

Compact Information Technology Enabled Systems

For

Intelligent Process Monitoring

By

Qaisar Ahsan

May 2006

Intelligent Process Monitoring & Management Centre

Cardiff School of Engineering

Cardiff University

UMI Number: U584875

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI U584875

Published by ProQuest LLC 2013. Copyright in the Dissertation held by the Author.
Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against
unauthorized copying under Title 17, United States Code.



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed Qaiser Akbar (candidate)

Date 25/5/2006

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated.

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed Qaiser Akbar (candidate)

Date 25/5/2006

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and inter-library loan, and for the title and summary to be available to outside organisation.

Signed Qaiser Akbar (candidate)

Date 25/5/2006

Acknowledgements

Praise be to GOD ALMIGHTY, the Creator and Sustainer, who made me capable enough to complete this thesis. I am also thankful to:

My supervisors, Dr. Roger I. Grosvenor and Mr. Paul W. Prickett for their knowledge, expert guidance and patience whilst assisting me throughout the research.

Mr. Waseem Amer, Miss Asma Jalal, and Dr. Alun Jennings for their companionship, support, and wealth of knowledge during the research work.

Ministry of Science & Technology, Govt. of Pakistan for their financial support, without which I would not have been able to engage this course.

My family, for its continued support allowing me to further my life long education.

ABSTRACT

The use of computers in industrial process applications is ever-increasing. Initially used to provide help to the machine operator, their application has evolved through automatic process control to monitoring of process health and performance. The latter, together with the quality control of the end product directly affect plant economics and ultimately the financial viability of the company. The research reported in this thesis is a contribution towards providing a cost-effective method of calculating a measure of the current health of a process and predicting any maintenance issues that may arise in the near future. Embedded systems are utilised and the monitoring system is designed to work automatically with a minimal input from the operator. This eliminates the need for peripherals such as keyboards, mice, and monitors thus reducing the overall system price and footprint. User interfaces are provided via the Internet and mobile phones giving remote access to multiple users. Single chip microcontrollers are at the heart of the embedded system rather than microprocessors, thereby reducing the relative system cost and size at the expense of localised processing power. The microcontrollers are distributed in a hierarchical network to attain the required processing power whilst minimising data storage and communications and to improve signal-to-noise ratios. The Controller Area Network (CAN) bus was selected, and used for the inter-microcontroller communications, for its robust performance in noisy environments.

In the developed system architecture, each microcontroller node acquires one of the required process sensor signals and applies initial signal processing. A novel sweeping filter technique is developed to perform frequency analysis using the microcontrollers. The processed data from all nodes are then combined using situation-based criteria to reach conclusions often not evident from single sensor data. The Internet-based system is provided with the capability to upload any monitoring software or updates. Plug & play capability of the monitoring nodes is also provided so that the system can be seamlessly adapted to new or changed applications. The design and development of the system are detailed along with its deployment on various applications. Fault detection, isolation, and prediction were achieved on batch and continuous processes. A machine tool application proved the frequency analysis and network traffic reduction capabilities. On-line monitoring of an industrial valve was also performed.

CONTENTS

DECLARATION	i
ACKNOWLEDGEMENTS	ii
ABSTRACT	iii
CONTENTS	iv
ACRONYMS	viii
LIST OF FIGURES	xiii
LIST OF TABLES	xix
CHAPTER 1. INTRODUCTION	1
CHAPTER 2. RESEARCH MOTIVATION	5
CHAPTER 3. LITERATURE REVIEW	10
3.1 Introduction	10
3.2 General condition monitoring	11
3.3 Techniques & their appropriate application	14
3.4 Monitoring with controller signal	25
3.5 Distributed systems for monitoring	34
3.6 Compact monitoring systems	42
3.7 Examples of web-enabled monitoring	50
3.8 Commercial services	55
3.9 Summary	58
CHAPTER 4. SIGNAL ACQUISITION AND ANALYSIS	59
4.1 Monitoring system overview	59
4.2 Processing element selection	62
4.3 PIC microcontrollers	63
4.3.1 PIC 18F458 microcontroller	64
4.4 Signal acquisition	66
4.4.1 One analogue signal per MCU	66

4.4.2	Digital signal acquisition	66
4.4.3	Analogue signal acquisition	68
4.4.4	Eight bit ADC results	69
4.4.5	Data storage	70
4.5	Signal analysis	71
4.5.1	Time domain analysis	71
4.5.2	Frequency domain analysis	72
4.5.3	Circuit design	79
4.6	Summary	81
CHAPTER 5. DISTRIBUTED MONITORING SYSTEM		82
5.1	Fieldbus	83
5.1.1	Controller area network	83
5.1.2	CAN in PIC 18F458	87
5.2	Synchronization and user interface	88
5.2.1	80C390 microcontroller as SUIN	89
5.2.2	Tiny InterNet Interface (TINI)	91
5.2.3	Brief introduction to protocols	93
5.2.4	Human interface	94
5.2.5	Mobile phone interface	96
5.3	Fault detection and isolation	96
5.4	CAN bus messages	98
5.4.1	Node identification	99
5.4.2	Message priority	101
5.4.3	Power-up sequence	101
5.4.4	Process monitoring	103
5.5	Plug & play	104
5.6	Software models for FENs and SUIN	105
5.6.1	Data acquisition mode	106
5.6.2	Monitoring mode	107
5.6.3	Software update mode	109
5.7	Network traffic reduction	111
5.9	Fault reporting by SMS	113
5.8	Summary	114

CHAPTER 6. PIPE BLOCKAGE DETECTION	116
6.1 Bytronic process rig	116
6.2 Batch process	117
6.3 Process monitoring	118
6.4 Fault simulation	120
6.5 Monitoring decisions	126
6.6 Monitoring results	132
6.7 Summary	137
CHAPTER 7. MULTIPLE FAULTS ISOLATION	140
7.1 Fault isolation	140
7.1.1 Leakage fault simulation	141
7.1.2 Leakage and blockage faults isolation	144
7.2 Multi-loop process monitoring	147
7.2.1 Continuous process monitoring	147
7.2.2 Batch process monitoring	151
7.2.3 Combined loop monitoring	155
7.3 Summary	156
CHAPTER 8. TOOTH BREAKAGE DETECTION	157
8.1 Tooth breakage detection theory	157
8.2 Sweeping filter application	159
8.2.1 Monitoring signals	159
8.2.2 Possible approaches	161
8.2.3 Threshold establishment	163
8.3 Monitoring decisions	166
8.4 Summary	168
CHAPTER 9. AIR FLOW PROCESS MONITORING	169
9.1 Process rig	169
9.2 Air flow process	171
9.3 Monitoring signals	173
9.4 Diaphragm condition monitoring	174
9.5 Pipe blockage monitoring	177

9.6	Fault isolation	180
9.7	Summary	182
CHAPTER 10.DISCUSSION		184
CHAPTER 11. CONCLUSIONS & FUTURE WORK		193
11.1	Conclusions	193
11.2	Future work	194
REFERENCES		196
APPENDIX A. Transducers and sensing techniques		205
APPENDIX B. Microcontrollers' details		206
APPENDIX C. System related details		217
APPENDIX D. List of Publications from this work		231

ACRONYMS

ACF	Autocovariance Function
ADC	Analogue to Digital Converters
AE	Acoustic Emission
AES	Advanced Encryption Standard
AMS	Asset Management Solutions
ANN	Artificial Neural Network
API	Application Programmer Interface
AR	AutoRegressive
ARMA	AutoRegressive Moving Average
ASK	Amplitude Shift Keying
BDD	Binary Decision Diagrams
BP	BackPropagation
CAN	Controller Area Network
CCPN	Complex Choice Petri Net
CGI	Common Gateway Interface
CLPA	Closed Loop Performance Assessment
CLPM	Closed Loop Performance Monitoring
CMA	Control/Monitoring Agents
CN	Control Nodes
CPU	Central Processing Unit
CRC	Cyclic Redundancy Code
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
DAMADICS	Development and Application of Methods for Actuator Diagnosis in Industrial Control Systems
DAME	Distributed Aircraft Maintenance Environment
DDG	Data Dependence Graph
DLC	Data Length Code
DMC	Distributed Measurement and Control

DSP	Digital Signal Processor
DTMF	Dual Tone Multiple Frequency
DWT	Discrete Wavelet Transform
EDF	Earliest Deadline First
EOF	End of Frame
EP	Embedded Processor
ETA	Event Tree Analysis
FDD	Fault Detection and Diagnosis
FDI	Fault Detection and Isolation
FEN	Front-End Node
FFT	Fast Fourier Transform
FIFO	First-In First-Out
FIR	Finite Impulse Response
FN	Fieldbus Nodes
FS	fieldbus server
FTA	Fault tree analysis
FTP	File Transfer Protocol
GPRS	Global Packet Radio Service
GSM	Global System for Mobile Communication
GSS	Ground Support System
HART	Highway Addressable Remote Transducer
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transmission Protocol Secure
HUMS	Health and Usage Monitoring Systems
I ² C	Inter-Integrated Circuit
IAE	Integrated Accumulated Error
ICD	In-Circuit Debugger
ICSP	InCircuit Serial Program

IDE	Integrated Development Environment
IFS	Intermission Frame Space
IIR	Infinite Impulse Response
IMP	Integrated Mail Processors
IP	Internet Protocol
IPMM	Intelligent Process Monitoring and Management
ISAPI	Internet Server Application Programming Interface
ISP	Internet Service Provider
ISR	Interrupt Service Routine
JDBC	Java DataBase Connectivity
JVM	Java Virtual Machine
KBS	Knowledge Based System
KPI	Key Performance Indicators
LAMDA	Learning Algorithm for Multivariate Data Analysis
LAN	Local Area Network
M2M	Machine To Mobile, Mobile To Machine
MAC	Media Access Control
MAC	Multiply-and-ACcumulate
MC	Message Centre
MCM	Motor Condition Monitor
MCU	Microcontroller Unit
MIPS	Million Instructions Per Second
MSPC	Multivariate Statistical Process Control
MSPM	Multivariate Statistical Process Monitoring
MSSP	Master Synchronous Serial Port
MVC	Minimum Variance Control
NCAP	Network Capable Application Processor
NN	Neural Network

ODBC	Open DataBase Connectivity
OLPI	Overall Loop Performance Index
OSI	Open System Interconnection
PC	Personal Computer
PCA	Principal Component Analysis
PCB	Printed Circuit Board
PDA	Personal Digital Assistant
PDU	Protocol Data Unit
PIC	Peripheral Interface Controller
PLC	Programmable Logic Controllers
PLS	Partial Least Squares
PSN	Programming and Supervision Node
PWM	Pulse Width Modulation
Q	Quality factor
QPT	Qualitative Process Theory
QSA	Qualitative Shape Analysis
QSIM	Qualitative SIMulation
RMU	Remote Measurement Unit
RISC	Reduced Instruction Set Computer
RTOS	Real-Time Operating Systems
S&H	Sample & Hold
SBC	Single Board Computer
SBR	Sequencing Batch Reactor
SCADA	Supervisory Control And Data Acquisition
SDG	Signed Digraphs
SEVA	Self Validating
SFR	Special Function Register
SIM	Subscriber Identification Module
SIMM	Single Inline Memory Module
SISO	Single-Input Single-Output

SME	Small to Medium Enterprise
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SNR	Signal to Noise Ratio
SOF	Start of Frame
SOFM	Self-Organizing Feature Map
SPC	Statistical Process Control
SPI	Serial Peripheral Interface
SQL	Structured Query Logic
SSI	Server Side Includes
STIM	Smart Transducer Interface Module
SUIN	Synchronizing & User Interface Node
TCG	Temporal Causal Graphs
TCP	Transport Control Protocol
TINI	Tiny InterNet Interface
TPM	Total Productive Maintenance
TTY	TeleTYpewriter
UML	Unified Modelling Language
USART	Universal Synchronous Asynchronous Receiver Transmitter
USB	Universal Serial Bus
VPN	Virtual Private Network
WAP	Wireless Application Protocol
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access
XML	eXtensible Markup Language

