often important to monitor the temperature of the air at various points in the air handling units. The function of these units in a large building is to use the chilled water supplied by the chillers to cool air taken from the building. This air is then distributed through ducts to the rooms and partitions inside the building.

Air handling units also draw some air from the outside in order to ensure that the building receives fresh air. Certain air handling units include an economy cycle which allows the unit to adjust the amount of outside air taken depending on the temperature. This is used in the case where the outside air is cooler than the air inside the building and therefore cooling can be accomplished without requiring the chilled water.

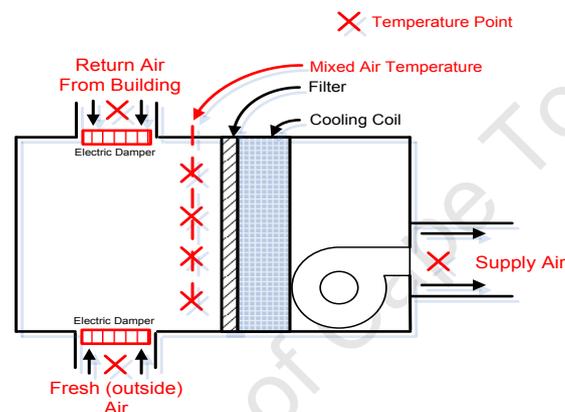


Figure 4-5: Temperature Points in an Air Handling Unit with an Economy Cycle

In devices such as these, it is important to know the temperature of the outside air, the temperature of the return air from the building and the temperature of the air leaving the air handling unit. The outside air temperature may in fact vary between air handling units in the same building as these devices may take the outside air from light wells or even ducts inside the building. These sensors can therefore be similar to either the ambient air temperatures or the more rugged construction of the outside air temperature sensors.

Often it is desirable to measure the temperature of the turbulent air in certain section of an air handling unit. This could be the case where air arrives from two inlets and must pass through a filter. The temperature of the air in the space in front of the filter is often very useful from a diagnostic point of view but cannot easily be measured with a single sensor. A possible solution is to use an array of sensors evenly distributed through the area and take an average temperature. It was felt that the averaged value was a better representation of the actual temperature and the system was designed to allow for such configurations.

4.2.1.4 Measuring Outside Air Temperature and Wet-Bulb Temperature

Measuring the outside air temperature is critically important in an air conditioning system as it is a good indicator of the expected load on the building. When combined with the humidity (or wet-bulb temperature) of the outside air, the time of year and the angle of the sun, these values can be used in predicting the loading of both components of the system and the system as a whole.

Outside air temperature is more difficult to monitor as it must be exposed to the outside air. Originally these sensors were placed at the air intake for the building as to reflect the temperature of the fresh air entering the building as part of the air conditioning process. These sensors were often placed on the grills of fans and began to show signs of corrosion. The current strategy is to place the outside air sensors in a small plastic box with holes drilled into the sides and bottom and mount this on the wall near to the air intake.

There are a number of different methods for measuring the wet bulb temperature. It can be calculated from other psychrometric factors such as humidity or dew point, if such sensors are readily available. Another method to measure wet-bulb temperature is to place a temperature sensor in a water-soaked wick and measure the temperature. The water on the wick must be exposed to the temperature of the area in which the wet-bulb is desired.

In the first prototype system, this was accomplished by constructing a small well with a reservoir in which the bottom of the wick was suspended. This allowed the wick to remain wet and accurate wet-bulb temperatures to be obtained. In future designs, this method was made obsolete as the wet-bulb temperature could be calculated from the humidity readings.

4.2.1.5 Monitoring Board and System Temperatures

Each Sensor Interface Board was designed to support one on-board temperature sensor to monitor the operating conditions in which the boards were placed. These temperature sensors are optional and are not required for the board to function. The inclusion of these sensors was primarily to allow early detection of possible problems arising from the fact that certain boards may be sealed in water-tight compartments.

Boards measuring current are a good example of where the on-board temperature sensor becomes important. The current is often measured using a current transformer, with the secondary winding connected to a large resistor capable of handling currents amounting to several amps. These boards were designed to be placed in a sealed box physically close to the operating equipment. The temperature sensor installed on the sensor interface board allowed the system to monitor the temperature inside the box, and as the power dissipated from the resistor was known, the rate at which the box itself dissipated heat.

These temperature sensors are identical to all the others used but are simply soldered directly to the sensor interface board. The actual reading is treated as any other and only distinguished by the diagnostic process on the server. The firmware on the board may use the value as an internal control or safety check but the host controller is not capable of distinguishing board temperatures from other readings. This was done to prevent the host controller becoming specialized and requiring different firmware for each configuration of sensor interface boards.

4.2.2 Sensor Technology used in the Temperature Sensors

There are a wide range of different commercial techniques for measuring temperature as it is one of the most commonly measured physical quantities. Generally these sensors fall into one of two categories; contact, or contact-less. Contact sensors require direct physical contact with the medium being measured whereas non-contact sensors circumvent this requirement by measuring the temperature using the emitted energy in the infrared spectrum.

During the design, only sensors requiring physical contact were considered as non-contact sensors do not produce good results with gasses due to their natural transparency [29]. Non-contact sensors were briefly considered for use in measuring water temperatures but were not pursued due to the high cost. However, once the preliminary systems were installed and the cost of installing sensor wells into water pipes was factored in, contact-less sensors are being reconsidered.

The temperature sensor chosen is a digital temperature sensor from Dallas Semiconductors which uses their 1-Wire® technology. The sensor, a DS18S20-PAR is a complete digital temperature sensor in a compact TO-92 package and has both an operating range and accuracy well within the specifications required for all the applications listed above [30].

One of the most compelling features of this sensor is the fact that it communicates and can draw power off a two-wire bus (ground and data). The protocol can be implemented using the pins of a microcontroller and only requires a weak pull-up resistor. This greatly simplifies the process of adding temperature sensors to microcontrollers as very little external circuitry is required. Another feature of the 1-Wire® protocol is that it supports multiple devices on a single bus, and therefore a single microcontroller pin can support a virtually unlimited number of sensors.

This feature is particularly useful in measuring aggregate temperatures inside air handling units (as discussed in section 4.2.1.3) as multiple sensors can therefore be manufactured into a single cable and will require no additional hardware on the sensor interface board.

The sensors also have additional functions such as a user-configurable low and high alarm conditions and 8-bit CRC generator. The sensor completes a temperature conversion in a maximum of 750ms but does not require any resources from the microcontroller during this time, which makes it very suitable for the reading acquisition process outlined in section 7.4.

The device is available in a special package in which the power supply pin is internally connected to the data pin to provide a simpler mechanical connection [31]. This greatly simplified the construction of the sensor as a bridging wire or small PCB was no longer required.

4.2.3 Acquiring Readings from 1-Wire® Temperature Devices

A microcontroller needs to be able to implement the 1-Wire® protocol in order to communicate with the temperature sensors. The protocol has strict timing requirements and therefore needs to be implemented with care. Special driver chips are available with an I²C interface which simplify the communication process but increase the total cost of the device. These devices may be used on the Host Controller or in special circumstances, but the sensor interface boards implement the protocol directly using port pins to allow maximum flexibility.

Each device contains a unique 64-bit ROM code which identifies it on the bus. The first stage in the process is to identify all the devices connected on the bus. If only a single device will be present, this stage is unnecessary and special commands can be used which greatly simplify the protocol implementation. Devices are detected using a special search algorithm which is outlined with code samples in a document from Dallas Semiconductors [32].

Command	Code	Description
SEARCH ROM	0xF0	This command is used to iteratively discover all slave devices on the bus through a process of elimination.
READ ROM	0x33	This command is only used when there is a single device on the bus. It can be used instead of the SEARCH ROM command.
MATCH ROM	0x55	This command, followed by a 64-bit ROM code allows a single device on the bus to be individually addressed.
SKIP ROM	0xCC	This command places all the devices on the bus in a state ready to receive a command. It can be used to address all the devices on a bus.
ALARM SEARCH	0xEH	This command is similar to the SEARCH ROM command except that only devices in which the alarm condition has been triggered will respond.

Table 4-1: Table showing the 1-Wire® ROM Commands

The table above outlines the basic 1-Wire[®] commands which are used to detect and address devices on the 1-Wire[®] bus. The microcontroller starts the process by issuing an initialization sequence to which devices on the bus respond to indicate their presence. The SEARCH ROM command is used to determine the ROM codes of all the devices connected to the bus. If only one device is present then the READ ROM command can be used to return its address.

The microcontroller then selects the devices which will respond to commands by using the MATCH ROM and SKIP ROM commands. The MATCH ROM command allows the microcontroller to select a device by specifying its ROM address. All the other devices will then stop responding until they receive the initialization command.

The SKIP ROM function provides the ability to send a broadcast message to all devices by placing each device in a state of readiness. If only one device is present on the bus, then this command can be used to issue commands to it without needing to store the ROM address. Implementing the protocol in this fashion greatly simplifies the code required.

There are also a number of ROM commands which are specific to the temperature sensor, these commands are used to initiate conversions and access the memory structure of the device. A table of the commands is given below:

Command	Code	Description
CONVERT T	0x44	This function initiates the temperature conversion process. If it is issued after a SKIP ROM command it will cause all the devices on the bus to begin the temperature conversion process.
WRITE SCRATCHPAD	0x4E	This command allows two bytes of data to be written to the internal scratchpad memory on the temperature sensor.
READ SCRATCHPAD	0xBE	This command returns the full contents of the scratchpad memory (all 9 Bytes) and can be stopped at any time with a reset.
COPY SCRATCHPAD	0x48	This command copies the 2 bytes of scratchpad memory (those written with the WRITE SCRATCHPAD command) to the internal EEPROM.
RECALL E ²	0xB8	This command copies the values from the EEPROM into the two bytes of memory accessible with the WRITE SCRATCHPAD command. It is performed automatically on startup.

Table 4-2: Table showing the 1-Wire[®] Command Specific to the DS18S20

Data on the temperature sensor is exposed through 9 Bytes of 'scratchpad' memory stored in the device. These bytes contain the result of the last conversion, two user definable bytes (which can be saved to EEPROM), two internally reserved bytes and additional values that can be used to attain a higher level of accuracy with the temperature sensor.

The values in memory can be access using the READ SCRATCHPAD command which instructs the device to return all 9 bytes of the 'scratchpad' memory. The master can terminate the reading process at any point as to only obtain the required data and the last temperature result is conveniently stored as the first bytes to be transmitted.

When the temperature device receives a CONVERT T instruction, it begins the temperature acquisition process and places the result in the scratchpad memory. As stated above, it is possible to issue this command to all devices on the bus simultaneously but the values can only be read back individually.

The device also has an alarm functionality which is implemented using the two user-configurable bytes of the scratchpad memory. This allows two 7-bit signed values to be stored, one to act as the high-trigger point and the other to act as the low trigger. If the temperature crosses these thresholds, the device will be placed in the alarm condition and respond to the ALARM SEARCH command accordingly.

These two values can be saved to the internal EEPROM memory using the COPY SCRATCHPAD function. It is important to note that there are special power requirements that need to be met when using this function under parasitic power. For this reason, the alarm functionality was never used in the system. A corresponding function is provided to recall the values from EEPROM although this is executed automatically on start-up.

It is important to note that these values can be used for general purpose values if the alarm functionality is not required. Each temperature sensor therefore can store two bytes of data to non-volatile memory if required. This function could be used to encode specific information onto sensors, possibly to allow them to become self-describing.

University of Cape Town

4.3 Current, Energy and Power Sensors

Energy consumption is one of the most direct measures of efficiency and is critical in the diagnostic processes run on chillers and compressors. Air conditioning systems contain a vast array of different electric machines that require a substantial amount of power. These include compressors, pumps, fans, and heating elements. Many of these devices are also 3-phase devices and therefore it is often important to measure other factors such as power factor in order to fully characterize the devices in question.

The currents drawn by the equipment in an air-conditioning installation can range from a few amps for small control systems to hundreds of amps in a large chiller. There are also a number of concerns relating to spikes and even the physical size of the cable becomes a problem. Increasingly there has been interest in expanding the energy measurement aspect of the system such that it can measure a building's energy load and break it down into components.

4.3.1 Application of Current and Energy Sensors in Air Conditioning Installations

Current measurements are extremely useful in the diagnostics system when combined with other readings. It can be used on virtually any device to monitor the efficiency, especially when compared with previous readings. An increase in current when all the other factors remain constant can be used to indicate problems such as clogged filters, damage to bearing or contacts or dropped phases.

Current is also critically important in characterizing the operation of a chiller as it is a very good indicator of the load placed on the chiller. Combining these values with knowledge of the configuration of the air conditioning system and the outside conditions can yield a very rough check of the efficiency and performance of the chiller.

At the moment the system only measure current on a single phase and therefore is mainly used to measure the current on each of the compressors in a chiller. A number of new boards are currently being developed which are capable of measuring three-phase current (and therefore real power and power factor) as well as special boards specifically designed to be mounted in electrical switchboards and capable of handling a wide range of current inputs.

4.3.2 Acquiring Current Sensor Readings

Unfortunately, the process to measure current involves both specialized hardware and firmware in order to obtain and process the current value. A current transformer is used to provide the raw measurement, with the secondary winding connected directly to a special Sensor Interface Board. As the devices being monitored can vary dramatically in size, the current transformers vary accordingly. A large power resistor on the sensor interface board can therefore be changed to accommodate a wider variety of current transformers.

As the current from the CT is alternating, the RMS voltage over the resistor needs to be converted to a DC value. A special IC from Analog Devices, AD536A was used that provides this functionality in a single package IC and is capable of handling signals with a DC offset [33]. This output is then buffered and clamped to a maximum of 2.7V before arriving at the pin of the microcontroller. The internal ADC converter is then used to obtain the value.

Due to the passive nature of the system, any changes in the current are automatically reflected up to the microcontroller such that all the microcontroller needs to do is perform another ADC conversion to obtain the latest value. Therefore the current acquisition process is extremely simple and mirrors that of the pressure sensor.

At the moment, only a single current per sensor interface board is supported. In order to add additional current sensors without requiring additional RMS to DC converters, a new board has been designed has a multiplexer before the RMS to DC converter. This solution is currently being tested

but does not allow simultaneous recording of multiple currents due to the settling time of the multiplexer chosen. Another alternative is to move the RMS to DC conversion to firmware and therefore remove the need to have an additional IC on the board.

4.3.3 Installation of Current Sensors into Chillers

The installation of current sensors is generally quite easy as it only requires a current transformer to be placed around the correct wire. Downtime is still required for the installation as the wire needs to be threaded through the current transformer. Care also needs to be taken as there are high currents involved and unconnected terminals on the CT may develop very high voltages across them.

The installation of a CT becomes more difficult when dealing with high-current cables. These cables tend to have a much greater diameter and often contain steel reinforcement. Even with a great deal of slack in the cable, it can still be very difficult to manipulate the cable to allow a CT to be slipped over it.

A possible solution to ease the installation in these cases is to use a split-core current transformer, in which a side of the current transformer can be removed in order for it to be installed without having to disconnect the cable. This type of current transformer is widely available but is much more expensive than a standard CT and is often far less accurate.



Figure 4-6: Current Transformers Affixed with Cable Ties in a Chiller

One interesting aspect that was originally overlooked was that most currents can be measured from the electrical switchboards which are generally installed close to each other. Therefore, it is possible to pick up a large number of different currents from multiple devices using a single board installed in the switchboard. This is the motivation behind designing a single board that can handle multiple currents.

4.4 Wet-Bulb Temperature and Humidity Sensors

At least two psychrometric properties need to be known in order to fully characterize air. Dry-bulb temperature is easily obtained using standard temperature sensor but another property, such as

wet-bulb temperature or relative humidity, is required⁵. There are a wide range of instruments that can measure relative humidity and wet-bulb temperature, but these devices do not integrate well into an autonomous monitoring system as proposed in this thesis.

The original design called for monitoring of the wet-bulb temperature by building a custom apparatus. A relative humidity sensor was later discovered that would integrate perfectly into the existing system and later replaced the wet –bulb temperature sensor.

Wet Bulb temperature can be measured by placing a sufficiently wet wick over a temperature probe. This method of measuring the wet bulb temperature is effective but difficult to automate in a reliable fashion. An early prototype system used this method in order to gain a rough estimation of the wet bulb temperature.

This was accomplished with a small dish containing water, the level of which was maintained with a small reservoir tank (essentially, a bottle with its top submerged in the dish). A wick was placed such that one end fell into the water. A temperature sensor was then placed in the other end of the wick. The apparatus provided an approximate estimation of the Web-bulb temperature

In order to improve on this, a humidity sensor was sourced that could measure the relative humidity of the air. The wet bulb temperature can be calculated (or at least estimated accurately) using the relative humidity and the dry bulb temperature. The humidity sensor sourced was a three terminal device with an output of 0.5 to 4.5V. The sensor therefore plugged directly into any of the pressure inputs on the sensor interface boards and only required a bit of calculation on the server in order to provide both the relative humidity and the wet bulb temperature.

Unfortunately, the humidity sensor corroded as a result of its location in the fresh air stream.

4.5 Running Time Monitoring

Although not technically a sensor, the running time of various devices in a plant room is of particular interest in the diagnostic process. The running time is used to ensure that devices are cycled correctly and to assist in the prediction of wear. It is therefore very useful when employing a strategy of planned preventative maintenance.

Air compressors provide a very good example of where running time is useful. Many old installations still use pneumatic control systems, which use compressed air to perform both the physical opening and shutting of valves and also implement a feedback control system. These pneumatic systems are supplied with compressed air from a central compressor which is usually located in the plant room.

As most of these systems were installed in the 1970s, corrosion causes the pipes, seals and joints to leak. By monitoring the running time of the compressor, the system can determine the rate at which the system loses air and can therefore monitor the development and severity of any future leaks that develop. Currently there is no other means of determining this without having a person watch the compressor over an extended period of time.

Some larger air compressors also have two small compressors feeding into a common tank. These compressors are supposed to be cycled in such a way as to allow each compressor to wear evenly. Unfortunately there is no method of determining when the compressors should switch other than using arbitrary time periods. Monitoring the running time of each compressor would allow the system to determine the best time for the compressors to switch and also to predict when then devices need servicing.

⁵ Three factors are actually required in order to calculate the remaining psychometric properties. One of these factors is atmospheric pressure, which is easily calculated based on altitude and remains constant.

5 Design of the Sensor Interface Boards

5.1 Overview of the Sensor Interface Boards

The Sensor Interface Board provides the connection between the monitoring system and the physical sensors. These are specially designed boards that initialize, calibrate and capture sensor data and then make it available to the rest of the system. Sensor Interface Boards are intended to be placed physically close to the devices or equipment on which sensors will be placed. This has the effect of reducing the distance that analogue signals need to travel before being captured and digitized.

Sensor Interface Boards also need to contain specialized circuitry to support the various sensors described in the preceding chapter. In order to keep the size and the cost of the board down, a number of different board configurations were designed. Each board appeared would have the same physical and logical connections to the rest of the system but would only acquire readings in a different fashion. This would allow the development of new Sensor Interface Boards without requiring any modification to the rest of the hardware or firmware in the system.

Although multiple configurations of the Sensor Interface Board were produced, they all share some common characteristics. The required features of each Sensor Interface Board are described below.

5.1.1 The Power Supply on the Sensor Interface Boards

It was decided prior to the development of the Sensor Interface Boards that a single cable, containing both power and data signals would be provided to each board. This would allow a board to be installed without requiring access to a power source. This is particularly useful when Sensor Interface Boards are installed in ducts or suspended ceilings, where power is not immediately accessible.

The power supply was chosen to be 24V DC due to the wide availability of such power supplies in existing air conditioning installations. The high voltage is also used to allow for the large voltage drops that may occur over longer sections of cable. Devices required a minimum of 5V DC in order to function as dictated by the RS-485 drivers. Many of the sensors used require higher voltages, which can also be supplied from the bus power supply.

Each Sensor Interface Board must therefore contain various regulators and protection diodes to safely produce the required voltages for the board. These typically include 5V for the RS-485 drivers and 3.3V for the microcontroller. The power supply must include adequate decoupling capacitors and protection against voltage and current spikes resulting from the electrical noise generated inside plant rooms.

It was originally decided to allow Sensor Interface Boards to support an external power supply in order to introduce a clean supply for the rest of the network. This would allow devices at the end of a long cable to use an external power supply instead on relying on the bus power supply. Although this system worked, it was no longer possible to power down the entire network from a single location (as in the host controller). For this reason a special power jumper board was developed to allow for this.

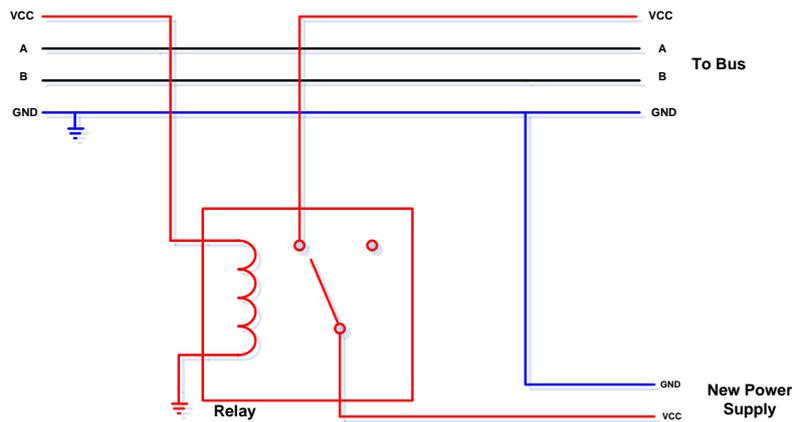


Figure 5-1: Schematic of a Simple Power Jumper Board

Figure 5-1 is a diagram of a passive power jumper board. It consists solely of a relay which uses the power supply of the bus to provide external power to the rest of the network. This has the effect of removing the new external power supply when bus power is disconnected, thus allowing the entire system to be turned off from the host controller.

More complicated power jumper boards are being considered which include a small microcontroller and are capable of monitoring such parameters as bus voltages and currents. These boards would provide the exact same functionality but allow for more flexibility in the implementation.

5.1.2 The Communication Hardware on the Sensor Interface Boards

Each Sensor Interface Board connects to the network using the same basic hardware configuration. This allows interoperability between every version and configuration of Sensor Interface Board produced. It also allows boards to be swapped and inserted at any point on the network. A diagrammatic view of the communication hardware is given below:

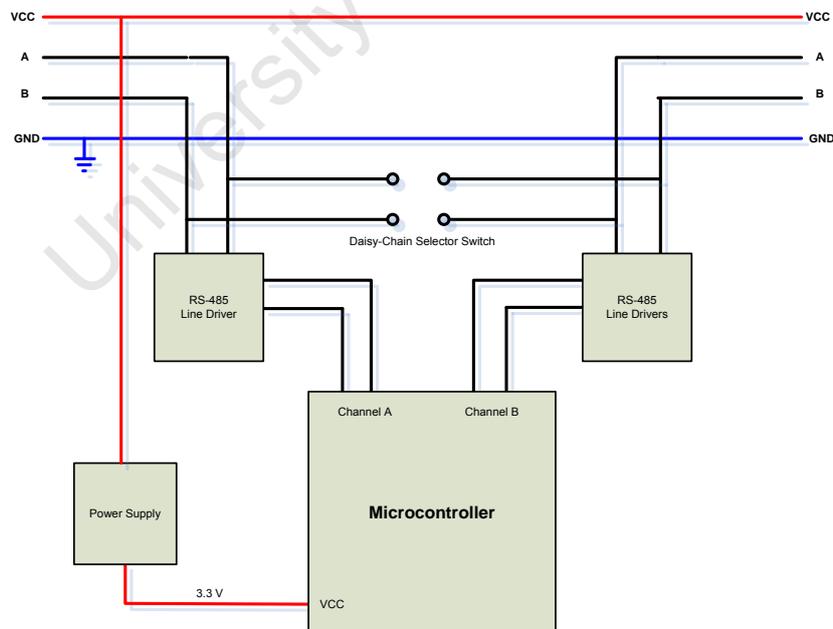


Figure 5-2: The Communication Hardware on the Sensor Interface Boards

The diagram shows the typical connections between the Sensor Interface Board and the 4-wire bus on which the network is based. Each side of the diagram represents one of the channels on the board. Power and ground are routed directly and do not pass through the microcontroller.

The design of the hardware originates from the development of the protocol and bus structure. This is discussed in much greater detail in Chapter 7. One of the more important points to note is that each board can be configured to work with either a daisy-chain network or a single wire bus network (hence the jumpers between the two data lines). When using the device in a single wire bus, only one of the line drivers needs to be installed. On a daisy-chain system, both line drivers are always required.

The line drivers are standard RS-485 drivers and include a terminating resistor (which is not shown on the diagram). The specifics of the controller are most important when using a single wire bus and a MAX3082 line driver was found to be suitable for all applications. The ICs were sourced in DIP packages and installed in an IC socket to allow them to easily be replaced if damaged.

The power supply to the board is connected to the power and ground line of the bus through a re-settable fuse. This is used to enforce the maximum power limitation placed on each board. Additionally, new boards are being fitted with trans-orbs across the supply to lessen the effect of high-voltage switching.

5.1.3 Lever Connectors and Mounting Holes on the Sensor Interface Boards

Special connectors were chosen for the Sensor Interface Boards in order to make the installation and wiring of the boards as easy as possible. These connectors feature a spring-loaded mechanism to hold the wire in place, removing the need for a screwdriver or a special crimper. Every connection on the board is required to use these connectors, resulting in a very quick and easy installation.



Figure 5-3: Lever Connectors and Mounting Holes on a Sensor Interface Board

The above image shows a Sensor Interface Board in a typical plastic housing. Notice how the connectors and the glands are lined up to make the internal wiring as neat as possible. In the above board, the bus is wired through the glands on the top and bottom of the box, with the sensors exiting through the glands on the side. Each PCB was carefully laid out to ensure that none of the tall components (such as the decoupling capacitors) were located in front of the connectors.

Asymmetrical mounting holes were used to ensure that the board is correctly installed in its plastic housing. This is important as it ensures that the glands and connectors align properly. The board is fixed with bolts running through the bottom of the board and secured with a nut (visible in the center of the image above). The boxes can either be stuck down using double-sided tape, or mounted directly to a surface by drilling through the back of the plastic housing.

5.1.4 The Board Temperature Sensors on the Sensor Interface Boards

A dedicated temperature sensor is included on each Sensor Interface Board in order to monitor its operating condition. These sensors are the 1-Wire[®] sensors mentioned in chapter 4.2.3 and are soldered directly to the board. On certain boards, these are used to measure ambient temperature and may be safely left out if not deemed necessary.

The motivation behind installing these sensors is to monitor the temperature inside the sealed plastic housings in which they are installed. Boards that measure current are often required to dissipate heat (as the current from the CT must pass through a resistor on the board) and it was felt that the temperature inside the box would be a good indication of developing problems.

Many Sensor Interface Boards also monitor the supply voltages. This is often required when using sensors that have a ratiometric output and therefore require a value for the supply voltage in order to produce accurate readings. Monitoring the supply voltages also provides some information regarding the state of the device and the on-board regulators.

Originally, it was felt that the boards should also measure the supplied bus voltage but this was not included due to concerns that it would allow electrical noise to couple into the processor (currently the only connection between the bus power and the board is through the power supply, which has been designed to protect the board against such occurrences).

5.1.5 The Firmware Framework on the Sensor Interface Boards

The firmware for the Sensor Interface Boards was designed in a very modular fashion to allow for the multitude of possible hardware configurations. The functioning of the device is divided into the three layers of the protocol, which is discussed in Chapter 7.2.2. These layers are the Messaging Layer, the Network Layer and the Sensor Interface Layer (which is the device-specific layer for the Sensor Interface Boards).

The Messaging Layer and the Network Layer are virtually identical for each board variation. Only the port mappings are different to accommodate the different multiplexed peripherals on each microcontroller. The Sensor Interface layer is the only layer that is board-specific and contains all the initialization, calibration and acquisition code for the board.

Two different versions of the firmware exist to support the two different addressing systems used in the monitoring system (see chapter 7.3). The firmware for Sensor-Based Address further increases the modularity by introducing a special structure to describe each sensor. This allows sensors and additional hardware to be added to a device with very limited alterations to the firmware code.

Devices on the bus are also expected to implement some form of watchdog timer to prevent a board failure from requiring a manual reset of the entire network. It has been found that power spikes can often cause the microcontrollers to stall or behave erratically. A watchdog timer can prevent this by ensuring a hardware reset when such an event occurs.

5.2 Sensor Interface Board Configurations

There are currently four different board configurations that have been produced. Each configuration was designed to suit a specific application and has undergone a number of different versions. Each board is named after the number of possible sensors that can be installed on it. Below is a short table that explains the naming convention.

Format of Board Name:	A{t} B{t} ...
Where	- (A, B....) are number indicating the number of supported sensors - {t} is the type of sensor from the list below
It is important to note that the number corresponds to the maximum number of sensors supported.	
Letter Abbreviation	Sensor Type
T	1-Wire [®] Temperature Sensors
P	Pressure sensors with output 0.5 – 4.5V
C	Current Input (with RMS to DC conversion)
L	Logic Input (either make/break configuration)
H	Humidity Sensor (with 0.5 – 4.5 V output)

Table 5-1: Naming Convention for Board Configurations

The boards were designed for specific monitoring applications in a typical large air conditioning installation. There are three broad locations at which sensor readings need to be acquired. These include boards installed on chillers, boards installed in Air Handling Units and boards installed in electrical switchboards. Each configuration is discussed below and includes details on the motivation behind the design, the specific sensors and the physical dimensions and requirements.

5.2.1 Design of the 4T1P Board

The 4T1P board was designed for use inside Air Handling Units where multiple temperatures and a single low pressure are required. These boards support either three temperature sensors and a pressure sensor or four temperature sensors. This is chosen by selectively soldering components as required. A temperature sensor requires a two-wire lever connector whereas a pressure input requires three inputs.

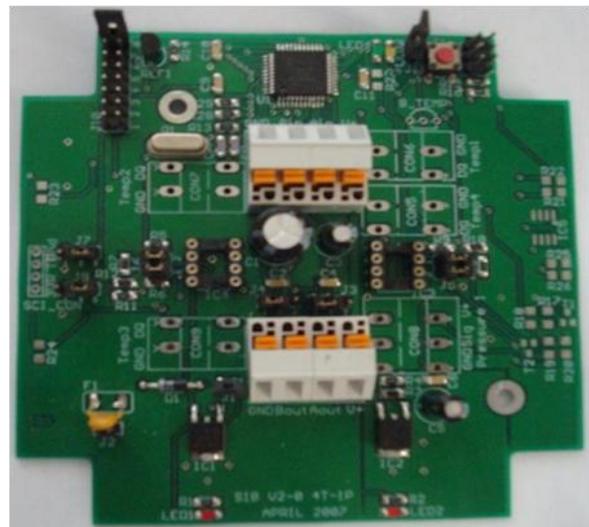
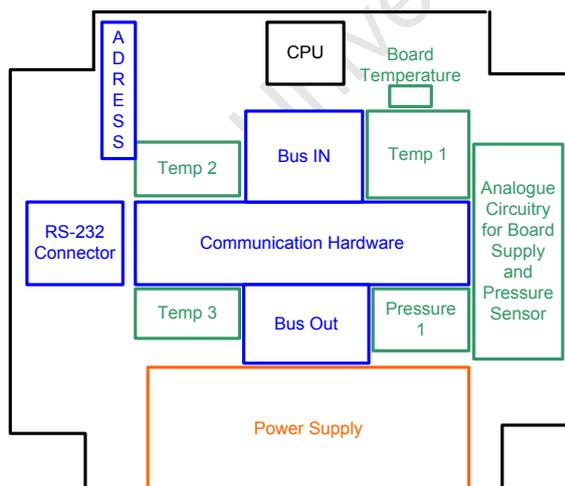


Figure 5-4: The 4T1P Sensor Interface Board

Although the board was designed for use in air handling units, it also used whenever temperature readings are required as it is the cheapest board to manufacture and produce. It is often used on

chillers to monitor the various water temperatures and also in ducts and open spaces to monitor the ambient temperatures.

The board contains a Freescale™ microprocessor which runs of a 16 MHz crystal. The internal ADC is used to acquire the reading from the pressure sensor and the 1-Wire© protocol is implemented directly from the I/O pins. The board also features an 8-pin header on the top-left of the board which was originally as a means of setting addresses but made obsolete by the implementation of the automated network discovery in the protocol (see chapter 7.3.3.1 and chapter 7.3.4.2).

The board was designed in a shape of a cross in order to fit inside the plastic housing selected for the boards. The board also contains a legacy RS-232 connection which was provided for debugging and development purposes and is therefore not used in normal operation.

In order to monitor the outside air temperature, a humidity sensor can be used to replace the low pressure sensor. Due to the compatibility between the outputs of the humidity sensor and the pressure sensor, this can be done with minimal effort, thus creating a version of the board known as the 4T1H board.

5.2.2 Design of the 3P1C Board

The 3P1C board was designed at the same time as the 4T1P board in order to provide an interface to the sensors inside the chillers. The board is very similar in design and layout but includes additional hardware to support both the 3 pressure sensors and a single current sensor. The board is also capable of handling a single temperature sensor if required.

The basic configuration of the board is identical to the 4T1P version. It has the same microprocessor, power supply, communication configuration, and connector layout. The board has special circuitry to handle the current input, including an on-board 1-ohm power resistor and a RMS to DC converter. These components also have additional power supply requirements.

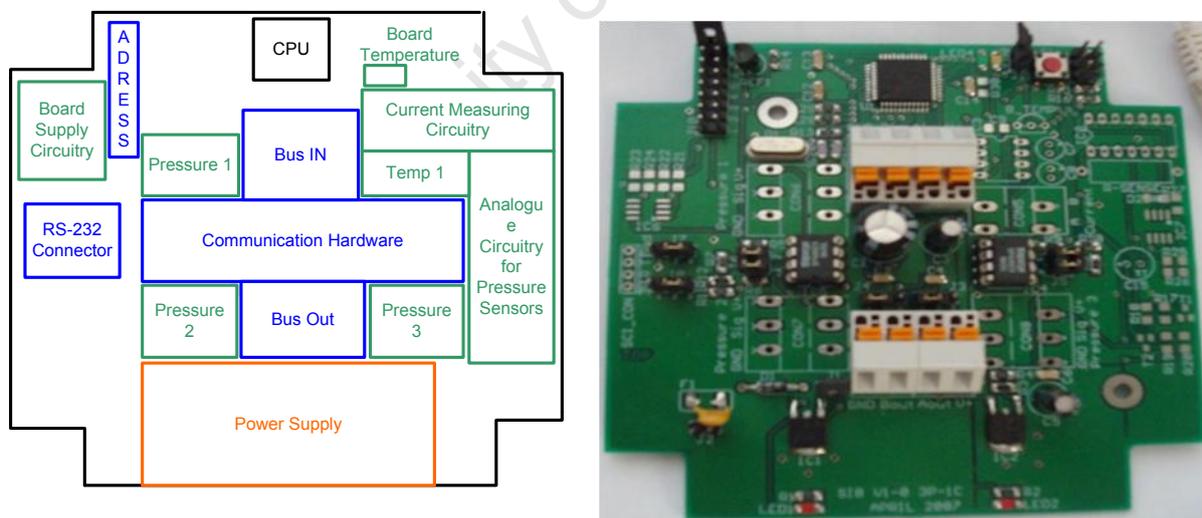


Figure 5-5: The 3P1C Sensor Interface Board

The 3P1C board draws considerably more power than the 4T1P boards and is more expensive in terms of component cost. For these reasons, these boards are used sparingly. The boards were designed to monitor a single compressor, which can have up to three pressures. Many chillers have multiple compressors and therefore use a single board for each one.

One of the most expensive components on the board is the RMS to DC converter from Analog Devices. It is also the only factor limiting the number of current inputs on the board to one. The

option of performing the RMS conversion in firmware was considered as an option to lower the total cost of the board. This was never attempted as a new board was designed that could handle up to 8 current signals. This board is mentioned in the following sections.

5.2.3 Design of the 3P1C4T Board

The 3P1C4T boards represented the second generation of boards developed for the monitoring system. These boards were designed taking into account all the shortcomings and problems of the first generation of boards. The 3P1C4T board was designed to be able to handle the roles of either the 4T1P or the 3P1C by simply selectively soldering components as needed. This was done as it only required a single PCB board to be produced instead of two different versions.