LIST OF FIGURES

Figure 3.1	Validation as a combination of technologies	27
Figure 3.2	Typical behavior of a control valve with static friction	29
Figure 3.3	Smart sensor	35
Figure 3.4	Modular system configuration with standard I/O	38
Figure 3.5	Controller performance assessment scheme	40
Figure 3.6	System architecture	44
Figure 3.7	Proposed architecture for system-wide FDI	45
Figure 4.1	Hardware architecture of proposed distributed monitoring system	61
Figure 4.2	Computing hierarchy of proposed distributed monitoring system	61
Figure 4.3	Basic parts of an ADC	68
Figure 4.4	Block diagram of sweeping filter system	74
Figure 4.5	MAX264 filter block diagram	75
Figure 4.6	Detection of 100mV 20Hz sine wave	77
Figure 4.7	Higher amplitude sine wave detection	77

Figure 4.8	Detection of 100mV 20Hz sine wave with Q values of 8, 16, and 64	78
Figure 4.9	Harmonic detection for Q 8, 16, and 64	78
Figure 4.10	Multiple frequency detection	79
Figure 4.11	Time based analysis circuit	80
Figure 4.12	Frequency based analysis circuit	80
Figure 5.1	The developed monitoring system with hardware layers	83
Figure 5.2	CAN Bus	84
Figure 5.3	Extended CAN data message bits	86
Figure 5.4	CAN protocol implementation in PIC 18F458	88
Figure 5.5	CAN message identifier	98
Figure 5.6	Message from node 1 to node 2	100
Figure 5.7	Broadcast message from node 1	100
Figure 5.8	Some example identifiers of CAN messages	102
Figure 5.9	CAN message sequence at system power-up	103
Figure 5.10	CAN message sequence after FEN time-out	103
Figure 5.11	Messages during process monitoring	104

Figure 5.12	Message sequence when a new FEN boots-up	105
Figure 5.13	FEN software model with data acquisition mode elaborated	106
Figure 5.14	SUIN software model with data acquisition mode elaborated	107
Figure 5.15	SUIN software model for Internet access to data	107
Figure 5.16	FEN software model with monitoring mode elaborated	108
Figure 5.17	SUIN software model with monitoring mode elaborated	109
Figure 5.18	FEN software model with software update mode elaborated	110
Figure 5.19	SUIN software model with software update mode elaborated	111
Figure 5.20	Hardware arrangements for SMS generation	114
Figure 6.1	Bytronic process rig	117
Figure 6.2	Control and monitoring signals in batch process	118
Figure 6.3	Typical normal condition signals	120
Figure 6.4	Typical control signals (power) for partial blockages	121

Figure 6.5	Typical tank level signals for partial blockages	122
Figure 6.6	Typical flow rate signals for partial blockages	123
Figure 6.7	Ripples effect of blockage	125
Figure 6.8	Turbulence effect of blockage	125
Figure 6.9	Power running sum values for normal and 40% blockage cases	128
Figure 6.10	Power running sum values at 30 sec time	128
Figure 6.11	Flow and level running sum values at 30 sec time	129
Figure 6.12	Threshold determination for power FEN	130
Figure 6.13	Blockage extent and its detectability with selected thresholds	131
Figure 6.14	Threshold determination for level FEN	133
Figure 6.15	Threshold determination for flow FEN	134
Figure 6.16	Monitoring results on web page	138
Figure 7.1	Leakage and blockage faults arrangement in batch process	141
Figure 7.2	Signals for leakage fault only	142
Figure 7.3	Enlarged flow and level signals	142

Figure 7.4	Power signal in normal and leakage conditions	143
Figure 7.5	Fault isolation processing	146
Figure 7.6	Continuous process	148
Figure 7.7	Fault detection procedure in continuous loop	149
Figure 7.8	SUIN reporting process condition	150
Figure 7.9	Continuous process monitoring signals	150
Figure 7.10	Partial availability	151
Figure 7.11	Decision making with partial availability	152
Figure 7.12	Detection of unexpected process conditions	152
Figure 7.13	Batch process	153
Figure 7.14	Level signal shows delay in batch completion for leakage	154
Figure 7.15	SUIN reporting process condition	155
Figure 8.1	Frequency spectrum calculated with FFT	158
Figure 8.2	R current profile for X-axis drive signal with new and broken tooth cutters	160
Figure 8.3	R currents during cutting in X-axes	161
Figure 8.4	Varying signal strength for new cutter	162

Figure 8.5	Tool rotation frequency (8.33Hz) for R and S signals at different times	164
Figure 8.6	Tooth passing frequency (25Hz) for R and S signals at different times	165
Figure 8.7	Frequency profiles for various depths of cut	167
Figure 9.1	Process rig	170
Figure 9.2	Process rig working schematic	170
Figure 9.3	Linearization of control command	172
Figure 9.4	Normal condition signals	173
Figure 9.5	Monitoring signals for normal process	176
Figure 9.6	Chamber pressure Signals for various diaphragm conditions	176
Figure 9.7	Developed circuit boards	177
Figure 9.8	Signals with outlet pipe pressure sensor	178
Figure 9.9	Signals with partial pipe blockage (flow sensor)	179
Figure 9.10	Developed decision making logic	180
Figure 9.11	Signals with partial pipe blockage fault	181

LIST OF TABLES

Table 3.1	Comparison of diagnostic techniques	24
Table 4.1	8-bit PIC microcontroller families' characteristics	64
Table 4.2	Achieved timings for various time domain analysis techniques	73
Table 5.1	Comparison of TINI with other Internet enabled embedded devices	92
Table 5.2	Protocols and acronyms	94
Table 5.3	List of used messages types	99
Table 6.1	Signal description for process variables	117
Table 6.2	Monitoring node identification numbers	119
Table 6.3	Further signal conditioning for MCU interfacing	119
Table 6.4	Mean values of acquired signals	127
Table 6.5	Variance values of acquired signals	127
Table 6.6	SUIN rule base	135
Table 6.7	SUIN monitoring decisions	135

Table 6.8	Blockage level detection with FEN processing and SUIN integration	136
Table 7.1	Statistical measurement for monitored signals	143
Table 7.2	Running sums for fault isolation	145
Table 7.3	Running sum values for level signal	154
Table 9.1	Control signal generation	172
Table 9.2	Monitoring signals	173
Table 9.3	SUIN decision table	182

INTRODUCTION

The ever-increasing and market led competition for better quality and lower prices is putting an enormous pressure on the manufacturing industry. The industry aim is to run process plants at their optimum level of performance to ensure the sustained quality of the end product. This is often hand-in-hand with attempts to reduce maintenance and running costs. These two contradictory demands may put a manufacturing operation in jeopardy. Researchers are increasingly using computers and information technology to address these problems. However, the complexity and scale of modern industrial plants often still makes it impossible or impractical to check the health of every component in a plant. Time-based maintenance is often employed rather than the more economic condition-based maintenance. There is a great need for automatic systems that monitor the plant and let the maintenance staff know when a problem is detected.

It may still be a very difficult task for maintenance staff to diagnose the real cause of the problem once an automatic system detects it. Monitoring systems are required that not only report a performance degradation but can also direct the maintenance staff towards a specific remedial action. Ideally, the automatic system will provide a prediction of each component's failure so that optimally timed maintenance activity can be planned whilst the plant continues to run efficiently to produce quality products all the time. Such a system would need to acquire signals from all the plant components with respect to all of their variables such as temperature, pressure, demand, etc. These signals would be considered relative to the particular specifications of the component and the working environmental conditions. The massive computational requirement is still impractical, even with current technology and computational power. The amount of knowledge (or comprehensive algorithms) required to interpret all this data may also not be available. This situation justifies the continuing research activities in the area.

The need for process monitoring stems from economic benefits and any expensive solution would be counter-productive. Low-cost monitoring solutions are therefore required that can be achieved with the current technology. The approach taken in this research is to reduce the number of additional sensors and to estimate the process and it's

component's status largely from the signals already available in the system. This reduces the cost of additional sensors and associated items such as cabling. Signal interfacing and input/output capabilities are required to connect the field signals with the digital processing system. The modern generation of microcontrollers were deemed to be useful devices in this context. A microcontroller is sometimes described as a 'single chip computer' as it contains on-chip memory and peripheral devices such as timers, counters and data ports. Communication modes are also supported so that the microcontroller may exchange information with its surrounding world. The microcontrollers, however, do not have the high processing power associated with today's high-end microprocessors. The delicate balance between the processing capability and the cost is investigated in this research.

The last decade has seen significant improvements in telecommunication facilities. An important development in this respect is the evolution of the Internet. It is an excellent medium for engineers to employ for remote monitoring applications. A monitoring system should ideally have Internet connectivity whereby it can provide remote access to the results. The monitoring system developed in the current work provides these facilities, using commonly available software.

The data from a process under continuous monitoring may become huge, especially if it contains analogue signals sampled at high rate. There may be several variables of interest from a maintenance point of view and putting all of the data for all of the variables on the Internet may not be feasible. The storage and communication of data is thus a consideration of paramount importance. There are technical constraints on the bandwidth of communication channels and large scale data transfers may have a financial impact for small companies. It was therefore considered not to be a practical solution to transfer all the data from a remote plant on the Internet for a maintenance engineer to examine. Rather, it was judged that the data acquisition system should provide some pre-processing and provide only the monitoring system results on the Internet rather than raw data. Raw data may be communicated remotely, on request, only if the monitoring system detects an abnormal process condition and is unable to diagnose it from its existing capabilities. Such data would then be provided to a higher tier of the monitoring system, or for expert analysis at some other location. The research motivation for the current study is more fully reported in chapter 2.

A literature review is provided in chapter 3, encompassing work reported towards various aspects of condition monitoring. Many approaches and techniques have been researched and are considered. The review also discusses various application areas. It summarises and introduces the numerous techniques developed and reports on some commercial applications and devices.

Keeping in mind the requirements for the various monitoring methods proposed within the reviewed research, a compact and hierarchical monitoring architecture is proposed in chapter 4. The suitability of embedded systems in process monitoring is discussed with currently available technologies. A distributed network of 8-bit microcontrollers was deemed appropriate and reasons for this selection are described. The acquisition of signals and their analysis in time and frequency domains are provided. A new frequency analysis technique, designated as the sweeping filter technique, is introduced. This enables 8-bit microcontrollers to perform frequency analysis in real-time.

The interconnection between various nodes in the proposed monitoring system is described in chapter 5. A Controller Area Network (CAN) bus is used for such connections and the reasons for this selection are detailed. A specialised node provides system synchronisation and the user interface via the Internet and mobile phone connectivity. Communication messages on this peer to peer network are detailed in this chapter and various software modes are explained. The chapter also explains the implementation of a plug and play functionality in the monitoring system. Measures are also taken to reduce network traffic while integrating information produced at various acquisition nodes. Overall this chapter gives a complete blue-print of the proposed system.

Chapter 6 introduces the deployment and testing of the proposed system on an application. Partial pipe blockage and tank leakage faults were introduced in a batch process to analyse the monitoring system's performance for fault detection. The chapter provides the implementation details of the work, the node and hierarchy details and the experimentation performed to confirm the proper functionality of the system. The detection and isolation of multiple faults in a process is elaborated in chapter 7. Data gathered by the monitoring system was analysed by the author and different symptoms

were identified for different faults. The monitoring system then used this knowledge, in its monitoring mode, and successfully isolated the faults. Multiple faults were also isolated successfully in a multi-loop process where one loop was dependant upon the other one. Details of locating different fault symptoms with the proposed monitoring system are provided in this chapter along with the achieved results.

An analysis based on time series data was used in the fault detection and isolation in the above mentioned processes. Chapter 8 provides an example of the deployment of the monitoring system on a different type of application. Its performance in the frequency domain was evaluated on machine-tool signals. Spectral differences between a new cutter and a broken tooth cutter in machine tool signals were already known from previous research. The monitoring system was deployed to detect such spectral differences using a novel sweeping filter technique and its details are provided in this chapter.

Chapter 9 explains the monitoring of an air-flow process involving an industrial pneumatic valve. Such valves are widely used in industry for automatic process control. The monitoring system successfully detected and isolated multiple faults introduced in the process such as diaphragm condition deterioration and partial blockage of air flow path. Finally, chapter 10 discusses the results obtained from the various implementations of the monitoring system and chapter 11 draws conclusions and sketches the roadmap for future research.

RESEARCH MOTIVATION

Industrial automation has been on a rapid rise in recent years. The industry is in search of optimum profitability and has aspired to near perfect manufacturing in this quest. Manufacturing equipment is produced to a very high standard and new process plant (or an entire factory) starts shipping quality products once it is properly commissioned. To maintain an on-going high level of quality is, however, an altogether different issue. Various components of plant/process experience the unavoidable wear and tear through daily usage. The mechanical friction, electronic aging, rusting, corrosion, variations in temperature and pressure, etc. affects the production equipment and it loses its initial performance capability over time.