The 3P1C4T board was designed to be installed in both plant rooms and air handling units. The boards are configured by selectively soldering components and setting jumpers on the board. It was intended that these jumpers could be automatically soldered as needed if the assembly process were to be automated.

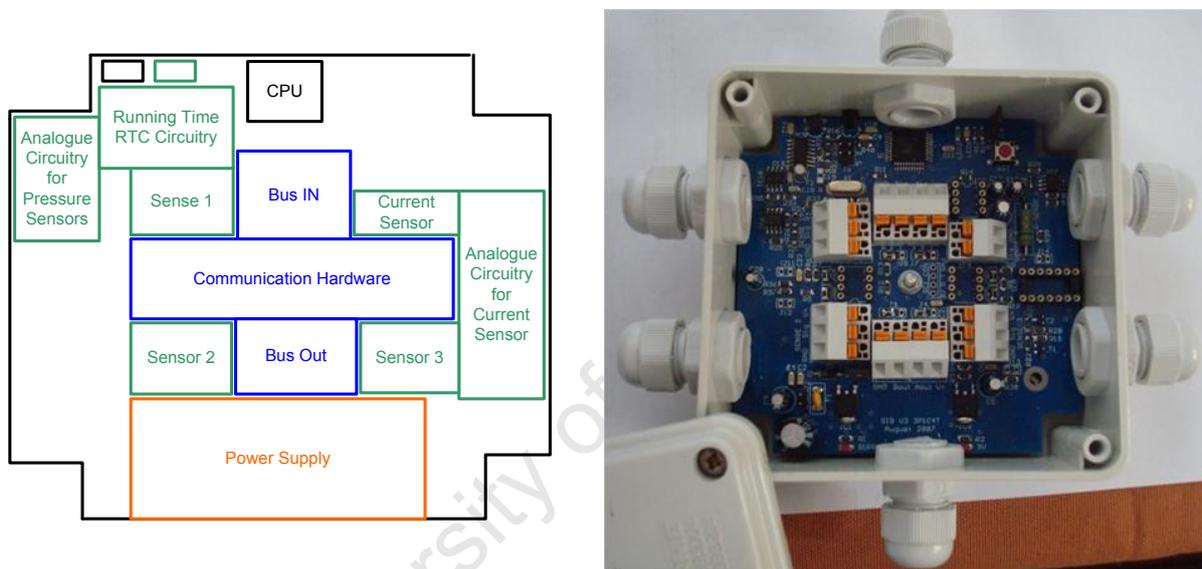


Figure 5-6: The 3P1C4T Sensor Interface Board

The second generation of boards retain the cross shape of the first generation but are slightly smaller in size as the dimensions of the plastic housings abruptly changed. The boards were therefore made smaller in order to compensate. This made the layout of the PCB more challenging but did result in a slight reduction in the cost of the PCB manufacture.

The 3P1C4T included additional components such as the ability to support an external crystal to implement RTC functionality. This was added along with a dedicated comparator on the current input to allow the board to estimate the running time of the device. Unfortunately, the threshold value for the comparator needed to be set manually and the running time value was calculated by the host controller issuing periodic checks of the elapsed seconds on the device. This required some calibration on the diagnostics server in order to estimate the running time accurately.

5.2.4 Design of the 8L8C Board

The most recent addition to the board configurations is the 8L8C board which is intended for use inside electrical switchboards. These boards have eight separate current inputs and 8 logic inputs on a single rectangular PCB. The purpose of this board is to monitor both the running time and current of a number of different devices in an installation.

The standard bus connections and power supply hardware are present on the right-hand side of the board. The left-hand side of the board contains 8 power resistors for each of the current inputs. These are then fed through an analogue multiplexer to a single RMS to DC converter which connects to the microcontroller. The microcontroller can therefore use the multiplexer to select any current but cannot measure all eight simultaneously.

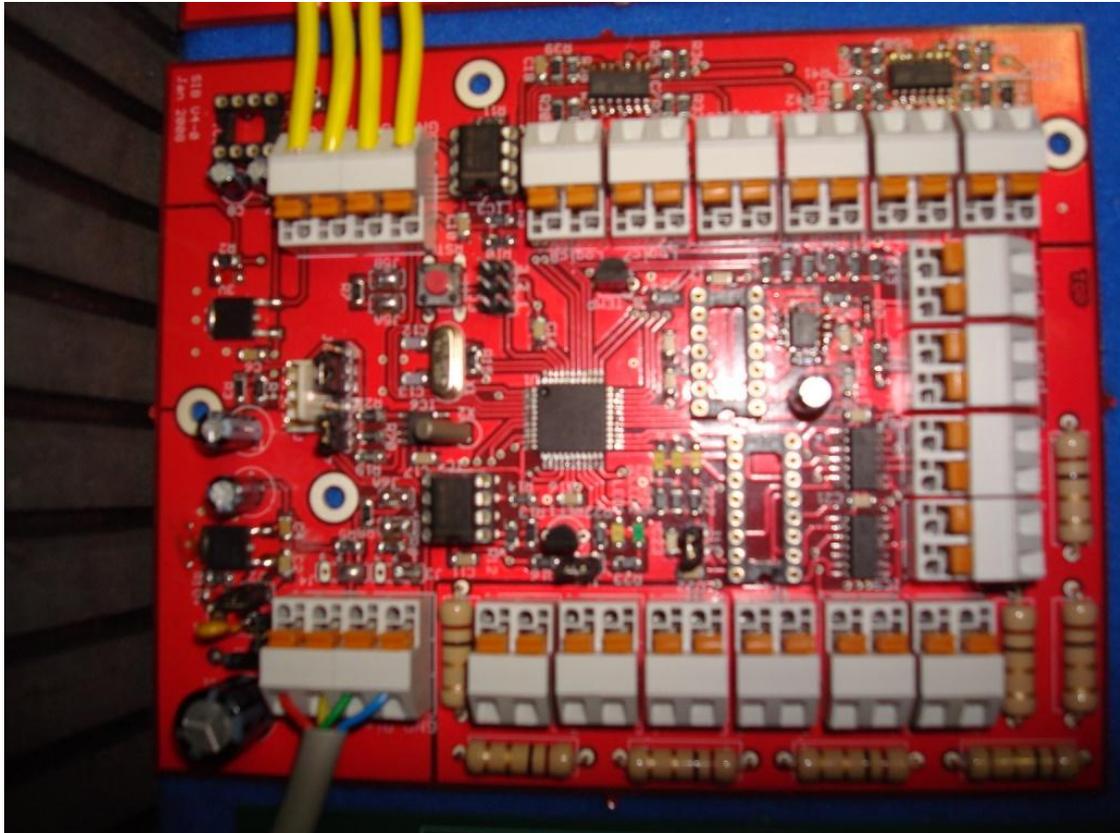


Figure 5-7: The 8L8C Sensor Interface Board

The logic inputs are configurable as either normally-open or normally-closed inputs through the use of pull-up resistors. The intended use for the logic inputs is to monitor the state of relays and contactors by utilizing the auxiliary inputs on such devices. This can then be used by the firmware in order to calculate the running time of up to eight different devices.

6 Design of the Host Controller

6.1 The Central Role of the Host Controller

The Host Controller plays a very significant role on the network as it is responsible for the coordination and control of all the Sensor Interface Boards on the network. It implements the addressing modes and is responsible for initializing, configuring and maintaining a network of varying lengths. It is also responsible for acquiring readings from sensors and then uploading them to the diagnostics server via a GSM data connection.

Besides the core tasks required by the network, the host controller also performs a number of other services such as providing a user interface and on-board memory to store readings. It also allows for other services to be implemented using the GSM link, such as the ability to transmit user-defined messages to and from the host controller.

The host controller also acts a source of power for the bus and therefore has a specialized power supply in order to handle the conflicting demands of a long network of devices and the strictly regulated needs of the GSM unit. The design of the host controller retains the same methodology regarding ease of installation and as a result, much effort was expended to ensure that the installation of the host controller was easy and required minimal configuration by the operator.

It is important to note that the host controller is the only device on the network that requires a custom installation as it needs to be assigned a unique identification so that data originating from it can be identified by the diagnostics server. This value can be as simple as a unique serial number programmed into the device during initial programming or could be expanded to allow installation-specific information to be included, allowing the host controller to be customized to the target installation. The only requirement is that the diagnostics server must be aware of the unique identification prior to the instantiation of a new installation.

6.2 Design of the Original Host Controller

Two different versions of the Host Controller were designed for this system. The second version of the Host Controller improved on many of the problems discovered when the first Host Controller was placed in real-world applications. Although two Host Controllers were designed, the majority of the development took place on the first version of the Host Controller as it was already in use in the field.

The host controller was initially designed to operate on a network implementing the device-based addressing system. The original version of the host controller was a physically large board as many of the components were untested prior to the assembly of the board and therefore the ease with which the board could be altered was valued above both size and cost.

6.2.1 Controls and the User Interface on the Host Controller Unit

Although autonomous functioning is one of the fundamental design ideologies for the system, it was felt that the Host Controller should support a user interface to allow a remote operator to interact with the system when necessary. The intention was to provide a means of accessing the system whilst simultaneously allowing the unit to perform autonomous collection and transmission of reading information.

Initially, the system was intended to operate as a “black box” and not support any interaction on site. This was later changed as it provided no means to diagnose problems with the actual monitoring system once it was installed at a location. Problems such as poor GSM reception or network problems would be impossible to diagnose otherwise.

An additional benefit of adding an interface is the ability to display messages to the client. Diagnostic information, such as critical errors or even the presence of a new diagnostic report, can be sent to Host Controllers and displayed on the interface.

The interface on the Host Controller consists of a six-line LCD screen, two push-buttons and rotary scroll-wheel. The intention was to provide a scroll-wheel to navigate through menus and use the two buttons as an “enter” and “cancel” button. In practice, however, a sufficient rotary switch was not found and the scroll-wheel concept was reduced to an up and down button.

A comprehensive menu component was developed to process the menu functionality on the device. The menu component is discussed in more detail in the following section.

6.2.2 Board Layout of the Original Host Controller

A photograph of an assembled and working Host Controller is presented below:

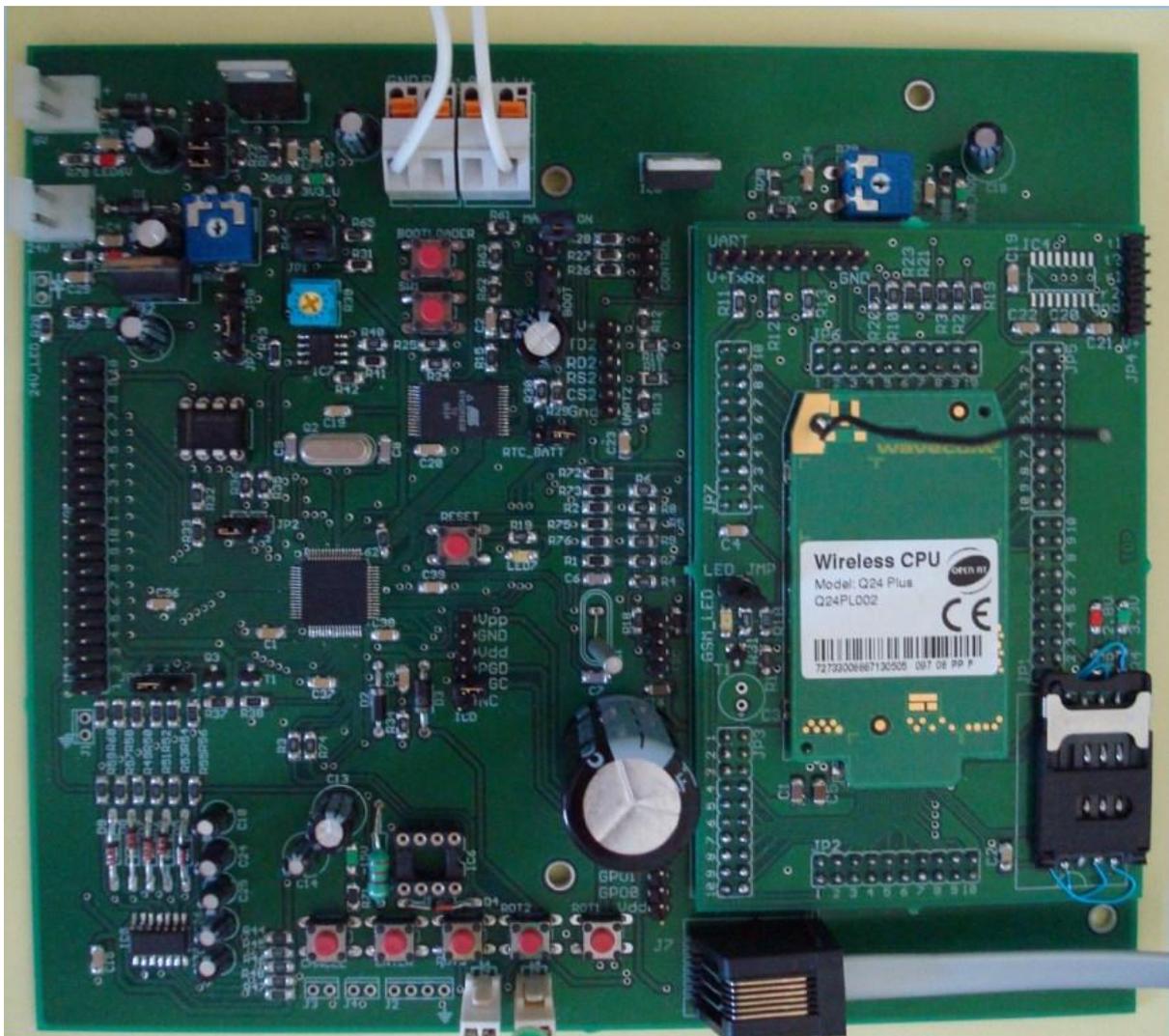


Figure 6-1: A Photograph of the Original Host Controller

The hardware on the Host Controller board was laid out in a logical fashion. The power supply is located at the top-left of the board and includes the ability to provide a separate power supply to the Host Controller and the Network. The GSM unit is supplied through a separate adjustable regulator (located at the top-left of the board) to meet the strict power requirement set out in the technical specifications for the GSM unit. The large capacitor in the lower-middle of the board is

used to supply peak power to the GSM unit. Beneath this connector are the reset buttons and jumpers for the GSM unit.

A lever connector is located at the top-middle of the Host Controller board and provides the connection point for the network bus. The large double-pin header on the left-hand side of the board is the connector for the Screen LCD and adjacent to that is the PIC microprocessor. The bottom of the board hosts the button circuitry, including on-board push-buttons and connectors.

On the right-hand side of the board is the GSM daughter-board, which is detachable. This board holds the GSM unit, the SIM card and an RS-232 level-shifter. The daughter-board was used to allow a different GSM unit to be substituted at a later stage without requiring a redesign of the Host Controller.

6.2.3 Hardware and Software Components on the Original Host Controller

The design of the Host Controller was kept as modular as possible. This allowed each component to be developed and tested separately before the final circuit board had been fabricated. A component view of the system is shown in Figure 6-2 and illustrates the process by which each component interacts. Most of the component in the above figure consists of both a hardware and firmware element. Each component is discussed in detail and outlines both facets of the component.

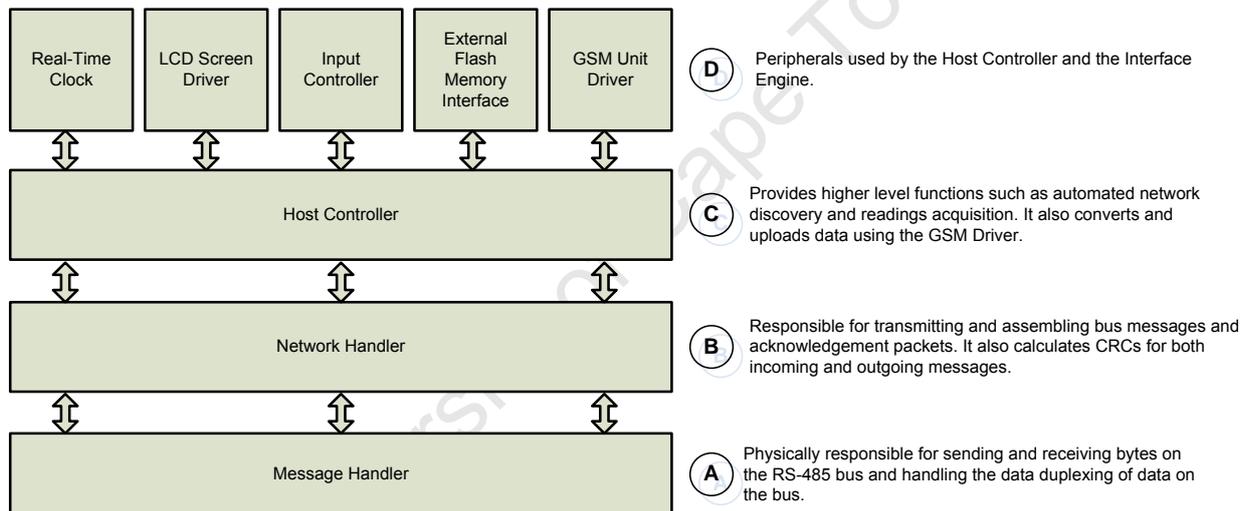


Figure 6-2: Components of the Original Host Controller Design

Most of the component in the above figure consists of both a hardware and firmware element. Each component is discussed in detail and outlines both facets of the component.

6.2.3.1 The Real-Time Clock Component

The Real Time Clock is used to provide timing and synchronization for almost all activities on the host controller. The real time clock is implemented using a 32 kHz crystal connected to timer module on the PIC microcontroller. The PIC includes special timer modules capable of driving a crystal oscillator which, when used in 16-bit mode, provides a method for generating very accurate interrupts at one second intervals. This method requires very few external components and can also be used by the microcontroller to provide a slower clock when in a low power mode.

The Real Time Clock in the system does not provide an accurate time and date. It simply provides an accurate time-base for all operations on the host controller. It allows processes to specify delays in terms of seconds and is used to generate interrupts at set intervals which are used as triggers for taking readings and uploading. The entire purpose of the real time clock is to track elapsed time

from power-on as opposed to the specific time (which is obtained from the server during the upload process).

Originally an I²C RTC from Dallas Semiconductors was considered as an option. It provided all the features of the on-board RTC and also provided a means to keep track of the date and time. It also featured a backup battery and persistent SRAM. It was later discovered that the GSM unit itself implemented a real time clock which could be used to provide all the features of the Dallas RTC if they should be required.

The GSM unit chosen also has an on-board Real Time Clock which is available through the GSM interface. The GSM unit provides the ability to attach a back-up battery in order to allow the time and date to persist when power is disconnected. This Real Time Clock was not used as it would still need to be set initially, and it was felt that an automatic method of determining the time would be better in-line with the requirements of the system.

6.2.3.2 The LCD Screen Driver Component

The Host Controller was designed such that it could be placed quite a distance from the installation it monitors. The intention behind this design choice was to allow the Host Controller to be placed in an easily accessible position and therefore allow users to frequently interaction with the system. System status and diagnostic messages could then be displayed on the screen and the menu system would allow routine maintenance and access to configuration data.

A large graphics LCD module was used on the first host controller and involved a large number of external components to provide backlighting, contrast and a -15V power supply required by the screen. The contrast was designed to be adjustable using a PWM signal from the microcontroller. Although the feature was successfully implemented, it proved to have very little benefit in practice.

In the next iteration of the hardware, the design was changed to support a 4-Line character LCD which was considerably cheaper. In order to simplify both the programming and to allow the LCD to be easily changed, a standard set of functions was defined to provide a common interface to the LCD. In order to change the LCD, one would simply need to change the method in which the functions execute. When conditional compiling is used, this allows the host controller to easily support a variety of different screen types.

The Host Controller is capable of supporting LCD screens with a backlight, although such features were not present on any of the LCD screens used in the first applications. Unfortunately, these screens were difficult to read in plant rooms, where poor lighting is a common occurrence and all future system will utilize backlight LCD screens.

The LCD screen was extremely important in the debugging process as it provided insight into the functioning of the host controller when installed on a site where using a PC for debugging was not possible. The LCD functions were designed to be compatible with the USART functions so that data could easily be rerouted to the LCD display. Both the GSM unit and the network were controlled and monitored in this way.

Care was taken when writing the functions for the LCD as it was discovered that the application can spend a significant amount of time writing to the display. If the functions are written badly, making many blocking calls to hardware, the application may not be able to process interrupt flags quickly enough and data loss can occur.

6.2.3.3 The Input Controller

The Input Controller consists of the hardware interface to buttons and the firmware to provide a reactive user interface on the LCD screen. The Host Controller has five button inputs and two LEDs which are used in conjunction with the LCD display to provide the interface to user.

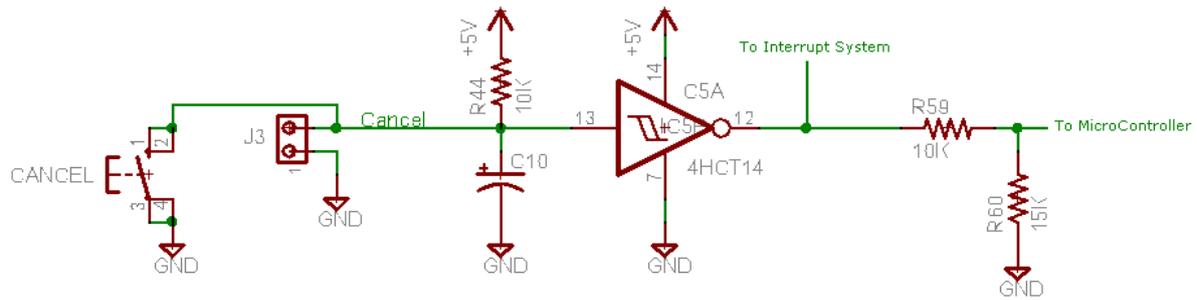


Figure 6-3: Hardware De-bouncing for the Host Controller Input Buttons

Each button input is hardware de-bounced using the circuit shown in the above image and a set of diodes is used to provide a rudimentary interrupt line. Due to the requirement of the LCD screen, the input system requires 5V whereas the microprocessor and GSM unit operates using a 3.3V supply. For this reason, the output of the De-bouncing circuit passes through a voltage-divider to ensure compatibility with the 3.3V logic on the microcontroller.

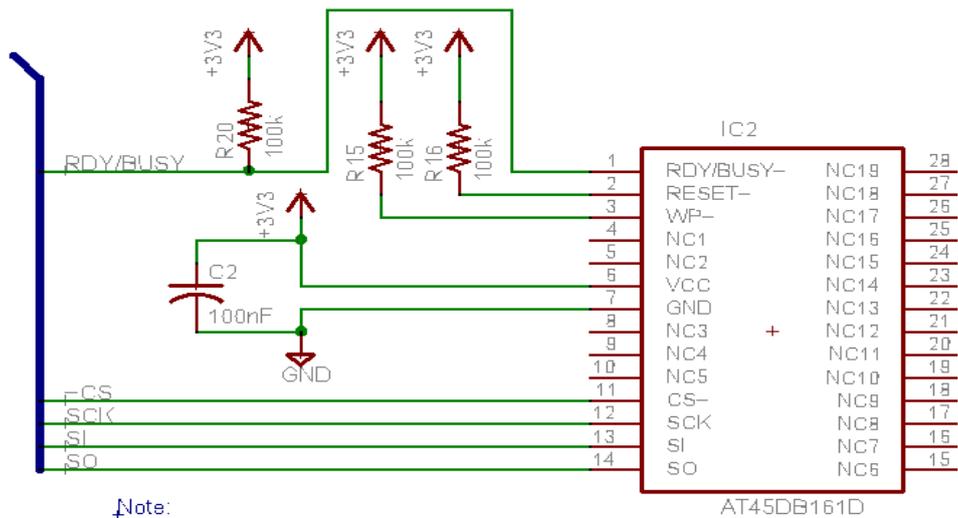
The Input Controller also manages the menu sub-system, which is a very efficient menu manager supporting multiple levels of menus and a screen saver without impacting on the timing requirements of the GSM or network interfaces.

6.2.3.4 The External Flash Memory Interface

An external flash memory chip was included in the design of the host controller for two reasons. Firstly, it was intended to lessen the RAM requirements on the microcontroller and therefore allow virtually unlimited number of boards to be used on the network. The flash memory chip chosen included two write buffers of 512 bytes, and it was decided to implement a paging system where 512 bytes of data were loaded at a time. Currently, no present network has reached the point where the paging system is required.

The second function of the flash memory chip is to provide non-volatile storage for data such as configuration settings. The flash memory could also be used to provide a means of profiling a certain reading over a much shorter time period (every second for 5 minutes, for example) and then uploading all the accumulated data. This functionality has yet to be implemented.

The flash memory IC used was an Atmel 16 Mbit ADT45DB16 Flash memory device. It has a serial output and is capable of running at voltages as low as 2.7V [34]. The IC was chosen due to its low cost, easy availability and its serial interface. It was felt that using a serial interface would allow the memory capacity of the device to be increased with minimal impact to the layout of the Host Controller board.



Note:
 CS does not need to be driven from SS
 SS is solely for use as a Slave Enable when the PIC is operating as a Slave device.

Figure 6-4: Circuit Diagram for the Atmel External Flash Memory

The above image shows the connections required between the flash memory and the microcontroller. The Flash memory uses a standard SPI interface, which attaches directly to the corresponding SPI interface on the microcontroller. There is also a Ready/Busy (RDY/BSY) line which is used to check whether the device is currently busy with a write or read operation.

The presence of the RDY/BSY line was an extremely important factor in the selection of the device as it allows write operations to the device to be implemented in a non-blocking manner. Many serial-based flash memories require constant polling when performing read or write operations, which would interfere with the timing requirements of the microcontroller.

6.2.3.5 The GSM Unit Driver

The GSM unit chosen was a Wavecom Q2402 unit with an external SIM card holder. Almost all the GSM units available implement a standard Hayes AT command interface which can be used to control the device [35]. Different vendors provide additional functionality by implementing proprietary AT command. Often high-level services such as TCP/IP, Email, FTP and even GPS functionality are exposed in this manner.

The Wavecom device supports a plug-in to allow TCP/IP sessions to be created and managed. An additional plug-in can be ordered to support FTP, POP and SMTP services directly from the AT command interface. The other factor that weighed in significantly was the on-board processing capabilities of the unit. The Wavecom unit has an on-board ARM7 processor, which supports user-specific applications through a framework known as OpenAT.

The OpenAT framework was used to develop firmware for the Wavecom unit in order to simplify and streamline the upload process. The first version of the firmware implemented the entire upload directly from the microprocessor, this involved configuring the Wavecom unit, opening the GPRS bearer, creating a TCP connection and uploading data.

One of the downfalls of this approach resulted from the method in which the Wavecom unit multiplexes commands and data. As there is only one serial connection between the GSM unit and the microcontroller, the unit uses a special escape sequence to switch between the two modes. The method worked successfully but proved inflexible as the configuration data for the GSM module needed to be hard-coded into the microcontroller [36].

Another issue plaguing the original method of operating the device was the manner in which the device operated. As there was no firmware running on the unit, it was turned off when not in use. Unfortunately, there are a number of documented cases in which the specific brand of Wavecom units spontaneously power up of their own accord [35]. This caused the configuration process to fail and resulted in a failure to upload.

In order to resolve these issues, firmware was developed to run on the GSM module. This allowed the module to act independently on the microcontroller, which resolved the sporadic power-on problems. The firmware added proprietary AT commands to allow the module to pass its sensor readings and request an upload, allowing all the functionality to be implemented through commands, bypassing the need to multiplex commands and data on a single channel.

A further benefit of porting the upload functionality to the GSM unit is the ability to remotely specify the configuration data via technologies such as SMS. Settings such as GPRS access information, the diagnostics server and any relevant usernames and passwords can be remotely configured. This simplifies both the installation and the maintenance of the Host Controller.

6.2.3.6 The Host Controller

The Host Controller component provides the connection between all the components on the device. It monitors the status of the system components and reacts accordingly. It is responsible for scheduling events based on the real time clock, responding to incoming network messages, managing the GSM driver and responding to user input from the input controller.

The method by which the Host Controller Component operates is described in more detail in the discussion of the design of the Host Controller firmware.

A PIC18LF6622 was chosen as the microcontroller for the Host Controller board as it provided two UARTs, abundant program and data memory and a wide array of peripherals such as timers and general purpose IO. The PIC is configured to run off a 16 MHz clock and is powered off a 3.3V regulated supply. PIC microcontrollers were chosen due to their low-cost, wide availability and reliability in industrial environments [37].

6.2.3.7 The Network Handler and Message Handler

The Network Handler and Message Handler components are responsible for interfacing with the physical network bus and the proprietary protocol designed for the monitoring system. These components are virtually identical to the corresponding modules used in the Sensor Interface Boards (the only adaptations being those required by the different peripheral interface) and are discussed in detail in Chapter 7.

The communication hardware is slightly different to that of the Sensor Interface Boards as only one channel is provided on the original Host Controller board. The interface therefore consists of a single RS-485 Level shifter connected to the one of the UARTs on the Host Controller board.

6.2.4 Design of the Firmware for the Original Host Controller

In order to understand how the individual elements combine and co-exist in the host controller, a short discussion on the manner in which the firmware was designed is presented below. It discusses the modular design and interrupt-based methodology used in developing the host controller, which ultimately enabled the component-based design.

The host controller is an event-driven system with a number of interrupt sources and event handlers. Interrupt routines are kept as short as possible and should ideally only set a flag and clear the interrupt. The main application loop is then responsible for checking whether any of these flags have been set and then branching to the appropriate functions (often referred to as an event handler) when necessary.

As a result of this, the event handling functions must be interruptible at any point (unless it can be guaranteed that no other interrupts can occur during the execution of the function) and must not make any uninterruptible calls to the hardware. If a function were to loop indefinitely, waiting for a user response for example, then it may cause interrupts to be missed. This manifested itself as issue with the screen driver as the firmware needed to wait for a hardware confirmation signal before sending the next byte of data.

In order to overcome the limitations imposed by the event-driven design, a strong state-based model was used to allow complex events involving delays and call-backs to be implemented. Components themselves may implement their own state-based systems, which is abstracted away from the main application through the use of separate header and code files.

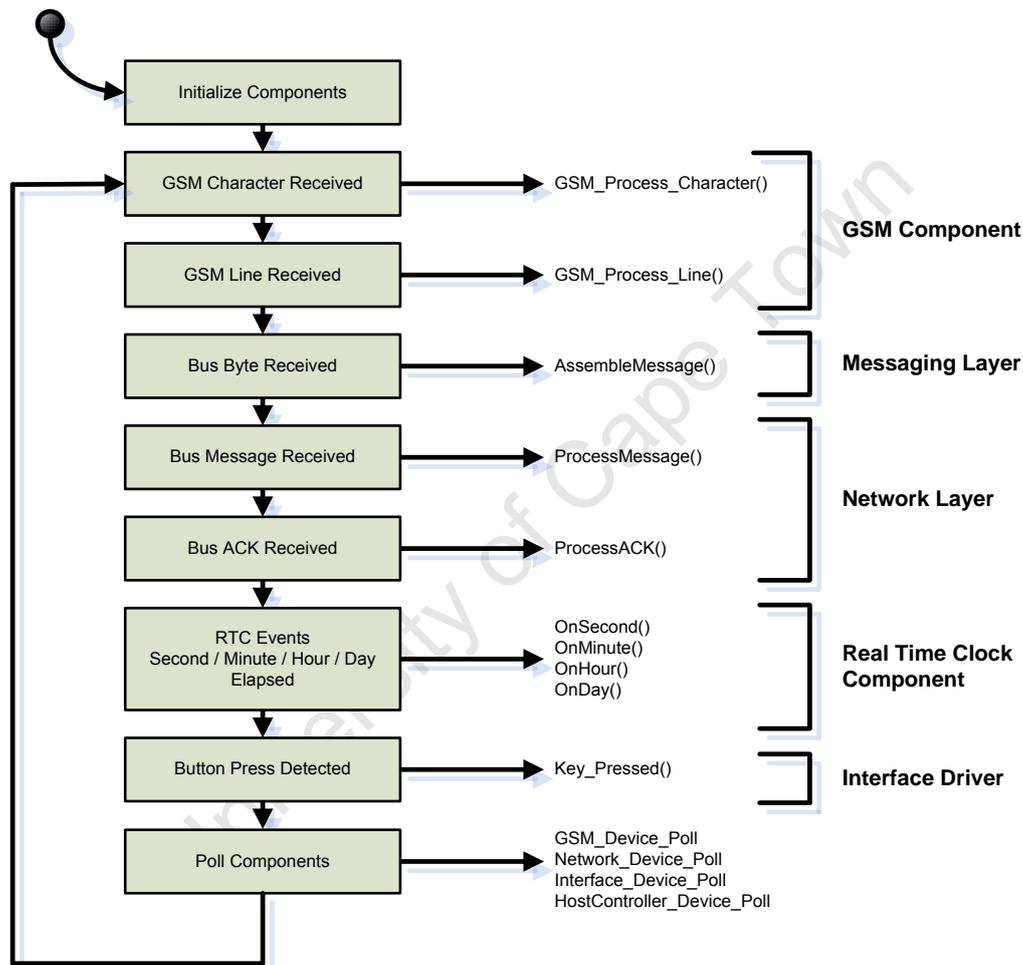


Figure 6-5: Steps in the Main Application Loop of the Host Controller Firmware

The above figure demonstrates the main processes the host controller executes during the main application loop. After power-up, all the components are initialized and configured before the application enters the main loop. The first stage of the loop checks all the interrupt flags to see if any events have occurred since the last iteration. If these events have occurred, then the appropriate event is raised. These events should appear to execute like interrupts, although with a small and unpredictable latency. For this reason, they should also be kept as short as possible and preferably only change the device state or set other flags.

It is important to note that not all the flags are set by interrupt sources. Certain flags are raised by other event handlers, making the order in which the flag are checks very important. A good example of this is the GSM character assembly process. When a character is received on the USART

responsible for GSM operation, an interrupt is triggered and sets the appropriate flag. When the main application loop executes, this flag is detected and the `ProcessCharacter()` function is called. This function places the new character into a string buffer and then checks to see if a valid line has been received. If so, it sets an appropriate flag which is detected by the very next check once the function returns to the main loop.

There are also a number of event handlers originating from the real-time clock to signify the passing of various time intervals. These are used extensively throughout the system and are described in more detail in the following section. Once all the event handlers have been called, the main application loop then examines the current state of the host controller. As the event handlers can change the state of the device, the main loop may start in a particular state but execute a different one after executing the appropriate event handlers. This is a very important factor to consider when debugging or tracing execution through the system.

The states of the host controller are very general, having usually only one state for each component in the system. If that state is detected, then the main application loop calls a function in the appropriate component and expects that component to respond accordingly. Each component must therefore implement its own state-based system within that function. Usually the host controller can only exist in one state at a time, although in a few cases a second state variable has been added to allow the device to handle multiple events simultaneously. If this system is used incorrectly this can create a host of problems including race conditions and deadlocks.

Delays are usually implemented using either the timer modules or the real-time clock. A component will often set up a time-out (using either of the methods previously mentioned) and then change state to a waiting state. Although the device cannot perform other operations, it can still respond to interrupts and perform non-critical tasks such as updating the display. A special flag is present which prevents the state from being changed during these timeout periods, thus preventing the possibility of hardware being left partially configured due to an unexpected (or user-initialized) state change.

6.3 Design of the Revised Host Controller

The initial Host Controller was designed to be flexible in order to ensure that alteration could easily be made after the board was produced. Many IO lines were exposed using jumpers and many settings were configurable either by the microcontroller or via dedicated override jumpers. The GSM module was installed on a special daughter board which plugged into the main board, to allow the module to be changed if necessary. All these extra components caused the Host Controller board to become rather large and resulted in a large number of unnecessary components.

Once the Host Controller was successfully implemented, a number of observations were made regarding the areas in which the Host Controller could be improved.

The first and foremost requirement for the new Host Controller was the support for multiple network channels. The original Host Controller only supported a single network connection, which led to unnecessarily long cable runs in certain building configurations (see section 7.3.4.1).

The second requirement for the Host Controller was the need to support a few sensors off the Host Controller directly. This would allow one or two sensors to be monitored using only a Host Controller board.

A number of improvements were made to the input and display components in order to reduce costs and size. Many of the features designed into the original Host Controller proved to be entirely unnecessary and were therefore removed.

A revised diagram of the components in the Revised Host Controller is shown below:

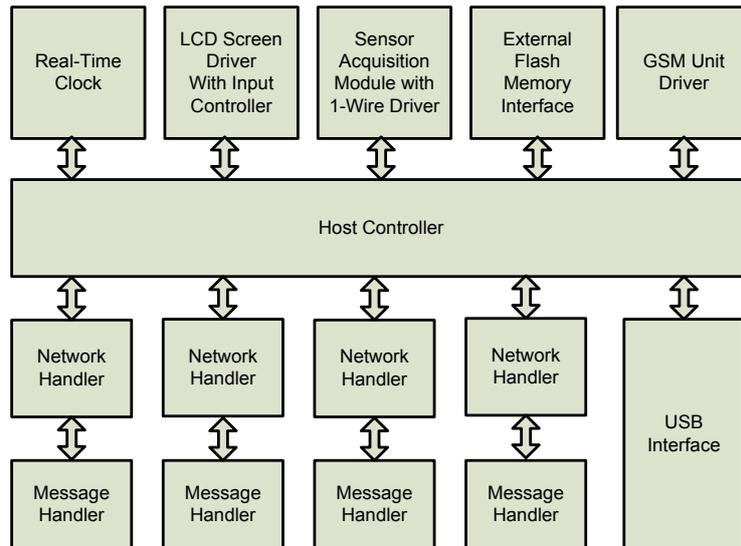


Figure 6-6: Components of the Revised Host Controller

The Revised Host Controller retains many of the components of the Original Host Controller. The Real Time Clock, Flash Memory and GSM driver remained and the LCD driver and Input controller were combined into a single component (known as the Interface Controller). The Revised Host controller adds a Sensor Acquisition Module consisting of firmware to support 1-Wire[®] devices. A 1-Wire[®] Analogue to Digital converter was also sourced to enable monitoring of pressures and currents using the Host Controller.

The Revised Host Controller supports four independent bus channels and therefore has a duplicate network and message handler for each one. A single Host Controller is responsible for managing each bus channel, amalgamating the outputs in order to form a single contiguous representation of the network as described in section 7.3.4.1.

The Revised Host Controller replaces the original PIC18LF6622 with the PIC18F87J50 which all the peripherals found on the PIC18F6622 and adds support for a USB connection. It was felt that the USB connection could be used to either provide streaming readings to a computer or allow additional upload methods to be implemented where a PC is available.

The Revised Host Controller is also a much smaller board as it only includes the necessary hardware peripherals. A newer GSM module was also selected which has the same interface as the original Q24 series but has features such as 5V tolerant UART inputs, which also serve to reduce the external pin count.

Figure 6-7 shows the PCB layout for the Revised Host Controller. The component density is much higher than that of the original Host Controller and the GSM daughterboard has been removed completely. The power supply circuitry is located along the left-side of the board, the bus connections are located on the bottom of the board, and the interface connections (USB, Screen and buttons) are located on the top edge.

The large gap on the right-side of the board is to accommodate any specific sensor acquisition circuitry that may be included in later versions. For 1-Wire[®] devices, these lever connectors are simply attached directly to port pins on the microcontroller.

Power to the board is supplied using the connectors on the bottom left and right sides of the board. The two jumpers allow the bus and the host controller board to be powered independently of each

other as the voltage drop over the regulators when using a 24V supply causes fairly high heat dissipation in the regulators.

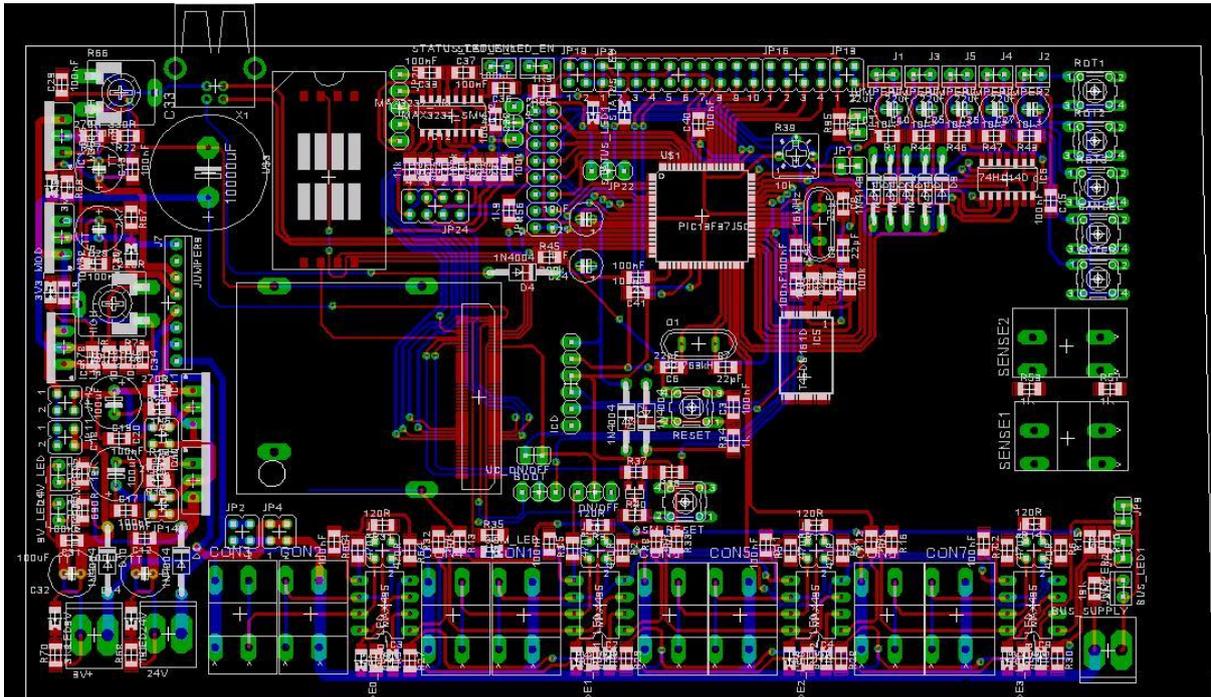


Figure 6-7: PCB Layout for the Revised Host Controller

In terms of size, the Revised Host Controller is approximately the same width as the Original Host Controller but only a third of the length. Special care was taken when routing this board to ensure that the majority of the routing was done on the top layer of the PCB. As a result, there are almost half the number of vias on the Revised Host Controller, thus further reducing the fabrication costs.

7 Network and Bus Design

7.1 Design of the Physical Bus

7.1.1 Requirements of the Physical Bus

One of the most important characteristics of the system is the ability to monitor a wide variety of different parameters across an entire installation. In a large air conditioning system this may include chillers, pumps, air compressors and air handling units which may be located at quite a distance apart. In order to cater for this, the bus must be able to handle long distances between Sensor Interface Boards.

Boards may also be installed in close proximity to each other and in this case it must not be necessary to provide separate power to each board. The bus must therefore provide both signaling and power to each of the units. The main requirements of the bus and the communications protocol are as follows:

- The bus must support long cable lengths between Sensor Interface Boards.
- The bus must provide power to devices but allow supplementary power to be added where necessary.
- The bus must allow for some degree of autonomous setup.
- The physical cable must be kept as inexpensive as possible.
- The protocol must be reliable and implement some form of error checking or correcting.
- The protocol should simplify the installation process as far as possible.

7.1.2 The Physical Transmission Medium for the Physical Bus

A number of different technologies were considered to provide the electrical interface between devices. Often the benefits of the features provided by a technology were outweighed by its cost or its failure to address a core requirement of the system. One of the major limiting factors was the range provided by the technologies which was often well below 30m.

RS-485 was chosen as the best technology to provide signalling between devices. It has a theoretical range of over a kilometre and a half-duplex solution can be implemented over just two wires [38]. Line drivers are both easily available and relatively inexpensive and can be driven from the standard UART hardware incorporated into most microcontrollers.

As the cost of cable increases with the number of cores, a four-core twisted-pair cable was chosen. Two of these wires would be used as the differential RS-485 pair and the other two would be used to provide power to the downstream boards. Although EA-485 uses differential signaling and twisted pair wire was chosen, shielded cable was used to further increase the reliability in an industrial environment.

7.1.3 The Choice of Bus Topography

The choice of topography is of great importance to a system such as this one. It determines physical properties such as the lengths of cable required, the ease of installation and physical wiring requirements. It also impacts the design of the protocol as different layouts provide different challenges for functions such as error checking and point-to-point communication.

As cable length is directly proportional to the cost of the system, a bus topography is often more cost effective than a star-type topology in terms of cost. The advantages of using a star topology is that each device connects directly to the host controller which allows the response time of each board to be easily characterized. Unfortunately, implementing a star topology using RS-485 either results in a large number of line drivers required at the Host Controller or complicated problems

relating to the impedance and termination of the RS-485 signals. It was therefore impractical for use in this system.

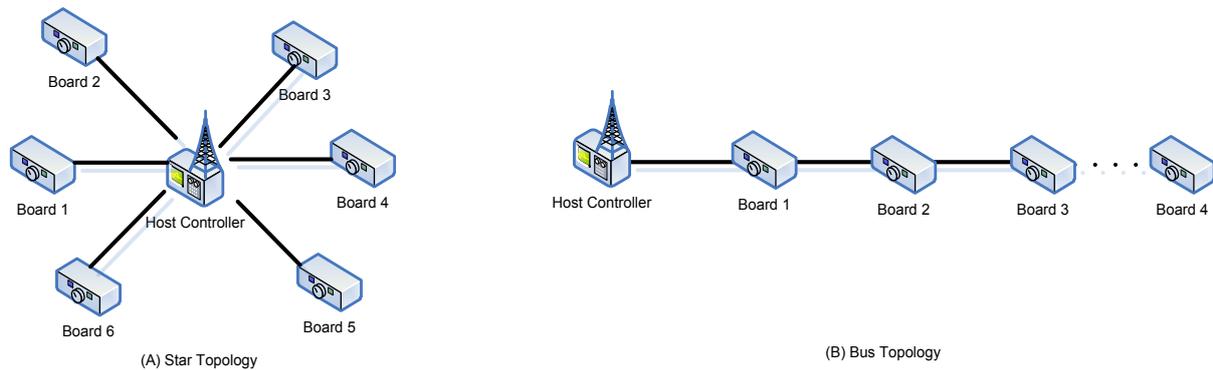


Figure 7-1: Examples of Star and Bus Topologies

A linear bus topology connects each device to a single physical cable. The most significant benefit of this method is that it can reduce the required cable length dramatically (although this may come at the cost of simplicity as cables may need to loop back on themselves. See 7.3.4.1 for an illustration and possible solution to this problem). The downsides are that such a technology can result in single points of failure and may suffer from propagation delays down the bus.

There are two methods by which a linear bus can be implemented. The boards can be connected to a single common cable, or connected through each board in a daisy-chain fashion. Each of these methods provides different advantages and disadvantages, which are discussed below.

7.1.3.1 The Direct-Drop Bus Topology

The RS-485 protocol was designed to support multiple devices on a single bus. This functionality is supported by carefully calculated characteristic impedances and terminating resistors such that a total of 128 devices can be used on a single bus (note that with $\frac{1}{2}$ and $\frac{1}{4}$ line drivers, this can be increased further). Each device simply connects to the common data signals and can act as both a transmitter and a receiver, although careful care must be taken to prevent two boards attempting to write to the bus at the same time.

In this configuration, the host would act as a bus-master and poll devices for data at regular intervals. Through careful control of the bus, the host controller could seek to prevent two devices communicating simultaneously. A further method of reducing the effects of data collisions arises from the nature of the RS-485 Line Drivers.

The RS-485 line driver provides both a Transmit Enable (DE) and a Receive Enable (\sim RE). These lines are often tied together to provide half-duplex control with a single line. If these two lines are driven separately, then it is possible to enable the receiver whilst transmitting and therefore receiving the data being sent on the microcontroller. This data can then be compared to the data being sent to check for corruption as it directly reflects the state of the physical bus and not the intended signal.

A further benefit of the Direct-Drop system is that a single board failure may not affect the rest of the network. With this topology, a board that enters an erroneous state would not influence the bus provided that its RS-485 line driver remains disabled or in receive mode. Additionally as each Sensor Interface Board contains a re-settable fuse, any short-circuit mode of failure would simply cause the board to 'disappear' from the bus until the fuse resets and therefore a single board failure would not affect the functioning of the rest of the system.

Direct Drop networks do suffer from problems regarding range and correct termination. Using a single RS-485 bus limits the total range of the system to that of a single RS-485 link (approximately 1.3 km) and is further reduced as the speed and number of devices is increased. The second problem involves termination as the last node is required to terminate the bus using an appropriately matched resistor to prevent reflections. This would require either a jumper to be set on the last board, or a special terminating connection for the last board. This would increase the complexity of installation.

7.1.3.2 Daisy-chain Configuration

The daisy-chain bus configuration does not utilize the ability of the RS-485 bus to support multiple devices but rather relies on it to provide point-to-point connections between connected devices. This design borrows from the designs of old daisy-chain priority interrupt systems, where each device passes the interrupt up to next device until it reaches the CPU. In a similar fashion, the Sensor Interface Boards pass data up to the next board in the chain until it reaches the Host Controller.

One of the immediate benefits of this system is that it each board effectively acts as a repeater, extending the range of the system without limit. It also solves the termination problem as each link is essentially a complete RS-485 link and therefore can be terminated on each board. The fact that each board can determine its distance (in terms of the number of devices) from the host controller gives rise to the ability to identify the position of a node (relative to one another) on the bus at any point. This property is exploited to perform the jumper-free autonomous network discovery (see Section 7.3.4.2) which greatly simplifies the installation requirements.

The drawbacks of a daisy-chain system are two-fold. Firstly, each board requires two RS-485 line drivers to provide communication to along the bus. This increases the cost of the Sensor Interface Boards and requires a microcontroller with two USARTs⁶. The second drawback is that each board now adds a propagation delay to the system which can become significant in large systems. The inherent nature of the daisy-chain bus results in a single board failure bringing down all the boards connected to it. This was not considered as important in a system of this nature as a single board failure would often invalidate the readings in any case.

7.1.3.3 Justification for Implementing the Daisy-Chain Configuration

The choice between implementing a daisy-chain or a direct-drop bus was not decided until after the initial boards were developed. For this reason, the initial boards were designed to support both configurations through jumpers. These jumpers remained in all future revisions of the boards, allowing either configuration to be implemented. Later board revisions also added a jumper to allow an SCI connection to a RS-232 connection for debugging purposes. A comparison of the features of the two strategies is given in the table below:

	Direct Drop Configuration	Daisy-Chain Configuration
No. of Line Drivers required:	1	2
Range of the network:	± 1 km	Limitless
Effect of a single board failure:	Limited to a single board	Brings down rest of network
Propagation Delay:	Constant	Different for each board
Total number of devices:	Limited by RS-485 loading limit	Limited by addressing size
Board Address Determination:	Set by physical jumpers	Determined from position
Bus Termination	Required at each end.	Each link fully terminated.

Table 7-1: Comparison of the features of Daisy-Chain and Direct-Drop Busses

⁶ It is possible to implement the same functionality on a microcontroller with a single UART although two line drivers are still required. This can be achieved using the separate read and write enable lines on the RS-485 drivers and can even support simultaneous receiving on one channel and transmission on another.

Both configurations proved to be effective in the network but the daisy-chain configuration proved to be more useful overall and was adopted as the primary strategy. The daisy-chain configuration allowed the protocol to be developed in such a way as to allow address-free setup of the bus and also allowed for other technologies to replace the RS-485 link if necessary.

The ability to support other technologies to link boards together arises from the fact that each link between two devices in a daisy-chain network is completely independent. As long as the data is successfully transported from one device to the other, changing the technology will not affect the functioning of the system. A corollary to this is that the daisy-chain methodology allows devices such as wireless bridges to be implemented with relative ease.

7.1.4 Daisy-Chain Transmission Methods

Each device in a daisy-chain network is required to pass messages along the bus. A direct-drop bus eliminates this problem by allowing signals to propagate to each end of the physical cable. In a daisy chain network, the message has to be passed along in software. This can be accomplished in one of two ways. The data can be passed along as it is received or each device can buffer an entire message before transmitting it to the next board

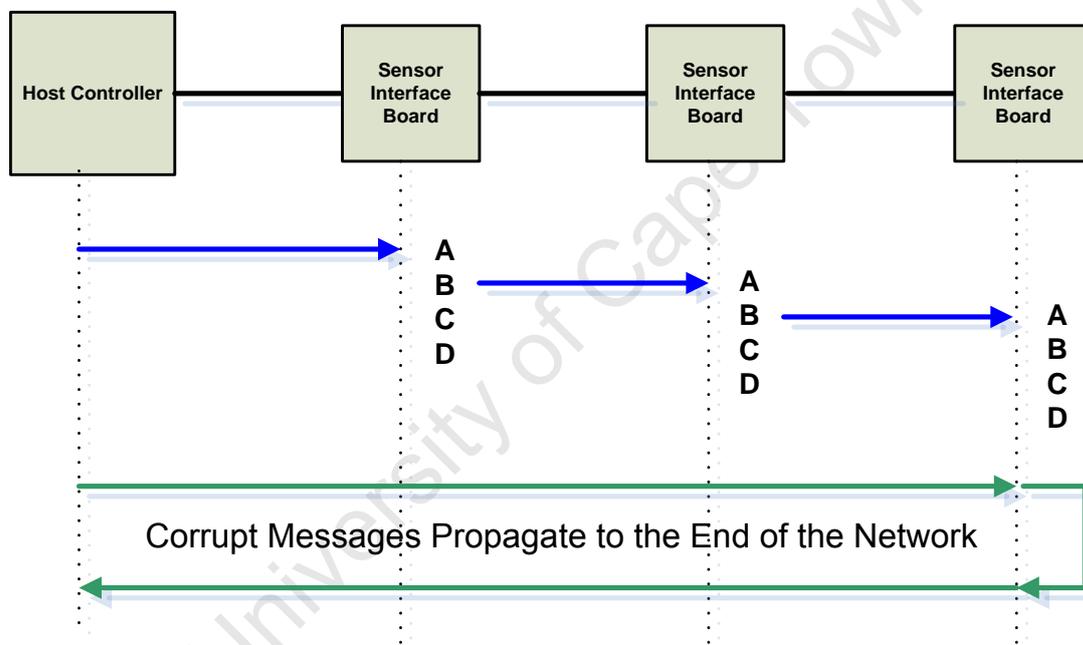


Figure 7-2: Illustration of Pass-through Network Propagation

‘Pass-through network propagation’ occurs boards transmit each byte along as it is received. This is illustrated in the figure above. In the example, a message containing the letters “ABCD” is transmitted to three nodes on the bus. Each device transmits to the next board after it successfully receives a character. This introduces a delay equal to the time it takes to transmit a byte, for each board.

This delay grows at a constant speed relative to the size of the network.

The main advantage of this approach is that messages propagate down the bus at the fastest possible speed. This becomes important in situations where readings need to be taken at the same time and the delays introduced in relaying the messages to the furthest board must be minimized. It also simplifies the firmware implementation as it requires less coordination and can be operated in either direction (although not both directions simultaneously).

This method does suffer from a number of pitfalls. The first problem is that any errors or corrupt messages propagate to the end of the network with no means of stopping the transfer at any point. This could cause synchronization problems if a message is corrupted and interpreted differently along the bus.

All messages sent on the bus are essentially broadcast messages and are received by each board on the network. Devices will simply ignore messages that are not addressed to them but this causes a constant delay when communicating with any node on the bus, regardless of distance from the Host Controller.

In order to overcome these problems, each board can buffer a complete message before retransmitting it to the next device. This is illustrated in the figure below. This greatly increase the time it takes for messages to propagate down to the end of the network as each device must receive the entire message before retransmitting it.

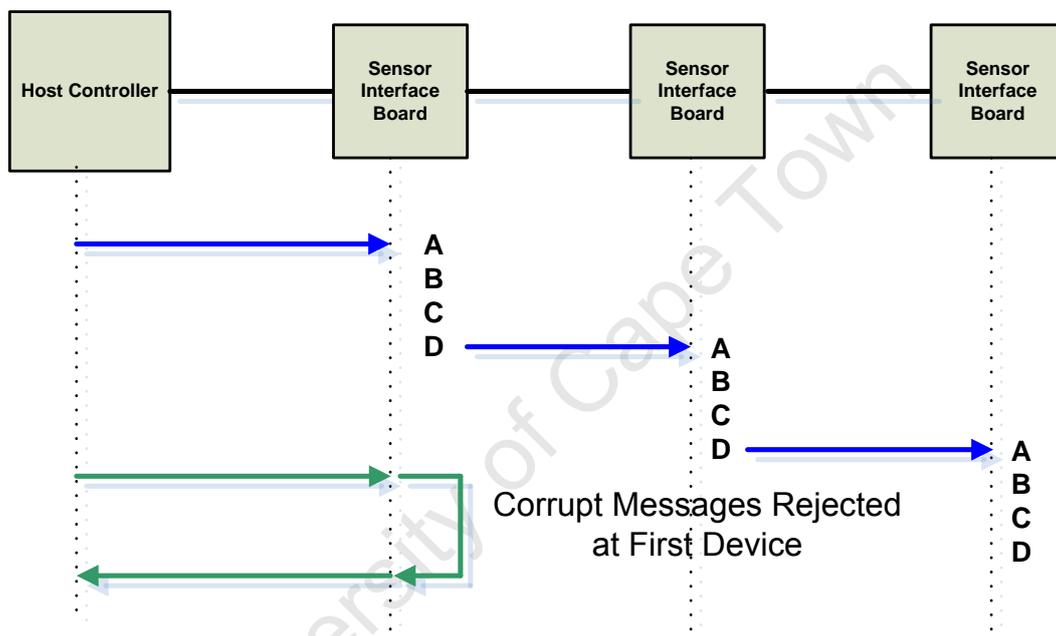


Figure 7-3: Illustration of the buffer-and-forward method of data propagation

This buffer-and-forward method allows each device to process packets before transmitting them to the next device. These devices can implement error checking to determine the integrity of the message before transmitting it along. As a result, errors and bad packets do not propagate to the end of the network and a retransmission system can be more easily and efficiently implemented.

The devices can also perform other functions on the data before retransmitting it. This is exploited by the communication protocol to guarantee synchronization on the network as devices can perform uninterruptible tasks before transmitting the message onto the next board. This further increases the time taken for a message to reach the end of the bus but does provide an elegant method of handling uninterruptible tasks of indeterminate length.

7.2 The Communication Protocol

7.2.1 Overview of the Communications Protocol

In order to cater for the specific requirements of the system, a custom protocol was developed to provide communication between the Host Controller and the Sensor Interface Boards attached to the network. The protocol needed to be implemented over a half-duplex daisy-chained RS-485

network and needed to support a limitless number of devices with minimal configuration required per device. These requirements forced the protocol to be developed from scratch.

In order to facilitate the development, which needed to be implemented on different hardware, the protocol was broken down into layers that build upon each other to provide the functionality of the system. These layers are discussed in section 7.2.2.

The protocol was designed around the notion that the host controller should be the sole master on the bus and the only device that can initiate communication on the network. This concept is modelled on the methods by which the USB Host Controller maintains control over all the devices connected to a particular USB port. The USB Host Controller implements all the scheduling and logic on the bus through a constant polling mechanism. If a device has data to transmit, it must wait for the data to be requested by the host controller [39].

In a similar vein, the Host Controller in the sensor network must then request data from all the nodes in the network. A device downstream should not transfer data unless it has been explicitly requested by the Host Controller. This behaviour is implemented using various message packets with special preambles to denote the direction of data flow. The details of these packets and the preambles are presented in sections 7.2.4 and 7.2.5.

The method by which devices are addressed on the network plays a very significant role in determining the operating parameters of the system. The addressing methodology used dictates factors such as the smallest addressable unit on the network and the maximum number of addressable devices allowed on one system. Although not a major factor in small installations, the importance of these value increases dramatically with network size.

Two different addressing systems were implemented on the network. Initially a device-based addressing system was developed, which assigned unique addresses to each Sensor Interface Board. It was used successfully but proved to have a number of disadvantages that were later addressed. A second addressing system, known as Sensor-Based Addressing was developed to address many of these issues. These addressing modes are discussed in section 7.3.

The primary function of the system is to gather sensor readings from the various Sensor Interface Boards located around an air conditioning installation. There are a number of complications regarding the timing and synchronization of this process, which is discussed in section 7.4.

7.2.2 Layers of the Communication Protocol

There are three layers to the communication protocol that are required to support communication along the bus. These provide levels of abstraction and make it possible to change underlying technologies at a later stage without having to change the layers above. Both the Sensor Interface Boards and the Host Controller implement these layers and they provide similar functions. These layers are the Messaging Layer, the Network Layer and the Device-Specific Layer.

7.2.2.1 The Messaging Layer

The Messaging Layer provides low-level functions on the bus such as the receiving and transmission of messages. These functions assemble and track messages sent and received on the bus and handle tasks such as multiplexing the upstream and downstream data on the outgoing busses.

The Messaging layer introduces the concept of channels to the system. Channels are logical representations of an endpoint on the physical bus to which all communications arrive at and depart from. The system was only designed to use RS-485 links, where a channel represents one end of a direct link to another board. The channel structure presented by the Messaging layer allows other technologies to be used instead of the RS-485 link with minimal changes.

The Messaging layer provides the additional service of identifying the upstream and downstream channels on the Sensor Interface board. This is important to prevent cyclic loops forming if the network is installed incorrectly. On the Host Controller, multiple channels are used to provide additional ports on the system.

7.2.2.2 The Network Layer

The Network Layer utilizes the Messaging Layer to implement higher-level functions related to the functioning of the network. The primary responsibility of the network layer is to process and respond to messages passed to it by the Messaging layer. This layer also tracks all the addresses associated with the device and is responsible for determining the action to be taken.

The Messaging layer passes all valid message received upwards to the network layer. It is the responsibility of the network layer to check the address of the message and the type of the message in order to either process it or forward it accordingly to another device.

The Network layer handles many of the low-level functions required by the bus but it does not perform any functions specific to the device in question. The network layer is simply responsible for identifying these messages and passing them up to the next layer.

The Device Specific Layer

The device-specific layer is responsible for performing the specific functions required on the device. For a Sensor Interface Board these functions would be related to the acquisition and processing of sensor information. On the Host Controller, these functions include the configuration of the network through a network discovery process, the scheduling of tasks and the initialization of transfers on the network.

The purpose of separating this layer from the network is to simplify the process of implementing different sensors as they are separated from the communication layers and as a result, it allows the network layer to remain separated from any device-specific requirements.

7.2.3 Communication on the Bus

Communication on the bus consists of two different types of data packets; Bus messages and ACK messages. Bus messages are used to transfer both command requests and data back and forth on the network. These messages can originate from any device on the bus and have a special header field in order to indicate the direction of data on the bus. Although any Sensor Interface Board can transmit a bus message, a device will only transmit a message in response to a request from the host controller.

Bus ACKs are used to report the status of data transfer on the bus. These short packets contain only a few fields and are used by the host controller to ensure data delivery and to track progress of transmissions down the bus. Bus ACKs are only sent from devices to the host controller as the host controller does not send acknowledgements when data is received. The rationale for this is that the host controller would have initiated the transfer and therefore will be waiting for data. Any errors in transmission would be detected through timeouts or checksum checks on the host controller.

Most operations on the bus adhere to the following outline. The process starts with the host controller issuing a bus message containing an instruction addressed to a particular device. The first device on the network receives the message in full, checks the checksum to ensure accurate reception and checks the address field. If the device is not the target of the message, the device sends an ACK message back to the host controller informing it that the message was successfully received and will be passed along to the next device. The device then transmits the message to the next downstream device.

The following device then performs the same process until the target device is reached. This device then sends an ACK message informing the host controller that the message has reached its intended target. The ACK may include additional information such as whether a reply (in the form of additional data packets) is pending.

A device sending an ACK message towards the Host Controller, in order to inform it of having passed along an ACK message, would be a redundant process. No data moving in an upstream direction (towards the host controller) is passed to the network layer and no ACK messages are generated.

Special preambles are used to differentiate between messages originating from the host controller and response messages from Sensor Interface Boards. This also allows a device to determine its orientation on the bus as the side receiving the host preamble is assumed to be facing the host controller. This channel will be referred to as the upstream channel. The other channel is the interface to the rest of the network and is therefore the downstream channel.

7.2.4 Structure of a Bus Message

Bus messages are the workhorse of the communication protocol. A standard message size was chosen for all communication as it was felt that constant length messages would make it simpler to ensure that the bus was in a known state. Potential communication errors caused by the industrial nature of application make it important to ensure that the devices at both ends of a connection are synchronized.

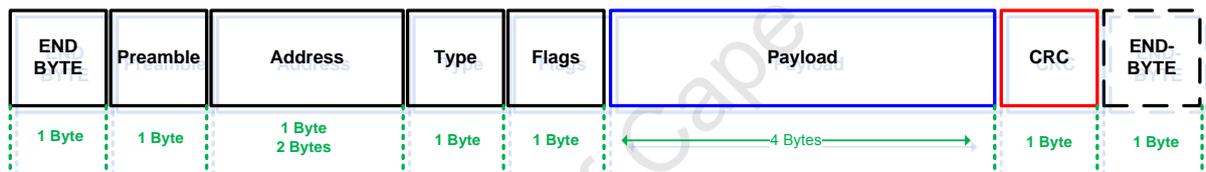


Figure 7-4: Structure of a Bus Message

The structure of the message packet is shown in the above figure. Each section of the message structure is discussed in the following sections.

7.2.4.1 The Initial END-BYTE

A special byte called an END-BYTE is always transmitted as the first byte of any message. This ensures that the receiver is in the correct state to process data. Receivers on the network ignore multiple END-BYTE characters when not processing a message which allows the message buffers to be flushed out with a sufficiently long sequence of END-BYTES.

7.2.4.2 The Preamble Field

The second field in the message structure is the preamble. This serves the dual purpose of marking the start of a message and indicating to the receiver the directionality of the message. There are three supported types of preambles in the current version of the protocol. These are given in the table below:

Preamble	Byte Value	Description
START_BYTE_HOST	0xFA	Indicates a message originating from the Host Controller and destined for a node on the network.
START_BYTE_CLIENT	0xAF	Indicates a message originating from a device on the network and destined for the host controller. Devices may selectively route these messages if it detects a cyclic loop forming.
START_BYTE_ACK	0xEE	Marks the start of an ACK message. Since the ACK message is of a different length, it is important to determine this as early in the message assembly as possible. As ACKs are only sent to host

		controllers, the direction is automatically implied.
--	--	--

Table 7-2: Valid Message Preambles

If the device receives any other value, it will abandon the message assembly and wait for a retransmission.

7.2.4.3 The Address Field

The address field is used to specify the destination of messages originating from the host controller. Messages destined for the host controller use this field to specify the source of the message. Devices on the bus use the preamble field to determine which direction to transmit messages and therefore prevent messages being transmitted back to their source.

The address field can be either one or two bytes wide, depending on the addressing scheme used. These are discussed in more detail in the section on addressing systems. Functionally, the two different addressing modes are implemented in the same fashion. The extended address field requires a slightly more complicated implementation on each device.

There are a number of reserved addresses on the bus which are used for network-specific operations. The zero address is reserved for un-initialized devices and is used extensively in the network discovery process. The 0xFF (or 0xFFFF for 2-byte addressing) is reserved for broadcast messages which are received and processed by every node on the bus.

When a device is powered up, it assumes the un-initialized address and responds to messages in the normal manner. The host will assign a specific address to the device during the network discovery process. Generally, every device on the bus must have a unique address. The exception to this rule is the zero address for un-initialized devices. When the network is initially powered up, all the devices will assume the zero address but will appear hidden as the first node will respond to all messages to the zero address. This is the premise behind the automated network discovery.

The Host controller on the bus does not have a specific address assigned to it and simply responds to all messages arriving on any of its channels. The design of the communication protocol is such that messages with the client preamble should never arrive at the host controller. If this occurs, then it is a clear sign of an error condition on the bus.

7.2.4.4 The Type Field

The type field specifies the command or return type of the message. The host controller usually issues commands to boards using specific values in the type field. Devices often respond to these commands with messages containing the same type field. Thus most types have slightly different definitions depending on the preamble.

Message type fields also specify how the payload data (see 7.2.4.6) is to be interpreted. The payload may contain additional information relating to the command being implemented. Certain instructions use the payload information to implement a wider set of functionality whilst other instructions may not utilize the field at all. In this case, the data is still required to be passed along in the event that future protocol revisions implement additional functionality and still remain backwards-compatible.

The type field is one byte in length to provide adequate room for future expansion. In earlier versions of the protocol, this field was split into two 4-bit fields; one to classify the type of command, and the other to store the actual command operator. It was soon realized however, that this was essentially the same as carefully organizing the allocation of byte values to type commands.

A summary of the commands implemented is given the table below. A more complete discussion of these commands is presented in the relevant sections.

Command	Byte Value	Description
PING	0x01	Used to check the presence of a device on a specific address. Clients respond to a PING message with a ping message containing their actual address in the payload. This can be used with a broadcast address to determine all the devices on a bus.
SET_ADDRESS	0x02	Host Controllers will issue a SET_ADDRESS message to assign a different address to a particular device. This is primarily used to assign a new address to an uninitialized device using the zero address.
POWER_SENSORS	0x03	Instructs a device on the network to power external sensors. This is used primarily for diagnostics purposes and must be used carefully as it may cause the power consumption of the bus to increase beyond the recommended limit.
INITILIZE_SENSORS	0x04	The INITILIZE_SENSORS is the first stage in the 4-phase sensor acquisition process. The entire process is described in detail in subsequent sections (see section 7.4)
BEGIN_AQUISITION	0x05	The second stage in the reading acquisition process.
AQUIRE_READINGS	0x06	The third stage in the readings acquisition process.
TRANSFER_READINGS	0x07	The transfer of readings from a device forms the last stage in the readings acquisition process. It is the stage where the devices are requested to place the acquired sensor data into the payload of a message byte.
TIME_ANNOUNCE	0x08	The TIME_ANNOUNCE command allows for periodic packets sent at various time intervals to allow devices implementing features such as running time or daily averages to update accordingly.
BUS_RESET	0x00	The bus reset command is used to force a device to reset its entire messaging and network layer to allow the network to reconfigure itself. This command works slightly differently from the other commands on the bus.

Table 7-3: Summary of the Different Message Types

7.2.4.5 The Flags Field

The flags field is used to pass various communication and network related flags between devices present on the network. The primary function of the flags field is to indicate the number of packets remaining on the outgoing buffer on the device. Certain operations may require more data than can be sent in a single bus message, in these cases, the messages are sent directly after each other, with the flags field used to indicate the number remaining. This concept is illustrated in Figure 7-5.

Only the lower half of the flags field is used for messaging tracking. The upper half is used to pass various communication flags between devices. These store information such as the number of retry attempts and the state and priority of the message.

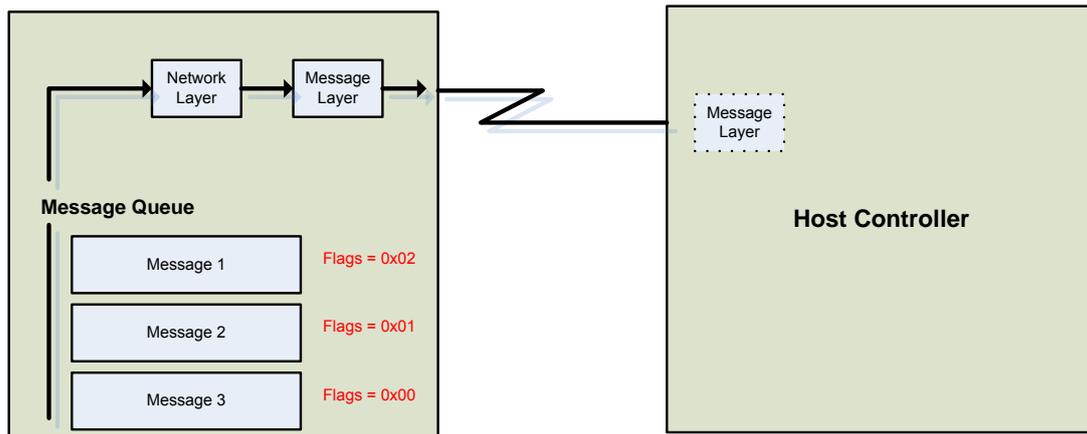


Figure 7-5: Illustration of Packet Numbering using the Flags Field

The majority of the flags bits have become redundant. They are a throwback to the original design of the bus which did not include ACK packets. In the original specification, ACKs were sent as standard messages or appended to existing messages through the use of the flags field. The new structure of the ACK message combined with a modified network approach has largely eliminated the need for this field.