Replacement of components is often undertaken on a time based schedule where components are expected to work above a threshold quality level for a certain period of time. It is difficult to precisely predict this life span of a component as it depends on its use as well as the production quality level of its manufacturing plant. Various components from same production batch would have differing life spans in practice. Precise knowledge of the conditions that a part experiences is required to reliably predict the failure time.

The economic impact of being able to obtain accurate knowledge about the health of the process and machines is vast. In particular, for the process industries, Trenchard et al (2002) mention that only one third of surveyed control loops are performing satisfactorily. Horch (2000) states that one of the most important problems with process industry control loops is the widespread presence of oscillations. Hagglund (1995) observes that the main reason for these oscillations is the bad health of the actuators in the loop. Annual losses due to such undetected problems may rise to millions. Trenchard et al (2002) provided some quantitative values, for example for an industry where \$100K was saved by detecting several valves that did not require scheduled maintenance. Another company claimed to improve plant run-time by 20% due to performance improvement obtained through efficient monitoring (Matrikon, ProcessDoctor web site). Venkatasubramanian et al (2003) state that the annual losses due to the lack of condition-

based maintenance and the resultant safety hazards may reach \$20 billion for the American petrochemical industry. Similar losses in the UK range up to \$27 billion per year (Venkatasubramanian et al, 2003). It has been claimed that improvements of 20% to 30% in process efficiency have been achieved using intelligent condition-based monitoring (Emerson process management, customer proven web site).

It is quite common that field visits by maintenance staff determine that the component is healthy and does not require any maintenance. Hartley (2002) recalls that 63% of the field visits to transmitters in a large chemical company were routine checks where no problems were found. A typical field visit costs around \$300 and such useless trips can actually out-cost the capital cost of the component. The knowledge of a healthy component is therefore as important as that of a faulty one. Automation World (web site) indicates that the cost for removing a valve from a process is \$2000 to \$3000 no matter whether the valve is good or bad.

Considering the size and complexity of a modern process plant, it is very difficult for maintenance staff to manually check the condition of all components. Data acquisition systems are used to acquire the data for the variables of interest and typically data trends are able to be viewed by the control room staff. It is however very difficult to locate a fault merely by looking at the data trends. Expert knowledge is required for this analysis, which is usually not at hand because of the shortage of engineers (Schafer and Cinar, 2004). Venkatasubramanian et al (2003) state that because of the complexity of modern process plants, it is not always possible for operators to provide a proper response to every fault condition and about 70% of the industrial accidents are caused by human errors in these circumstances. Thus, they identified the need to develop automated fault detection and diagnosis systems to tackle this problem. An automated system is required to provide monitoring results, thereby reducing and ideally minimising the load on the engineers. A detailed review of the research in the development of automatic monitoring systems is provided in chapter 3. In spite of all of this research, the development of effective and automated diagnostics is still considered, by Harris (2004) for example, as an unresolved technical challenge, especially for multivariate systems.

Potentially this means that a continuous monitoring system is needed, measuring all the process and environment variables all of the time. This is a gigantic task even with the

latest state-of-the-art computers. Logistically, sensors are needed to detect various process variables, along with a cabling network to connect the sensor signals to the processing computers. The situation thus provides scope for the development of a monitoring system that could provide the current process health status with few additional sensors. This system would comprehend the holistic situation from only the key parameters and would be able to match them with fault symptoms. A departure from the nominal behaviour in a signal indicates a performance degradation which would be investigated by the monitoring system towards determining a root cause. Such a system can provide the information about the proper time to replace a plant component. The system should identify the faulty sub-section in the plant and may ideally provide information about each component.

A practical solution to this requirement is a system that can track the signals from a few available sensors in the process and evaluate the other process conditions indirectly. If such a system can be developed, only then it is possible to reliably let a component work in the process plant, or replace it with full confidence that a replacement was really required. Such a system is feasible in economic terms only if the money spent is worth the saving it provides; or if it results in a competitive advantage in the market by shipping better quality products all the time.

A significant amount of research has been reported but industry is still facing numerous problems in achieving comprehensive process monitoring. The detection of a fault is still an issue, especially for soft faults where the process is still likely to be running but with degraded performance levels. Finding the root cause of the problem is also an issue. The availability of signals, mounting and connecting sensors in the plant, the hardware architecture of the monitoring system, etc. are all issues that need to be considered. There is a fine compromise between missing a fault condition and generating false alarms. Clarke (1999) identified that between half and three-quarters of shut-down time is due to false alarms caused by lack of confidence of the process operatives in the measurement information provided. Many researchers have tackled these issues in the local sense but have not yet established comprehensive systems. The few systems that claim to provide efficient solutions are often too costly to be widely deployed in industry on a large scale. Big multinationals may afford such complicated monitoring solutions but it is almost impossible for small to medium enterprises (SMEs) to install these expensive systems.

The Mattec Corporation (Mattec corporation THE-MAN-A-ger web site), for example, claims to provide a low-cost solution where the basic unit price is about \$15000. Such solutions are therefore not commonly adopted in industry.

Another issue is the generality of the monitoring system. It is difficult to provide a generic solution because each process plant is essentially a unique one. Similar process plants may be using different types of sensors to measure the same variables. Each process may have different set points and thresholds according to the requirements of that particular plant. A monitoring system should therefore be able to adapt to such plant related changes easily. A generic solution can be taken by any company as the company is still not bound to any one service provider. Paulonis and Cox (2003) for example, state that lack of generality was an important reason for their company's decision to develop its own monitoring system rather than going for a commercially available one.

The common availability of the Internet these days makes it sensible that it should be exploited for remote monitoring. Eisenreich and Demuth (2003) urge the use of the Internet as an easily available communication medium predicting a trend towards Internet appliances, i.e. electronic devices connected to the Internet. Potentially then the process manager can check the plant performance from anywhere in the world and take appropriate management decisions. Decisions do not remain pending only because the manager is not in the office. It is also useful where the office is located remotely from the plant. Such connectivity reportedly provided better plant management which in turn improved profitability. The use of generic software was again favoured over proprietary software. The use of common Internet browsers, such as Internet Explorer, was deemed to be effective as any computer can then be used for browsing the process information.

The total price of the monitoring solution obviously plays a very important role in the decision about whether such a system will be installed or not. The system should be compact and use low-cost components to make it a cost-effective solution. The developments in microcontrollers over the last few years have made them much more powerful than a small chip was traditionally expected to be. Bolic et al (2001) suggest that today's microcontrollers are a feasible option for low-cost distributed systems. These microcontrollers integrate signal acquisition and communication facilities with a limited

processing capability on the same chip. The compactness of the solution results in lower overall cost and is potentially very attractive for monitoring systems.

Another benefit of microcontroller-based monitoring systems is their relatively small physical size. A limited number of peripheral components are required to build such solutions and small printed circuit board designs are achievable. The reduction in size provides an ease of installation in the factory environment. Small sized modules can be easily placed close to the required monitored part, often reducing any potential noise interference problems. The acquisition of cleaner signals provides the potential for improved fault detection and reduces uncertainty. This helps in reducing false alarms and generates more confidence about the system performance.

The need for a monitoring system with all the above mentioned capabilities is still a research challenge. The amount of money involved and the widespread applications of such a system in diverse fields of industry thus provided the research motivation to investigate into the possibility of developing a complete monitoring solution fulfilling the requirements of diverse industrial processes. This thesis provides the details of such an investigation, the recommended solutions, the existing technologies to be utilised and a discussion of new techniques and technologies.

LITERATURE REVIEW

3.1 INTRODUCTION

This chapter provides a review of monitoring techniques and applications from a wide range of research publications. Monitoring is defined as to observe, supervise, or keep under review; to keep under observation; to measure or test at intervals, especially for the purpose of regulation or control (Oxford English dictionary web site).

The scope of the research in this thesis is limited to engineering processes in industry. Section 3.2 reviews monitoring examples and set the scene regarding applications. Section 3.3 examines monitoring techniques and provides examples of their use. Published review papers are also included in this section. These are used to discuss the suitability of given techniques for particular process types. Section 3.4 then provides a detailed review of the narrower field within which controller signals are utilized in process monitoring. Control engineering is a well established field and there is a significant amount of work available for control loop performance assessment (CLPA). There is an emerging trend to link the degradation in loop performance with process health and the resulting control loop performance monitoring (CLPM) approach can provide very important information. This section explores the various aspects of this emerging field.

The role of distributed systems in monitoring is increasing and thus section 3.5 reviews various approaches for tackling problems specific to distributed process monitoring. Distributed systems are more efficient for geographically distant sensor data acquisition and provide hierarchical process facilities. Section 3.6 covers compact systems, where a compact system is defined (in this thesis) as a small yet comprehensive system that can be easily placed in a process plant. It is essentially an embedded system, preferably based on a microcontroller unit (MCU) with very little hardware other than the MCU. The small number of components may also decrease energy consumption and the MCU itself may go in power-down mode whenever possible if implemented with appropriate intelligence.

The Internet provides a range of services such as web-hosting, live media streaming, email, news-groups, file transfers, remote terminals, etc. Commonly used browsers are well understood and easily used by non-technical people. The development of document structures, components, and containers of the Internet may be complicated but is easy to use. These features have made the Internet a preferred mode of communication. The use of the Internet for monitoring applications is accordingly on the rise. Section 3.7 covers and reviews this aspect. Finally, section 3.8 briefly describes a selection of commercially available monitoring systems. It includes complete monitoring solutions as well as the data acquisition systems provided by various companies who process the acquired data and provide the results to their respective clients.

3.2 GENERAL CONDITION MONITORING

This section demonstrates the diversity of applications where monitoring techniques are being employed. Process performance monitoring provides technical and economic advantages and its benefits are manifold. Various fields of industry are therefore interested in process monitoring and there are researchers making efforts to improve process performance by applying various techniques. Plesnyaev and Pazderin (2003), for example, applied monitoring techniques in the Russian electrical power industry where a large number of consumers resort to meter tampering in order to reduce energy bills. A method for improving the accuracy and validity of the measured data on energy consumption was presented based on the mathematical modelling of energy flows using state estimation techniques.

Yang et al (2003, A) applied monitoring techniques to a metal forming process. They proposed a friction source location detection system using three acoustic emission (AE) sensors positioned at the metal forming tool. Multiple signals generated by the distributed AE sensors were acquired and analyzed using frequency analysis techniques such as Fast Fourier Transform (FFT) and Discrete Wavelet Transform (DWT). The deviations of the arrival time of the AE signals were used for friction source location using a pre-constructed source-location database.

Studzinski (2004) described the use of monitoring techniques in environmental engineering. Three examples were quoted for modelling and maintaining environmental processes using computers. The first example considered a wastewater treatment plant where a computer aided system was developed to support decision making by the process operator. The development of the models and their adaptive validation was possible only with an efficient monitoring system, and such an automatic system was installed. Their second example considered an integrated computer system developed to support operational decision making in a communal water network. The system had three modules cooperating with each other and water pressure and flow were measured at nine points. The data transmission from the deployed system to the computer was accomplished using GSM telephony. The third example was developed to provide missing data for atmospheric parameters such as environmental temperature. A model was produced using a neural network whose output closely matched the real temperature data. The model output data was used by the expert systems in a case where actual data was not recorded due to a power failure.

Tokatli et al (2005) described the use of process monitoring in the food industry. Product safety is controlled in many food processing operations by checking the end product by microbiological and chemical methods. A major problem with this approach is the associated delays. Tokatli et al considered this method expensive because the product might have been shipped to retailers before any contamination detection. The cost of recalling the product from the market adds to the economic loss caused by the problem. Real-time monitoring of critical control points in the process was suggested using multivariate statistical methods for the early detection of the problem. Separate fault diagnosis methods were used once a fault was detected. A discrepancy was observed in that multivariate statistical charts did not indicate the variable causing the fault. A model-based fault diagnosis technique, referred to as a parity space technique, was used to monitor a high-temperature short-time pasteurization pilot plant in this study. Plant sensors and actuators were interfaced with a computer using a data acquisition system. The actual sensor measurement values were modified by adding numbers to them for generating sensor faults.