A potential use of the flag bits involves the change from a fixed-length messaging format to a variable-length one in which the flags field is used to specify the number of messages and use the upper 4-bit value to specify the number of payload fields included in the message. The other option is to remove the field completely and adhere to a strict limit of a single-message response. The divergence is a result of the different requirements of the two addressing systems.

7.2.4.6 The Payload Fields

The payload field contains the actual data sent in the message. The structure of the data within the payload is subject to the type of message being sent and the version of the protocol running on the network. In terms of the layers of the communications protocol, the header information is processed by the network layer whereas the actual data is always processed by the Device-Specific layer.

It is important to note that the device-specific layer will often examine the message type when processing the data contained in the payload fields. Unlike many layered protocol models, headers and frames are not removed when passing messages up to higher levels. The distinction is rather that the lower levels will not modify data destined for higher layers. As a result, the network layer will never alter the contents of the payload when passing messages between devices, as this is entirely a function of the device-specific layer.

7.2.4.7 The Checksum Field

The checksum field contains a simple checksum to ensure the accuracy of the data being received. It is calculated over both the headers and the data and is checked at the network level of the protocol. Therefore, bad messages are automatically reported (via ACK messages) to the Host Controller for retransmission and should never affect the Device Specific Logic. This allows the Device Specific layer to assume that any messages received are the valid for processing.

The checksum field is currently implemented as a simple modulus addition of all the byte values contained in the message. This process is relatively fast but only provides rudimentary error detection. Room exists for a more intricate checksum formula to be implemented but the current method has proven effective enough for the application.

7.2.4.8 The Terminating End-Byte

All bus messages should be terminated with an End-Byte character although it is not strictly enforced. A message is considered fully assembled when the checksum field is received. It is considered valid once the checksum field has been verified successfully.

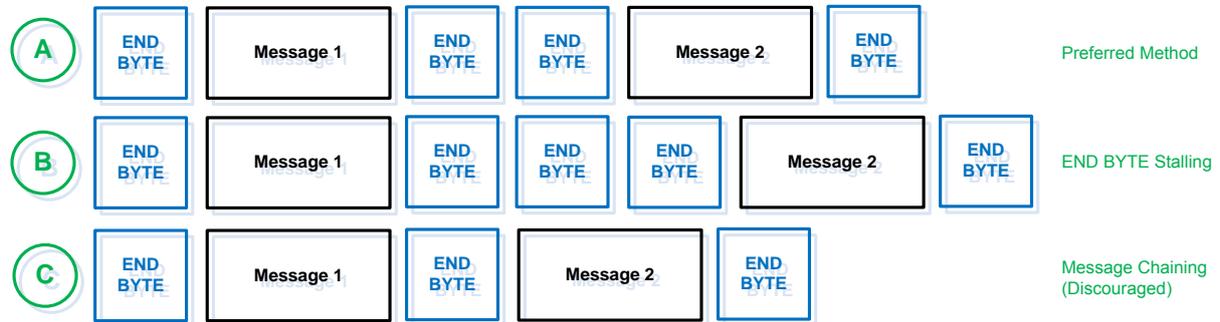


Figure 7-6: Illustration of the different methods to terminate a bus message

The purpose of the terminating END-BYTE is to ensure that the target receiver is ready to receive the next message. The figure above demonstrates three valid means of framing a message on the bus with END-BYTES.

The first method adheres to the strict definition of the message structure. Each message is framed between two END-BYTE characters. One of the advantages of this method is that it makes it far simpler to analyze traffic on the bus. It also proves to be more reliable, especially if there is a noticeable delay or direction-switching occurring between messages.

The second method (B) indicates multiple END-BYTES being sent between messages in a process called END-BYTE stalling. This allows a device to hold the channel open between messages and still ensure that messages are detected correctly. END-BYTE stalling is often used in situations where both channels are being used simultaneously or when a device is recovering from a bus error.

Devices are expected to ignore multiple sequential END-BYTES in between messages and use the preambles to detect the start of a message. This allows devices to effectively flush the message buffers of the receiving device by sending a sequence of END-BYTES the length of a message.

The third example demonstrates a technique called message chaining. In situations where messages are transmitted sequentially (using the decreasing-count technique mentioned in the Flags section), the device can use the last END-BYTE of the previous message as the first END-BYTE of the next message. In the case where the END-BYTE is a single character, this is a saving of only a single byte, but when a more complicated string is used for the END-BYTE (as has been tested on a few long links), this can become a significant saving.

7.2.5 Structure of a Bus ACK

ACK messages are specialized short messages sent from devices on the network to the host controller. These messages are used to provide information to the host controller regarding the state of communications on the bus. The structure of an ACK message is given below:

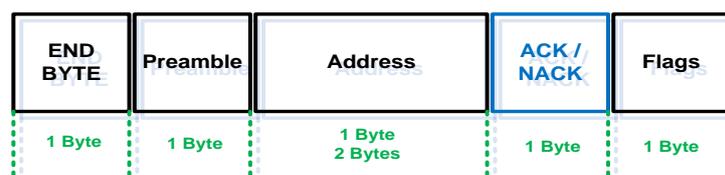


Figure 7-7: Structure of a Bus ACK

7.2.5.1 The Initial END-BYTE

All messages transmitted on the bus require an END-BYTE in order to synchronize the channel prior to transmission.

7.2.5.2 The Preamble Field

There is a special preamble to identify ACK messages on the bus. This is the same value for both ACK and NACK messages and is simply used by the receiver to determine the message assembly process to follow. This needs to be determined at the earliest possible step as ACK messages have a completely different structure to a normal bus message and follow a completely different path through the messaging layer.

Any character that is not a valid preamble will force the device into an error state. The only exception to this is an END-BYTE sequence which will simply be ignored (to allow END-BYTE stalling). Unlike the preamble in the bus message, the ACK field does not explicitly specify directionality; it is implied as bus ACK messages only travel from devices to the host controller. Therefore, ACK preambles can be considered to be a special case of the START_BYTE_CLIENT preamble (see Table 7-2).

7.2.5.3 The Address Field

The address field in an ACK message is identical to its counterpart in the bus message. It can be either one or two bytes wide depending on the addressing scheme in use. As messages always originate from devices on the bus, the address field always contains the address of the source.

7.2.5.4 The ACK/NACK Field

The ACK/NACK field is an entire byte dedicated to the reporting either a success (an ACK) or a failure (a NACK) of an operation on the bus. Although it is essentially a Boolean value, an entire byte is dedicated to the value to make implementation on certain types of devices simpler.

Most devices on the bus do not parse or interpret ACK messages being received. As a device is never the destination of an ACK message an ACK will simply be passed along on the other channel to an upstream board. The nature of the communication protocol is such that ACK messages should never travel downstream (i.e. away from the host controller) and therefore this action should never cause cyclic loops to form.

A substantial amount of consideration was given to how the bus and network would be expandable both in size and function. As a network grows, it might be necessary to incorporate devices that process messages on a higher layer (in a similar fashion to a router or bridge on a local area network). The ACK/NACK byte was kept as a full byte in order to make the interpretation of the result of an ACK message simpler and faster.

7.2.5.5 The Flags Field

ACK messages also include additional information regarding either the success or failure of the operation in question. This data is transmitted using the flags field. Each bit in the Flags field corresponds to a particular situation applicable to the ACK message. Through the use of bitmasks and bit-fields, these can easily be chained together on the bus and interpreted on the Host Controller.

The Flag field has the following structure:

Bit	Flag	Description
0	ACK_SUCCESS	Indicates that the requested operation on the bus was a success. This is often used to indicate that the message has reached its final destination and signifies that the communication to the target was a success. The host controller may choose to interpret the precise

		meaning of this flag based on its current state.
1	ACK_TIMED_OUT	If a message could not be delivered to a device downstream, the transmitting device might return an ACK message to indicate that the transmission failed. This is often used to determine the last node on the network.
2	ACK_PASSED_ALONG	Devices may choose to inform the host controller when a message has been passed along to a downstream device. This is often useful in debugging and pin-pointing problems in the network. The use of this particular flag is heavily influenced by the choice of addressing mode.
3	ACK_END_OF_BUS	As multi-drop networks must terminate the last receiver on the bus, it becomes easy for the device to determine that it is the last node on the bus. The device would send an ACK_END_OF_BUS message to inform the host controller that it has reached the last node.
4	ACK_RETRYING	When a device attempts to retransmit a message (due to a possible collision or unbalanced line condition), it sends a special ACK message to indicate this condition. The ACK may not necessarily represent a message failure; it simply indicates a problem transmitting on a single channel.
5	ACK_CRC_FAILURE	When a message fails a CRC check, the device detecting the failed CRC will inform the host controller using this flag. This flag is only sent with a NACK.
6	ACK_PENDING	When a device receives a message, it may need to perform an action before returning data. In order to allow for this, the device may send an ACK message with the ACK_PENDING flag to indicate that the message was successfully received and further bus messages will follow.
7	ACK_ERROR	When a requested operation fails, a device may use an ACK message to indicate an error. Depending on the state of the host controller, it may request further error information or handle the error internally.

Table 7-4: Summary of the Bus ACK Flags

As this is a bit field, these flags can be combined as needed to indicate the state. Although any combination of these bits will constitute a valid message, not all of the flags can be combined in a logical fashion. The decision and priority of one flag over the other is done purely at the discretion of the host controller and therefore the outcome should not be considered either reliable or reproducible.

7.2.6 Network Layer Messages

Most of the messages received on a device are passed up the layers of the communication protocol until they arrive at the device-specific layer. These messages tend to be high-level commands which access the functionality provided by the device. There are a number of commands on the network which are central to the functioning of the communications network and are therefore not passed to the device-specific layer, but rather handled on the network layer itself.

These messages are low-level commands which are required on every device on the network, regardless of the higher-level functionality provided. These commands are therefore implemented in the network layer to ensure that the strict division between physical interface code, network interface code and device-specific code.

These messages are also used by both addressing systems, although with a few minor differences. These messages are discussed in detail below as they are critical to the functioning of the entire network.

7.2.6.1 The Ping Message

Ping messages are used primarily to test if a device is present on the specified address. Ping messages originate from the host controller, Ping reply messages can originate from any device on the network. The Host Controller sends out ping request messages to specific addresses. A device at the specified address will return a ping reply message and uses the payload fields to return additional information.

A Ping message has the following structure:

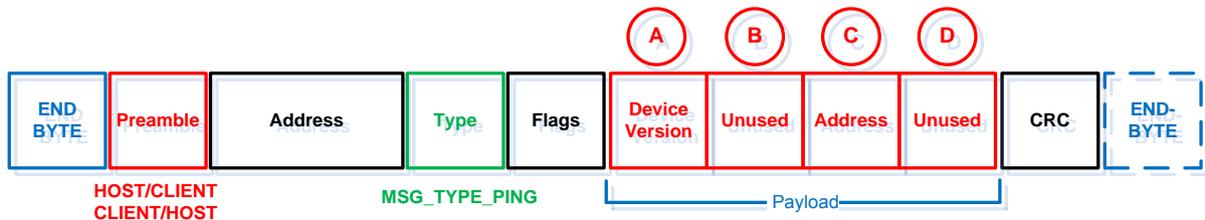


Figure 7-8: Structure of a Ping Message

The payload of a ping message contains additional information regarding the originating device. The first byte (A) is used to allow a device to provide versioning information such as the type of the device. The second byte (B) has no specific use but is used in some addressing situations to return the number of sensors connected to a device. The last two bytes are used to store the address (which can be either 8-bit or 16-bit in resolution).

A host controller will send a Ping message utilizing only the device version portion of the payload. As the host controller does not have an address in the same manner as a device on the bus, it does not use the address bytes of the payload. Commonly the host controller sends the version number of the firmware in the device version byte. This is not commonly used by devices on the bus and is included primarily for completeness sake.

Devices on the bus make better use of the fields in the ping message. The device version field is used to specify either the type or the version of the device responding to the ping. This data is used by the host controller to perform specific initialization for devices on the bus. The second field (B) is used in network addressing to indicate the number of sensors attached to a specific sensor interface boards. If sensor-based addressing is used, then this field is reserved for future use.

The device will also load its current address into the address fields of the payload. It is important to note that a device will only issue a ping message in response to a ping request from the server. The address field is included in the ping reply to handle the case where the device is responding to a broadcast message (see section 7.2.4.3).

In the case of a specific ping request sent to a single device on the bus, the host controller will presumably know the address of the intended target of the device. The host controller can also issue a ping command on the broadcast address, in which case it will be received by every device on the network. All devices will respond with a Ping reply and as the Host Controller can examine the payload fields of each reply to determine the addresses of all devices present on the network.

7.2.6.2 The Set Address Message

The Set Address message is a bus command issued only by the host controller. It is used to force a device to change its address to a value specified in the payload. It forms the basis of the automated network discovery process. The Set Address message has the following structure:



Figure 7-9: Structure of a Set Address Message

The Set Address message must be issued to a specific address and cannot be sent to the broadcast address. The reason for this is that this would cause multiple devices on the bus to assume a single address and violate the unique address requirement on the bus.

The only exceptional case would occur if the un-initialized address was sent as a broadcast message. This type of operation is extremely important to the functioning of the network and is actually provided by a separate message type called a bus reset. There are a few differences in the manner in which a bus reset is implemented and therefore a Set Address message is never allowed to be sent to the broadcast address.

The host controller uses the payload data to store the new address for the device on the bus. Depending on the length of the addressing mode in use, this can occupy either the first byte or the first two bytes. The other bytes are currently unused.

A device receiving a Set Address message is required to change its address and then send an ACK message from the new address. Therefore, if a Set Address message succeeds, then the ACK will contain the new address, otherwise, in the case of failure, it will contain the previous address.

7.2.6.3 The Bus Reset Message

The Bus Reset command is a message issued only by the host controller. It is used to instruct devices on the bus to perform a soft-reset in order to force the entire network into a known state. This process is complicated by the fact that certain aspects of the device cannot be disconnected whilst a bus message is being transmitted.

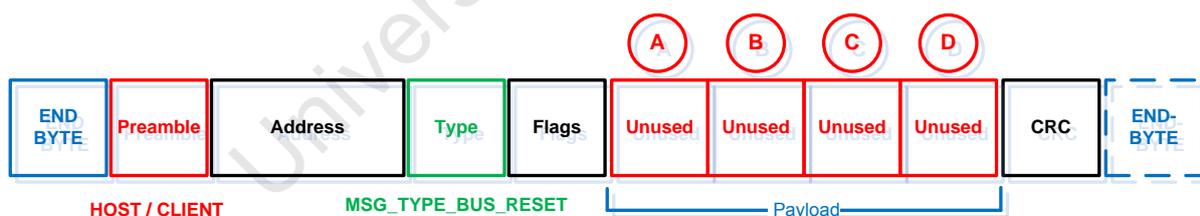


Figure 7-10: Structure of a Bus Reset Message

Bus Reset messages form an exception to the common rules on the network. These messages are implemented in the same fashion on all devices, regardless of addressing scheme in use. The bus reset is the lowest level command that can be issued by the host controller as it is the only command that affects the entire protocol stack.

The bus message has an exceedingly simple structure. It can only be issued from a host controller and does not contain any additional information in the payload fields. A bus reset message can be addressed to either a single address but is more commonly sent to the broadcast address. Sending a bus reset message to the broadcast address is often used to reset every device.

Bus Reset messages are handled in a completely different way to all other messages on the bus. The actions of all other commands on the bus are dictated by the addressing method implemented. Bus

Resets on the other hand, are always handled as if the bus implemented a device-based addressing system. In essence, the bus reset command acts on the physical hardware rather than the logical structures presented to the network.

The reason for this is that the bus reset is a very low-level command which requires a device to reset many of the internal variables required to implement the network protocol. A single device may implement multiple logical addresses and therefore resetting logical addresses may result in the situation where a device resets before all logical devices are reset. This would result in messages that cannot be delivered which would result in increased traffic and delays in the bus.

The method in which bus resets are handled depends on the addressing system implemented on the network. For this reason, the specific information on how the reset is performed is discussed in more detail in the section on Device-Based Address (section 7.3.3.3) and Sensor-Based Addressing (section 7.3.4.3)).

7.3 Addressing Modes

Two different addressing systems were developed during the design and testing of the system. The addressing system needs to ensure that each device is uniquely addressable and it must also be able to infer the location of the devices based on the address. The location of a device is determined by ensuring that it holds the same address for the entire operating duration.

Although the addressing system is technically a protocol-level task, it has been separated from the protocol design to highlight the fact that both techniques are interchangeable network layer technologies and utilize the same base communication protocol.

7.3.1 Overview of the Addressing Modes

Many network technologies use a physical system of addressing. These may include preset addresses or physical jumpers on each device. The first version of the sensor interface boards included an 8 bit header to allow an address to be specified using jumpers. Although it proved an effective method of implementing addresses it suffered from a number of problems relating to the installation.

The primary concern was that the installer would need to set each address correctly and this would involve specific configuration for each board on the network. The protocol provides no means of ensuring a unique address in this situation and will also be unable to detect address conflicts if an address appears multiple times on a network. Adding extra devices to the network would also require that the addresses on each board be checked.

It was decided that a dynamic addressing system would need to be developed to eliminate these problems. The system would need to ensure that it assigns the same addresses to each device provided that all the other parameters remain unchanged. This would remove the need for a hard-wired address on each device and make the installation of the network far simpler.

The first system developed implemented a device-level addressing system. It proved to be very effective and is currently being used on at least two sites. There were a few shortcomings in the design of the system, primarily resulting from the manner in which it evolved into its final state. In order to cater for a number of new features required, a second system was developed that implemented a sensor-level addressing system. These two systems are discussed in detail in this chapter.

7.3.2 Cascading Effect of the Addressing System on the System as a Whole

The choice of addressing system has a dramatic impact on the performance of the system as a whole. This results from the fact that the address is the only unique identifier present on the bus and

therefore is the basis for determining the source of data on the bus. The server receiving data from the unit must use the addresses to interpret the incoming information and process it accordingly.

The requirement that a device must receive the same address under the same circumstances therefore allows this mapping of sensor information to address values. This mapping is can therefore be considered static and needs only to be changed when the physical network is altered through the addition or removal of devices. The creation of the mapping and relevant database on the server is the most labor-intensive aspect of configuring a new site (apart from the installation of the system).

The nature and the complexity of this mapping are directly influenced by the choice of addressing system. Device-based addressing resulted in an indirect mapping which becomes increasing complicated as the size of the network increases. This was one of the primary reasons for the development of the sensor-based addressing system. Both these methods are discussed in detail below.

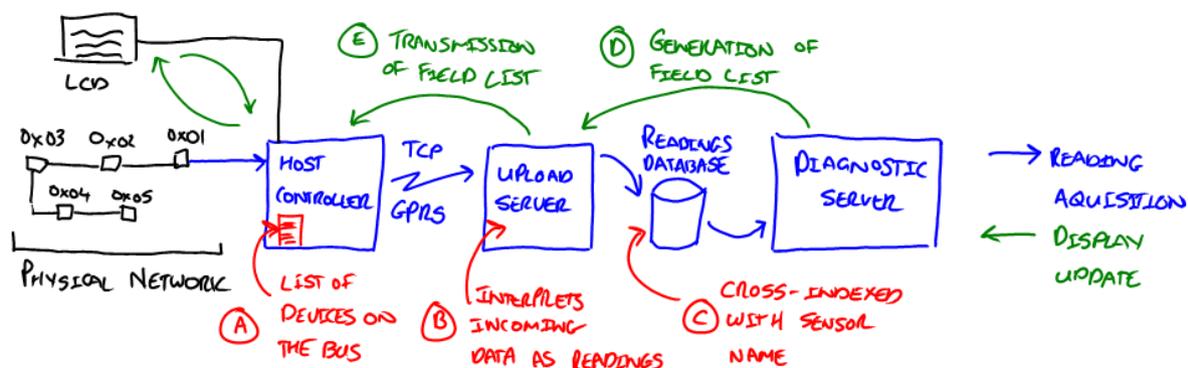


Figure 7-11: Propagation of Addresses through the System

The above figure illustrates how the address is used throughout the system. It is important to note that although the addresses are always assigned by the host controller, it does not actively know of the mapping between sensor, device and address on the network. The purpose behind this design choice was to prevent specialized code on the host controller being written for specific sensors. It was felt that this would cause the need for the host controller to be specially configured for each installation and therefore increase both the complexity and cost of each new installation.

The host controller acts purely as a data router instead. It is capable of detecting and configuring devices on the network but is prohibited from interpreting any data in messages from the device-specific layer. The host controller simply obtains a list of all the devices on the network, request values from these devices periodically and then uploads the data to the server for processing. Although the host controller does not know the specific function of a device on the network there are provisions for a device to request certain parameters such as polling time.

The server responsible for handling the incoming data from a host controller must interpret the data stream and break it down into individual sensor information. This is currently accomplished with a static mapping file that is loaded once the host controller has identified itself to the server. The upload server then stores the sensor information in a relational database and cross-references it by sensor name to allow the diagnostics server to operate on logical fields as opposed to unintuitive addresses.

The upload server essentially provides the service of converting addresses into relevant sensor information. It is also the only component responsible for maintaining and updating the readings database used by the entire system. Each new installation will require a new mapping file and a new database table in order to handle the incoming data. If diagnostics services will be provided, then

the appropriate diagnostics must be configured in the diagnostics server. If these already exist, then there is no need to duplicate them as long as the upload server can map the addresses to the same fields as the previous installation. This allows much improved re-usability in the diagnostics server.

One interesting point is the manner in which the host controller displays information on its LCD screen. Diagnostic information can travel back through the upload server to the host controller to allow it to display readings, diagnostics or user-defined messages. This may give the impression that the host controller is tracking the actual sensor information on the bus but is rather only a reflection of the processes running on the server.

7.3.3 Device-Based Addressing

Device-Based Addressing evolved along with the system as it transitioned from a multi-drop bus with fixed addresses into its current state. It therefore has a number of characteristics which resulted from ideas and concepts that are no longer present in the system. Device-Based addressing is however very effective and is currently in use on at least two different sites.

The smallest addressable component in Device-Based Addressing is the Sensor Interface Board. Each board is assigned a unique address during a process called a network discovery. This address persists throughout the lifespan of the system (provided that no physical changes to the bus occur). The host controller therefore communicates with the physical board and issues commands to the hardware directly.

7.3.3.1 The Automated Network Discovery

In order to remove the need for jumpers or switches on each sensor interface board, the network discovery process was developed to automatically assign addresses to each device on the bus. This process automatically determines the number of devices connected and proceeds to assign addresses in a predictable fashion to ensure that a device will receive the same address each time the network discovery is performed.

The network discovery process exploits the daisy-chain nature of the bus in order to detect devices on the network. The process is similar to the USB enumeration process and also borrows certain concepts from the daisy-chain method of implementing priority interrupts. In order for the network discovery to function, each device must assume the default un-initialized address on start-up.

The nature of the daisy-chain bus gives rise to the problem of shadowing, which occurs when two devices occupy the same address on the network. In this situation, the board furthest from the host controller (in terms of links, not physical cable length) will be completely hidden by the closer board. This is precisely the situation that occurs when power is applied to the system and all the devices assume the un-initialized address.

The network discovery process exploits this shadowing nature as the first board on the network essentially hides all other devices on the bus. The host controller will then set up this device and assign it a new address, which will reveal the next un-initialized board on the network. The host controller can then rely on the first configured board to pass all messages destined for the un-initialized address to the second board and therefore can proceed to configure the second board. This process is repeated until there are no more un-initialized devices on the network.

Figure 7-12 gives a graphical view of the first few steps of a network discovery process. The host controller uses the PING command in order to determine if there is an un-initialized device on the network. If a valid response is received (note that a device includes its own address in the payload field of a PING message to allow this to be verified), the host controller will then issue a SET ADDRESS command with a new valid address.

A device must respond to a SET ADDRESS command with an ACK or NACK and change its address on successful completion. Originally the device would simply refuse the address and remain un-initialized but this resulted in phantom boards appearing when an invalid command was detected. The formal ACK and NACK system allows the retransmission system to function which eliminates that problem.

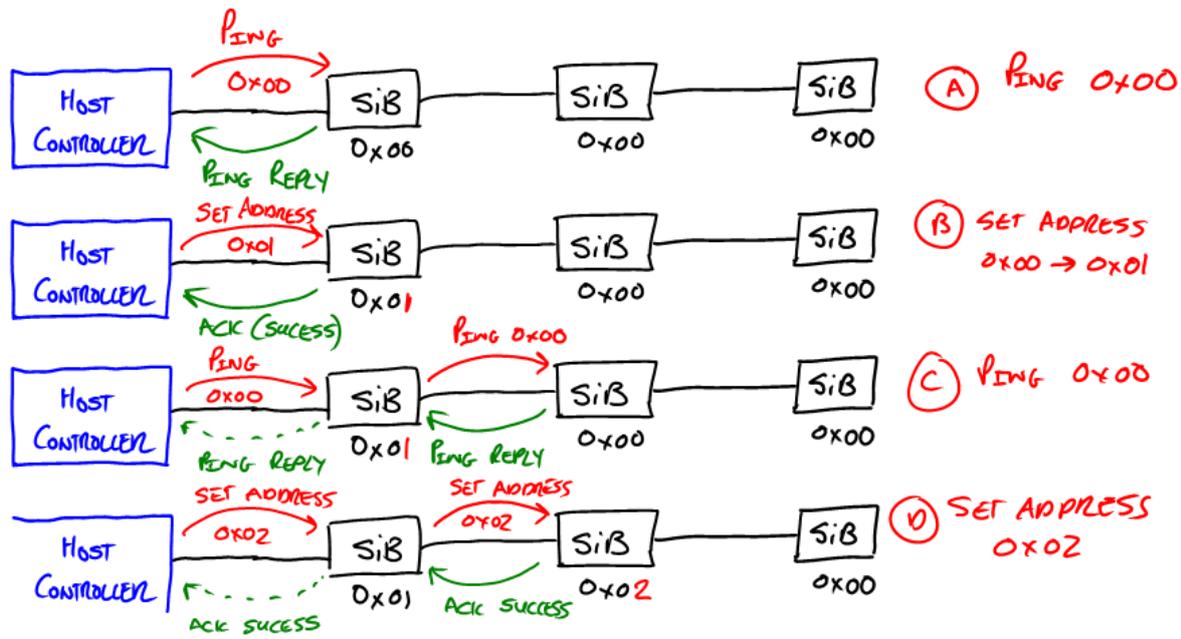


Figure 7-12: Illustration of the Network Discovery Process

If the host controller receives a valid ACK indicating success from the new address, it will then continue the network discovery process by issuing another PING to the un-initialized address. The first board, which is now configured with a new address, will ignore the message and pass it along the next device on the network. This device will then respond in a similar fashion to the first board with the first board acting only as a relay for messages and ACKs.

This process will continue until the host controller does not receive a response to a PING message on the un-initialized address. This indicates that the last device has been successfully configured and the network discovery process can terminate.

The amount of time the host controller waits for a ping response is a predefined constant and must be long enough to ensure that the propagation delay on a network with a large number of attached device does not prematurely terminate the network discovery process. The value of this parameter was determined experimentally.

The host controller builds up a table of devices during the network discovery process which is used in all further bus operations. This table includes information gathered from the payload packets of the PING command and reserves space to store future readings. When a device captures a reading it places the data into a memory structure and must wait for the host controller to read the data. This is analogous to the manner in which a USB host controller reads data from device endpoints. The received data is stored in the table established during the network discovery.

In order to store data for a large network, an external memory can be used to store the table information. In order to facilitate this, the size of each entry in the table must be static. When using device-based addressing, each record stores twelve data bytes, dividable in any manner required by the sensor interface boards.

7.3.3.2 The Network Inventory Process

The network inventory process is largely a diagnostic procedure as it has been surpassed with the introduction of the network bus reset. The network inventory process is a means to rebuild the table of devices by examining all the addresses in use on the network. It can only be run once a complete network discovery has been performed and was originally used to ensure that the host controller is synced with the rest of the network.

The network inventory uses a PING message sent to the broadcast address to illicit a response from each device attached to the network. This relies on the means by which a broadcast message is handled by devices on the network as the each device will process the message before passing it to the next device. In the case of the PING message, this will cause the board to respond to the ping before passing the message along.

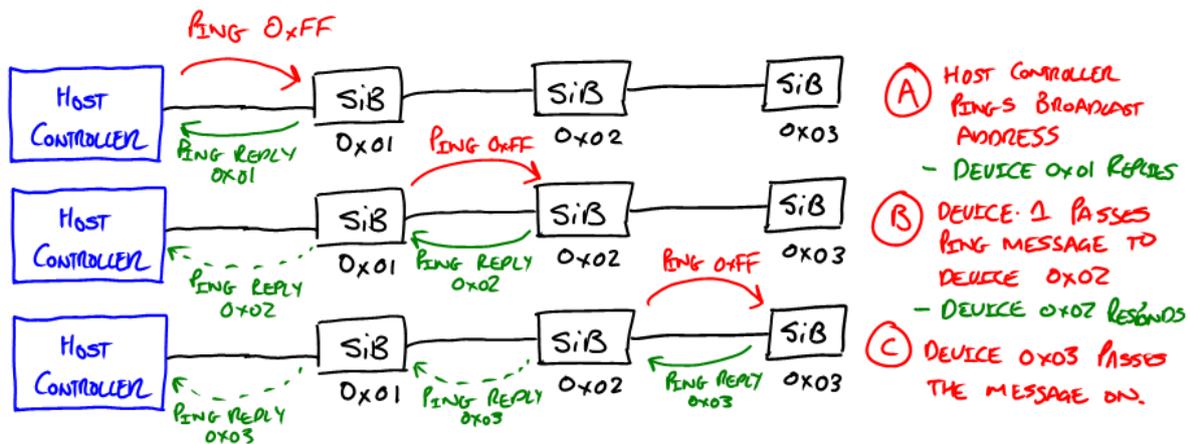


Figure 7-13: Illustration of the Network Inventory Process

The above figure illustrates the network inventory process. When the first device on the network receives the PING message addressed to the broadcast message, it first processes the ping and replies before passing it along to the second device. The network inventory process relies on the fact that a device includes the address in the payload fields of the PING reply which informs the host controller of the address of the device.

The Host Controller simply creates the table based on the order in which the addresses are received. The means in which broadcast messages are handled on the bus eliminate the possibility of the PING replies arriving in the incorrect order. The only problems occur when a PING message is not successfully transmitted to the next device and the network inventory process prematurely terminates.

As with the network discovery, there is a timeout that must elapse before the network inventory process terminates. In the case of the network inventory, it is the time since the last valid PING response was received. This timeout must be greater than the total network propagation to ensure that the process does not stop while a message is being relayed to the host controller.

The network inventory process originates from the original network design where each board had a fixed address. In that configuration, it was necessary to perform an inventory of all devices on the bus prior to start-up. One benefit of this approach is that it is capable of detecting duplicate addresses on the network. The primary reason for the inclusion of this process is that it is extremely useful for debugging the system as it allows a host controller to rebuild the entire device table without having to perform either a hard or soft bus reset.

7.3.3.3 Bus Resets

Occasionally it is necessary to reset the network and perform a complete network discovery to rediscover all the devices on the bus. This situation may occur when a device is connected or removed from the network or when a device stops responding. In the original system it was assumed that disconnecting the power to the host controller would cause the entire network to reset.

The 'hard' reset works with small networks involving short links but cannot handle longer networks which may have separate power supplies on the end of long cable runs. In this case, powering down the host controller will not result in all the devices on the network resetting their addresses and this will cause the network discovery to fail. For this reason a special broadcast message, known as a bus reset, was introduced to cause all attached devices to perform a soft reset.

A soft reset does not require the device to disconnect and reconnect power. It simply requires the device to restore all the layers of the communication protocol to their power-on state. The most important requirement is that the device reset its address to the un-initialized address. Devices are also required to flush all message buffers and discard any sensor data stored in outgoing sensor records.

Due to the nature of the bus reset message, it is handled differently on the network. A device receiving a bus reset message must assume that the sending device has just performed a bus reset and therefore cannot relay messages up to the server. For this reason, no ACK messages or retransmission mechanisms are used during a bus reset. This results in the bus reset message being more unreliable than other messages on the network.

Two different methods have been devised to overcome this problem. Neither solution guarantees success and therefore it was decided to do a complete bus reset and a network discovery at a regular interval to ensure that intermittent problems do not persist indefinitely. On particularly problematic networks, a network inventory can be performed prior to readings being taken to ensure the integrity of the device table.

The first method is to simply perform multiple bus resets in a row. This technique is only effective when the problem arises in the message relaying mechanisms on the bus. Situations such as this can arise when the message assembly buffers become corrupted. In the case of a faulty link, where there is certain percentage chance of a transmission failure, there is no benefit to performing multiple bus resets as the last reset is just as likely to fail as the previous one. It is important to note that this is more likely as none of the message retransmission mechanisms operate when performing a bus reset.

The second method is somewhat more effective in practice but is both limiting and not easy to expand. Instead of issuing the bus reset to the broadcast address, the host controller issues a bus reset message to each valid address on the bus. This is performed in a descending fashion to ensure that the device furthest from the host controller is reset first.

This process has proven more effective in resetting the bus but can only be used with 8 bit addressing before the bus reset process takes a significant amount of time. The process does not overcome the problem with faulty links but does allow the retransmission mechanisms to be used as the devices are now reset in a descending order.

7.3.3.4 Accessing Sensor Data using Device-Based Addressing

Device-Based addressing assigns each sensor interface board a unique address and therefore the host controller must send a request to a board to acquire readings. As a sensor interface board usually hosts multiple sensors, the host controller must either request all the sensor information at once, or query each sensor individually.

In order to query each sensor individually, the host controller must therefore be aware of the number of sensors on each board. As the device table in the host controller must be a static size, the number of sensors attached to each board needs to be limited to a constant number. The host controller would also need to query each sensor on each board, which dramatically increases the complexity of the host controller code. For this reason, it was decided not to allow the host controller to query individual sensors when using device-based addressing.

Instead, the host controller simply requests all the data from the sensor interface board at once. The device then transmits the data as a sequence of messages, sequenced using the Packet Numbering technique described in Section 7.2.4.5. Currently the sensor interface boards each return three messages in sequence and allow the device-specific logic to pack the data into the payload bytes in any combination.

These packets are then stored in the device table on the host controller and uploaded directly to the server without any processing or interpretation by the host controller. It is the task of the upload server and the appropriate mapping file to decode the payload data into sensible sensor information. Unfortunately, as a result of this process, the mapping can become complicated and it becomes difficult to detect problems.



Figure 7-14: Example of how a Pressure Board packs data into the payload fields

The above image demonstrates how a 3P1C (3 pressures and 1 current) sensor interface board would pack data into the three message packets in order to send it back to the host controller. The type field has been deliberately left out of the image above as the reading acquisition process is described in greater detail in the section below.

7.3.3.5 Shortcomings of Device-Based Addressing

Device-Based Addressing evolved with the development of the system and therefore has a number of issues which impact its effectiveness. It remains an effective means of addressing devices on the bus and is currently in use on a number of different sites. The limitations and scalability concerns lead to the development of an alternative addressing method.

The first limitation with device-based addressing was the difficulty in packing data into the payload fields in an effective manner. This has the knock-on effect of increasing the complexity of the mapping process on the upload server which dramatically increases the work required to add a new host controller to the system.

The design of the system also limits the number of sensors that can be installed on any particular sensor interface board as there are only a limited number of payload packets available. If the number of message packets sent from each is increased, this has the effect of increasing the total memory required on the host controller and also increases the amount of data to be uploaded to the server.

Sensor-Based Addressing was designed to eliminate many of these issues and extend the functionality of the system to incorporate a number of features that were not possible using device-based addressing.

7.3.4 Sensor-Based Addressing

In Sensor-Based Addressing, the smallest addressable unit changes from the sensor interface board to the specific sensors connected to the network. The immediate effect of this change is to change the role of the sensor interface board from an active device on the network to a discrete routing node. The Sensor Interface Boards simply acts as a hub on a star network, which has the effect of removing many of the limitations on the number of devices that can be attached to a sensor interface board.

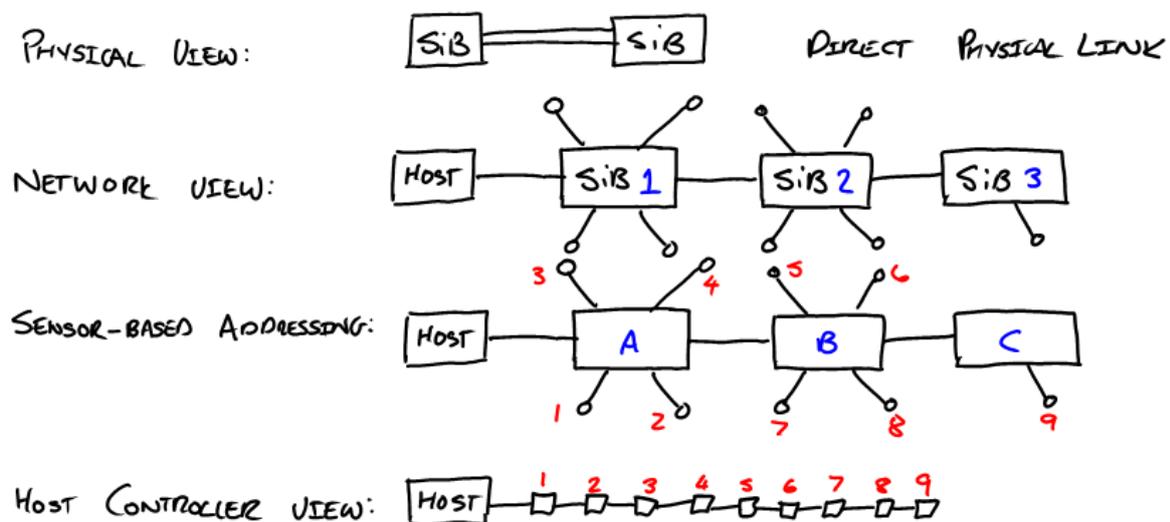


Figure 7-15: The Network Perspectives viewed by each layer of the protocol

The purpose of the addressing system is to provide a logical and device-independent view of the network for the upper layers of the communication protocol. This allows the sensor-specific logic to be functionally separated from the physical communication protocol. The figure above demonstrates the method by which the addressing mode and communications protocol simplify the nature of the bus for the higher level functions.

The physical view is a function of the daisy-chain nature of the bus and represents a physical connection between two devices. Devices are functionally physically connected together to form the network view, which is seen by the network layer of the protocol. Device-based addressing can be viewed as a network-layer addressing system that simply assigns a unique address to each physical device on the network. This has the effect of simplifying the network by hiding the sensor attached to each device.

Sensor-based addressing extends this process further by exposing the sensors to the host controller directly instead of hiding them. In order to accomplish this, the sensor interface boards become transparent and therefore are not visible to the host controller. When using sensor-based addressing, the host controller simply sees a bus consisting of sensors, each with a unique address.

It is important to note that sensor-based address also keeps track of the number of sensor interface boards attached to the system. The devices themselves are not addressable though, and cannot be the target for bus messages and commands. The purpose of tracking the number of boards is purely to assist when something goes awry on the network. Each sensor interface board is associated with

a range of addresses and therefore can be used to pinpoint the location of a fault. This process is entirely transparent to the device-specific layer of the protocol.

7.3.4.1 Support for Multiple Channels

One of the most important additions included in sensor-based addressing is the support for multiple channels on the host controller. Originally the host controller was designed to support a single bus connection over a single cable. This single-channel solution worked well in small buildings or plant rooms but the single cable solution proved difficult in large installations with long cable runs.

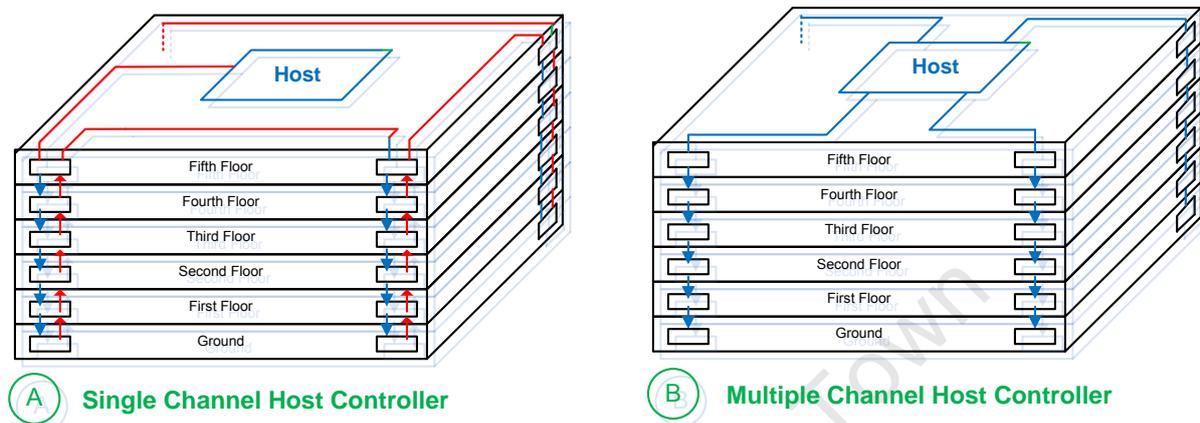


Figure 7-16: Example of the benefit to supporting multiple channels on the host controller

A practical example of the use of multiple channels on a host controller is a large building such as the one shown in the figure above. The building in the figure has a central plant room on the roof and four air handling units in the corner of each floor. This setup is typical of a large building with a central air handling system and requires units in each air handling unit and the central plant room.

The host controller was situated in the plant room as it is the central point in the building and has pre-existing conduits to each of the air handling units in the building. The first figure on left shows the cabling required if a single channel host controller is used. As each device needs to be on a common cable, the cable needs to loop upwards and across to reach each corner of the building. Cable could have been saved if the cable was run along the ground floor instead of rising up but this is difficult to do in practice as the ground floor does not have a suspended ceiling.

A multiple channel host controller solves this problem by allowing four distinct channels (and therefore four separate bus cables) to leave the host controller. In this case, each corner of the building is handled by a separate channel, which dramatically reduces the cable length and also the length of each bus segment. Multiple channels increases the complexity of host controller and most beneficial in large installations where long cable runs are present.

Channels are implemented solely on the host controller. Devices on the network are not aware if they are installed on a multiple-channel host controller as it does not impact on their performance or the characteristics of the communications protocol. Host controllers implement each channel as a separate network for most tasks on the network such as acquiring readings. Network discoveries are performed across individually on each channel with the exception that the addresses are sequentially assigned across the channels. Therefore each device on the network still receives a unique address on the network.

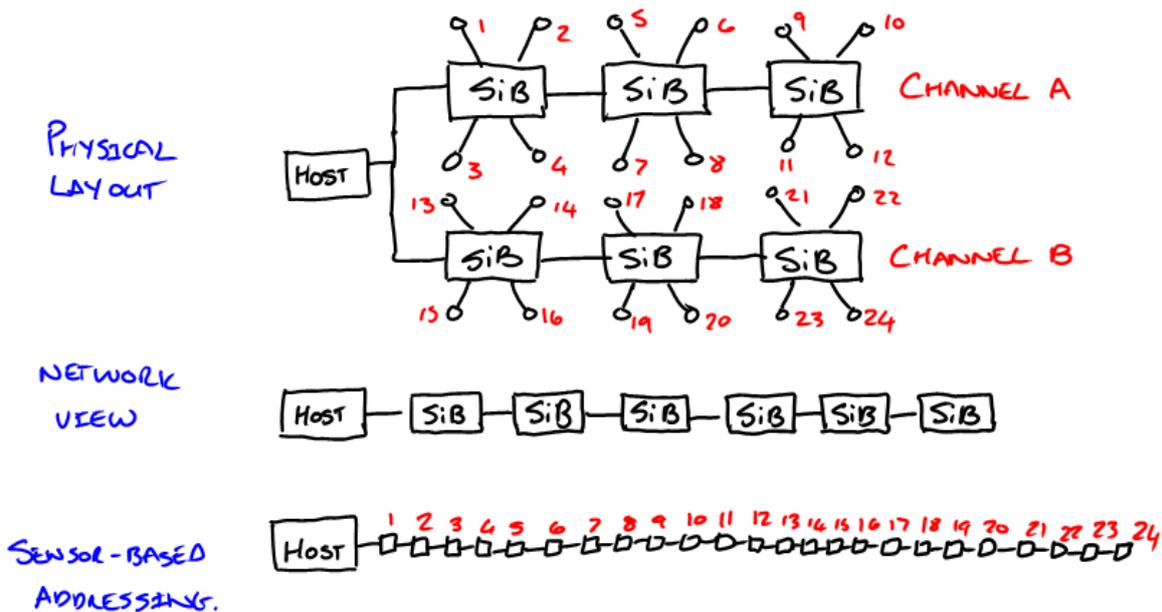


Figure 7-17: Channel-Merging in Multiple Channel Host Controllers

When referring to a multiple-channel host controller, the term 'network' refers to the all the devices on every channel in use on the host controller. The figure above demonstrates how the host controller is responsible for ensuring that the device-specific layer on the host controller is presented with the same bus structure as a result of the sensor-based addressing mode. The result of this is that, with the exception of the network discovery process, all high level processes remain unchanged between the addressing modes.

7.3.4.2 The Network Discovery Process

The network discovery process under sensor-based addressing is very similar to its counterpart under device-based addressing. The process is essentially the same, only the implementation on the sensor interface boards is changed to accommodate the new addressing mode.

In device-based addressing, each sensor interface board responds to the PING request and is assigned an address by the host controller. In sensor-based addressing, the sensor interface board needs to maintain a list of the sensors attached to it and implement each one as a separate device.

The sensor interface board does this by storing a special record for each sensor implemented on the device. These records are stored in a sensor device table on each sensor interface board and each record can be assigned a valid address during the network discovery. When the sensor interface board receives a message, it checks the address against all the entries in the sensor device table and will only pass the message onwards if it doesn't match any of the entries in the table.

When the device starts up, or a soft-reset is performed, all the addresses in the sensor device table are set to the un-initialized address. This is the only case where the sensor device table is permitted to have duplicate addresses. The sensor device table is checked sequentially, which results in the first un-initialized sensor shadowing the other ones on the device. This characteristic is analogous to the method in which un-initialized sensor interface boards shadow other boards under device-based addressing.

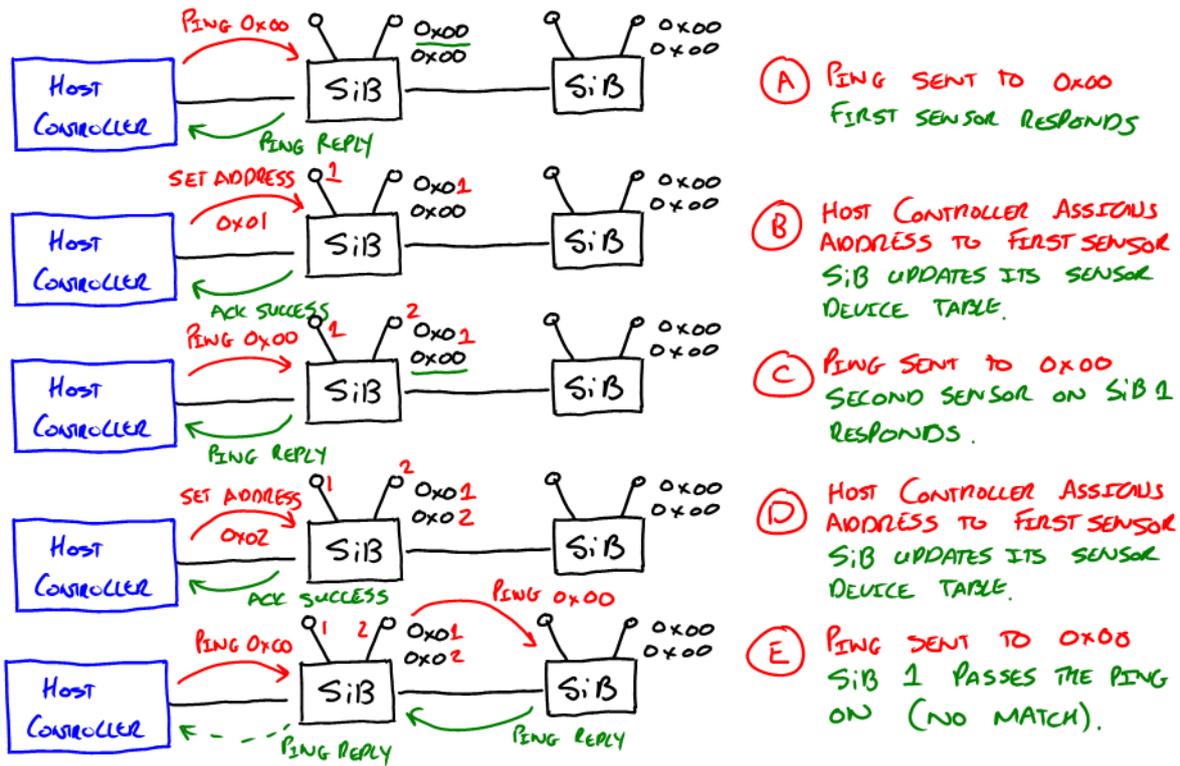


Figure 7-18: Illustration of a Network Discovery using Sensor-Based Addressing

The figure above illustrates this process on a small network involving two sensor interface boards, each with two sensors attached. The process followed by the host controller is essentially the same; it sends a PING request to the un-initialized address and waits for a response. The first un-initialized device responds and the host device issues a SET ADDRESS command to assign it an address. This process is repeated until the host does not receive a response to the PING packet.

The sensor interface boards do not respond to the PING message directly. When a sensor interface board receives a PING message, it checks the sensor device table on the device for a matching address. If a matching address is found, then the sensor interface board responds on behalf of the sensor and uses the information in the table to fill the payload packets of the PING reply.

If a PING message is received and an address match is not found in the sensor device table, the message is passed on to the next device.

7.3.4.3 Bus Resets under Sensor-Based Addressing

Bus resets are performed in the same fashion under both sensor-based and device-based addressing. Even under sensor-based addressing, the bus reset command is handled by the sensor interface board and not the individual sensors on the bus. The reason for this is that the concept of a bus reset does not apply directly to a sensor.

A bus reset is intended to reset all the layers of the communication protocol on a device. Under sensor-based addressing, the sensor interface board implements the lower two layers of the communication protocol. The sensor itself represents only the device-specific layer of the protocol and therefore if a sensor were to receive a bus reset, it could not perform a proper reset. For this reason, the bus reset is detected by the sensor interface board and the board itself performs the reset action.

The sensor interface board does this by restoring the sensor device table to its start-up value. This has the dual effect of resetting all addresses and clearing any previously acquired sensor data. The sensor interface board also performs all the other tasks involved in a bus reset, such as clearing out buffers and resetting state variables.

One consequence of this change is that bus reset messages can only be sent to the broadcast address when using sensor-based addressing as sensor interface boards are no longer addressable. This removes the possibility of individually resetting each address on the network, even though this is not a feasible solution due to the 16-bit address range.

In an attempt to improve the success rate of the bus reset, certain retransmission techniques can be implemented by the sensor interface boards when dealing with bus reset messages. As the actual sensor interface board is transparent to the network, it can perform retransmission functions as long as it does not impact on the functioning of the bus. There has been limited success in implementing these on the network so far.

7.3.4.4 Accessing sensor data under Sensor-Based Addressing

Sensor-Based Addressing was designed to streamline the process of accessing data on the network. As each sensor becomes addressable on the network, data is requested from each sensor individually. Each sensor interface board reserves space in its sensor device table for values obtained from the physical sensor. When a sensor receives an instruction to transfer any pending readings, it packs the readings from its entry in the table into the payload of the response message.

Unlike in device-based addressing, only one message packet is returned for each reading transfer request. Each sensor is allowed to transfer four bytes worth of data, divided up in any manner the device sees fit. For example, a sensor may send its value and the voltage on the supply in the message packets.

7.3.4.5 Time Announcements on the Bus

Another improvement added in sensor-based addressing is the regular time broadcasts on the network. These are implemented using a special message marked by the TIME ANNOUNCE type and containing the type of announcement and relevant values in the payload. The purpose of these announcements is to allow the sensor interface boards to become aware of the relative time-base used on the system. This is not intended to allow coordination on the bus but rather to allow processes running on sensor interface boards to include temporal aspects in their calculations.

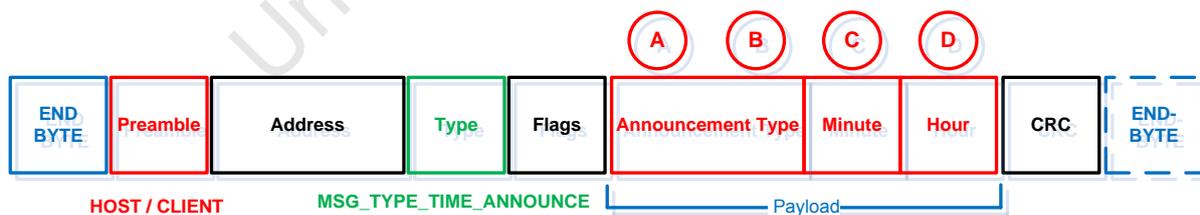


Figure 7-19: Structure of a TIME ANNOUNCE message

7.3.4.6 Virtual Sensors

The fact that each sensor interface board can support an unlimited number of sensors gives rise to the ability to support sensors that are not directly tied to a hardware device. This type of sensor appears to the host controller as a physical sensor but is implemented entirely in software on a sensor interface board. This provides a mechanism to provide information such as averages, maximum values and simplifies the implementation of compound readings such as running time. In order to differentiate these from standard sensors on the bus, these sensors are referred to as virtual sensors.

Virtual sensors are essentially just entries in the sensor device table residing on a sensor interface board. These entries are identical in structure to normal sensors and cannot be directly detected by the host controller. Further querying of the sensor device table may inform the host controller as to the nature of the sensor, but this data should never be used by the host controller to directly identify the nature of the sensor attached.

One of the most important characteristics of virtual sensors is the fact that they can be updated at any time during the operation of the sensor interface board. Unlike traditional sensors on the bus, which can only acquire readings during the strictly controlled reading acquisition process, virtual sensors can be updated at any time so that they represent accurate information when read.

Virtual sensors greatly simplify the implementation of features such as running time, which was implemented rather poorly under device-based addressing. The sensor interface board can simply update the values in the sensor device table every second that the target device is running and ignore the direct requests during the sensor acquisition process. The transfer of the reading data will be taken care of automatically as it would with any other sensor.

Another interesting option available is the ability to store the minimum and maximum values of a physical sensor. This is implemented on the sensor interface board by checking any newly acquired readings against the value stored by the virtual sensor and updating the stored record accordingly. In order to allow the network to provide averages, the sensor interface boards can respond to the time-based host controller broadcasts. This can be used to provide daily and weekly averages, maximum and minimum values during these periods and the ability to count discrete events over a set period.

7.4 The Reading Acquisition Process

The physical act of acquiring sensor data is the responsibility of the sensor interface board. The reading acquisition process is responsible for orchestrating and synchronizing the act of taking readings on the network. Acquiring readings on the network involves obtaining readings from a number of different sensors each with different characteristics. Care needs to be taken to ensure that all the readings are taken at approximately the same time and that the power limitations on the network are not exceeded.

There are two complications surrounding the process of acquiring readings on the network. The first is related to the time taken to physically acquire the readings. The time taken to acquire a reading can vary from virtually instantaneous (in the case of digital points) to 800ms for the 1-Wire[®] temperature sensors. The second problem relates to the amount of power drawn when sensors are powered as the increased current causes higher voltage drops over the cables. The network therefore cannot power all the sensors simultaneously without running the risk of exceeding the power requirements of the bus.

Each type of sensor imposes a slightly different time requirement. Temperature sensors require a total of 800ms to perform a complete sensor acquisition. It is important to note that there are times during this 800ms in which the device is idle and waiting for the sensor to return its value. During this time the sensor interface board can perform other tasks such as message passing and bus activity. In order to streamline the reading acquisition process, each sensor's timing requirements are broken down into interruptible and non-interruptible sections.

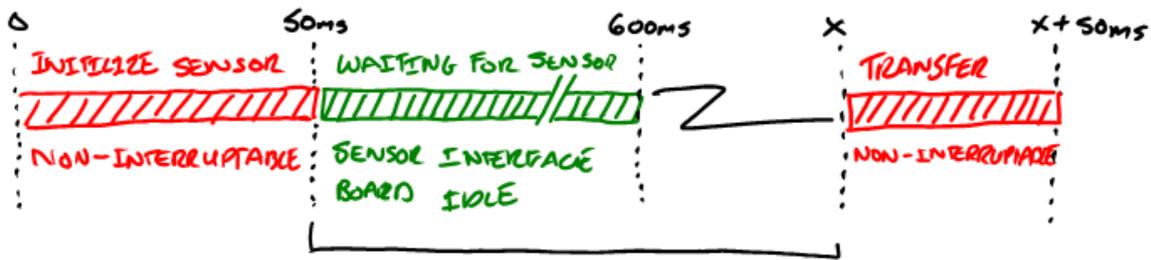


Figure 7-20: Timing diagram for a temperature Sensor

Temperature sensors, for example, require 20ms initially while the sensor interface board configures and starts the sensor. During this time, the sensor interface board cannot be interrupted as strict timing is required to implement the 1-Wire[®] protocol. The device then waits 600ms while the 1-Wire[®] temperature sensor stabilizes and takes the temperature reading. During this period, the sensor interface board can perform other tasks. After 600ms the data is ready to be read but it can be read at any time from there on. It takes a few milliseconds of uninterruptible time to transfer the readings from the 1-Wire[®] device back to the sensor interface board.

Pressure sensors output a constant voltage proportional to the pressure under test. This value is sampled by an ADC on the sensor interface board's microprocessor. This introduces only a few milliseconds of delay but is repeated as many times as possible over the period of 0.02 seconds to eliminate any noise resulting from 50 Hz electrical noise. Interestingly, this is one of the few factors that need to be changed in order for our product to operate overseas. The current drawn by the pressure sensors represents a significant factor of the total power used the sensor interface board and therefore is only turned on when readings are taken.

There are other sensors, such as running time, which are updated constantly and therefore do not require any specific time to acquire the reading. These sensors however still need to ensure that they take their current value at the same time as the other values in the system. These sensors generally do not impose specific power requirements on the network as they are either internal or constantly running.

A different process for acquiring readings was implemented for each addressing system. Device-Based Addressing implemented a very simple system of acquiring readings which had a number of shortcomings relating to speed, efficiency and scalability. The improvements made in the sensor-based addressing system allowed a more comprehensive and streamlined process to be developed. The two methods are discussed in detail below.

7.4.1 Reading Acquisition under Device-Based Addressing

The reading acquisition process under device-based addressing is a very straight-forward and linear process. A host controller can only request readings from the network once it has performed a successful network discovery and it can therefore be assumed that the host controller is aware of every valid address on the network.

The host controller can implement a reading acquisition in one of two ways. It can either send an ACQUIRE_READINGS broadcast message to all the devices on the bus, or it can individually poll each device for readings. The effect of either system is the same, as each device will complete the entire acquisition process before the next device can start acquiring readings. Therefore each device must suffer the full time delay for each sensor attached (i.e. the full 800ms for a 1-Wire[®] temperature sensor).

The nature of device-based addressing lends itself to certain improvements in this process. As each sensor interface board may host multiple sensors, many opportunities exist to streamline the

reading acquisition process in the firmware of the sensor interface board itself. For example, a sensor interface board may execute 4 separate 1-Wire[®] temperature conversions on 4 different devices simultaneously, effectively reducing the time taken by a factor of four.

Unfortunately, these gains are only achievable on a board-by-board basis and each board will only start acquiring readings once the previous board has completed in its entirety. On a long network, this delay could increase to a number of seconds which may become cause time skew in the acquired readings. The improvements also require specialized firmware for each sensor installed such that each board configuration must be independently optimized.

The reading acquisition process is a single step under device-based addressing. The host controller simply issues the command and the device responds with the appropriate readings when ready. The process of storing and transferring readings was described above in section 7.3.3.4. If the host controller chooses to use the broadcast address to acquire readings, it must carefully examine the response messages and the ACKs to determine if a successful acquisition has occurred. A major point of concern in this system is that a single lost packet can cause an otherwise successful acquisition to appear unsuccessful. This is especially problematic as ACK packets do not have any retransmission mechanisms or error checking.

The host controller can alleviate this problem somewhat by polling each device on the bus individually. This requires the host controller to consult the device table generated during the network discovery and to ping each device in the order in which they were detected. Using this system, an error can easily be detected and its precise location can be determined. It is important to note that the devices are polled in the order in which they appear so that the process can still acquire some readings even if a device on the network stops responding.

7.4.2 Reading Acquisition under Sensor-Based Addressing

Sensor-based addressing allowed a number of improvements to be implemented in order to streamline the reading acquisition process. Many of these improvements were made possible by the new functionality inherent to the sensor-based addressing system but a number of improvements originated from observations of the functioning of reading acquisition process under device-based addressing.

The improved reading acquisition process was designed to reduce the time taken to acquire readings through a four stage process in which sensor operations can essentially be pipelined by allowing multiple sensors to perform various tasks simultaneously. The process was designed to be as flexible as possible and allow a wide range of future sensors to be developed.

The four-stage process relies on the fact that there are periods during the time taken to acquire a sensor reading when the physical hardware can perform other tasks such as message passing. The process takes this into account by providing three different signals to allow devices to perform various configuration and initialization tasks while the previous devices are waiting for their sensors to either become ready or to actually take the reading.

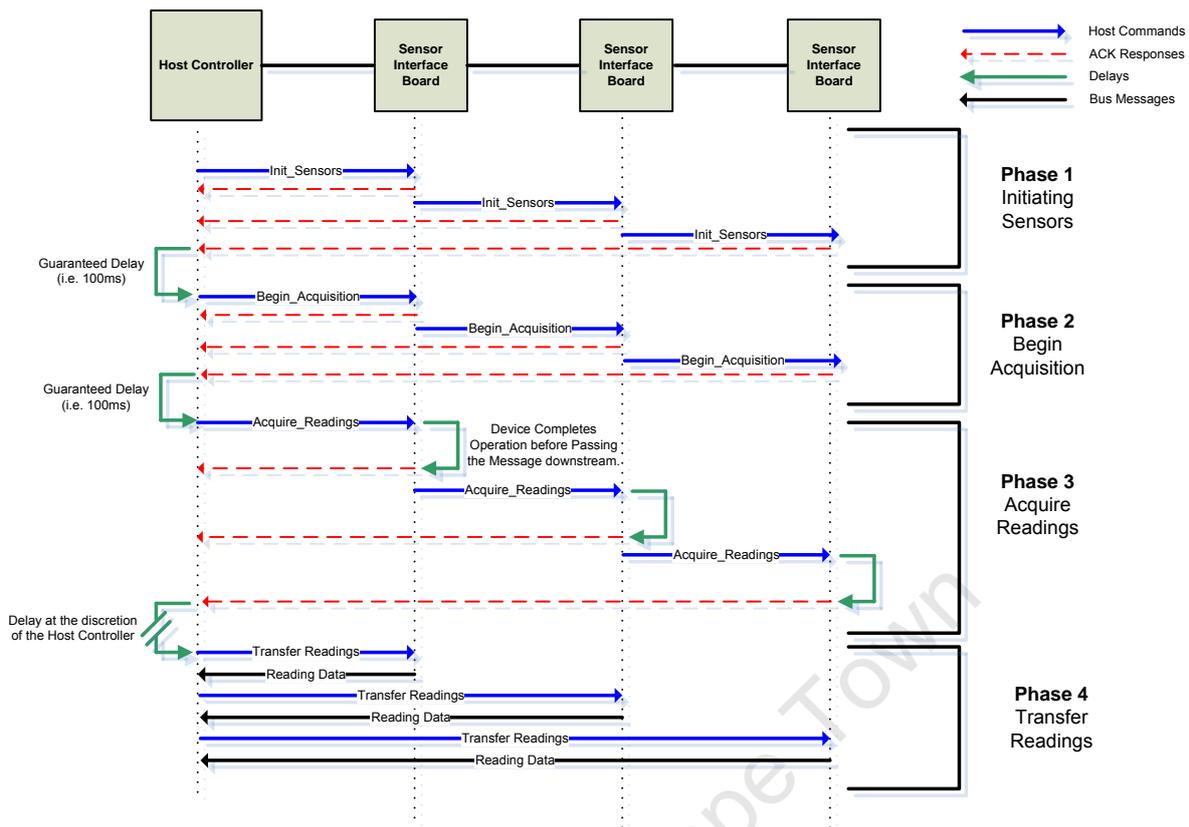


Figure 7-21: Illustration of the 4-stage Reading Acquisition Process

The above figure demonstrates the four-phase reading acquisition process. Each phase of the process is separated by a guaranteed delay which is known to all devices on the network. This allows all devices on the bus to be guaranteed of a minimum amount of time between phases. As the message is passed along the bus from the host controller, it is the last device on the network that experiences the shortest delay equal to the guaranteed delay specified by the host controller. The first board will therefore experience the longest delay, the length of which is not determinable.

The first phase is intended to give the devices time to initialize any attached sensors. Note that under sensor-based addressing the sensor is the target of the address and therefore multiple sensors can actually be initialized without the transmission delay if they are all attached the same sensor interface board. Operations performed during this step usually include powering sensors and issuing initialization commands to external peripherals. The guaranteed delay is used to allow for factors such as the time taken for a power supply to settle.

If a device needs to implement a routine which is time-critical and cannot be interrupted by bus messages, it may process the message before transmitting it to the next device. This increases the delay in the system and therefore must be kept to a minimum. This is often used when implementing protocols in software where pin timing becomes critical. It is important to note that the system was designed to support such actions for legacy reasons and it is not encouraged as it does lower system performance.

The second phase is executed in an identical fashion to the first phase. It is essentially a second opportunity to perform initialization and configuration tasks. Devices can choose not to implement any activity during this phase and can even use it an additional delay for the first phase (as may be the case where the sensor has unusually long settling times). Another guaranteed delay is provided

at the end of this phase which is significantly longer than the one after the first phase. It is intended that devices acquire readings from non-blocking sensors during this phase.

The acquire readings phase differs from the first two phases. Sensors are intended to perform any blocking activity during this phase and obtain readings. The phase is also terminated by a guaranteed delay but adds the requirement that all sensors must have some data ready by the end of the delay.

The final stage of the process is the act of transferring readings back to the host controller. This is performed in any manner the host controller chooses to allow the host controller to read only certain sensors. One significant advantage of sensor-based addressing is that the data returned from each sensor during this phase is a single message, unlike the multiple messages sent under device-based addressing. As the transferring is done after all devices have successfully acquired readings, there is no time restraint and therefore no possibility of time skew. For this reason, the host controller can poll each device individually and therefore gain the benefit of all the error handling. This increases both the efficiency and robustness of the entire process.

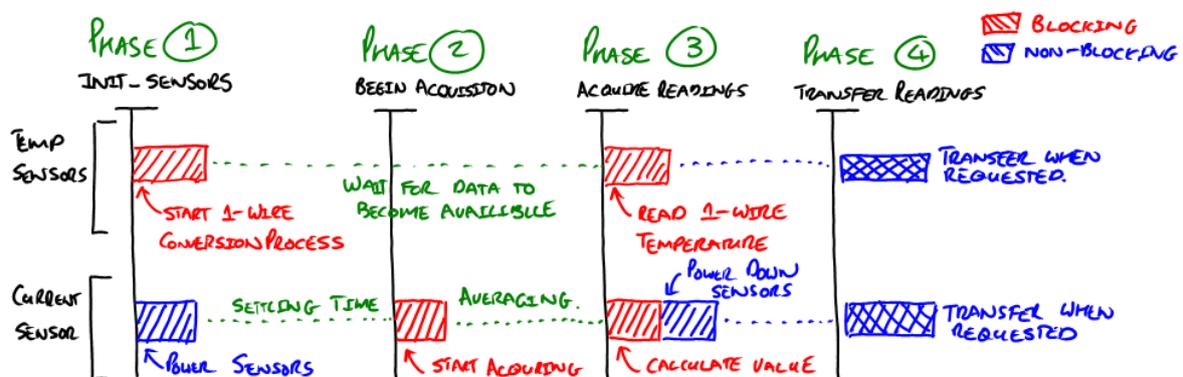


Figure 7-22: Acquiring Current and Temperature readings with the 4-Phase Acquisition Process

The figure above illustrates how current and temperature readings can be acquired with the four phase reading acquisition process. These two sensors were chosen to highlight the versatility of the process and its ability to handle a wide range of different requirements. The timing and logical components of the acquisition process for the 1-Wire[®] temperature sensor can be found in the previous section (refer to Figure 7-20).

In the case of the temperature sensor, the first phase is used to issue commands to the 1-Wire[®] device instructing it to begin the temperature conversion process. This process can take up to 800ms and therefore the host controller performs no action during the second phase of the process. At the end of the guaranteed delay after the second phase, the temperature data is read from the 1-Wire[®] device and is prepared for transmission to the host controller. As with the first stage, this is a blocking section as the device needs to implement very accurate timing to detect pin changes.

The second sensor displayed in the figure is a current sensor implemented using an ADC on the microcontroller. This process was chosen as it is very similar to the process used to acquire values from the pressure sensors and also demonstrates a number of features of the process. Unlike the temperature sensor, the interface circuitry for the current sensor needs to be powered when required and therefore uses the first phase of the process to allow the power supply to settle.

In this configuration, the current sensor uses the ADC on the microcontroller to continually take readings and average them over 20ms (in countries using 50 Hz electrical supply). This process can be carefully written to allow interrupts and therefore it can initialize the process at the start of the second phase and proceed to acquire readings up until the start of the third phase of the process.

Once the device receives the message indicating the third phase, it can then perform a calculation if necessary (such as calculating the RMS value or factoring in the supply voltage) and then proceeds to power down the current interface circuitry.

In general, any form of sensor can be catered for using this system as long as it adheres to the timing requirements in each phase and ensures that some data is available at the end of the third phase. In the case where the sensor could not obtain a value it must still return a value. As the host controller does not interpret the value, the reading acquisition process will still succeed and the value should be uploaded to the server. It is therefore up to the design of the sensor data and the diagnostic process to indicate an unsuccessful reading acquisition.

If the host controller receives a NACK from a device during the reading acquisition process, then the reading acquisition process will fail and all data obtained during the process will be lost. The data will therefore not be uploaded and the diagnostic server will only be informed that a process was attempted and failed. Depending on the configuration of the host controller firmware, it may attempt to retry the reading acquisition process immediately or wait until the next reading is required.

7.5 Network Simulation

Three different tools were developed to assist in designing and testing the network, the sensor interface boards and the host controller. Each tool was developed to either test or simulate a specific part of the system and was altered along with the underlying technology. The three tools together allow any part of the system to be isolated and tested under strict and controllable circumstances.

The three tools used are the Protocol Control Panel, the Network Simulator, and the Upload Simulator. Each one is discussed in more detail below.

7.5.1 The Protocol Control Panel

The protocol control panel was the first tool developed specifically for use in the system. It is a Windows application that utilizes a standard RS-232 connection to a specialized sensor interface board in order to perform the functions of a host controller on the network.

The Protocol Control Panel is capable of issuing commands and receiving both bus messages and ACK packets from the network. It can also be used to perform higher-level function such as a network discovery and acquiring readings. This tool was critical in the design and testing of the sensor interface boards, the communications protocol and new sensors for the bus.