Kimmich et al (2005) reported on fault detection for modern diesel engines using signal and process model-based methods. Appropriate signal processing of measurable signals generated residuals and symptoms using these models in this modular design. The algorithms were implemented on a rapid control prototyping system with Matlab/Simulink. The differences between the signal and the reference model generated the residuals. Further processing on these residuals gave the symptoms which indicated faults by considering variations from thresholds. Several faults were temporarily introduced in the engine to check the algorithms. They used semi-physical models where dominant characteristics were modelled physically and secondary effects with neural networks.

Patton (2005) described the monitoring of an electro-pneumatic valve in a sugar juice evaporation plant as part of a Development and Application of Methods for Actuator Diagnosis in Industrial Control Systems (DAMADICS) project. A range of actuator faults were considered and the necessity of advanced diagnostics was emphasized for fault detection and isolation. Benchmarking was considered a useful approach in fault detection and multiple benchmarks were described. Examples included are benchmarks based on simulation of the actuator behaviour (in normal and faulty states) and also a data driven approach based on process data acquired from the evaporation and steam boiler stations.

Hawkins (2004) considered the pros and cons of using Health and Usage Monitoring Systems (HUMS) in defence applications. It was stated that condition monitoring dates back to 19th century and a bewildering array of monitoring techniques have emerged. The most prominent condition monitoring example within the UK ministry of defence is the installation of HUMS to the Chinook and Merlin helicopter fleets. These systems, which have been in gestation for many years, were deemed to be beginning to deliver dependable safety-related management information. Confidence in these systems is still below the desired level but the future deployment of HUMS to the Sea King, Puma, and Lynx fleets was reported as a certainty. Adherence to open standards was also deemed important in defence applications monitoring. It was observed that web-enabled HUMS environment is a rapidly developing field.

3.3 TECHNIQUES & THEIR APPROPRIATE APPLICATION

Process and condition monitoring is becoming an important part in today's industrial setup. A number of monitoring techniques have been developed over the years and applied to various applications. These techniques range in complexity and specialised modifications have been suggested to the techniques for particular applications. Various methods have been tried in different combinations in order to overcome the modern industrial processes' complexity. Fault tree analysis (FTA) is one of the commonly used techniques for fault detection and isolation (FDI). A top level event is specified in the tree and all of the associated elements in the system that could cause that top level event to occur are identified (Relex software web site, FTA). Raaphorst et al (1995) considered a fault tree based diagnosis system for modern trains. They considered that fault tree generation, consistency checking, and maintenance for such a complex system were very difficult tasks for humans to perform. Accordingly, they proposed the use of an expert system with case-based-reasoning as the inferring mechanism. The fault tree was implemented using a graph structure where input nodes were associated to fault symptoms. Existing symptoms posed questions to the network and the answers provided were used for fault code matching. Three train modes, driving, on-platform, and stand-by, were defined and diagnosis was attempted accordingly.

Andrew and Dunnett (2000) considered the use of Event Tree Analysis (ETA) in FDI. ETA is a visual representation of all the events which can occur in a system (Relex software web site, ETA). They considered traditional FTA methods inaccurate and inefficient especially for nontrivial situations with dependencies amongst the branch point events. They proposed use of Binary Decision Diagrams (BDD) where a '1' branch showed occurrence and a '0' branch showed the non-occurrence of an event. This technique was stated to provide quicker results with increased accuracy.

Hu et al (2003) consider fault source location very important as about 80% of downtime is spent on locating the source and only 20% is consumed in the actual repair. They consider FTA as a mature and efficient method that identifies the cause of a system fault hierarchically from the system level to the part/component level. They combined FTA with sequential and logical diagnostic models to achieve good results in fault source

indication for systems based on programmable logic controllers (PLC). The sequential model comprised of the states the system undergo sequentially whereas the logical model provided fault source indication by matching the actual controller's signals against the expected ones using the digital sources. The two models were used as complementary to each other and not as the alternates to one another. A failure in a machining centre was investigated with this approach as a case study and the system correctly diagnosed the cooling system as the root cause. The cooling oil pipeline was blocked and the system could not provide high enough pressure making the motor drive behavior abnormal.

Isermann (1997) supported the use of heuristic approaches for fault diagnosis stating that the underlying physical laws for a process are either often not known (in analytical form) or are too complicated for calculations. He provided an introduction to the field of Fault Detection and Diagnosis (FDD) emphasizing that a monitoring system cannot provide useful alarms unless it does more than checking the measured variables against thresholds. He argued that the monitoring system should calculate features and hence generate fault symptoms. Classification methods are then proposed for the mapping of symptom vectors into fault vectors. He also provided a detailed review of techniques for fault detection. It was observed that binary variables based FTA did not prove successful because of the continuous nature of faults and symptoms. Approximate reasoning methods were therefore considered more appropriate.

Au et al (2004) described a monitoring technique that uses the time difference between successive arrivals of an acoustic emission signal. Statistical distribution analyses were adopted and applied to monitor bearing conditions. The time difference between successive acoustic emissions signals (above a certain threshold) provided a measure that was able to detect wear in the bearings. Kaewkongka et al (2004) applied and tested this technique for bearing monitoring and claimed good results.

The Intelligent Process Monitoring and Management group at Cardiff School of Engineering (IPMM group web site) have developed and refined a Petri-net based process and condition monitoring approach. Petri nets were originally deployed as a dynamic graphical tool to show the current status of the process to the operator. Davey et al (1996) applied this technique for a machine tool failure diagnosis. Normal times were associated with each event and time-out exceptions were used to indicate a fault. A program was

developed for monitoring the machining cycles. The fault cause was diagnosed by analyzing the event that caused the time-out. The authors claim that this method is very effective and provides rapid, accurate and appropriate fault condition information. They state that the process can be managed using an expert system which was also developed by a partner working on the same DTI funded project.

Turner et al (2001) describe use of Profibus fieldbus for process monitoring and control. Process parameters are communicated to a personal computer (PC) acting as bus master. The controller implemented on the PC controlled a batch process on a laboratory test rig. Object oriented programming was used and graphical user interface was provided. They stated that controller-signal monitoring is a useful technique to enable fault detection. Also within the IPMM work, Prickett (1997) proposed and reported a link between the Petri net monitoring system and maintenance management tools. He suggested that events history can lead to the diagnosis of particular failures and fault isolation should be based on the particular signal that prevented an event from happening. This approach was tested on a machine tool failure diagnosis. He considered it possible to detect variations in the manufacturing cycle that may not initially stop the machine but will often do so if left uncorrected. Production downtime can be significantly reduced using this approach. The overall lost production time in a single year was estimated at almost 450,000 hours in a study and around 35% of the reported faults in that period related to causes where operator recovery under a TPM initiative was possible. The 35% reduction in the downtime will have huge economic benefits in this scenario.

An evolution in approach by the IPMM group was indicated by Frankowiak et al (2001) who stated that the cost of monitoring systems has often prevented their wide spread adoption. The commercially available low-cost 8-bit microcontrollers were therefore suggested as primary components for process monitoring applications. An implementation of a Petri net modelling technique using 8-bit microcontrollers was reported. The resulting system comprised of 24 digital, 4 analogue, and 2 pulse inputs and was implemented with a MicrochipTM Peripheral Interface Controller (PIC) microcontroller as the front-end device. The PIC communicated with a PC via a serial communication port. A dedicated application in the PC submitted the gathered event information to a remote server-side database with time and date stamping. The PC provided Ethernet connectivity to the remote database (as it was considered a challenge to

provide Internet access protocols and monitoring tasks running on the same system). They suggested using two separate inter-communicating systems for these two tasks as the most promising way forward. The developed system initially used the RS-232C serial port protocol for communication between the PIC and the PC but subsequently Controller Area Network (CAN) bus communication were determined to be a better option.

Additional knowledge about process parameters may be invaluable for monitoring but extra sensors would be required for that. Companies generally tend to avoid this because of the additional cost and installation issues. Grosvenor and Prickett (2003) evaluated this situation on the basis of experiences learnt from various machine tool monitoring projects. They cited MSc projects within the IPMM research group and established that it is timely to incorporate more sensor inputs into the distributed monitoring systems. Several possible monitoring applications were identified where this approach would be beneficial.

Amadi-Echendu et al (1992) used frequency components analysis in detecting faults in flow processes. They included a pulsator assembly in the water flow loop and observed the signals from a flow sensor for normal as well as perturbed flow. They observed a difference in the frequency components for the two cases although the time series plots looked similar. Similar results were obtained by changing the pump types and their number of blades. It was also observed that such differences appear at frequencies higher than those used for control purposes, which are generally suppressed. The sensors used in this study were slightly modified to gain access to the unconditioned signals. Higham and Perovic (2001) stated that sensor signals are typically filtered to suppress frequencies greater than 5Hz in order to provide a stable signal to process controller. They advocated the use of wide-band sensor signals for monitoring so that the information-rich higher frequency components can also be used. They analysed un-filtered signals from pressure and flow sensors in flow loops using various types of pumps. They observed differences in characteristic frequency peaks for normal and various abnormal conditions including cavitations, partial blockages, and incipient faults. They considered it feasible to identify the type of fault and its level of severity by using this method. They claimed that their conclusion is applicable, in a generic way, to a wide variety of circumstances.

Another commonly used method for process monitoring is Statistical Process Control (SPC) where numerous control charts check process parameters against tolerance ranges. Out-of-tolerance behavior from any chart raises a monitoring alarm. Goulding et al (2000) recognized the presence of highly correlated variables in a typical process plant and urged researchers to use the correlations. They recommended the use of Multivariate Statistical Process Control (MSPC) techniques in FDI, as they provide better monitoring capabilities than univariate SPC by examining process parameters in a cumulative way. They warned that the word 'control' in MSPC may be misleading as the major thrust of the technology is for fault detection and isolation. Two MSPC techniques, namely Principal Component Analysis (PCA) and Partial Least Squares (PLS), were used in their research for fault detection in continuous processes. They found PLS suitable whenever plant variables could be partitioned into cause and effect variables. They favoured the joint analysis of cause and effect stating that its benefits had been observed practically. PLS was therefore used to predict the process outputs from input values. The prediction errors were then analyzed using PCA, which reduced computational effort by converting large number of process variables into a smaller number of principal components without losing significant information. A four-input, four-output plant with three feedback loops was used to initially test the techniques. The performance of these techniques was evaluated by introducing faults in the plant. Simulation and industrial data from a reactor in a chloro-carbon production plant was later used to evaluate the effectiveness of MSPC in FDI. They stated that PLS may provide a better indication of a changing process condition than PCA.

Lennox et al (2001) developed a condition monitoring system for a fed batch fermentation system. Linear procedures, such as conventional PCA and PLS, have limited effectiveness to non-linear batch fermentation problems. Multiway MSPC was therefore used to remove non-linearity in batch data. Fault detection and isolation was performed using multiway PCA. Multiway PLS was used for final product composition estimation, which was in turn used to classify a batch among various low and high yield batches. A warning was issued when the PLS model predicted that the current batch would not produce the required results. Lennox et al suggested the use of an Artificial Neural Network (ANN) in the future as soft sensors with MSPC techniques to provide a suitable controller for this application.

Isermann and Balle (1997) explored the suitability of various techniques in FDI for various types of applications. This was undertaken on the basis of a review of 165 publications between 1991 and 1995. According to this review, electrical and mechanical processes are more investigated than others; with DC motor as the target of the most applications. 70% of applications used observer-based (OB) or parameter-estimation (PE) methods; 50% sensor faults were detected by OB methods. OB methods were preferred for actuator faults. The detection of process faults was mostly carried out with PE methods; nearly 50% of publications used them in such applications. For processes with non-linear models, OB methods were most frequently applied, with PE and neural network (NN) also playing important roles. For processes with linear or linearized models, OB and PE methods were often employed. Isermann and Balle concluded that OB and PE methods were the most frequently applied methods for fault detection whereas fault isolation was often carried out using classification methods for which the use of NN methods is growing. They also provided the definitions for the commonly used terms in FDI in order to enable people to use consistent terminology. Different definitions are given for seemingly similar terms such as fault, failure, malfunction, and error.