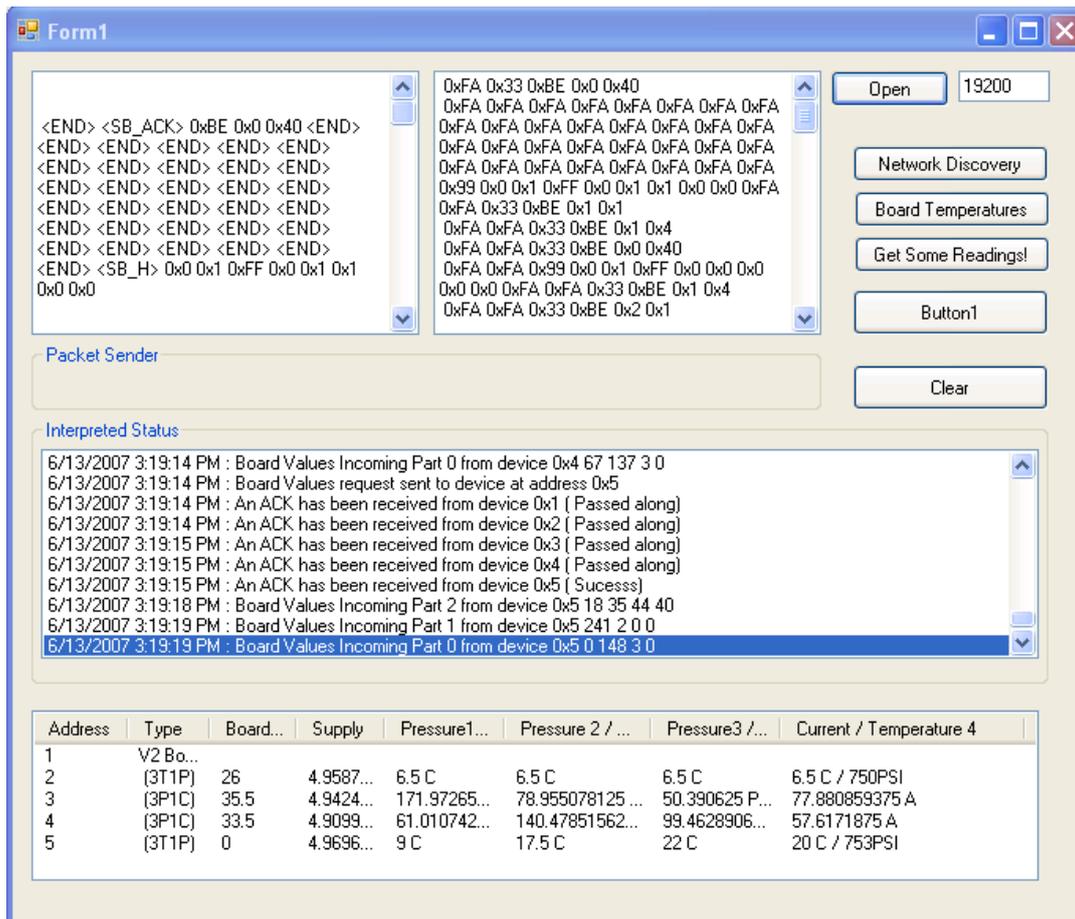


Figure 7-23: Screenshot of the Protocol Upload Panel

The above figure is a screenshot of the Windows application. The application itself is divided into three major sections. The top two text boxes display the raw data being received by the application. The box on the right displays the bytes being received and the box on the left shows the results of the message interpreter as it detects various special characters in the message such as Start-Bytes and End-Bytes. This top section represents the physical layer of the communications protocol.

There is also an option to send messages using the frame and button directly underneath these two boxes. This allows messages to be sent at the lowest possible level and at any time, regardless of the current process being performed on the network.

The frame labelled 'Interpreted Status' displays the message in a more easily understandable format. It is the result of the message being interpreted by the simulated network layer and can be used to determine how the host controller would interpret the message. For completeness sake, both outgoing and incoming messages are displayed in this window.

The interpreted status also displays received ACK messages in detail. This is particularly useful when debugging the communications protocol as the precise location of the message can be determined from the ACK messages received.

The hardware required by the protocol control panel is simply a modified sensor interface board with an RS-232 connection instead of a RS-485 connection. This requires only minor modification to the board as RS-232 is a full-duplex communication protocol, whereas RS-485 is only half-duplex. This removes the need for the direction control lines (which can simply be ignored) and allows the RS-232 line driver to connect to the microcontroller in the same fashion as the RS-485 line driver. The original board design, which had been designed to only accommodate a direct-drop bus, was set

up in this fashion and was often used to allow the protocol control panel to function as it included a RS-232 line driver on the board.

Most Sensor Interface Boards have jumpers and a connection point to allow it to function as a special node for the protocol control panel. An external RS-232 level shifter board is required to jump the signals and the jumpers and connectors are often not assembled on the boards.

7.5.2 The Network Simulator

The network simulator functions in the opposite direction to the protocol control panel by allowing networks of any size to be simulated in software on a desktop computer and then accessed and controlled via an RS-232 connection. The network simulator provided two important functions; firstly it provided a means of implementing and testing network concepts and comparing different processes on the bus. The second function it provides is the ability to test a host controller with any sized network and any conditions.

The network simulator was written as an object-oriented Windows application. Each sensor interface board was represented by a class stored in a double-linked list in order to simulate the daisy-chain nature of the protocol. Data channels were simulated using functions and events of the classes and actual link characteristics, such as random noise and propagation delay, are simulated in these functions.

Sensors were also implemented as classes contained within the sensor interface board class. Two different versions of the network simulator exist to cope with the two addressing modes present on the bus but due to the object-oriented nature of the bus, this change is relatively minor. Messages and Bus ACKs are also implemented as classes to simplify the action of passing messages from device to device. The address field in both classes has a conditional size declaration depending on the chosen addressing system.

The visual interface to the Network Simulator is almost identical to that of the Protocol Upload panel as it provides virtually the same service. Additional information is given in the lower pane regarding the boards present on the network and allows the adjusting of parameters on almost all objects.

The network simulator was critical in the design of the 4-stage reading acquisition process as it is capable of emulating the delays resulting from sensor acquisitions and allowed the behaviour of the system to be predicted without the need to test it with a very large number of devices.

One aspect that is not modelled by the simulator is the power requirement of each board on the network. In a real network, there are voltage drops across long cables and the amount of current each device draws can have a significant impact on the functioning of the bus. This feature may be added in the future if power requirements become a significant factor in the planning of an installation.

7.5.3 The Upload Simulator (aka Protocol Control Panel Featuring Upload)

The third tool developed to test the complete system was an upload simulator. Implemented as an addition to the Protocol Control Panel, it allowed different upload mechanisms to be tested on site through any available communication medium. The upload simulator represented a complete software simulation a host controller.

The upload simulator was first used as a proof-of-concept to demonstrate the end-to-end functionality of the system. At the time it was developed, the host controller hardware was not finalized and it was unclear whether GSM would be applicable to the system. A small network of sensor interface boards was connected to a computer running the upload simulator. A wireless internet link was added to allow basic upload methodologies to be tested. This wireless link was later replaced with a serial GSM unit to add credibility to the proof-of-concept model.

The upload simulator became useful again during the development of the firmware on the host controller as it allowed different upload mechanisms to be tested quickly and without the myriad of hardware complications arising from implementing it directly on the host controller hardware. These factors include the size of the buffers, the physical control of the unit, the difficulty obtaining state and error information, the multiplexing of data and commands and the inability to test the system in low-signal conditions.

8 Uploading Methodologies

8.1 Introduction

One of the primary tasks of the Host Controller is to periodically upload data to a central server for processing. The majority of the diagnostics applied to the data is done on a secure server which is far removed from the physical hardware. The advantage of this approach is that it limits the ability of a competitor to reproduce the system from the hardware alone.

The Host Controller contains an embedded GSM unit capable of connecting and transferring data using GPRS. One of the criteria involved in choosing an appropriate unit was that it supports at least a single TCP data connection. Many other units also provide higher level services like email or FTP either through embedded applications or specific plug-ins.

The task of establishing the connection falls to the hardware of the GSM unit (and in some instances the high-level protocol implementation as well) but the task of exchanging data using these methods involves a number of other factors such as complexity, length and speed. The importance of these factors is determined by the nature of their application. A discussion of the factors relative to this particular implementation is given below.

8.2 Requirements of the Uploading System

The upload component of the system must meet certain base requirements in order to be effective. These requirements are not necessarily stringent and offer enough flexibility for a wide range of different techniques to be applied successfully. The requirements listed below include those made on the physical hardware, on the quality of the connection and on the devices on the receiving end.

As the system has not been designed to implement any control functionality, the majority of the requirements involve transferring data from the monitoring equipment to the host. When not explicitly stated, it can be assumed that data transfer in the other direction is not critical.

8.2.1 Processing Latency

Processing latency in this context refers to the length of time between the physical acquisition of sensor data and the point at which it becomes available for the diagnostics system to process. This time may correspond with the time taken to physically exchange data with a host but it may be substantially longer in cases where data is then relayed or needs to be periodically checked.

As the system does not perform any control functions, the actual duration of the processing latency is not as important as its consistency and characterization. The quantity of readings required per day is substantially smaller than the maximum number that can be acquired and therefore large gaps between readings can be tolerated.

An interesting facet of this requirement regards the method by which the devices establish a time-stamp on a particular set of readings. There are a number of real-time clocks embedded in the system to allow for intervals and delays of a predictable and useful nature. The primary purpose of these real-time clocks is to provide delays of a specific number of seconds rather than accurately track the current time. The reason behind this decision is to prevent the system installer having to set the time on the equipment which would increase the installation complexity. Instead the system relies on the upload system to apply a time-stamp to the data upon receipt.

The possible pitfalls of this manner of time-stamping data arise when there is sufficient delay between the acquisition of the reading and the point at which the time-stamp was applied. For some technologies, such as SMS, this is done as the message is received by the network but other technologies, like email may not necessarily apply the time-stamp in a timely manner.

A final issue which has arisen from actual tests is that the host may not necessarily reside in the same time-zone as the device sending the data. For this reason, it is important to know the location of both the installation and the physical hardware on the other end of the data exchange. Some technologies may further complicate the problem by employing virtual servers which may cause the time-stamp to vary in an entirely unpredictable way.

8.2.2 Supported Upload Frequency

The diagnostics system requires only a few valid readings per day. A valid reading can either be obtained through careful selection on the Host Controller or by uploading a far higher number of readings to the diagnostics server and selectively choosing readings in the diagnostic process. A common method (which can be implemented on either the server or the physical hardware) is to monitor running time to ensure that the device is in a stable state.

An additional benefit to a higher number of readings is that it allows more careful monitoring for alert conditions, such as a spike in oil temperature or a failure of a compressor or pump. In these cases, a faster response is desirable.

The chosen upload method must be able to support transferring data at least five times more frequently than required by the diagnostics system. The factors that may impede this include the minimum time between connections (i.e. latency on the server-side), minimum times required for retrying a failed connection and, to a lesser extent, the actual time taken to transmit that data.

8.2.3 Reliability and Consistency

The reliability of the data exchange is crucially important to the system performance. The benefit of accurate sensors is effectively eliminated if it is transmitted via an unreliable medium. Every effort has been taken in each component of the system to ensure the integrity of the data arriving at the Host Controller. It is the responsibility of the upload process to ensure that the data arrives at the server correctly.

The worst possible outcome of an upload operation is that the invalid or incorrect data is passed to the diagnostics server for processing. It would be preferable for the data transfer to fail completely, than to upload incorrect or incomplete data. For this reason, the upload method must at least support error checking. Error correction would improve throughput but should not improve the accuracy or reliability of the data transfer.

System reliability will often come at a cost of either speed or increased overhead in both transmission and computation. For this reason, error detection is of paramount importance but error correction is not a requirement.

8.2.4 Implementation, Maintenance and Running Costs

The physical hardware on the Host Controller represents only half of the system when viewed in its entirety. The cost of the physical hardware is predictable and constant (only decreasing as a result of price breaks and minimum quantity orders) but the cost on the receiving end of the data exchange is more complicated.

There are costs involved with the implementation of both the server and any technologies on that. Technologies that can run on standard hosted web servers would offer the advantage of allowing a single server to handle both the responsibility of acquiring the data, processing it and hosting a website to allow external access.

Some technologies may require additional requirements of the server such as a fixed IP address or additional open ports. These requirements often cause additional security concerns which may increase the cost of hosting the server. Ultimately, a dedicated physical server would provide the

most flexibility but there are increasing benefits to utilizing a hosted service where tasks such as backup, servicing and security are included in the cost.

Although the costs of the server are aggregated over the entire system, the cost of each additional unit must also be considered.

8.2.5 Scalability

The ability of the chosen uploading system to handle increasing number of devices is also an important consideration in the system. The system must also be able to increase the number of connected devices without requiring any updates or interactions on existing systems.

The scalability does not necessarily have to be transparent. Partitioning the system or the physical assignment of devices to servers are all valid options as long as they do not affect other units in the system. The requirement is only that the technology be expandable to support any number of additional units.

8.3 Comparison of Uploading Technologies

A number of different protocols were considered when designing the upload mechanism. The majority of these rely on the TCP services provided by the GSM unit. The hardware on the Host Controller was designed to allow a wide variety of protocols to be implemented and was not a factor in deciding which protocol to use. A description of the various upload methods considered is given below.

8.3.1 Uploading using and SMTP Mail Server

Email is a very convenient method of transferring information as it a very widely supported medium. Email servers also have a relatively simple protocol for sending email and many GSM units also have plug-ins which assists in sending emails from the device.

The process for uploading data from the host controller would involve the composition of an email containing the relevant data which would be sent to as predetermined email address. A standard SMTP mail server would be used to send the email and a client program would be written to periodically check the receiving mailbox and interpret the messages.

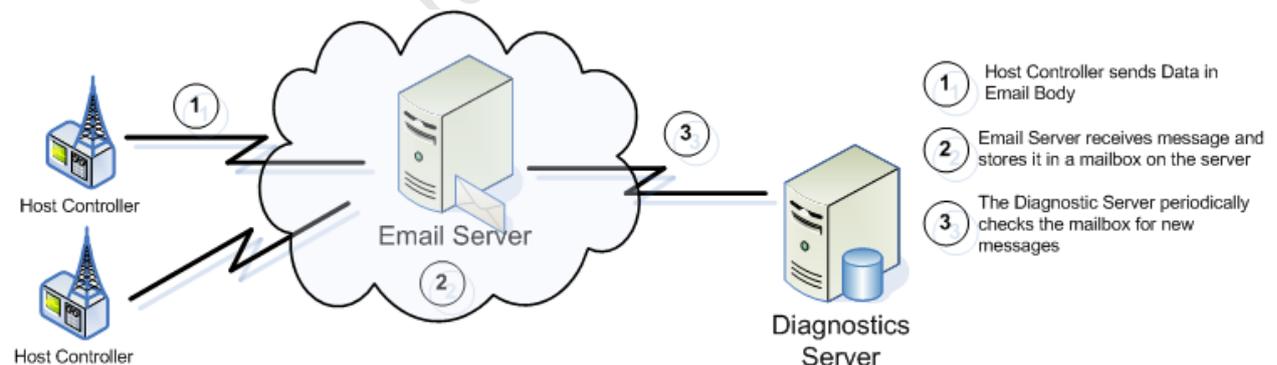


Figure 8-1: Diagrammatic view of uploading data using a centralized email server (SMTP Server)

The figure above illustrates the process of uploading using an email server. The cloud in the figure represents the internet. The devices on either side (the host controllers and the diagnostics server) are physical devices which simply attach to the internet and do not require any special connection. The email server requires a fixed IP address as it acts as a central point for the other two devices. This is an asynchronous process as the mail server will store email messages in a mailbox until the diagnostics server downloads them.

The most desirable attribute of this method is that it does not require a dedicated server as it can use existing SMTP servers. The diagnostics process can be run on any computer with an internet connection as the mail server acts as a known entity for both devices. A hosted website would still be required in order to host the client side of the system and it would almost certainly include a mailbox capable of performing the services required.

There are a number of problems with implementing the system using this technique. The first problem is one of reliability. Email traffic on the internet is not necessarily a priority and it can incur substantial and unpredictable delays. Data may not arrive in the correct order and there is no simple mechanism for ensuring that data reaches the diagnostics server. There are also increasing problems with SPAM filters which often employ a list of safe and unsafe domains, emails originating from certain places (or certain networks) may be blocked by SPAM filters and never reach the destination.

The overhead to sending an email is relatively low, only a few initialization commands need to be passed between the SMTP server and the device. The process is interactive, forcing the client to wait for the server to respond before continuing to send data. The email system is easily scalable in theory, although increased traffic would probably require more complicated transactions such as mail server authentication when sending.

Receiving data on the Host Controller using email technology is not feasible when using a GPRS connection. Mail is usually retrieved using the Post Office Protocol (POP), which can easily be implemented due to the simplicity of the protocol [40]. Unfortunately, as there is no direct way of controlling who can send to an email address, factors such as spam, make this solution impossible to implement.

8.3.2 Uploading to an FTP server

The FTP protocol is a widely used protocol for uploading and downloading files from a remote computer. It is well supported and is often provided as a standard feature on any hosted web server. Many GSM units also support FTP access through plug-ins which serves to simplify the process further.

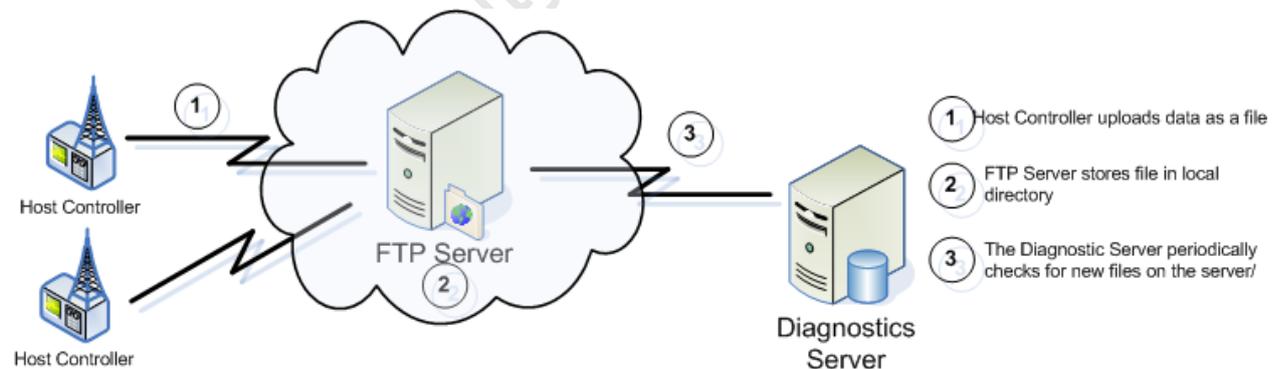


Figure 8-2: Diagrammatic view of uploading data using a FTP server

Uploading data using an FTP requires access to an FTP server which provides the common point between the diagnostic software and the device as is visible in the above illustration. The FTP server can be hosted on a standard web server as long as it has a valid resolvable domain. Devices can then connect to the server and upload their data as files, which can later be retrieved by the diagnostics software.

The FTP protocol itself suffers from two problems which restrict its usefulness. It is a verbose protocol and therefore introduces unnecessary overhead. The second problem is that it requires two separate TCP connections, one to transfer data and one for commands. This makes implementing

the protocol from a microcontroller very difficult as it needs to multiplex two simultaneous connections.

The control connection is primarily responsible for initiating transfers and often remains idle whilst data is transferred over the data connection. When transferring a substantial amount of information (relative to the connection speed), the control connection may remain idle for long periods. This may result in the connection timing out and the data connection prematurely severed.

One further complication with this method involves the coordination between the devices and the diagnostics server. If both devices attempt to access a file at the same time, the results may be unpredictable. The scalability of the system is also a factor, especially in terms of coordination.

8.3.3 Uploading to a Proprietary TCP Server

All the existing protocols considered have been developed for purposes somewhat different from those required by this system. As the hardware provides a standard TCP connection, it is possible to develop a proprietary TCP server and implement a custom protocol. This protocol can be designed to provide the optimal combination of features. The protocol can also be made extremely simple to implement on the client side, simplifying the development of the host controller firmware. This can be used to justify the development time required for the TCP server.

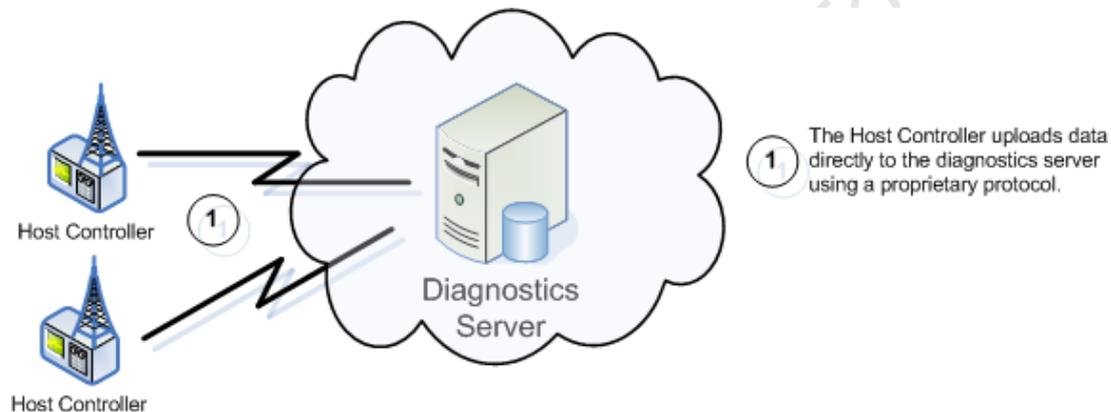


Figure 8-3: Diagrammatic view of uploading using a proprietary TCP server

As the above figure illustrates, the direct benefit of developing a custom TCP server and communication protocol is to radically simplify the upload procedure. As the host controller uploads data directly to the diagnostics server, there are no coordination concerns or problems with mislaid data. This process also allows the diagnostics server to interact with the reading acquisition process, yielding more control over the system.

The cloud in the above figure represents the internet and the diagnostics server must have a resolvable DNS address in order for the Host Controllers to connect and relay data. The ramifications of this are two-fold. Firstly, the server connected to the internet must now allow a non-standard application to run, which drives up the cost of the server as most hosted companies will not permit this on standard hosting packages. The second problem is one of security as the entire diagnostics process now resides on a public server. Additional TCP ports must be opened to allow the application to run and it is possible for the system security to be breached allowing unauthorized access to the diagnostics routines.

One of the most appealing aspects of this approach is that the entire exchange of data between the Host Controller and the server can be tailored to provide maximum functionality at the lowest possible overhead. The protocol can also be expanded to later add features such as security and authentication. This flexibility comes at the cost of development time as a proprietary protocol

requires implementation on both ends of the communication exchange. Changes are therefore more difficult and costly to apply.

8.3.3.1 The TCP Server Application

The proprietary TCP server must be run on a server with a fixed IP address. As it does not use any of the standard web protocols (i.e. HTTP, FTP, POP etc), it requires the use of TCP ports which are usually blocked by firewalls for safety. The server component is currently a Microsoft C# application which runs in the background and listens on a single port for incoming connections. This port is configurable but must be known by both the server and the connecting clients.

When an incoming connection is detected on that port, the server then spawns a client on a separate thread to handle the new connection. The TCP connection is then bound to a different, random port as to allow the server to continue listening for other incoming connections. The advantage of this system is that only one additional TCP port needs to be left open (for the incoming connections) as the other ports used are only opened from the host and therefore will not be blocked by the firewall.

Each client thread implements a buffer which fills until a linefeed character is received as this marks the end of a valid command. At this point, the buffer content is then parsed and the message processed. An appropriate response is formulated and the buffer is cleared.

Due to the multi-threaded nature of the server component, care must be taken to prevent race conditions or deadlocks. As multiple clients may be uploading simultaneously, it is possible that the server component may attempt to access either the mapping files or the database simultaneously. For this reason, the protocol contains certain features to implement a crude form of congestion control.

The TCP server application proved to be difficult to maintain on the hosted server. The hosting company provided a spare port and allowed the application to run. Unfortunately, due to restrictions on the server, the application could not start when the operating system was started and had to be started manually. This became increasingly problematic and eventually led to the rewriting of the application as a Windows NT service, which is still ongoing.

8.3.3.2 The TCP Server Protocol

The protocol between the host controller and the proprietary server was designed to closely resemble the AT commands used to communicate between the processor and the GSM unit. This would allow the microprocessor to communicate directly with the TCP server using the same process used to configure and create the connection.

In order to achieve this, the commands are built to resemble AT commands and the server responses mirror those of the GSM unit. Therefore, the microprocessor can simply stream commands to the GSM unit and the TCP server without needing to distinguish between the two. In other words, the microcontroller does not need any additional firmware to be able to upload data as it appears as an extension of command list stored in the microprocessor. A detailed breakdown of how the microcontroller communicates with the GSM unit can be found in sections 6.2.3.5 and 8.3.4.2.

As GPRS is usually billed based on the quantity of data transmitted, the protocol should be kept efficiently compact as possible. A balance was struck between saving space and allowing the data logs to be easily interpreted. The implemented protocol is far from compact but does produce log files which are clear and easy to understand. The general structure of a client request is shown in the image below.



Figure 8-4: Structure of Client Requests to the Diagnostics TCP Server

The structure of the request message is very straight-forward. It consists of a four character command word followed by the data in a comma-delaminated format where applicable. Each command is terminated with a linefeed character. A valid message need only have a command but the server is expected to parse the message directly after receiving the linefeed character and ensure that message contains all the required information for the particular command.

Command Name	Data Parameters	Description
HELO	None	The HELO command simply tests the presence of a connection with the server. The server will acknowledge this request if it can handle a new incoming connection.
NAME	1 Parameter: [Device Name]	The NAME command is used to identify the device attempting to upload data. It is required before any data can be saved.
TIME	2 Parameters: [Date], [Time]	The TIME command allows a device to upload an optional time-stamp for the incoming data. This is used to upload stored values.
DATA	2 Parameters: [Name], [Diagnostic data]	The DATA command is used to upload a line of data to the diagnostic engine. Each segment has a name, to indentify the 'chunk' of data being sent. The actual format of the diagnostic data is determined by the uploading device and the diagnostic server.
SAVE	None	The SAVE command forces the TCP server to process and store the readings. The result of this command is used to determine whether or not the server accepted the incoming data.
QUIT	None	The QUIT command simply ensures that the connection is destroyed. No response is sent.

Table 8-1: Proprietary TCP Server Request Commands

The above table contains a list of all the valid commands used in the protocol. The current version of the protocol does not include any security and only rudimentary error checking. As TCP is a connection-oriented protocol which guarantees data delivery, the only error-checking is provided by the diagnostic process which validates the new data once the SAVE command is issued. Security remains an interesting issue with this method as it is open to abuse by simply monitoring all TCP traffic between the host controller and the server. Methods such as IP address tracking and client-server rolling codes were considered but were never implemented.

The server responses are designed to be compatible with the V24-V25 response codes returned by the GSM unit. The V24-V25 protocol can produce either verbose text-based responses or simply numerical codes to represent the outcome of an AT command. The TCP server is therefore capable of generating both sets of responses depending on the implementation required. These two outputs are shown in the figure below.



Figure 8-5: Verbose and Abbreviated TCP Server Responses

There are only three responses which the server issues in response to messages. These are summarized in the table below:

Response Name	Verbose Response	Abbreviated Response	Description
OK	[CR LF] OK [CR LF]	0 [CR LF]	Indicates success
ERROR	[CR LF] +TCP ERROR {num} [CR LF]	4 [CR LF]	Indicates that the operation failed.
BUSY	[CR LF] +BUSY [CR LF]	9 [CR LF]	Indicates that the server is busy

Table 8-2: Proprietary TCP Server Responses

The verbose output is useful when debugging or configuring a modem as it produces responses that are clear and easy to interpret. The verbose output is very difficult to interpret using a microcontroller due to the possible variations and string manipulation required. The abbreviated output method uses numerical codes (where applicable) to indicate the response and therefore lends itself to easy interpretation on the microprocessor.

8.3.3.3 Uploading Data to the Proprietary TCP Server

The procedure for uploading data to the server was designed to be as simple and straight-forward as possible. Once the GSM unit establishes a TCP connection with the target server, it enters the 'data' mode in which all bytes received on the UART are assumed to be data destined for the TCP connection. This transition is seamless as a multiplexed UART is used to provide both the command and data channels.

As the protocol acts in the same vein as the AT commands, this switch does not affect the means by which the microcontroller issues instructions. It can therefore communicate with the TCP server in the same way in which it issues AT commands. The only complication occurs when the connection is closed and the UART needs to be switched back to accepting commands. This is performed with an escape sequence which is issued after the QUIT command (which ensures that the server is no longer listening).

```
>> HELO
OK

>> NAME dev001
OK

>> DATA board1 ; 1,2,3,4,5
OK
>> DATA board2 ; 6, 7,8,9,10
OK

>> SAVE
OK

>> QUIT
<connection closed>
```

Figure 8-6: Example of Performing an Upload to the Proprietary TCP Server

Once the connection to the server has been established, the server waits for the client to initiate the communication. The server does not automatically announce its presence so that the client has ample time to prepare for the incoming data. Once the client is ready, it initiates the communication by issuing the HELO command.

When the server receives a HELO command, it can assume that a client is now connected. The server will respond with either a SUCCESS or a BUSY. The SUCCESS response indicates that the server is ready to accept new information and that the client can proceed at will. The BUSY response is a simple implementation of congestion control and informs the client that the server cannot handle the request at the current time. The client can either keep the connection open and re-issue the HELO command periodically, or disconnect and try again later.

The second stage of the protocol requires the client to identify itself to the host using the NAME command. The purpose of the name is to allow the diagnostic server to determine which physical device is uploading data. The given name is purposefully not checked at this point to prevent brute-force attacks on the system. The name is only evaluated when the SAVE command is issued later in the protocol.

The actual data is then passed to the server through the DATA command. This command requires that the data being passed has a valid reference name. This is used later to interpret the received data and break it down into relevant sensor information. There are a few important points regarding the use of the DATA command to pass data. As the data uploaded is mostly numerical, care must be taken to ensure that no special characters appear in the second parameter of the DATA command. These include carriage returns, line feeds, semi-colons and commas.

One solution is to convert the numeric value to its string equivalent and transmit the digits separately as ASCII characters. This solution works well as there is no limit to the length of each data line and the length does not need to be known prior to uploading.

The final stage in the upload process is to commit the data to the system using the SAVE command. This command forces the system to evaluate all the input received in the session and to attempt to submit it to the diagnostic server as a valid set of new readings. The SAVE process will fail if either the data is incomplete or if the supplied device name cannot be found. The exact reason for the failure is not reported to the client device to prevent brute-force attempts to determine valid device names.

Once the client has received a response from the server, it can either terminate the connection using the QUIT command or proceed to upload more data by issuing either the NAME or TIME command. The connection can therefore be considered persistent as it should not be closed by the server (unless the device stops responding and the session times out). This functionality was added to allow a single device to upload to multiple sources using the single protocol as the data could be dynamically routed based on the name specified.

8.3.3.4 Extensions to the Protocol

A number of different extensions were planned for the TCP protocol but were never implemented fully due to problems with opening additional TCP ports on the hosted server. These extensions involved adding new instructions to allow the client to request information from the server and extensions to improve the security and reliability of the transmission process.

The primary purpose of the protocol is to provide an upload mechanism that is both versatile and easy to implement. The protocol was not initially designed to allow the client to request data back from the server, although this functionality has since become desired in order to improve the user interface on the host controller. An INFO command was proposed to allow the client to request a specific piece of information from the server (which would use the device name to provide different

information for each device). This information could range from diagnostic messages to be displayed on the screen of the host controller, to periodic time or status requests for use on the host controller.

8.3.4 Uploading to an HTTP Web Server

The last uploading method utilizes the most commonly used internet communication technology by performing the upload procedure using standard HTTP commands. Although these commands were not directly intended to be used in this fashion, the upload process uses the same standard HTTP commands used by a standard internet browser and therefore requires no additional technology or services on the host.

The concept behind utilizing existing HTTP commands is to pass data between the Host Controller and the server over a medium which requires no unusual or specialized configuration. The HTTP commands invoke scripts that directly process the data. Unlike the case of a static upload using a protocol such as FTP, the server is fully aware that it is receiving specific data and therefore can apply correct time-stamps and store it accordingly.

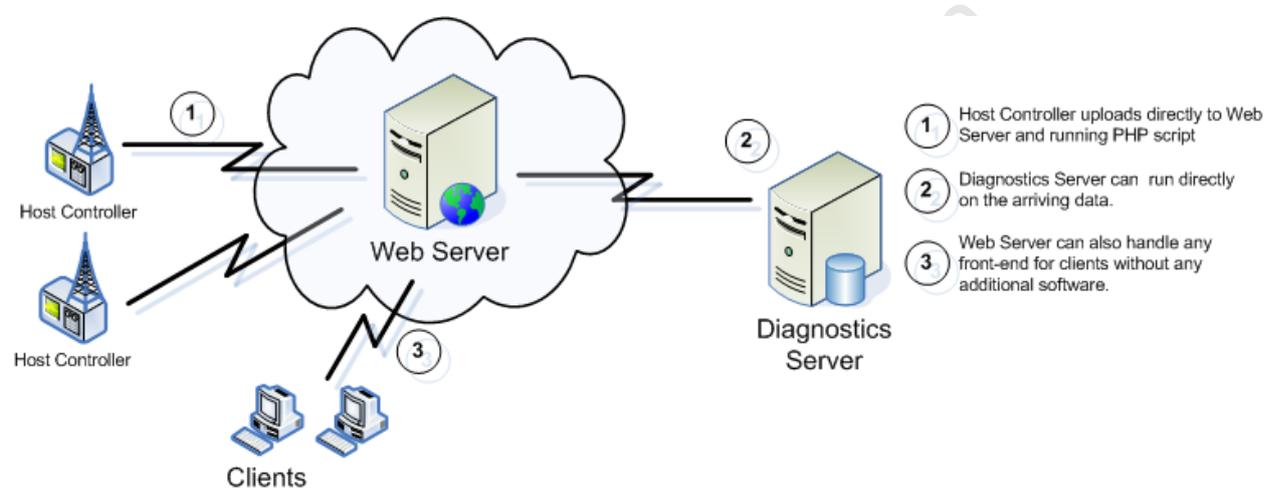


Figure 8-7: Diagrammatic outline of the Uploading using HTTP Commands

The diagnostics server can reside in one of two places. It can be installed directly onto the web-server or it can be completely disconnected from the internet and periodically check the uploaded data. Locating the diagnostics server on the web server (inside the internet cloud show below) is much safer than in other methods as Web Server technology has evolved stringent and well-understood security mechanisms. The Web Server requires only the ability to execute either PHP or ASP scripts (a standard feature from most hosting companies). The only point of entry into the diagnostics program is through these established scripting languages.

The HTTP protocol is a stateless protocol and therefore treats each new connection as a new entity. This allows multiple Host Controllers to upload simultaneously in one cohesive operation. All transactions revolve around a request sent to the server containing data and various headers, and a response from the server indicating the result of the request. The connection is terminated directly after the response.

The downside to using a method such as this is that data needs to be formatted specially for transmission. There is also an increased overhead in the form of various header packets that need to be transmitted.

Currently the HTTP upload method is being used in all devices as it has proved to be the easiest and most versatile means of uploading data. The current configuration involves a standard web server

provided by a web hosting company which contains a PHP application and a MySQL database. A domain name was registered and linked to the server to allow the host controllers to locate the server. The web server also hosts the client PHP application, which allows viewing of both the readings and any diagnostic data produced.

The host controller implements a basic HTTP client using a single TCP connection provided by the GSM unit. The host controller creates a connection to the server using the standard port for HTTP traffic and interacts with the web server using standard HTTP requests. The structure of these requests is given below.

8.3.4.1 Structure of an HTTP Request and a Response

There are a number of complications regarding implementing an upload mechanism over HTTP. Many of these problems arise from the structure of the HTTP protocol, which is not well suited for implementation on a microprocessor. In order to explain the issues surrounding the upload process, it is necessary to provide a brief overview of the structure of an HTTP request.

An HTTP server does not store information about the client between connections. Each session is considered unique, and various headers in the HTTP request message are used to convey information between sessions. For this reason, HTTP is known as a connection-less protocol. An HTTP device will initiate a connection to a server and send a request. The server will then return a response which contains the result of the request sent to the server.

In order to facilitate internet browsers, the server does not necessarily close the connection after a successful response has been sent [41]. The client may request that the connection remain open so that additional requests can be sent. This is known as a persistent connection and can be further optimized through the use of a technique known as pipelining. Pipelining allows the client to send requests at any time during the session. The server then responds to the requests and returns them back-to-back. This further reduces any delays or idle-time during the session.