The use of neural network provides a relatively simple way to deal with non-linear processes. A neural network comprises of at least one hidden layer which is implemented through some non-linear function. The weights of the links between the layers are adjusted on the basis of actual data obtained from the process. The neural network is therefore trained without knowing the exact mathematical details of the process. Tansel et al (2000) investigated the use of a backpropagation (BP) type neural network in monitoring tool wear in a micro-end-milling operation. They investigated the relationship between tool usage and the cutting force by presenting data to a NN in two different encodings. One of the encodings was based on simple force-variation and the second encoding was based on a more complicated segmental-average. Experiments were performed on aluminum and steel to include the effects of the material being cut on the process. They observed that the optimization of the NN parameters was extremely difficult for micro-milling but extensive training would create a compact and representative model. They claimed to get excellent wear estimations using this approach.

Ruiz et al (2004) described the monitoring of a pilot sequencing batch reactor (SBR), which is a complex process used in waste water treatment plants. It has highly non-linear

and time-varying behavior and is subject to significant disturbances. MPCA was used for situation assessment to detect abnormal batch behaviors. The information obtained from MPCA alone was not found sufficient and automatic fuzzy classification was also used for better situation assessment. The procedure used in this study combined numeric and symbolic classification algorithms with fuzzy logic and hybrid connectives.

Rengaswami et al (2001), while observing that there are very few papers explicitly dealing with the problem of detecting faults in control loops, described a qualitative approach for detecting sinusoidal, square and triangular oscillations in a control loop. Friction and hysteresis in control valves are the most common reasons for such oscillations and detection of oscillation type leads to cause determination. They proposed Qualitative Shape Analysis (QSA) for detecting an oscillation and its type in a control signal. Variations in the signal were checked for any of the pre-defined primitive shapes. Three basic primitive shapes were defined as Increasing (In), Decreasing (Dn), and Steady (S), and their regular repetition was detected as an oscillation. The primitive identification problem for noisy data was solved by using a feed-forward neural network recovering shapes from data with up to 20% noise. A time window was defined in which the presence of a primitive was checked. The window width was considered very important for correct primitive detection and it was shown to depend on the particular application. These primitive shapes, when combined together, provided a signal profile. A profile consisted of the primitive type followed by the number of consecutive time windows for which it was detected. The algorithm pattern-matched the detected oscillation to decide its shape. Square oscillations were easy to detect being predominantly composed of primitive S followed by sudden In or Dn primitives. The alternate repetitions of In and Dn primitives showed triangular oscillation. An oscillation was considered sinusoidal if it was detected but not identified as a square or a triangular one. The algorithm was checked on simulation results as well as on the actual industrial data and provided satisfactory performance for all the test cases. Rengaswami et al claimed that their approach worked quite fast and was suitable for real-time applications.

Various quantitative methods have been applied to FDI for exact analysis. Many soft techniques were also used providing monitoring solutions in fuzzy situations. Both methods have their own advantages and disadvantages. Biswas et al (1997) advocated using combined qualitative and quantitative systems for FDI. Their proposed system uses

qualitative methods at the first stage to reduce the number of variables, or potential fault candidates, and then applied quantitative methods on the reduced set. The use of qualitative methods was urged as development of fault models for continuous physical systems was considered very difficult because of the very large range of possible behaviors. The process was assumed to be in a steady state in this methodology and residuals were calculated from the observed and the nominal plant behaviors. The residuals were marked (0), (+), or (-) for normal, above-normal, and below-normal behavior respectively. Fault candidates were then generated by employing a heuristic best first search technique in the tree structure of the hypothesis space. Multiple fault candidates thus generated were then refined by the quantitative method of transforming the analytic model into a set of linear equations and looking for hypothesis contradictions. The candidates with no contradictions were reported as the possible reasons for the fault. The algorithm was implemented in C and executed on ground-station-based Unix workstations for finding faults in the thermal bus system of the space station Freedom. FDI was ground-based because of the limited processing power on-board. Measurement data was available to ground station every 30 seconds.

Kerkeni et al (2003) considered the monitoring problem from a software point of view and proposed an agent-oriented framework for complex monitoring systems. Various autonomous entities, called agents, were defined in the system that controlled and monitored well defined subsets of the production system at a given abstraction level. Multiple agents shared information with each other through intelligent blackboards. A blackboard had a shared memory area for its associated agent where required local and remote information was stored in a data structure. An intelligent blackboard with updated data informed all other blackboards in the system about the update, thus maintaining consistency in the system. A central information system containing the global view was also updated. The system was defined in a hierarchical way where father and child Control/Monitoring Agents (CMAs) communicate with each other through the intelligent blackboard. The bottom most CMAs in the hierarchy were interfaced with the physical agents. The monitoring results were stored on the blackboards under various contexts such as production, processing, physical resources etc. The proposed system was developed on a multi-agent Java-based platform and JDBC was used for connection to the information system.

Soderholm and Parida (2004) emphasize the need to interlink the stakeholder requirements and the key performance indicators (KPI) in performance measurement. A conceptual framework was provided for this purpose with examples in the context of a modern fourth generation combat aircraft. The combat aircraft is taken as the example because it is a highly complex and safety critical system with stringent requirements on low life cycle costs. The JAS 39 Gripen aircraft, for example, consists of more than 20 subsystems for flight control, weapons, hydraulics, display, etc. These systems communicate with each other and together they build a system with a theoretically infinite number of possibilities. Such complex technical systems often have a rather long life time and the requirements on these systems change over time due to the technical development and changes in the needs of the stakeholders. The proposed stakeholder based health management system framework established traceability between stakeholder requirements and corresponding health measurements which support requirements validation, verification, continuous improvement, and modification. Soderholm and Parida considered it very important that all stakeholders know the health status precisely as incorrectly performed maintenance might result in unwanted effects. They urged that an on-board health management system must be connected to the other information systems.

Venkatasubramanian et al (2003 A, B, C) provides a very detailed review of monitoring methods used in various kind of applications. They stress the importance of FDD citing failure examples causing huge human and property losses. Minor accidents in industry occurring on day to day basis accumulate to huge annual sums such as \$20 billions for the American petrochemical industry. Similar losses in the British economy range up to \$27 billions every year. Because of the complexity of modern process plants, it is not always possible for operators to provide a proper response to every fault condition. About 70% of the industrial accidents are caused by human errors in these circumstances. They urge the need to develop automated FDD systems to tackle this problem. Ten most desirable characteristics for such a system were listed including reliability, robustness, quick response, and adaptability. It was noticed that these features are contradictory to each other and cannot all be provided at optimal level simultaneously; a good compromise between these was however desired. They classified the FDD techniques in three broad areas, i.e., quantitative model based techniques, qualitative model based search strategies,

and non-model based techniques that use only the historical data of the process. Each part of this three part series of papers concentrate on one of these methods.

The first part provided a detailed review of work done on quantitative model based techniques and described various such techniques. It discussed analytical redundancy, residual generation and evaluation, parity relations, Kalman filters, and parameter estimation approaches in detail. Directional residuals and structured residuals were stated as two enhanced residual generating techniques that have attracted much attention. These techniques provide a set of residuals that collectively behave differently for different faults and hence provide an indication of a particular fault. Various limitations for quantitative model-based approaches are also reported such as the need of accurate modelling, the problem with non-linear processes, and the detection of faults that have not been modelled.

The second part provided a review of qualitative models and search strategies used in the area. Signed digraphs (SDG) were considered very efficient in the graphical representation of qualitative models; where SDG is a graph with signed directed arcs from 'cause' nodes to 'effect' nodes. A cause-effect graph is obtained from SDG containing only the nodes showing abnormal behavior, thus indicating the fault reason. Several extensions to standard SDG were also reported including FTA, which uses different logic nodes rather than the predominant use of an OR node by SDG. Qualitative physics or common sense reasoning is another approach used in qualitative FDD. A review was provided covering the work from several researchers using several techniques in this area including qualitative simulation (QSIM) and qualitative process theory (QPT). A system may be divided into sub-systems to reduce the complexity of the problem. 'Structural' and 'functional' decompositions were regarded as the two most popular hierarchical decomposition techniques. The techniques for the search in fault diagnosis space were classified as either topographic or symptomatic. Topographic searches perform malfunction analysis using a template of normal operation whereas symptomatic searches look for symptoms to direct the search to the fault location.

The third part reviews techniques and implementations concentrated on the use of historical process data. Various features are extracted from the data and used for FDD. Feature extraction may be done using qualitative techniques, such as expert systems or

trend modeling methods, or statistical or non-statistical quantitative methods. A neural network is an important example of a non-statistical method whereas PCA, PLS, and pattern classifiers are important statistical techniques in this respect. Suitable classification and process trend analysis can detect faults earlier and lead to quick control. Qualitative trend representation can pave the way for efficient data compression. Zero-crossing of trends was considered to be an important sign of change in trend. Some research was reported trying to combine multivariate statistical methods and model-based approaches. An application combining neural network with wavelets was also described. K-means clustering was recognized as the most popular clustering algorithm whereas back propagation was considered the most popular supervised learning technique in neural networks. The relative advantages and disadvantages of the described techniques were provided and are presented here as Table 3.1.

	Observer	Digraphs	Abstraction hierarchy	Expert systems	QIA	PCA	Neural networks
Quick detection & diagnosis	Y	?	?	Y	Y	Y	Y
Isolability	Y	X	X	Y	Y	Y	Y
Robustness	Y	Y	Y	Y	Y	Y	Y
Novelty identifiability	?	Y	Y	X	?	Y	Y
Classification error	X	X	X	X	X	X	X
Adaptability	X	Y	Y	X	?	X	X
Explanation facility	X	Y	Y	Y	Y	X	X
Modelling requirement	?	Y	Y	Y	Y	Y	Y
Storage & computation	Y	?	?	Y	Y	Y	Y
Multiple fault identifiability	Y	Y	Y	X	X	X	X

Table 3.1: Comparison of diagnostic techniques (Venkatasubramanian et al C, 2003)

Venkatasubramanian et al observed the scarcity of literature on industrial applications of diagnostics systems and identified the proprietary nature of in-house developments as a possible reason. They considered easy deployment and adaptability to future requirements as necessary for industrial solutions. The development of hybrid monitoring systems was favored as none of the techniques sufficiently covers all the requirements on its own. A brief review of hybrid solutions was also provided including the blackboard-based DKit architecture which is adopted by Honeywell ASM Consortium for its next generation intelligent control systems called AEGIS and MSEP. They also observed that researchers generally treat diagnosis and control as separate problems, in spite of their close

connection, and suggested their integration for real progress in this area. Several important challenges were also highlighted for future research such as reasoning without assuming accurate models, ability to cope with data explosion, implementational issues for large scale industrial applications, etc.

3.4 MONITORING WITH CONTROLLER SIGNALS

This section concentrates upon approaches based upon process controller signals and identifies how such signals may be used to distinguish between normal and abnormal process conditions. The premise is that, as the controller signals vary to control the process under faulty conditions or in response to disturbances, monitoring the actions of the controller will indicate the current health of the process. A review of the use of controller signals in process monitoring is given in this section. It also gives a brief introduction to the rapidly developing Closed Loop Performance Assessment (CLPA) techniques and the integration of such techniques with traditional FDI.

Harris (1989) considered the issue of assessing control loop performance benchmarking. He suggested that an estimate of the best possible control can be obtained by fitting a univariate time series to process data collected under routine control if the process time delay is known. The performance of any control loop can therefore be assessed on the basis of how close it is to the theoretically best achievable performance. The theoretically best performance can be assessed using the minimum variance control (MVC). Such a controller is not used practically because of the extensive control action it exerts on the actuators resulting in their excessive wear. However, its calculations are beneficial for the sake of comparison to what performance the actually implemented controller is achieving. MVC performance can thus be used as a benchmark and the ratio of actual loop variance to the MVC loop variance is usually referred to as the Harris index. The minimum value of the Harris index is unity and is the best possible performance from a controller. Any performance improvement is not possible by re-tuning the controller in this case and a process change is required for an increase in process efficiency. Larger values of Harris index shows that improvements can be achieved by re-tuning the controller. The paper heralded a new era in loop performance analysis that has subsequently developed in various ways.

Horch (2000) reviews developments and improvements based on this work. He states that the concept has been extended to feedforward loops by Desborough and Harris (1992, 1993). Tyler and Morari (1995,1996) have modified the idea to apply it to unstable and non-minimum phase processes and have used statistical likelihood ratio tests. Lynch and Dumont (1996) used a Laguerre network to evaluate the performance index. The initial idea covered single input single output loops but this has been extended to multivariable loops by others such as Harris et al (1996), Huang et al (1997), and Ettaleb (1999). Bezergianni & Georgakis (2000) compared the actual control with both minimum variance control and open loop control for their performance index. The evaluation of such performance indices requires knowledge of any process dead-time. Horch (2000) introduced event-triggered estimation for process dead-time estimation from normal operating data.

Desforges et al (2002) advocated interaction between the process controller and the condition monitoring system. They emphasized the need for continuing process operation under identified fault conditions and suggested a two level hierarchy at sensor and process levels. Self validating (SEVA) sensors, providing status signal as well as data, were supported at the sensor level monitoring. A SEVA sensor would self-sense a fault and decide whether the fault is permanent or not. It continues to provide data, based on an estimate from previous data, until a decision is reached about its fault. At sensor fault confirmation, process level condition monitoring would start generating estimated data, which is likely to be more accurate in the medium term. The process can thus continue under an identified sensor fault. Desforges et al reported the development of a toolbox with advanced multivariate statistical process monitoring (MSPM) methods tailored to the process control environment. The system was claimed to be successful in identifying sensor and process faults in a case study on a simulated fluid catalytic cracking unit.

Fu and Dumont (1995) reported an implementation of their algorithm to evaluate a control loop performance linking it with the problem cause. They modified a previously available program to calculate the loop oscillation index by using oscillation period estimation. The algorithm was claimed to be successful in detecting the oscillating loop by using minimum achievable output variance and the oscillation index. The performances of two loops with strong interaction were evaluated in a simulation and one of the loops was found problematic. It also correctly indicated that the problem was not

due to poor controller tuning but because of the poor health of a controlled valve. In a separate application, the algorithm picked a poor performing loop out of the 22 loops monitored in a mill trial. The algorithm was claimed to provide fast on-line evaluation and that the detection of the problem source was possible.

Clarke (1999) highlighted the importance of the generic validation of information provided by the sensors, actuators and the control loops and its reporting to the next higher level. He observed that economic pressures are de-skilling the maintenance work force and therefore suggested that an automatic information validation process should be built into the components. He observed that embedded microcomputers are of use for sensor validations and recommended information sharing on generic standards such as fieldbus. This would reduce economic limitations on manufacturers and the product can be used in several applications by various companies. Sensors would communicate the validity index along with the data they sense. He proposed four status levels for a sensor: clear; dazzled; blurred; and blind. This validity information can be used to switch the modes of the loop controller, if required. Sensor validation leads to the structure of a 'SEVA' sensor whose synergy can be represented by the intersection of the circles as shown in figure 3.1. The importance of the actuator validation was also highlighted, being the mechanism actually implementing the controller decisions. It was considered important to provide validation results to the respective higher level for decisions in a broader sense. Clarke observed that some crude form of validation was already being built in the sensors but actuator and loop validations were still in their infancy.

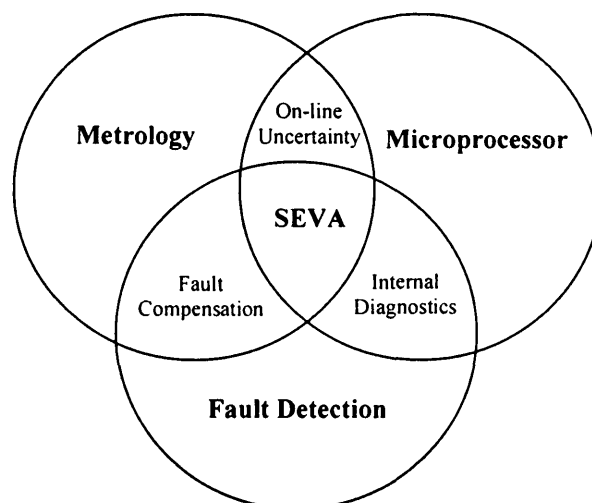


Figure 3.1: Validation as a combination of technologies (Clarke, 1999)

Gustafson and Graebe (1998) concentrated on detection of the loop performance degradation. Particular attention was paid on detecting whether an observed deviation from nominal performance was due to a disturbance or due to what they termed 'a control relevant system change'. The system was perturbed by sinusoids of the frequencies of interest with amplitude well above the expected noise level. The prior knowledge of the frequencies of interest was therefore very important and a useful discussion was provided on this topic. The stability margins were defined in terms of a computationally convenient clover like region for their algorithm rather than the conventional circle. These stability margins were monitored rather than parameter drifts or jumps in the algorithm. The algorithm was successfully tested on simulation data as well as in real-time on a DC motor. The implementation was done on a TMS320C30 digital signal processor (DSP) based dSPACE real-time system. The algorithm requires an injection of an exogenous signal that perturbs the normal operating conditions and is, therefore, of a limited use only.

Hagglund (1999) proposed an index for detecting sluggish control loops. He observed that in case of sluggish loop response, both the control signal and the process output drift in the same direction for a very long time. An idle index was therefore suggested to detect such situations by studying the correlation between the two signal increments (or decrements). The time periods when the correlations between the signal increments were positive, t_{pos} , and negative, t_{neg} , were calculated and the idle index, I_i , was computed as $I_i = (t_{pos} - t_{neg}) / (t_{pos} + t_{neg})$. The value for I_i would be close to +1 for sluggish loops and close to zero for reasonably well tuned loops. The values close to -1 generally showed well tuned loops but were also observed for sluggish loops with oscillation. The presence of oscillation in the loop should therefore be detected separately. Another limitation of the idle index was observed with loops with overshoots where the index was not reliable.

Hagglund (1995) described possible reasons for oscillations in control loops and considered friction in actuator valves as the most common reason. He proposed an integrated accumulated error (IAE) index for oscillation detection using the magnitude of absolute error for detecting load disturbances. Load disturbance was declared when the integrated value of the absolute error between successive zero crossings was greater than threshold value (IAE_{lim}). This load disturbance detection procedure was then used to detect oscillation in the loop. Oscillation detection was announced if the frequency of the

load disturbance detection became greater than a threshold value, say n_{lim} disturbances detected in t_{sup} supervision time. The method can detect an oscillation irrespective of its shape as only the zero-crossing time is taken into account. On oscillation detection, further tests may be conducted on process valve to confirm its health.

Considering widespread presence of oscillations as the most important problem in process industry, and high static friction in valves as an important reason for that, Horch (2000) reviewed the methods for automatic detection of static friction in the actuators. Figure 3.2 explains the phenomenon of oscillation generation as a result of static friction. The cross-correlation between controller signal and the process output was used to distinguish whether a detected oscillation was caused by the valve friction or not in a non-integrating plant. Another method using the second derivative of the process output provided the same information for integrating plants.

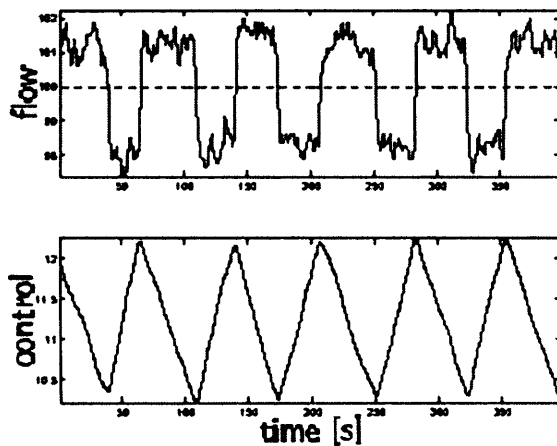


Figure 3.2: Typical behavior of a control valve with static friction. Top: Measure flow (solid) and flow set point (dashed). Bottom: Control signal (Horch, 2000)

Horch (2000) observed that Harris index may provide misleading results for oscillatory control loops. He suggested to detect the oscillation first, if possible, or otherwise using autoregressive (AR) modeling rather than autoregressive moving average (ARMA) modeling. Once the oscillation was detected and removed from the time series, the modified method was used to distinguish between oscillations generated within the loop or induced from external sources. This knowledge directs the maintenance staff towards the root cause of the problem. Another method for this distinguishing was proposed when a simple process model was available. Using the model and an estimate of the controller, the ultimate frequency was calculated and compared to the actual oscillation frequency.

Thornhill et al (2001) discussed the detection of a root cause loop for a plant-wide oscillation among several control loops in the plant. They investigated the effectiveness of detecting non-linearity in the control loops for finding the root cause loop as the valve static friction is a non-linear process. The dynamic behavior of physical processes gives a low-pass filtering affect and therefore reduces non-linearity in loops away from the root cause. A loop with maximum non-linearity is therefore the prime candidate for being the root cause. Two possible non-linear measures were used and their respective effectiveness was compared. The first measure used a distortion factor, D , which was calculated from powers in the controller signal and its fundamental component. The second measure, N , was the non-linearity statistics calculated from time series trend. Any time series with $N > 1$ was classed as non-linear. Larger values of N showed more non-linearity in the loop. Both measures were found useful in a case study on a hot water flow valve in a stirred tank pilot plant. However, N was found more responsive than D . Similar results were found in an industrial study where N measure showed better performance in the presence of noise.

Xia and Howell (2003) defined control loop status monitoring as near-real-time declaration of what a loop was actually doing at that time. They qualitatively classified control loops in seven categories according to their current statuses. Various statistics were defined to identify a PI or PID control loop status out of the seven categories. These statistics were based on the variances in deterministic parts in a measured time series for controller signal, output signal, and noise. A 30th order d-step ahead AR model with least square estimation was used to estimate the deterministic components from the time series. Each status was assigned a number to calculate a quantitative index. A loop with poor performance would have a higher index than a good one and the loop with the highest index in a group would be the first candidate for attention. Current status information was also used to narrow down the number of possible faults to be investigated. This approach was considered suitable for PI and PID loops only and not for proportional or open control loops.

Schafer and Cinar (2004) noticed that the availability of a small number of control engineers makes the analysis of raw data virtually unmanageable and urged automatic detection of problems. They favoured the classification among possible root causes to reduce the processing required. They proposed a performance measure based on the ratio

of historical and achieved performance for monitoring. The diagnosis procedure was started when a decrease in performance was indicated and the ratio of design and achieved performance was suggested for problem diagnosis. They tabulated the possible root causes in two main groups which contained further sub-groups for detailed diagnosis. A group was first established for the fault and the root cause was searched in that group according to the given procedure. A commercially available knowledge based system was used to monitor an evaporator in a case study. The software modules were developed in Matlab from where C code was generated. This study focused on Model Predictive Control (MPC) systems and diagnosis was limited to distinguishing between root cause problems associated with the controller and problems that were not caused by the controller. The diagnostic sequence assumed that only one source cause could occur at one time to reduce complexity.

Mosca and Agnoloni (2003) suggested continuous control loop monitoring so that performance degradation is detected as early as possible. A measure is therefore required for performance analysis which can indicate a problem without perturbing normal operating conditions. They aimed to find such a measure under typical control system conditions where set-point changes are infrequent and the process, actuators and sensors are noisy. A statistic was proposed and computed as the ratio between the norm of an I/O regression vector and the maximum absolute value of a nominal output prediction error. It was claimed to detect the divergence trends very quickly enabling the operator to promptly switch to a more suitable controller. They warned that as a single scalar-valued measure, this test should only be used as an early warning system. Other more elaborate tests could be initiated for detailed investigation upon receiving this early warning.

Kendra and Cinar (1997) emphasized the use of frequency domain techniques for controller performance assessment. They stated that time averaged measures provide little information about the performance of the system. A system-identification based method was proposed for assessing the performance of multivariable closed loop systems using measures that coincide with classical and modern frequency domain design specifications. In particular, two parameters 'Sensitivity' and 'Complementary sensitivity' were proposed (in s domain form). These parameters were obtained by exciting the reference input with a zero-mean random binary sequence and observing the process output and error responses. A closed-loop model was thus obtained. Comparison

between the model performance and the design specifications then provided the performance measure. Matlab system-identification toolbox was used to assess the performance.