All these techniques are used by internet browsers to improve the speed at which HTML content can be displayed as web pages tend to contain many objects which must be retrieved using separate requests. The uploading system is far simpler and hence only uses a small subset of the features described above. The uploading mechanism uses non-persistent and therefore non-pipelined HTTP sessions. The host controller simply opens a connection, sends a request and retrieves the response.

It is also important to note that HTTP requests are sent in plain ASCII and are therefore relatively easy to generate on a microprocessor.

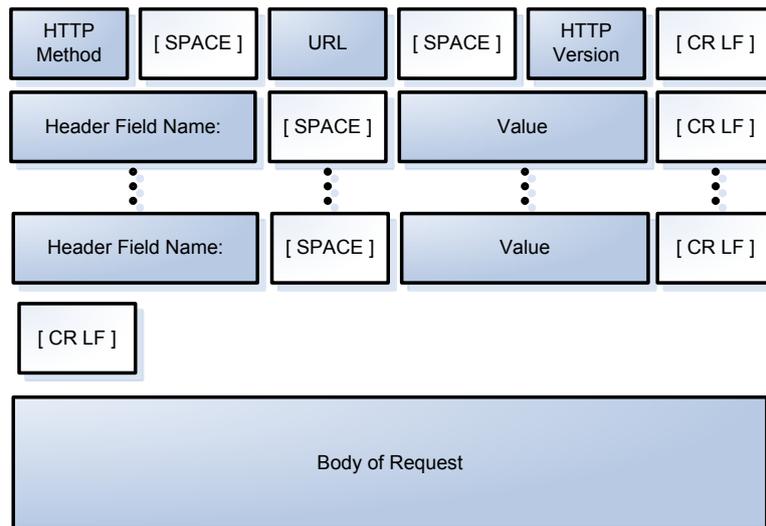


Figure 8-8: Structure of an HTTP Request

An HTTP Request consists of three major sections. There is a request section which outlines the type of operation and the object being requested. The second section contains headers which are used to control various aspects of the connection. Headers are also used to pass information to the server in addition to the body of the request. The last section of the HTTP request is the body which contains any data relative to the request type. New-line characters are used to separate the various parts of the message.

As the structure of the request is critical to the uploading process, each part is discussed in relevant detail in the following sections.

8.3.4.1.1 The Request Line

The request line determines the nature of the message and contains three fields. The first field is the type of HTTP request (also known as the HTTP method), the second field is the URL being requested and the third field is the HTTP version, which allows the server to determine the capabilities of the client. An example of a request is given below

```
GET /upload.php HTTP/1.1
```

The above example shows the HTTP request line for retrieving a webpage with the name upload.php. The forward-slash before the name indicates that the file is located on the root folder. The last part of the request simply states the version of the HTTP protocol implemented on the client. Currently only versions 1.0 and 1.1 are supported; with version 1.2 is currently being developed.

The HTTP request is a character string (terminated by a whitespace character) denoting the type of operation to be performed. These can be divided into safe and potentially unsafe methods [42]. Safe methods are not capable of modifying resources on the server directly (although they may have very harmful indirect effects). Unsafe methods are used to modify resources on the server and are less commonly used as a result. A table of the eight HTTP Methods is presented below.

Method Name	Safe	Description
GET	YES	The GET method is used to retrieve a resource (specified in the URL field) to be returned in the body of the response from the server. It is intended only for data retrieval but is often misused in web applications to interact with data. It is the most commonly used HTTP

		method.
HEAD	YES	The HEAD method is used to return the same response as a GET request without the response body. As a result only the response codes and the headers are returned by the server. This is useful when only the meta-data sent in the headers is required.
PUT	NO	The PUT method is used to upload a resource to the web server.
DELETE	NO	The DELETE method is used to delete a remote resource on the server.
POST	NO	The POST method is used to pass data from HTML forms to the web server for processing. The POST method also requires a valid resource (specified in the URL field) to which the form data will be sent for processing. The result can be a change to the existing resource, or even the creation of a new resource.
OPTIONS	YES	The OPTIONS method causes the server to return a list of the supported operations. It can be used by the client to check the capabilities of the server.
TRACE	YES	The TRACE method causes the server to echo back the request it received. This is useful when intermediate web servers are present on the connection as they may append headers to the HTTP request. The TRACE method can then be used to examine the headers being attached.
CONNECT	NO	The CONNECT method is used to establish a transparent TCP/IP tunnel to allow HTTPS (SSL-Encrypted HTTP traffic) through a standard web proxy.

Table 8-3: HTTP Request Methods in HTTP Version 1.1

In order to upload data to the web server, the host controller can use either GET or POST requests. The host controller makes a request to a special resource on the server (usually a file called upload.php, accessible through a special URL) and passes the data along in the request. The means by which the data is passed differs between the two HTTP methods.

The GET method is not supposed to be used to alter the state of the server but it is often used in this manner due to the relative ease by which it can be implemented. The web server only examines the headers associated with a GET request and ignores the body of the request. For this reason, data is passed as part of the URL field. The data is appended to the end of the URL, usually with a special character like a question mark. The exact syntax varies with the web server software but a very common configuration is as follows:

```
GET /upload.php?parameter1=value1&parameter2=value2 HTTP/1.1
```

In the above code, the resource being requested is a file called upload.php. The PHP extension indicates that the file contains a script and the file will then be processed by the PHP runtime engine on the server. The question mark character denotes the beginning of the parameter information (often referred to as a query string) which will be passed along to the PHP engine when the script is executed and can be compared to the method by which command line arguments are passed to an application.

The data in the query string is arranged in name/value pairs separated with the ampersand character. Any special characters (including spaces and punctuation) need to be encoded when placed in the query string. The HTTP server does not perform any checking on the query string (unless it contains white space or new line characters, which would cause the request to fail).

The protocol specifications do not provide a maximum limit for the length of the URL field but it is important to note that this has resulted in different technologies imposing various limits that need to be considered [43]. RFC 2068 cautions that servers should not implement URLs over 255 characters in length as older clients and proxies may not be able to support them [44]. During testing of the uploading mechanism, URLs of over 1024 characters were used successfully although there is no guarantee that these results can be repeated.

A better alternative to the GET command is the POST command. The original purpose of this command was to pass data from HTML forms to the web server for processing. Unlike the GET command, the contents of the request body is examined as it contains the data being sent. POST commands can therefore send far greater amounts of data. The structure of the POST command request line is very similar to that of the GET command. An example is shown below:

```
POST /upload.php?parameter1=value1&parameter2=value2 HTTP/1.1
Content-Length: 35
```

```
Parameter3=value3&Parameter4=value4
```

The above example shows the complete (although minimal) POST request and therefore includes both the header and body section. These are included for the sake of completeness and are described in detail in the next sections. It is important to note that the Content-Length header contains the amount of data in the body of the POST request.

The query string parameters in the GET command are actually considered to be part of the URL and do not form part of the HTTP request. For this reason, parameters can also be passed in the same way with a POST command. In addition the body of the request can also contain data in the same name/value pairs. In PHP, both sets of parameters are easily accessible.

Due to the potential limits associated with the length of the query string in the GET command, the POST command was used when uploading from the Host Controller. Both the query string (in the URL) and the body of the request were used. The URL would contain the device name and all the relevant data was placed in the body. It was felt that this arrangement would scale well.

8.3.4.1.2 The Header Line

The second section of the HTTP request contains the headers or meta-data which are used to specify additional information about the connection, caching and nature of the request. A header entry consists of the name of the header, a colon the desired value and a new line character. There can be any number of headers between the request line and the body of the request and the section is terminated with two new line sequences (one to end the last header and a second to end the header section). A list of the relevant headers is given below:

Header	Example	Description
Connection	Connection: close	The connection header informs the server whether or not the client would like a persistent or non-persistent connection.
Host	Host: abc.com	The host header is used to allow multiple websites to be hosted on a single server. The host header specifies the host name being requested.
User-Agent	User-Agent: Dev001	The user agent allows the client to specify the name of the client application making the HTTP request. This is commonly used allow web-servers to provide web pages tailored to the user's browser.
Cache-Control	Cache-Control: no-cache	This header is processed by all the devices on the path between the client and the server. It specifies whether

		or not these objects may cache the requested resource.
Content-Length	Content-Length: 100	Specifies the length (in Bytes) of the data contained in the request body. Most web servers will not process any additional information in the request body.
Content-Type	Content-Type: image/gif	Specifies the type of content contained in the request body. These types are defined as Multipurpose Internet Mail Extensions (MIME) and are detailed in RFC 2046 [45].

Table 8-4: HTTP Headers required in the Upload Process

The HTTP headers play a significant role in the functioning of the upload mechanism. The above table only shows a small subset of the possible header fields that were relevant to the uploading process. Each of the headers played a significant role in the development process and often behaved differently when executed on a remote web server. The details of these headers are discussed below:

8.3.4.1.2.1 The Connection Header

The connection header is used to specify the nature of the connection requested between the client and the server. This field was not included in the original HTTP/1.0 specifications, although it existed in an experimental form. The header was included in the HTTP/1.1 specification and it was felt that it should be implemented in the communication between the host controller and the server as the default connection type of the web server varies.

Due to the nature in which the GSM module multiplexes data and commands on a single UART, it was found that it improved reliability to have the server disconnect the connection before the GSM module. For this reason, the client expressly requests a non-pipelined connection in which the server disconnects immediately after issuing a response.

The first version of the GSM controller did not receive responses from the host controller and simply disconnected after sending the HTTP request. Although this method worked well, it did not provide any means for the host controller to determine if the upload procedure succeeded. The second version of the host controller firmware would wait for the server to return a response before closing the TCP socket.

8.3.4.1.2.2 The Host Header

The Host header is used to specify the hostname being requested by the client and is provided to allow support for multiple web sites (each with a separate domain) to be hosted on a single web server. In such a case, the host name for each web site will resolve to a single IP address. When a client connects to that IP address, the server has no means of determining the host name through which the client accessed the server and therefore cannot display the requested page.

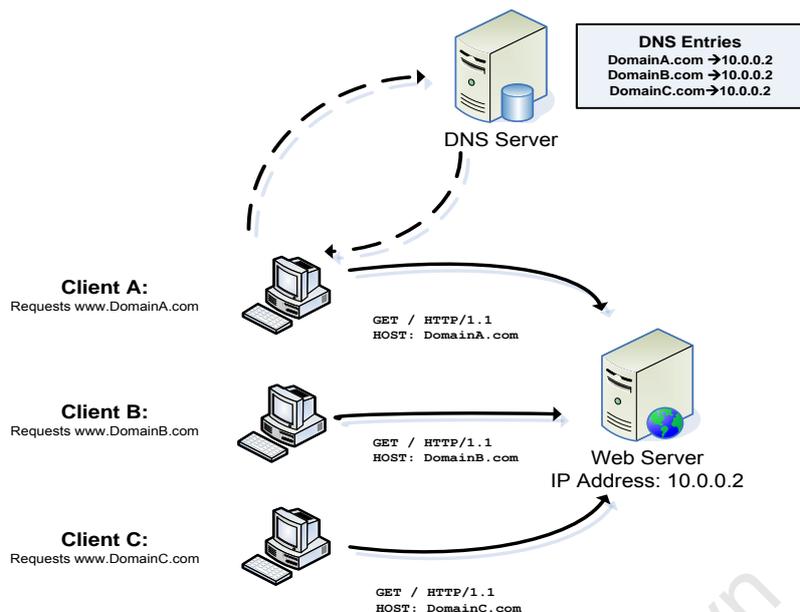


Figure 8-9: The Use of the Hosts Header when accessing Virtual Web Servers

The above figure demonstrates the method in which the Host header allows multiple domains to co-exist on a single web server. The figure shows three separate clients, each requesting the default web page from separate domains. Each of these domains is hosted on the same physical server. When a client makes a request, it must first resolve the hostname into a valid IP address by querying a DNS server.

DNS servers are essentially lookup tables that match host names to IP addresses. This mapping is not necessarily a one-to-one relationship as multiple hostnames can resolve to a single IP address (as shown in the figure above). Each hostname will therefore direct requests to the same IP address. TCP sessions set up between the server and the client are only aware of the source and destination IP addresses. The original hostname is never passed along by the TCP protocol and server cannot determine which hostname the client visited.

In order to allow the server to provide the correct web page, the Host header is used and specifies the name of the domain being requested. The web server can then provide the correct web page. This header provided some difficulty when developing the system as it is not required in version HTTP 1.0 and therefore was not included when the system was being tested off an internal web server.

This resulted in the entire system working on an internal server, but not working on a hosted server. The need for the Host header was only discovered after monitoring outgoing HTTP traffic between a web client and the hosted server.

8.3.4.1.2.3 The User-Agent Header

The User-Agent header is used to provide the server with a character string identifying the client being used to access the HTTP resource. Internet browsers use this field to identify themselves to web servers to allow the server to send an optimized version of the page to the client. There are no requirements as to how this field is used, and many web servers will simply ignore the header.

As each host controller needs to identify itself to the server when uploading, it was thought that the User-Agent header could be used to send such information. Each device would place its unique name (or number) into the User-Agent header to allow the PHP script to determine the source in that manner.

The original method used involved transmitting the device name in the request body. The server would then interpret the message and extract the key. This method works for uploads performed using the POST command but cannot function on features that require the GET HTTP command as it does not allow the use of the request body. As devices may make GET requests periodically to obtain information such as the current time, the ability to identify these devices when making a GET request allows the server to stay better informed as to the current status of the device.

8.3.4.1.2.4 The Cache-Control Header

The Cache-Control header is used to prevent any devices between the client and the web server caching the particular resource being requested. Caches are employed on the internet in order to improve the response times by storing copies of static web content on a server closer to the client. This is often used by internet service providers to conserve international bandwidth and improve the response time for clients. Internet browsers themselves also cache web resources locally to further improve performance.

Most caching systems will only cache content that is deemed to be static and unlikely to change frequently. The Cache-Control header is used to explicitly define the caching behavior allowed on the web resource and is used by the diagnostics system to explicitly prohibit any caching from taking place.

This proved to be a problem when performing GET requests to a server as the pages were not updating correctly until cache control was added. A number of supporting headers are also provided to give finer control over the caching process, including the maximum amount of time that an object can be safely cached. These were not used as none of the data travelling between the host controller and the upload server is static in nature.

8.3.4.1.2.5 The Content-Length Header

The Content-Length header is used in POST requests to specify the amount of data in the body of the request. Many web servers will stop processing data after this number of bytes has been exceeded. As the length of the body is sent in a header, the request body cannot be dynamically generated as it is sent. It must be calculated before the request is sent.

It is not always possible to generate the upload data for the request body in its entirety as it requires a large amount of RAM. When the upload procedure is performed from the microcontroller on the host controller board, it simply cannot reserve that much memory for the upload procedure. Instead, the host controller calculates the length of the request body prior to sending and dynamically generates it when required. As a result, the format in which sensor data is uploaded to the server is extremely rigid.

8.3.4.1.2.6 The Content-Type Header

The Content-Type header is used to describe the type of data contained in the body of the request. This header is then used to determine how the web server processes the body of the request. This field might require additional fields in order to provide more information to the server.

In the case of a POST request, the Content-Type is set to *application/x-www-form-urlencoded*. This indicates to the server that the data is encoded using the name/value pair syntax described in chapter 8.3.4.1.1.

8.3.4.1.3 The Request Body

The request body is transmitted directly after the new line characters that signal the end of the header section. The request body is not required by many of the HTTP requests, such as GET, HEAD and TRACE. It is used when performing a POST operation to transmit the data intended for the server as it can be up to 4 Mb in size (although many servers do not support requests of that length).

As HTTP requests are sent using ASCII characters, the data in the request body must be encoded in such a fashion. This created a problem for the upload mechanism as the numeric data (representing sensor values) needed to be converted to a character string when uploaded. Functions such as `itoa()`, which convert integers to character strings, cannot be used as they produce variable length strings depending on the value of the number.

Variable length strings caused a problem as it prevented the host controller from being able to calculate the length of the request body before generating it. This essentially prevented the host controller from generating the request body when required. The solution was to convert the number to its hexadecimal representation as it would always be represented by a four character string.

8.3.4.2 Performing an Upload from a Host Controller

HTTP requests are generally transmitted using a standard TCP on port 80 in ASCII format. HTTP connections require no hand-shaking and no exchange of configuration data as the client simply connects and makes a request. These factors make the HTTP protocol relatively easy to implement using a microcontroller.

The GSM unit used in the project provided an HTTP library which was accessible for firmware applications written in the OpenAT environment. This library was found to be incomplete and incapable of properly implementing the Hosts header. For this reason, a custom HTTP library was developed.

As mentioned in chapter 6, the upload process was divided between the PIC microprocessor and the firmware on the GSM unit. This was done to avoid the need to multiplex data and commands on the single interface between the GSM unit and the microprocessor.

The upload process can therefore be divided into two separate phases; the uploading of readings to the GSM unit, and the Uploading of readings to the Diagnostic server.

8.3.4.2.1 Uploading Readings to the GSM Unit

An upload is only initiated by the Host Controller once it has successfully acquired readings from its network of Sensor Interface Boards through the Reading Acquisition process described in chapter 7.4. The Host Controller stores these received readings in an array which implemented using the flash memory (when necessary). The first stage of the upload process requires the microcontroller to transfer these readings to the GSM unit.

This is accomplished using a set of proprietary AT commands exposed by the firmware on the GSM unit. These commands allow the microcontroller to pass the readings using the command interface (since the data is passed as a parameter in the AT Command). The microcontroller uses the standard set of AT commands to initialize the GSM unit and then waits until the device is ready.

When the GSM unit is fully operational, the microprocessor begins transmitting the proprietary AT commands. A table of these AT commands is presented below:

AT Command	Parameters	Description
AT+ADDREADING	<address>,<value>,<reference>	<p>The ADDREADING command is used to pass a reading to the GSM unit. It takes three parameters corresponding to the three values associated with each reading stored on the Host Controller.</p> <p>The <i>Address</i> is the address of the sensor assigned during the network discovery.</p>

		<p>The <i>value</i> is the actual reading taken. It is a 16-bit hex value.</p> <p>The <i>Reference</i> is the second value obtained with each reading and is used in certain readings to improve accuracy.</p> <p>The result of the command is successful if the reading is successfully added.</p> <p>Example: AT+ADDREADING="0001","0002","0003" OK</p>
AT+CLEARREADINGS	None	<p>Clears any stored readings on the GSM unit.</p> <p>Example: AT+CLEARREADINGS OK</p>
AT+DELETEREADING	<address>	<p>Deletes a reading with the specified address.</p> <p>Returns successful if the reading was found and deleted. The command is unsuccessful if a reading matching the supplied address is not found.</p> <p>Example: AT+DELETEREADING="0001" OK</p>
AT+GETREADINGS	None	<p>Displays a list of all readings currently stored on the GSM unit. Primarily used as a debugging tool.</p> <p>Example: AT+GETREADINGS +GETREADING: "0001","0002","0003" +GETREADING: "0002","0003","0004"</p>
AT+GETREADING	<address>	<p>Returns the reading information for a reading matching the given address.</p> <p>Example: AT+GETREADING="0001" +GETREADING "0002", "0003"</p>
AT+STARTUPLOAD	None	<p>Instructs the GSM unit to upload the given set of readings. The stored readings are not deleted when the upload process is completed.</p> <p>If the upload is successful, the command will output a message containing the</p>

		<p>current time obtained from the diagnostics server. This is used by the microcontroller as a basis for timing future uploads.</p> <p>Example: AT+STARTUPLOAD +STARTUPLOAD: Opening GPRS Bearer +STARTUPLOAD: Opening TCP Connection +STARTUPLOAD: Upload Complete +TIME: 13:34:23</p>
--	--	---

Table 8-5: Proprietary AT Command Implemented on the GSM Unit

The microprocessor simply loops through the array of stored readings and transmits an AT+ADDREADING command for each one. As the data is buffered on the GSM unit, there is no timing requirement which allows external memory access to be implemented in a simpler fashion.

Once the Host Controller has successfully uploaded all the readings, it issues the AT+STARTUPLOAD command to initiate the upload procedure on the GSM unit. The unit will respond with status messages which are simply ignored by the microprocessor. If the upload is successful, the GSM unit will send a special +TIME response containing the current time on the server. This time is then used by the Real Time Clock as a time-base to synchronize future uploads.

The device will perform its first upload once it has powered up and discovered the attached network. If successful it will obtain the current time on the server and begin uploading readings at the required times. This method allows the unit to keep track of the time without requiring it to be set manually.

It is important to note that additional information can also be passed between the Diagnostics Server and the microcontroller by employing additional unsolicited responses. These responses would have a similar structure to the +TIME response but contain other information. A +INFO message is often used to pass a message intended to be displayed on the Host Controller screen.

8.3.4.2.2 Uploading Readings from the GSM Unit to the Diagnostic Server

Once the microcontroller has uploaded readings to the GSM unit and instructed it to proceed with the upload process, the firmware application executes. The GSM unit performs the following actions when uploading data to the Diagnostics Server:

- **Opens the GPRS Bearer:** In order to exchange data using GPRS, the unit must perform a GPRS attach. This can only be accomplished in areas where there is GPRS coverage. During this process the network will assign the GSM unit an IP address and will return the successful once the device is fully attached to the network.
- **Create a TCP Connection to the Diagnostics Server:** The firmware then attempts to create a TCP connection to the diagnostics server. The server is usually specified using a hostname which is either hard-coded into the GSM unit, or received via SMS. The TCP/IP stack on the Wavecom unit is capable of either accepting an IP address or resolving a hostname using DNS. The GSM unit always assumes the default port to be 80.
- **Issue the HTTP Request Line:** If the connection is successfully made, the firmware will begin to issue the HTTP request line. Usually this is a POST message accessing an upload script on the server. The firmware passes the GSM units IMSI number in the query string of the request, in order to uniquely identify itself to the upload server.

- **Issue the HTTP Headers:** The firmware then needs to transmit the HTTP headers. A few standard headers regarding the type of connection and caching are transmitted, along with the content length and type. The content length is determined from the number of readings transferred the GSM unit. The firmware also sends the appropriate Host Header, which it determines from the requested hostname. If an IP address is used, the Host header is not sent.
- **Transmit the Readings in the Request Body:** The firmware then encodes each reading into a query string and transmits it to the HTTP server. Once all the readings have been transmitted, the firmware waits for the HTTP server to respond or close the connection. If the upload has been unsuccessful, the firmware will disconnect the connection after remaining idle for 30 seconds.
- **Interpret the Server Response:** The upload script on the upload server has been designed to return the current time on the server in the body of the HTTP response. The firmware first analyses the HTTP result to determine if the POST was successful and then, if available it extracts the time and passes it to the microcontroller through a +TIME message.

Once these steps have been completed, the GSM unit will remain idle until it is instructed to either enter a low power mode, or turn off.

8.3.4.3 Processing Data on the Server

The server component of the HTTP uploading process requires a web server running either PHP or ASP.NET. As PHP is an open-source initiative, it was chosen as the server technology for use in the diagnostics system. An HTTP request needs to be targeted at a specific resource, such as a webpage or an image, and therefore the Host Controller needs to make a request to a specific PHP script on the web server.

The web server performs the majority of the work in receiving the uploaded data. As it is a standard HTTP POST command, the web server receives, parses, interprets and then passes the data to the PHP script. In fact, PHP as a language exposes the data received from a POST command in an associative array, making the retrieval as simple as possible.

The upload script needs to examine the unique identifier specified by the Host Controller during the upload process and then load the appropriate sensor mapping file. The mapping file relates the addresses assigned by the Host Controller to fields in a database. This file (and the associated table in a database) represents the only customization required for each site.

The PHP script then creates an SQL insert query and pushes the data into a relational database. The Diagnostics system will then pick up the readings from there and produce the appropriate reports. The only requirement of the PHP file is that it must return the current server time if the upload is successful.

8.3.4.4 Authorization, Encryption and Security

One of the biggest problems with the HTTP method as it is currently implemented, is that the data is sent in plain-text and is therefore not secure. One way in which the system attempts to mitigate the security issues is by relying on the diagnostics engine to perform any calculations on the reading information. Therefore, intercepted readings would require an in-depth knowledge of the installation in order to extract any meaningful information.

HTTP does provide a number of mechanisms which could be used to improve security in the upload process. These are HTTP authentication and Secure Socket Layer (SSL) HTTP connections. Both these technologies are widely used on the internet to provide secure transactions and would be relatively easy to implement on the Host Controller.

Such technologies do come at a cost, both financial and in terms of quantity of data transferred. The current upload process was designed to minimize the data uploaded. Technologies such as SSL would require multiple requests and negotiation between the server and the HTTP client. This would effectively minimize many of the benefits of using HTTP uploads.

8.4 Justification for Uploading using an HTTP Server

The HTTP upload mechanism proved to be the most effective and efficient means of uploading data from a Host Controller. Although it lacks the streamlined efficiency of the proprietary TCP protocol, it does not require additional ports to be opened or a custom server application to be run. Unlike the FTP and SMTP upload methods, HTTP does not suffer from asynchronous problems and potential time delays.

The HTTP solution provides the most flexibility and requires the fewest server resources to implement. Although it may require more resources on the client than the proprietary TCP protocol, it benefits from the ease with which PHP websites can scale. The TCP server developed implemented a basic multi-threaded server which would require much tuning before being ready to handle a large number of clients.

University of Cape Town

9 Results

The monitoring system designed was first installed on a single chiller in a building on the Foreshore in Cape Town. The building is 7-story office building with a centralized chilled water system. There is a large plant room on the top floor of the building in which there are four chillers. There are four smaller plant rooms on each floor which house Air Handling Units, these are located on the four corners of the building.

The plant room contains four chillers, one large Carrier chiller and three smaller Daikin machines. The smaller Daikin chillers were originally purchased to replace the large Carrier chiller, which operated on a banned refrigerant gas. The Carrier chiller has since been converted to run off R143a refrigerant and re-commissioned in order to boost the capacity of the chilled water system.

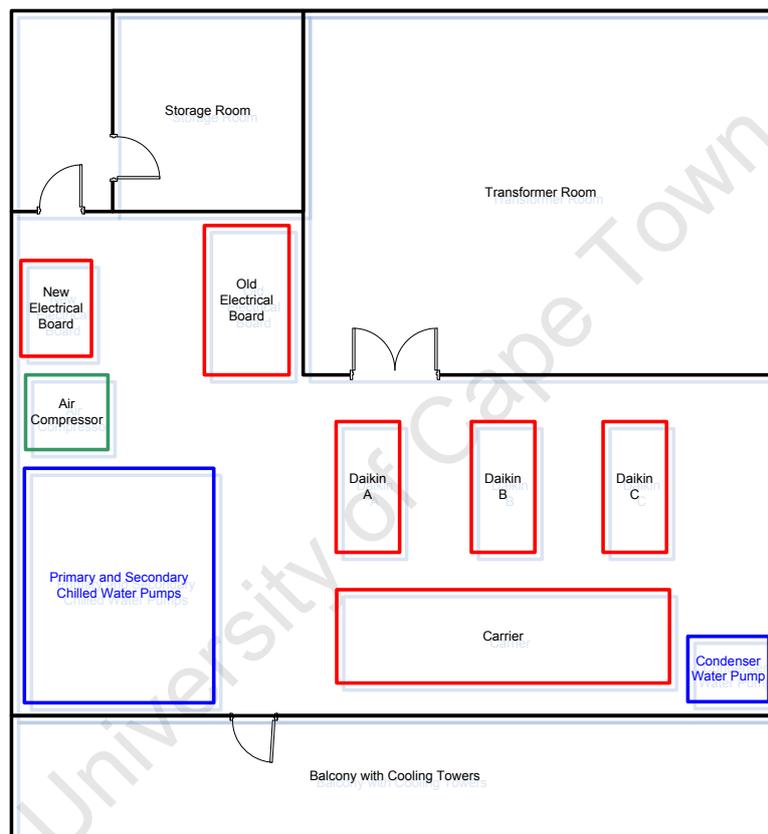


Figure 9-1: Layout of the Plant Room at the Installation Site

The above figure shows the layout of the plant room. The monitoring system was installed on “Daikin B” initially. A second phase of the installation includes the monitoring of the Carrier chiller, the air compressors and various pumps in the plant room. These systems could not be installed until the Carrier machine was fully re-commissioned but all the hardware has been developed and is ready to be installed.

The Daikin Chiller being monitored is a water-cooled chiller with two compressors, each fitted with an unloader. All three Daikin chillers are connected to a common cooling tower. As a result, these chillers share a common Condenser water return pipe and common condenser water pumps. The pump shown in the bottom right of the figure is a condenser water pump for the Carrier chiller, which uses a separate cooling tower.

In order to monitor the Daikin chiller, Four Sensor interface Boards were installed on the chiller itself. A dedicated board was installed for each compressor which monitoring the evaporator pressure, the discharge pressure, the oil pressure and the motor current. The other two boards are used to monitor the temperatures of the condenser water and the chilled water as it enters and leaves the unit. A summary of the network structure is shown in Figure 9-2.

A fifth board was installed on a louver opening onto the balcony in order to monitor the outside air conditions. The board contains a temperature and a humidity sensor. The Host Controller was placed in the storage room off the corridor to allow access to it without having to enter into the plant room.

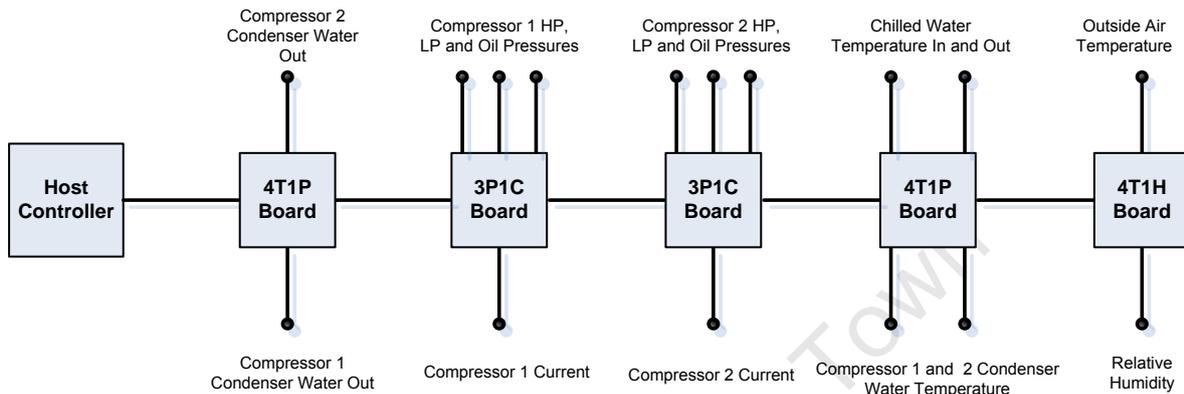


Figure 9-2: Network Configuration for "Daikin B" Chiller

The three Daikin machines are operated from a single controller and are controlled in accordance with the demands of the building. The control system implements simple three-stage cooling by selectively bringing in compressors as the load increases. Further control is provided by the chillers themselves through the use of the unloaders, which are controlled by the chiller in response to the chilled water temperature differential.

It is common to cycle the chillers in such a control system so that each unit wears equally. It would be found that the control system always turned chillers on in the same sequence, so that the Daikin Chiller being monitored operated almost continuously, with the other two being used when needed.

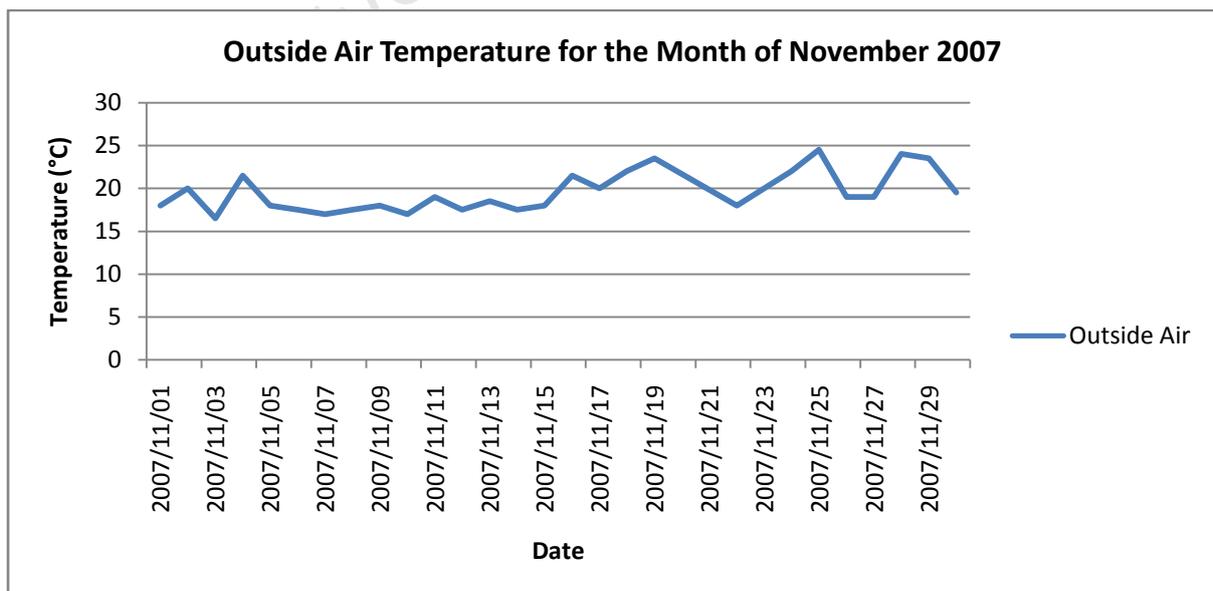


Figure 9-3: Outside Air Temperature for the Month of November

During the month of November, the Carrier chiller was not operational and the Daikin machines ran continuously. All three of the Daikin machines were running for the majority of the month. For this reason, this month has been used to provide diagnostic results. A short summary of some of the results is given below.

The graph above shows the daily temperature (taken at 10:20 AM each day) for the entire month of November. The average temperature for the month was 20° C. January and February are typically the hottest months of the year and therefore require the air conditioning system to operate at close to the designed conditions.

Due to the increased load of the building, and the fact that the Carrier chiller had not become fully operational, the plant was not able to handle the peak conditions in the months of January and February. For this reason, the readings taken were from the month of November, in which the plant operated normally.

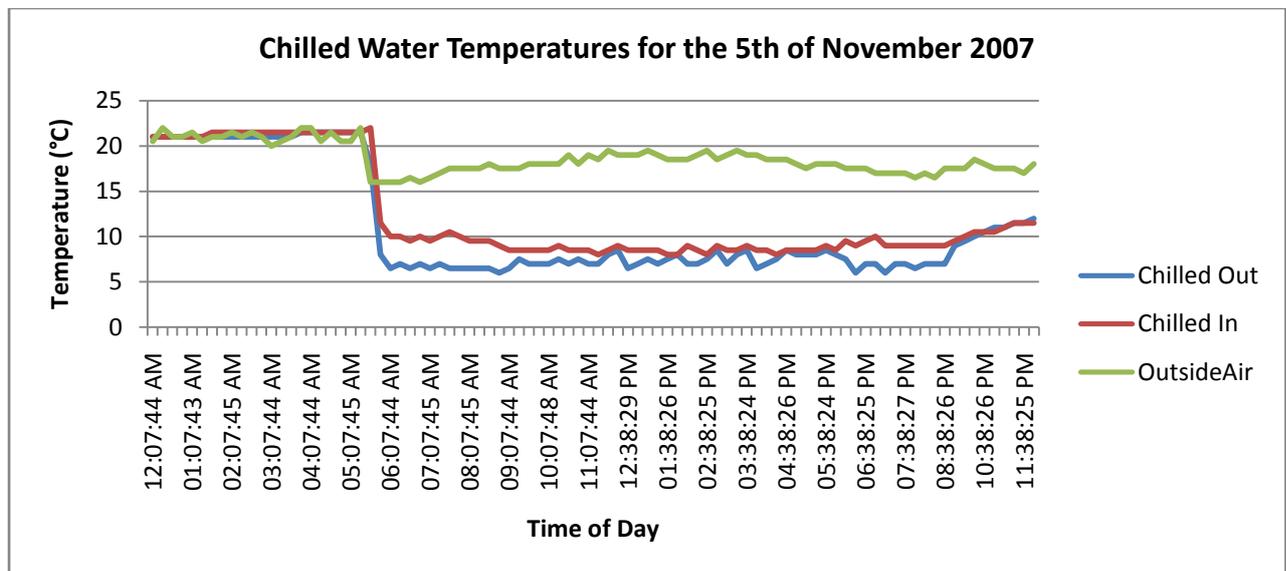


Figure 9-4: Graph of the Chilled Water Temperature for the 5th of November

The above graph shows the chilled water temperatures for the 5th of November 2007. The graph also includes the outside air temperature in order to assist in interpreting the graph. In the early hours of the morning, the entire plant is off and therefore the chilled water settles to approximately the temperature of the surrounding air. The plant turns on at 5:30 AM, which is incidentally one of the coldest points in the day. It is clear from the graph that the difference in chilled water temperature (the distance between the blue and red lines) decreases as the outside air temperature increases. As the flow rate in the chilled water system is known (it is a constant volume system), this differential can be used to estimate the instantaneous demand of the building.