Thornhill et al (2003, A) reported the detection of a plant-wide oscillation in an Eastman Chemical company plant and the isolation of its root cause. Detection was achieved using data-driven analysis of routine plant data stored in a database. The plant had 15 control loops and information was available on the set points, process variables and controller outputs. Plant data was sampled every 20 seconds and time trends were available for visual inspection, which showed the presence of an oscillation with a period of nearly 2 hours. The oscillation affected many process variables and controller outputs and was considered a plant-wide oscillation. A non-linearity index was used to detect the root cause for this oscillation among all the oscillating loops. A root mean square (rms) value of error from non-linear prediction using matching of nearest neighbors in an m-dimensional phase space known as an embedding was used. The embedded matrix, Y , contains E columns and has successive rows of the same data with a time delay. In case of oscillation, later rows of Y will be similar to the earlier rows and are called near neighbors. An earlier row can thus provide a prediction for a later row. It was recommended that 25 to 35 samples per cycle, S , are taken into account and that data for at least 10 cycles are used. Consistent and robust results were claimed using $E = S$ and $H = E$ where H -step ahead predictions were used. A value of 8 was considered as a cautious and robust selection for the number of nearest neighbors. Surrogate data was derived from these pre-processed time trends and used for the non-linearity tests. Non-linearity index, N , was defined as a three-sigma statistic and non-linearity was inferred when $N > 1$. The loop with the highest non-linearity index was the prime candidate as root cause. Process knowledge about inter-affecting loops was also used to confirm the result. Other loops with high values on N were also investigated. Combination of process knowledge and non-linearity test indicated that maintenance was required for a particular valve. Further testing of the valve confirmed the result and valve was scheduled for maintenance at next plant shutdown.

Thornhill et al (2003, B) presented an automated method for oscillation detection in a control loop. Regularity of zero crossing of a filtered autocovariance function (ACF) of time series data was used. The ACF was used in order to avoid noise induced zero

crossings in the time series. An oscillation was considered regular if the standard deviation of the period was less than one third of the mean value. Only the oscillations with large magnitudes were considered to avoid noise effects. The presence of multiple frequency oscillations would disturb the regularity of zero crossings and affect the detection. Thornhill et al used digital filters to separate various frequency bands to avoid this disturbance. They observed that frequencies close to the filter boundaries can cause false detections and recommended rechecking such results using different filter boundaries. Manual selection for filter boundaries was used in developing the technique although an iterative automated algorithm was also provided. The technique was initially devised using a pilot plant data and was proven for an industrial data set.

Burkett and Thornhill (2002) provided an end-user's view about industrial multivariable control and addressed the question of its performance assessment. They reviewed the use of multivariable control in BP Chemicals with special emphasis on MPC. It was noted that MPC is useful in petrochemical, refining, and polymer production industries but its benefits could still be improved in many ways. The monitoring and benchmarking of these controllers were suggested to improve plant-model mismatches and problem diagnosis. It was also stated that diagnostics tools were very limited in SISO systems. Commercially available tools for the monitoring of industrial MPC typically use performance measures such as up-time and availability, constraints active, percentage of time against constraints, LP targets, model prediction errors, an overall dynamic performance indicator, and checking of controlled and manipulative variable limits.

Hartley (2002) provided another end user's view of process monitoring stating that many companies have reported paybacks of only a few months for strategically placed vibration-monitoring systems. He emphasized the early detection of anomalies in the plant and proposed adopting complementary approaches for increased plant availability and efficiency. He considered that the knowledge of an instrument's health was equally important as the knowledge about its failure and examined industrial data which gave the probabilities of component failures. He noticed that rotating equipments was the least reliable type of instrument in industry and that transmitters were the most reliable. It was observed that about 20% of any maintenance budget was still spent on the inspection of transmitters. He emphasized on using automated process and condition monitoring to reduce such work. Hartley favored the use of smart field devices over centralized

approaches for this purpose. He stated that smart field devices consisted of sensor and electronics modules, and considered how more and more tasks are being accomplished in the electronics section. Hartley described the PlantWeb™ architecture as an example of such an automated process and condition monitoring system. He described PlantWeb™ as a system for enhanced measurement, advanced diagnostics and control in the field. It was built on the network of intelligent field devices, scalable control and systems platform, and used integrated modular software. Hartley defined asset management as maintaining product equipment properly so as to deliver maximum performance and service life at minimal cost. He reported the use of Asset Management Solutions (AMS) software, which is based on HART and fieldbus. AMS was claimed to be the leading PC-based software for providing on-line diagnostics for equipment and process monitoring. It was claimed that smart field devices reduce process variability and result in better asset management.

3.5 DISTRIBUTED SYSTEMS FOR MONITORING

The quest for effective monitoring systems is resulting in new and unique solutions. A major development in the field is to make different processing elements collaborate with each other to solve a problem. Modern computers have efficient communication facilities and networking speeds now allow data transport in real time, although some volume limitations still apply. These distributed systems provide several benefits and researchers have explored their use in monitoring applications. This section provides various solutions to the problems related specifically to the distributed monitoring systems.

Ehrlich et al (1997) described a generic model for distributed data acquisition architecture. They stated that distributed architectures become attractive as soon as the instrumentation domain size and the number of measurement points increase. They provided a comparison of the system cost based on centralized and distributed systems and considered the cost of system controller (or centralized DAS), measurement point(s), and wiring as important factors. It was observed that wiring length can be greatly reduced in a distributed architecture. The generic model described a smart sensor, system controller, and communication network as three major system components. Ehrlich et al defined a smart sensor as a microsystem located in the vicinity (around 10 cm) of a

transducer or a group of transducers dedicated to conditioning, sampling, calculation, and communication. Figure 3.3 depicts a smart sensor as proposed in this model. The system controller provides management tasks in the network including data storage and user interface. A Data Dependence Graph (DDG) produced a data flow under the event-based control of smart sensors where both local and external event were used. The model performance was checked by simulating its components on a single computer.

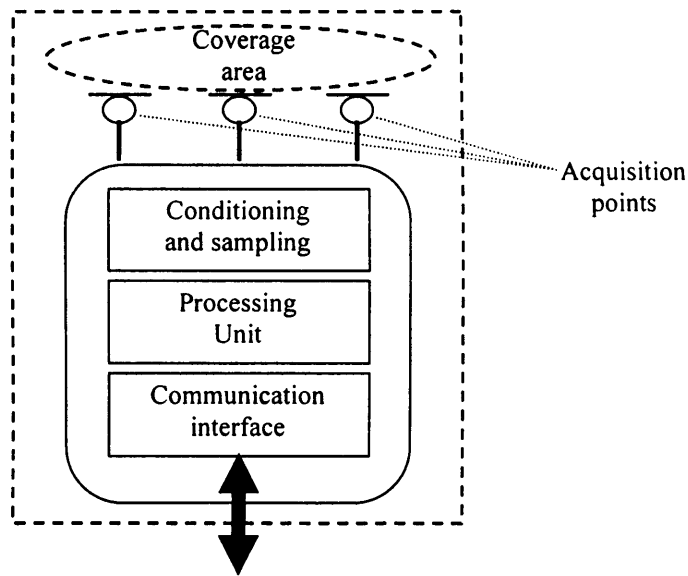


Figure 3.3: Smart sensor (modification from Ehrlich, 1997)

Nieva and Weggmann (2002) provided a conceptual model of a generic data acquisition system. “The conceptual model is a formal definition of a system, from the object perspective, that shows the relevant concepts and relationships that make up the system. Using a conceptual model of a system makes it easier to understand the system, because the model only focuses on the main aspects of the system by hiding low-level details”. The model was presented in unified modelling language (UML) and contained various packages. The ‘device items’ package represented a real world device and a ‘device models’ package characterized a set of device items. Monitoring criteria were defined for device models and device items and included in the generic model as separate packages. Operational-level and knowledge-level concepts were distinguished in the model using different UML representations for them. Further UML diagrams were used to elaborate model concepts such as observations and monitoring reports, mapping policy, time condition, etc. A unique global identifier was associated with each device model and device item. Nieva and Weggmann stated that a generic DAS model must provide a plug

and play facility and they included an auto-configuration scheme in their model. The generic DAS model can be used to implement a real DAS and an example was provided where a DAS for railway equipment was developed. The provided conceptual model specified only the static concepts of a system and further work was ongoing for the dynamic behaviour of a generic DAS using role-based use-case modelling.

Jennings et al (2001, A), within IPMM research group, considered the problem of getting required processing power to analyze data obtained from the acquisition system. They proposed to use the computers available on the local area network (LAN) for processing while they were idle. A PC in screen saver mode was considered idle. This distributed data processing system comprised of task manager and task processor modules. The task processor module was installed on the PCs as screen saver. It was launched by the operating system when there was no user activity observed for a certain time. The module then connected to SQL server via an ODBC driver and obtained the first job from a task list. The processing tasks were divided into small jobs so that the computer would become available to a normal user on any keyboard or mouse activity. The jobs were designed to finish between 7 to 20 seconds, but this time could be reduced. This performance was measured using Windows NT on Pentium 2, 450 MHz, 64 MB memory computers and better computers would reduce the processing times significantly. Increases of 37% in CPU usage and 36% in memory usage were observed in the study. A significant increase in server CPU load was observed with increased number of workstations on LAN.

Kandasamy et al (2005) stated that embedded systems are being increasingly used in safety critical mechanical and hydraulic systems and multiple processors are available in such distributed systems. Steer-by-wire is an example of such systems where a traditional steering system is replaced by a microprocessor-controlled networked system without any mechanical backup. Such advanced vehicle control applications are typically realized as real-time distributed systems where sensors, actuators, and processors interact through a common communication bus. It is important to detect faulty actuators quickly before the system reaches an unsafe condition. Kandasamy et al addressed distributed failure diagnosis under resource and deadline constraints and proposed cost reductions using software-based redundancy rather than hardware-based. They developed a software-based approach where multiple processors agreed on the fault status of an actuator using

multiple and possibly diverse behavioural models. The use of multiple independent detection points provided redundancy and faults were quickly detected even with a faulty detection subsystem. This provided the information about the failure of a monitoring subsystem as well, which would be removed from future decision making. Tasks were duplicated on various processors to provide this redundancy. Only the very critical failure modes were monitored in this study to reduce computational overhead. They did not address recovery action once a faulty actuator is shut down considering it a user responsibility.

Mittal et al (2003) observed that several algorithms have been proposed for real-time multi-hop networks but not much work has been done for real-time multiple access networks. They cited several schemes for real-time communication in multiple access networks but regarded them as suitable only for soft real-time cases. They considered that probabilistic collision resolution protocols were not suitable for hard real-time communication. Accordingly, they proposed two guarantee based protocols for real-time channel establishment to support periodic and aperiodic messages. It was assumed that channel access was time-slotted and transmission could start only at the beginning of a slot. Both protocols worked in two phases where resources were reserved in the first phase and transmission was done in the second phase. The first protocol, called the earliest deadline first (EDF), calculated the deadline for any given message and scheduled messages according to the earliest deadline, pre-empting any other scheduled messages if required. The second protocol, called the BUS protocol, was a modification of a backplane bus scheduling algorithm, used by several hardware modules communicating through a backplane bus under centralized control. No centralised control was required in the modified BUS protocol. The performance of the two protocols was compared. It was observed that the EDF protocol offers higher schedulability than the BUS protocol as it accommodates a higher number of periodic messages. The BUS protocol provided faster response to aperiodic messages.

Suzudo et al (2003) recognized the problem of software module integration. The need for software portability adds to the complexity of the task. Different modules in an application may be written in different languages and run on different hardware platforms with different operating systems. TCP/IP is an important communication protocol for distributed systems but several languages do not provide any built-in support for it.

Variations in development tools usually make it very difficult to put all the enabling components together in one working application without modifications. Suzudo et al proposed the use of standard input/output (keyboard/monitor) to alleviate this problem. Each module would read from a keyboard and write to a monitor only. Their idea stemmed from the way a user controls all the tasks on the desktop PC. A super-humanly fast operator was required to synchronize all the modules in a monitoring system. An operating system like tool, Expect, was used to do it automatically, which was available as free software. Expect functions can be called from C/C++ and Tcl languages. Tcl is a portable interpreter language suitable for relatively small programs. Tcl and Expect were used to develop a neural network based anomaly detection system. The neural network program, written in Fortran, was already available as high reliability software but did not provide a TCP/IP Application Programmer Interface (API). Each module was therefore designed using the above concept and figure 3.4 shows the resultant.