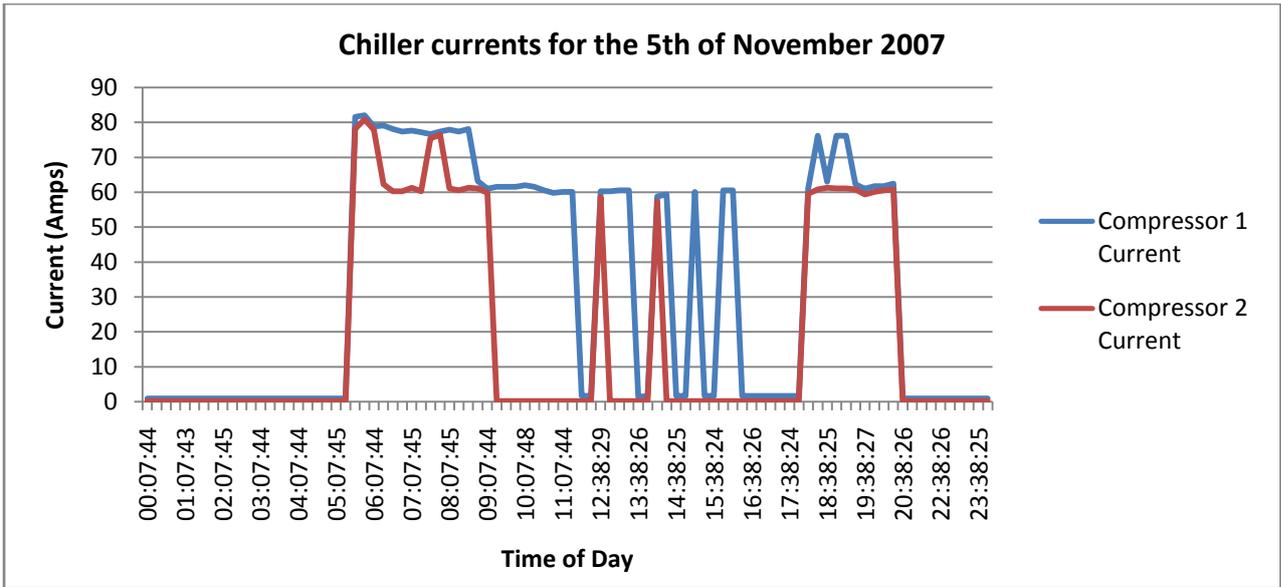


Figure 9-5: Compressor Currents for the 5th of November

The graph above shows the current drawn by the compressors on the same day. The action of the unloader is clearly visible as the sudden drop in current from 80 A to 60 A on both compressors. It is also clear that compressor 1 is operating more frequently than compressor 2. The peaks, during which both compressors are operating, occur at 12h00 and 13h00 and correspond with the highest outside air temperatures for that day.

If the control system does not periodically cycle the order in which uses compressors, the first compressor will wear faster than the second. If left in such a manner, it would shorten the operational lifespan of the chiller. This behaviour was observed over the whole month of November, indicating a problem with the compressor cycling in the chiller.

It should also be noted that the plant operated until approximately 20h30 each night. As this is an office building, the load should dramatically reduce outside of business hours but it can be seen that both compressors run between 17h30 and 20h30, which is an example of performance losses due to the HVAC equipment being left on unnecessarily.

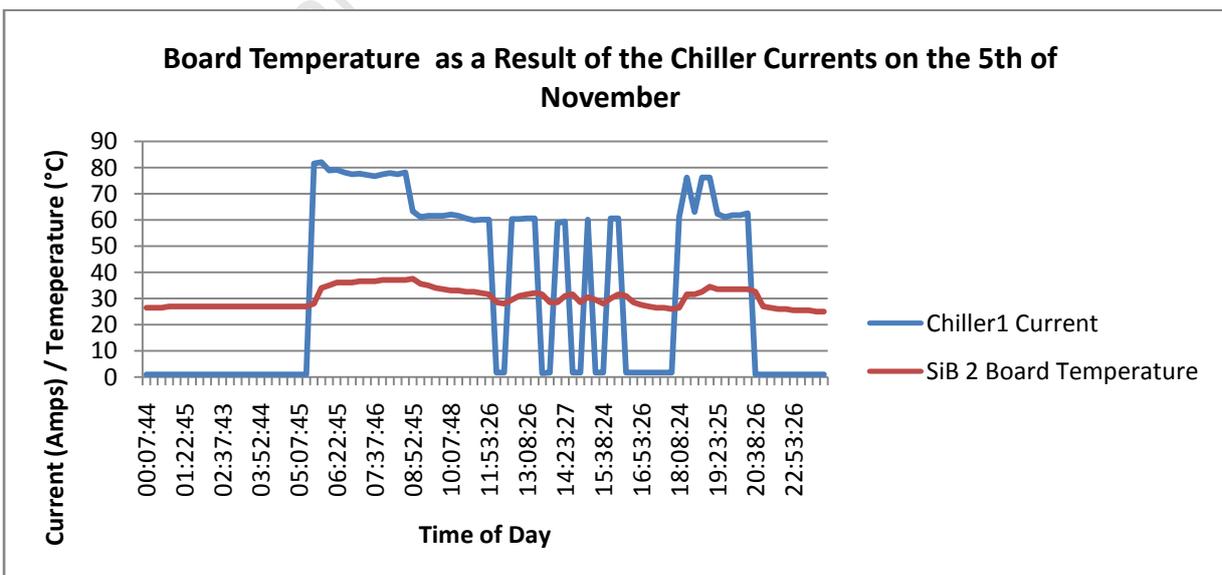


Figure 9-6: Graph of Board Temperature and Compressor Current

An interesting result was obtained when comparing the board temperatures to the compressor current. It was discovered that the board temperatures on the Sensor Interface Boards responded proportionally to the changing compressor current. This was discovered to be a result of the 1-Watt power resistor used to convert the current from the CT into a voltage. As the compressor current increased, the power dissipated in the resistor increased, and since the Sensor Interface Boards are sealed in plastic boxes, this caused the board temperature to rise accordingly.

The temperature remained well within the designed operating range of the Sensor Interface Hardware but it was noted that such a problem could manifest itself when monitoring larger chillers. To put the problem in perspective, the compressors on the Daikin chiller draw between 60A and 80A each. The large Carrier machine has a single compressor which draws between 150A and 400A.

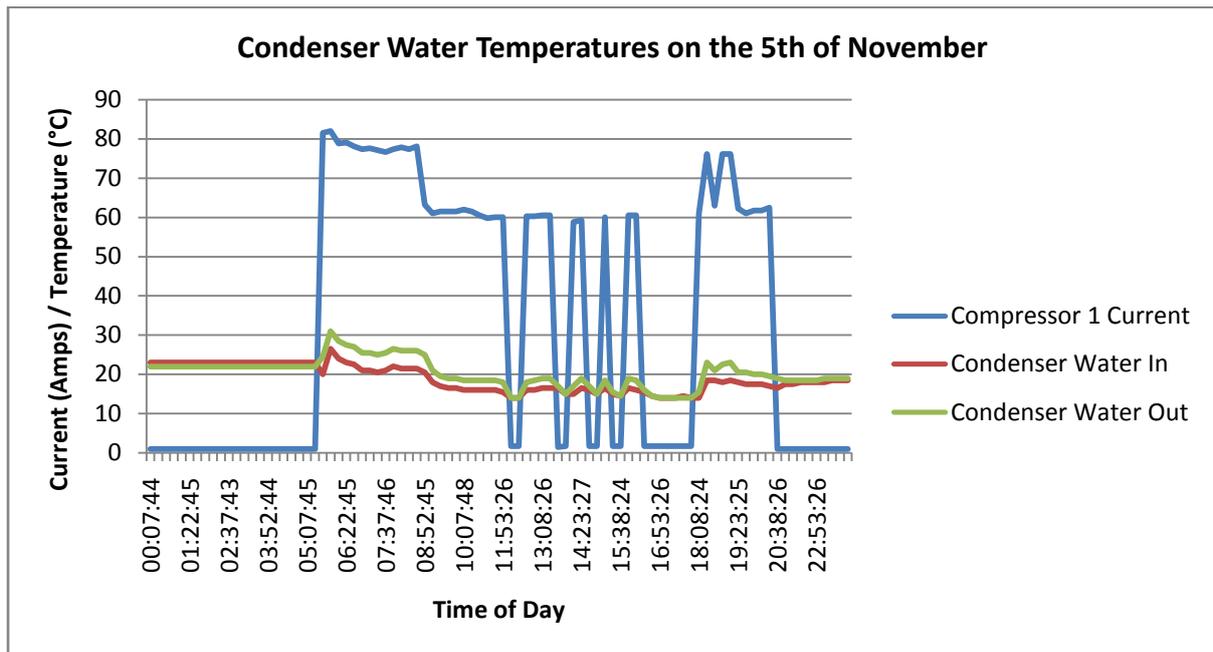


Figure 9-7: Graph of Condenser Water Temperatures for the 5th of November

The above graph shows the change in condenser water temperature against the change in compressor current. The compressor current can be used to estimate the expected performance of the compressor, and as a result, the expected temperature difference in the entering and leaving condenser water. If this calculated value is far below the expected temperature, then it is indicative of a condenser requiring cleaning. The condensers on this chiller appear to be relatively clean.

The second phase of the monitoring system is currently being installed on the Carrier chiller. The Carrier chiller operates at a different pressure range to the Daikin machines and therefore required different pressure sensors, which had to be specially ordered. A new humidity sensor was also sourced, which is supplied with a chemical epoxy coating to prevent corrosion.

The second phase of the installation will add six more boards to the existing network. The details of the boards are given below:

- **3P1C4T Pressure Board:** Installed on the chiller to monitor the suction and discharge pressures. It also has the capacity to monitor the chiller current, although it was not possible to monitor the current at the chiller due to the wiring of the start-delta starter.

- **3P1C4T Pressure Board:** A second pressure board was installed on the chiller to monitor the oil pressure before and after the oil filter. These pressures were located on the other side of the machine and therefore required a second board in order to pick up the temperature values.
- **3P1C4T Temperature Board:** Installed on the front of the chiller to monitor the condenser and chilled water inlets and outlets.
- **3P1C4T Temperature Board:** Two additional temperatures required monitoring on the Carrier. These were the refrigeration temperature and the oil temperature. A second temperature board was installed to monitor these two values.
- **2 8L8C Current Boards:** Two 8L8C boards will be installed in the two electrical switchboxes in the plant room. This will allow the system to pick up the currents and running time of the chillers, compressors, pumps and fans in the plant room. These two boards will essentially be capable of handling 16 different currents and 16 running times.

A new outside air temperature board will replace the existing one, which has survived a number of floods in the plant room. It is being upgraded to a newer version of the board to allow it to benefit from the lower power consumption and more stable firmware.

10 Conclusions

10.1 Successful Design and Implementation of the Proposed System

The proposed monitoring and diagnostics system was successfully implemented on a number of different sites around Cape Town. The monitoring system has been in operation since November 2007 and has suffered no significant failures or problems.

The design has proven to be easy to install and does not require any configuration on site. The system is fully autonomous and does not impact the functioning of the air conditioning plant in any way. The cost of the system was also kept within reasonable limits and the entire system will only cost a fraction of a building management system capable of performing the same diagnostic functions.

The system has worked exactly as proposed and is currently being installed on additional air conditioning plants around South Africa.

10.2 GPRS is an Effective Method of Providing Time-Insensitive Monitoring

The system has successfully demonstrated the effectiveness of utilizing GPRS in a remote monitoring application but has highlighted the potential unreliability of using GPRS networks. GPRS is suitable when monitoring slow-changing systems such as air conditioning system but may not be suitable for application which requires data at regular intervals as network congestion and unpredictable downtime can prevent such an application from operating correctly.

The vast majority of the problems encountered on site were GSM network related. Besides some network downtime, most of the problems arose from the use of prepaid airtime instead of proper data contracts. Although prepaid SIM cards work, there are a number of restrictions which are not made clear at the outset, such as the requirement to load air time within a certain period. These problems make prepaid SIM cards an infeasible option for a remote installation.

10.3 The Energy-Saving Potential of the Proposed System

The monitoring system was designed around the recommendations laid out in the TIAX report. Below is a reproduction of the table in chapter 3.1.1, showing the energy losses resulting from building faults. Highlighted in blue are the faults that are directly linked to air conditioning installations.

Fault	Fault Type	Energy Consumption (quads)
Duct Leakage	Air Distribution	0.30
HVAC Left on When Space Unoccupied	HVAC	0.20
Lights Left on When Space Unoccupied	Lighting	0.18
Airflow Not Balanced	Air Distribution	0.070
Improper Refrigerant Charge	Refrigeration Circuits	0.070
Dampers not Working Properly	Air Distribution	0.055
Insufficient Evaporator Airflow	Air Distribution	0.035
Improper Controls Setup / Commissioning	Controls	0.023
Control Component Failure / Degradation	Controls	0.023
Software Programming Errors	Controls	0.012
Improper Controls Hardware Installation	Controls	0.010
Air-Cooled Condenser Fouling	Refrigeration Circuits	0.008
Valve Leakage	Waterside Issues	0.007
Total		0.993 quads

As the table shows, all but one of the building faults are related to parameters can be directly monitored using the diagnostics system presented in this thesis. Each of these faults is discussed below:

- **Duct Leakage:** Although duct leakage cannot be detected directly using the monitoring system, the severity can be determined through the ability to accurately model the heat load on the building and the internal conditions.
- **HVAC Left on When Space Unoccupied:** This can be directly monitored using the system as it has been designed to monitor running time.
- **Airflow Not Balanced:** The effects of poor air balancing can be determined using the same methods for estimating duct leakage but the ability to measure static pressure in ducts at any point allows the monitoring system to more accurately estimate the effects, especially in variable air volume (VAV) systems.
- **Improper Refrigerant Charge:** The specialized Sensor Interface Boards for chillers allows all the parameters required to detect an improper refrigerant charge to be gathered. Diagnostics to check for such problems have already been developed.
- **Dampers not working properly:** The Sensor Interface Boards for Air Handling units were specially designed to assist in determining whether the dampers in Air Handling Units are functioning correctly. Therefore this fault is directly detectable by the system.
- **Insufficient Evaporator Airflow:** The monitoring of the operating parameters on a chiller will allow the monitoring system to detect such conditions.
- **Improper Controls Setup / Commissioning:** The monitoring system provides a form of continuous commissioning and therefore will pick up any control or commissioning errors over time.
- **Control Component Failure or Degradation, Software Programming Errors or Improper Controls Hardware Installation:** The Monitoring system is completely independent of the control systems and therefore should not be affected by control failure, software issues or incorrect hardware installation and will detect any deviations from the designed operating conditions.
- **Air-Cooled Condenser Fouling:** The ability to monitor the pressure, temperature, currents and flow rates on an air-cooled chiller or package unit will allow such a problem to be detected.
- **Valve Leakage:** Valve leakage can be directly monitored though the use of flow sensors or determined via other parameters modelled by the system.

The designed system is therefore fully capable of monitoring the parameters required to prevent many of the building faults listed by the TIAX report. As a result, such a system should allow a building to improve the efficiency of its air conditioning plant, lower energy consumption and prolong the life of the equipment.

10.4 Successful Design of a Comprehensive Monitoring Framework

Although the system was designed specifically to monitor air conditioning installations, the nature in which it was designed allows it to be applied to a wide range of other applications. A comprehensive framework was first developed to acquire, process, and interpret sensor information. It has simply been customized for air conditioning installations.

Adapting the firmware for new Sensor Interface Board configurations requires changing only the Device-Specific layer of the framework. This results in the ability to rapidly adapt the framework to suit any large-scale industrial process.

10.5 The Proposed System Does Address the Barriers of Entry Facing Diagnostic Systems

The Proposed system was designed to address the shortcomings and barriers to entry that many diagnostics systems face. The TIAX report summarized in chapter 3.1 formed the basis of the requirements of the system and the majority of the concerns expressed in the report have been address.

The designed system is a low-cost alternative to the expensive building management technologies already available. It requires minimal expertise to install and maintain and does not interfere with existing control systems. It can also be retro-fitted to an existing plant at far less expense than other such technologies.

The low-cost of both a new installation and retro-fitting an existing installation acts to lower the required financial gains that the system needs to prove. It is also not vendor-specific and is therefore capable of monitoring equipment from a wide range of different manufacturers and suppliers.

These factors serve to complement the energy saving and maintenance benefits of the proposed system.

11 Bibliography

- [1]. **ASHRAE**. *2001 ASHREA Handbook - Fundamentals (SI Edition)*. s.l. : American Society of Heating, Refrigeration and Air-Conditioning Engineers, Inc, 2001.
- [2]. **Jones, W.P.** *Air Conditioning Engineering*. s.l. : Elsevier Butterworth-Heinemann, 2001.
- [3]. **Honeywell**. *Engineering Manual of Automatic Control for Commercial Air Conditioning*. s.l. : Minneapolis-Honeywell Regulator Company, 1958.
- [4]. Psychometrics on Wikipedia, the free Encyclopedia. [Online] [Cited: 7 March 2008.] <http://en.wikipedia.org/wiki/Psychometrics>.
- [5]. **Carrier Air Conditioning Company**. *Handbook of Air Conditioning System Design*. s.l. : McGraw-Hill , 1965.
- [6]. **Nall, D. H.** Looking across the water: Climate-adaptive buildings in the United States and Europe. *The Construction Specifier*. 2004, Vol. 57.
- [7]. **Trane**. *Trane Air Conditioning Manual*. s.l. : The Trane Company, 1965.
- [8]. **The Chartered Institute of Building Services Engineers**. *TM13 - Minimizing the Risk of Legionnaires' Disease*. London : s.n., 1987.
- [9]. **TIAX LLC**. *Energy Impact of Commercial Building Controls and Performance Diagnostics: Market Characterization, Energy Impact of Building Faults and Energy Savings Potential*. U.S. Department of Energy. 2005.
- [10]. **U.S. Department of Energy**. *Annual Energy Review 2006*. s.l. : Energy Information Administration, 2006.
- [11]. **Energy Information Administration**. *Annual Energy Outlook 2003 with Projections to 2025*. 2003. DOE/EIA-0383(2003).
- [12]. **Friedman, Hanna and Piette, Mary Ann**. *Comparative Guide to Emerging Diagnostic Tools for Large Commercial HVAC Systems*. s.l. : California Energy Commission, 2001.
- [13]. **Sellers, David A.** 21st-century HVAC controls: Progress and paradox. *Networked Controls*. July, 2003.
- [14]. **Brown, Thomas**. Fire Protection System Integration. *Security Technology & Design*. 1998, Vol. August.
- [15]. **United Nations Conference on Trade and Development**. *Information Economy Report 2007-2008*. 2007.
- [16]. **Alabandan, B. A. and Oyewo, O. A.** Temperature Variations within Wooden and Metal Grain Silos in the Tropics during Storage of Maize (Zea Mays). January 2005, 6.
- [17]. **Miyamoto, Hiroshi, et al.** Development of a New Seminested PCR Method for Detection of Legionella Species and Its Application to Surveillance of Legionellae in Hospital Cooling Tower Water. June 1997, Vol. 63, 7.
- [18]. **Olken, Frank, et al.** *Remote Building Monitoring and Control*. s.l. : Lawrence Berkeley National Laboratory, 1996.

- [19]. **Olken, Frank, et al.** *Object Lessons Learned from a Distributed System for Remote Building Monitoring and Operation*. s.l. : Lawrence Berkeley National Laboratory, 1999.
- [20]. **Ann, Piette Mary, et al.** *Model-based chiller energy tracking for performance assurance at a university building*. 1997.
- [21]. **Perez de Vinaspre, M, et al.** *Monitoring and Analysis of an Adsorption Air-Conditioning System*. 2004.
- [22]. **Hayter, S, Torcellini, P and Judkoff, R.** *Optimizing Building and HVAC systems*. s.l. : ASHRAE, 1999.
- [23]. **Gagliarducci, M, Lampasi, D.A. and Podesta, L.** *GSM-based monitoring and control of photovoltaic power generation*. s.l. : Department of Electrical Engineering, University of Rome, 2006.
- [24]. **Peersman, C., et al.** *The global system for mobile communications short message service*. s.l. : IEEE, 2000.
- [25]. **Kalaitzakis, Kostas, Koutroulis, Eftichios and Vlachos, Vassilios.** *Development of a data acquisition system for remote monitoring of renewable energy systems*. Crete : s.n., 2003.
- [26]. **Bushby, Steven T.** *BACnet: A Standard Communication Infrastructure for Intelligent Buildings. Automation in Construction*. 1997 .
- [27]. **Continental Automated Buildings Association for the Natural Research Council, Canada.** *Technology Roadmap for Intelligent Buildings*.
- [28]. **Sinclair, Ian R.** *Sensors and Transducers*. s.l. : Newnes, 2001.
- [29]. **Newnes.** *Sensor Technology Handbook*. [ed.] Jon S. Wilson. s.l. : Newnes, Elsevier, 2005.
- [30]. **Dallas Semiconductor.** DS18S20-PAR: 1-Wire Parasite-Power Digital Thermometer Datasheet. www.maxim-ic.com. [Online] 2007.
- [31]. —. Application Note: DS18B20-PAR/DS18S20-PAR/DS1822-PAR Advantages for Remote Temperature Sensing. *Dallas Semiconductor*. [Online] http://www.maxim-ic.com/appnotes.cfm/appnote_number/203.
- [32]. **Dallas Semiconductors.** Application Note - Interfacing the DS1822 1-Wire Temperature Sensor in a Microcontroller Environment. [Online] http://www.maxim-ic.com/appnotes.cfm/appnote_number/162.
- [33]. **Analog Devices.** AD536A: True RMS-to-DC Converter Datasheet. [Online] 2007. www.analog.com.
- [34]. **Atmel.** ADT45DB161 16-megabit 2.7V Only Serial Dataflash Datasheet. www.atmel.com. [Online] [Cited: 23 April 2007.]
- [35]. **Wavecom.** *Application Note - Q24NG Power On*. 2007.
- [36]. —. *AT Commands Interface Guide for 6.57 Release*. s.l. : Wavecom, 2006.
- [37]. **Predko, Myke.** *Programming and Customizing PICmicro Microcontrollers*. s.l. : McGraw Hill, 2002.
- [38]. **Axelsson, Jan.** *Serial Port Complete*. s.l. : Lakeview Research LLC, 2000.

- [39]. **USB-IF.** *Universal Serial Bus Specification Revision 2.0.* 2000.
- [40]. **Myers, J. and Rose, M.** *RFC 1939 - Post Office Protocol - Version 3.* March 1996.
- [41]. **Kurose, James F. K and Ross, Keith W.** *Computer Networking: A top-down approach featuring the Internet.* s.l. : Pearson Education, 2005.
- [42]. **IBM.** *TCP/IP Tutorial and Technical Overview.* s.l. : IBM Redbooks, 2006.
- [43]. **Network Working Group.** *RFC2616: Hypertext Transfer Protocol -- HTTP/1.1.* s.l. : The Internet Society, 1999.
- [44]. —. *RFC 2068 - Hypertext Transfer Protocol - HTTP/1.1.* s.l. : The Internet Society, 1999.
- [45]. —. *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types.* 1996.
- [46]. **S. Katipamula, S. Gaines.** *Characterization of Building Controls and Energy Efficiency Options Using Commercial Building Energy Consumption Survey.* 2003.
- [47]. **A Dexter, J Pakanen.** *Demonstrating Automated Fault Detection and Diagnosis Methods in Real Buildings.* s.l. : International Energy Agency.

Appendix A – List of Abbreviations

ADC	Analogue to Digital Converter
AHU	Air Handling Unit
ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
ASHRAE	American Society of Heating, Refrigeration and Air-Conditioning Engineers
ASP	Active Server Pages
BACnet	Building Automation and Control Networks
BCS	Building Control Systems
BMS	Building Management System
BTU	British Thermal Unit
COP	Coefficient of Performance
DB	Dry-Bulb
DCE	Data Communications Equipment
DNS	Domain Name System
DTE	Data Terminal Equipment
EDGE	Enhanced Data rates for GSM Evolution
EEPROM	Electrically Erasable Programmable Read Only Memory
EMCS	Energy Management and Control System
FCU	Fan Coil Unit
FTP	File Transfer Protocol
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communication
HTML	Hypertext Markup Language
HTTP	Hyper Text Transfer Protocol
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
LAN	Local Area Network
LCD	Liquid Crystal Display
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Messaging Service
NBR	National Building Regulations
PHP	PHP Hypertext Processor
PIN	Personal Identifier Number
POP	Post Office Protocol
RTC	Real Time Clock
SGSN	Serving GPRS Support Node
SiB	Sensor Interface Board
SIM	Subscriber Identity Module
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
USSD	Unstructured Supplementary Service Data
URL	Uniform Resource Locator
VAV	Variable Air Volume

VPN	Virtual Private Network
VRV	Variable Refrigeration Volume
WAP	Wireless Application Protocol
WB	Wet-Bulb
WML	Wireless Markup Language
XML	eXtensible Markup Language

Appendix B - Unit Conversions and Physical Constants

Pressure Conversions and Constants

Common Pressure Conversion Table

	Bar	PSI	cm H ₂ O	Inch H ₂ O	ft H ₂ O	kPa
Bar	1	14.50377	1019.74	401.474	33.4562	100.0
PSI	0.06894757	1	70.3089	27.6807	2.30673	6.894757
cm H ₂ O	0.000980638	0.0142230	1	0.393701	0.0328084	0.0980638
Inch H ₂ O	0.00249082	0.036126 3	2.54	1	0.0833334	0.249082
ft H ₂ O	0.0298898	0.433515	30.48	12.0213	1	2.98898
kPa	0.01	0.1450377	10.1974	4.01474	0.334562	1

1 kilopascal is equal to

- = 0.01 bar
- = 0.750061682704 centimeter mercury (0°C)
- = 10.1974428892 centimeter water (4°C)
- = 0.334887209988 foot water (60°F)
- = 0.29529987508 inch mercury (32°F)
- = 4.01864651985 inch water (60°F)
- = 0.0101971621298 kilogram/square centimeter
- = 101.971621298 kilogram/square meter
- = 0.000101971621298 kilogram/square millimeter
- = 7.50061682704 millimeter mercury (0°C)
- = 20.8854342331 pound/square foot
- = 0.14503773773 pound/square inch (PSI)
- = 0.0104427171166 ton/square foot
- = 7.2518868865E-5 ton/square inch
- = 7.50061682704 torr

1 PSI is equal to

- = 0.06804596391 Atmosphere (sea level)
- = 0.0689475729318 bar
- = 5.17149325716 centimeter mercury (0°C)
- = 70.3088937322 centimeter water (4°C)
- = 2.30896603345 foot water (60°F)
- = 2.03602096738 inch mercury (32°F)
- = 27.7075924015 inch water (60°F)
- = 0.070306957964 kilogram/square centimeter
- = 703.06957964 kilogram/square meter
- = 0.00070306957964 kilogram/square millimeter
- = 6.89475729318 kilonewton/square meter
- = 6.89475729318 kilopascal
- = 51.7149325716 millimeter mercury (0°C)
- = 6894.75729318 Pascal
- = 144 pound/square foot
- = 0.072 ton/square foot
- = 0.0005 ton/square inch
- = 51.7149325716 torr

1 Bar is equal to

- = 0.986923266716 Atmosphere

=	75.0061682704	centimeter mercury (0°C)
=	1019.74428892	centimeter water (4°C)
=	33.4887209988	foot water (60°F)
=	29.529987508	inch mercury (32°F)
=	401.864651985	inch water (60°F)
=	1.01971621298	kilogram/square centimeter
=	10197.1621298	kilogram/square meter
=	0.0101971621298	kilogram/square millimeter
=	100	Kilopascal
=	750.061682704	millimeter mercury (0°C)
=	100000	Pascal
=	2088.54342331	pound/square foot
=	14.503773773	pound/square inch
=	14.503773773	Psi
=	1.04427171166	ton/square foot
=	0.0072518868865	ton/square inch
=	750.061682704	torr

Pressure Variation at Different Altitudes

Altitude (ft)	Altitude (m)	Inch Hg	Pascal	PSI
0	0	29.92	101.2792	14.68931
1000	304.8	28.86	97.6911	14.1689
2000	609.6	27.82	94.1707	13.65831
3000	914.4	26.82	90.7857	13.16735
4000	1219.2	25.84	87.4684	12.68622
5000	1524	24.89	84.25265	12.21981
10000	3048	20.58	69.6633	10.10381
15000	4572	16.88	57.1388	8.287282
18000	5486.4	14.94	50.5719	7.334834

Pressure Constants

Standard Temperature: 0°C = 273.15 K

Standard Pressure: 101.3 kPa (1 atmosphere at sea level)

At Standard Temperature and Pressure, one mole of an ideal gas occupies 22.4 litres.

Power and Energy Conversions

Common Energy Conversions

	Kilojoules (kJ)	Watt Hours (Wh)	Kilowatt Hours (kWh)	BTU
Kilojoules (kJ)	1	0.2778	0.0002778	0.9485
Watt Hours (Wh)	3.6	1	0.0001	3.414
Kilowatt Hours (kWh)	3600	1000	1	3414
BTU	1.054	0.2929	0.0002929	1

Common Power Conversions

	Watts	Kilowatts	BTU/hr	Horsepower	Tons
1 BTU/s	1054	1.054	3600	1.414	0.3
1BTU/min	17.57	0.01757	60	0.02357	0.005
1 BTU/h	2.929	0.002929	1	0.003928	0.0008333
1 kW/s	36 000 000	36 000	122 900 000	48 280	10 240
1 kW/min	60 0000	600	2 049 000	804.6	170.7
1 kW/h	1000	1	3414	1.341	0.2845
1 J/s	1	0.0001	3.414	0.001341	0.0002845
1 J/min	0.01667	0.00001667	0.05691	0.00002235	0.000004742
1 J/h	0.0002778	2.778x10 ⁻⁷	0.0009484	3.725x10 ⁻⁷	7.904x10 ⁻⁸

1 calorie is equal to

- = 0.00396832071933 BTU
- = 2.61332001727E+19 electron volt
- = 3.08802520669 foot-pound
- = 4.1868 Joule
- = 4.1868 Newton-meter
- = 3.30693393277E-7 Refrigeration ton-hour
- = 0.001163 watt-hour
- = 4.1868 watt-second

1 calorie/s is equal to

- = 14.2859545896 BTU/hour
- = 0.23809924316 BTU/minute
- = 0.00396832071933 BTU/second
- = 0.00561459128474 Horsepower
- = 15072.48 joule/hour
- = 251.208 joule/minute
- = 4.1868 joule/second
- = 0.0041868 Kilowatt
- = 0.0011904962158 Refrigeration ton
- = 4.1868 watt

Appendix C – Glossary

Air Cooled Condenser	A condenser which uses a fan to cool the coil using outside air.
Air Handling Unit	An Air Handling Unit is a device which moves and conditions air into the ductwork of a building.
BTU (British Thermal Unit)	A BTU is a standard unit of measurement defined as the amount of heat required to raise the temperature of one pound of water by one degree Fahrenheit.
Centrifugal Compressor:	A dynamic compression device in which a rotating impeller is used to compress a refrigerant vapor.
Condenser Coil	A coil, consisting of a network of tubes, through which refrigerant is pumped in order to exchange heat with either water or air. Condensing coils are primarily used to remove heat from a refrigeration system.
Cooling tower	A cooling tower is used to lower the temperature of water using evaporation. It is primarily used to lower the temperature of water heated by a condenser coil.
Damper	A damper is a mechanical device found in ductwork which is used to control the flow of air in a duct. Dampers can either be manually operated or electrically controlled.
Dehumidification	The process of removing water vapor from air by cooling air below its dew point or via chemical means.
Device-Based Addressing	One of two addressing modes used by the monitoring hardware to address Sensor Interface Boards. Device – Based Addressing assigns a unique address to each Sensor Interface Board present in the system.
Dew-Point Temperature	The dew-point temperature is the specific temperature of air at which moisture vapor in the air would begin to condense.
Direct Expansion Systems:	A direct expansion system is an air conditioning system in which air is cooled directly from a refrigerant gas, as opposed to using a distributed water system to distribute cooling.
DNS	DNS is a technology which provides a method to resolve hostnames into IP addresses. This allows an internet site to be accessed using a readable alias as opposed to a static IP address
Dry-Bulb Temperature	The Dry-bulb temperature is the sensible heat of air and is the value commonly measured using a standard thermometer.
Enthalpy	Enthalpy is a measurement of the total heat content of air, taking into account both the sensible and latent heat. It is measured in kilojoules of energy per kilogram.
Evaporator Coil	A coil, consisting of a network of tubes, through which refrigerant is pumped in order to exchange heat with either water or air. Evaporator coils are commonly

Fan-Coil Unit	used to absorb heat into a refrigeration system. A Fan-Coil unit is a small air handling unit which uses chilled water to provide cooling. Such units may also contain an electric heater in order to provide heating as well.
Filter	A filter, in air conditioning terms, is a device installed in the flow of air to remove any particles or dirt.
Freon	Freon is a term used to refer to a group of partially or completely halogenated simple hydrocarbons containing fluorine, chlorine or bromine and is commonly used as a refrigerant.
GPRS	GPRS is a GSM technology which allows the transmission of packet data over GSM networks. It is commonly used to provide internet access to mobile devices.
Heat Exchanger	A device responsible for transferring heat from one medium to another.
Host Controller	The Host Controller is the component in the proposed monitoring system responsible for collecting sensor data from Sensor Interface Boards and uploading it to the diagnostics server.
HVAC	Heating, Ventilation and Air Conditioning.
Infiltration	Heat from the movement of air into a space through cracks and small spaces in the physical construction.
Latent Heat	The heat energy required to change the state of a substance without changing its temperature
MySQL	MySQL is an open-source relational database engine which is both extremely efficient and scalable.
Network Discovery Process	The Network Discovery Process is a technique described in this thesis to assign addresses to all the devices connected to a Host Controller.
Packaged Unit	A single device containing all the components necessary to provide air conditioning.
PHP	PHP is a web-scripting language which is commonly used on web-servers to provide dynamic content.
Relative Humidity	Relative humidity is analogous to the ratio of water vapor to moist air in a given sample of air. It is a dimensionless quantity.
Sensor Interface Board	Sensor Interface Boards are a hardware component in the monitoring and diagnostics system presented in this thesis. These devices are responsible for acquiring data from sensors and making them available to the monitoring equipment using the proprietary protocol described in this document.
Sensor-Based Addressing	One of two addressing modes used by the monitoring hardware to address Sensor Interface Boards. Sensor-Based Addressing assigns each sensor in the network a unique address.
Ventilation	The process of supplying or extracting air from an enclosed space.
Water-Cooled Condenser	A condenser which uses a separate water-loop in order

Wet-Bulb Temperature

to cool the coil. This water is then cooled using a cooling tower.

The temperature of air if it were cooled by water evaporating into it, until the air is saturated.

Appendix D – Additional Graphs for April 2008

This appendix contains a selection of graphs for the month of April 2008 in which a new graphing engine was developed for the system. This system allows virtually any combination of parameters to be graphed on a single plot, across varying time-scales. Reproduced below are a number of graphs for various days in the month.

The graphs are calculated and created dynamically from a web-server and reflect data actually stored in the database. The data come from the same installation that was used to gather the data for the graphs in section 9.

University of Cape Town