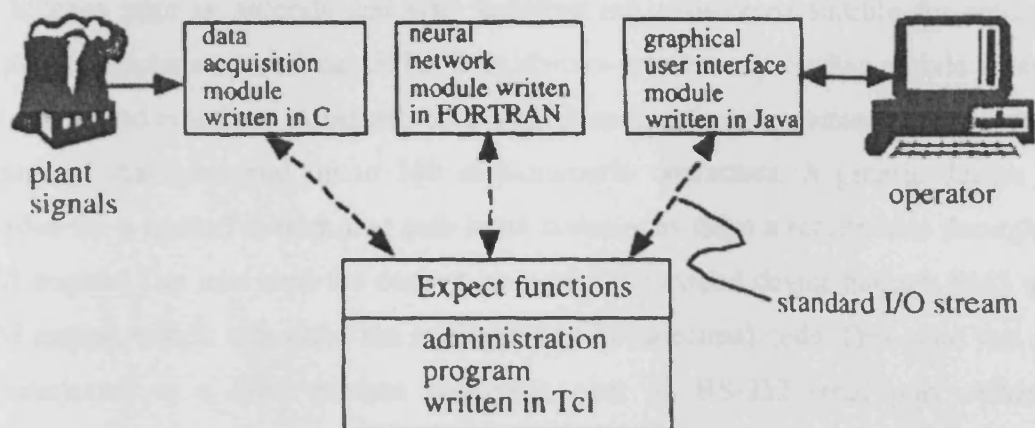


Figure 3.4: Modular system configuration with standard I/O (Suzudo et al, 2003).

Mounce et al (2003) considered data fusion from sensors measuring flow and pressure in a treated water distribution system. About 20% to 30% of transported water was reported lost through pipe leakages in UK during 1990s. Flow sensors were therefore mounted on the pipes recording the water flow every 15 minutes. The data from the sensors was used to audit in-flow and out-flow of water for leakage detection. The sensors' data was manually collected from the field and was thus available after several days in most cases. A supervisory control and data acquisition (SCADA) system was considered in this study to reduce the delay. Use of an artificial neural network (ANN) was proposed to detect the leakage. A number of ANNs, arranged in parallel and hierarchical fashion, were required

in complex water distribution systems. The ANNs were trained using time series data collected from the sensors for several weeks or months. Pressure sensors were also used in the system and data from 14 pressure loggers were combined in test trials. Pipe bursts were simulated by opening outlet valves and system performance was evaluated. The resultant three dimensional pressure drop map was reported successful in the accurate indication of fault location.

Alheraish (2004) considered a mobile phone as a commonly used device and urged its use as a widely available remote user interface. He reported commercial availability of Mobile-to-Machine (M2M) engines that interface computers with Global System for Mobile Communication (GSM). An M2M engine contains Subscriber Identification Module (SIM) and provides mobile services such as Dual Tone Multiple Frequency (DTMF), Global Packet Radio Service (GPRS), and Short Message Service (SMS). DTMF sends multiple frequency tones for a key pressed and is used in telephone banking etc. It takes time in seconds and was therefore not considered suitable for automatic machine to machine operations. GPRS is an always-on service providing mobile access to the Internet and email etc. Alheraish used SMS messages (more commonly known as text messaging) that can send up to 160 alphanumeric characters. A generic design was provided for a control system that gets input commands from a remote user through the M2M engine. The user sent the desired state of a controlled device through SMS to an M2M engine, which converted the message into hexadecimal code. This code was then communicated to a local process controller over an RS-232 serial port. Alheraish favoured microcontrollers for such implementations but used a PC for his test system. Controllers were implemented for an on-off lamp and 3-speed fan as examples of a system having many practical applications in industry, business maintenance, customer service, and security.

High end distributed monitoring systems use large-scale wide-area computer networks. An example of such systems is Distributed Aircraft Maintenance Environment (DAME) project (Fletcher et al, 2004). It provides a Grid based environment for aero-engine condition monitoring where engine data is captured, stored and used for fault diagnosis and prognosis. Engine data is captured during the flight by on-wing monitoring system, "QUICK", and stored on-board. The QUICK monitoring system is the result of the collaboration between Rolls-Royce and Oxford University and performs analysis of data

derived from continuous monitoring of broadband engine vibration for individual engines. Known conditions and situations can be determined automatically by QUICK and its associated Ground Support System (GSS). A remote expert system analyzes current and historic data for less well-known conditions. Data is therefore captured from the aircraft's on-wing system once the aircraft has landed on the airport. Each aircraft flight can produce up to 1 Gigabyte of data per engine. Considering the size of fleet, this data is in order of Terabytes per year and is captured in a distributed way at different airports. Storage of this data requires vast repositories that may be distributed across many geographic and operational boundaries. Processing of this data requires a distributed diagnostic infrastructure whose requirements were captured and developed via use case analysis in collaboration with the industrial partners. The "DAME" diagnostic infrastructure provides a GRID based environment where users in different organisations and locations can access this distributed data and work together using a variety of tools and processes to determine a diagnosis. The infrastructure includes several specialist software packages for diagnosis involving techniques and tools for signal analysis, advanced pattern matching, case based reasoning, data visualization, and very fast searches on extremely large data sets. It provides a workflow manager for systematic FDI progression from simple to complex faults. It also incorporates complex scenarios resulting from specialists' need of additional tests to confirm a diagnosis.

Another large-scale PC based distributed system was reported by Paulonis and Cox (2003) providing information about thousands of control loops in various Eastman Chemical Company plants worldwide. The system comprises of various blocks such as a data interface, computation engine, web server, and User interface. Each block may consist of several computers for large systems. Figure 3.5 shows the schematic diagram of the complete system.

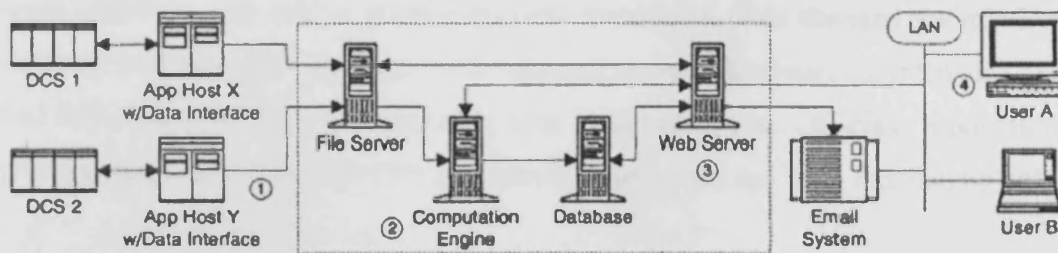


Figure 3.5: Controller performance assessment scheme (Paulonis and Cox, 2003).

Processing power at each control system was used to collect and push data to an assessment system. The computation engine performed various checks including time series trends, setpoint crossings, closed loop impulse response, power spectrum, extended horizon performance index, oscillation detection, and cross-correlation between error and output. The generated results were written in a database in numeric and text formats. Interested users browsed via a provided web interface and navigated through the system using hyperlinked web pages. In addition, a user could subscribe to a variety of reports generated by the system on a daily and monthly basis. These reports were sent to the subscribing users via HTML emails. A report was generated for tuning changes daily as a change in tuning, being easy and cheap, is usually attempted as a first remedial measure for solving a problem. However a change in tuning may not be the best response to an event, and the daily report therefore promptly notifies management as it takes place. A user can pull up a web page for the problematic loop by clicking on its hyperlink provided in the reporting email and check for details. A detailed monthly performance report was also generated containing performance indicators for various loops. The worst performing loops in the process area were thus identified enabling technical staff to concentrate on them. Another monthly report provided loop statuses of similar loops in various plants so that a plant manager could compare his/her plant's performance with others'.

These reports were found beneficial in locating process problems and their extents. For example, a loop in a process was found to be in a 'fair' state of performance. It was a critical loop expected to perform better and was investigated further. Contents of the detailed report suggested a hardware problem. A quick check of the valve in the loop showed adequate supply pressure and reasonable output pressure and the valve "looked OK". However, in an in-depth check, the valve showed hysteresis and poor calibration and was scheduled for maintenance in next shutdown. Paulonis and Cox claimed that results obtained from the system showed that many of the poor performing loops had hardware problems with valves, positioners, and transducers. This changed the mindset in the company and increased emphasis was reported on loop hardware maintenance. They claimed 66% reduction in troubleshooting time in the company. Off-class production in one process area was reduced by 53% and standard deviation had been reduced by 38%.

Although large monitoring systems have been implemented and reported to be successful, there are still monitoring issues requiring further research. For example, Paulonis and Cox

discarded an available “LoopScout” software package because it worked on Honeywell control systems only. This shows the requirement of a generic monitoring system that can work in all situations. Small to medium enterprises (SMEs) do not have specialist departments to construct big monitoring systems and require ready-made solutions. They can only afford generic systems that are configured to their specific requirements at installation time and may be updated from time to time. Specific configuration of a generic system would always be required but the main system components should either remain the same or should be modular and easy to change. Another issue with this system is the requirement of expert engineers to decide what maintenance action is required. This, once again, is possible only in big companies having their own control engineer employed specifically to locate maintenance issues from the generated reports and acquired time series trends etc. Low cost generic and modular solutions are thus required for SMEs that can provide automatic guidance alarms toward specific maintenance actions. Research reported in this thesis considers such requirements and a cost effective generic solution is proposed and developed for SMEs.

3.6 COMPACT MONITORING SYSTEMS

A compact system is defined in this thesis as a small sized system that can be placed anywhere in the process plant. It is preferable if the monitoring system can be embedded in the main processing system without consuming additional space on the plant floor. Embedded modules provide ease of installation and work close to the sensors improving the Signal to Noise Ratio (SNR). Researchers have developed embedded modules working in collaboration with a PC and an increasing trend in the literature can be seen towards total system implementation through embedded modules only without any additional computer on the shop floor.

Feng et al (2002) emphasized the use of PC technology in an embedded environment. They stated that PC technologies are commonly understood and are easier to use thus reducing development cost and time. The requirements of an embedded system are different from a desktop PCs as more real-time operations with greater reliability are required. Feng et al reviewed some common operating systems with an eye on real-time capabilities and identified some real-time operating systems (RTOS). They tabulated a

comparison between Windows 95/98, Windows CE, Linux, QNX, and DR-DOS operating systems. DOS was observed to be the most popular operating system for embedded applications with more than three million copies of DR-DOS sold since 1997. RTOS are, however, more expensive than their normal counterparts, except Linux which is freely available. The availability of hardware supporting PC technologies in embedded systems was also discussed. Various available options, such as PC/104, compactPCI, and SBC (single board computer), were detailed. The ease of using high level languages with networking protocols available as built-in libraries was found encouraging for quick development times. They observed that use of such tools reduce software development cost significantly which was otherwise the most costly part of an embedded system. Several processors may be connected to solve a complex problem giving rise to a distributed system of hybrid multiprocessors. Ethernet was favored over serial connection for inter-processor communication because of its higher bandwidth and the wide availability of its protocol implementations. Real-Time Ethernet and Real-Time Publish Subscribe were quoted as examples of newly emerging real-time Ethernet protocols. They proposed the implementation of HTTP servers in the embedded systems in addition to the work these systems are currently doing and supported the use of embedded PCs as a hardware platform for such embedded systems. The security of these Internet connected embedded systems is an important issue and Feng et al recommended the use of proxy servers to protect them from unauthorized access. This would also enable the system to use private Internet Protocol (IP) addresses instead of a public IP address for each embedded device. They stated that “to control or monitor the embedded system through a proxy server that runs an HTTP server and has access to the embedded system behind it, both Common Gateway Interface (CGI) and Internet Server Application Programming Interface (ISAPI) interfaces can be used” for Windows based servers. The ISAPI extension was considered more powerful than CGI having much better performance.

De Frutos and Giron-Sierra (2002) reported a distributed system implementation comprising of a PC and various distributed nodes on embedded PCs. The idea was to use the processing power of an embedded PC while using the ease of programming an Intel-compatible system. Flashlite boards by JKmicrosystems were used to implement the distributed nodes. The board was based on V25 NEC microprocessor with 512 Kbytes of RAM and 256 Kbytes of flash memory. It provided six digital inputs, six digital outputs, an eight-channel 12-bit Analog to Digital (A/D) converter and two 12-bit Digital to

Analog (D/A) converters. The program was written on the PC and then downloaded to the node. Object oriented programming was used for the system where a user interface was designed using visual tools for PC monitors. The PC was connected to a Bus Control Unit through a modem on a RS-232 port. The bus control unit connected the PC with the embedded nodes attached to the bus. A baud rate of 1200 was used for the system but higher baud rates were possible. Figure 3.6 shows the system connectivity model with an embedded node block diagram.

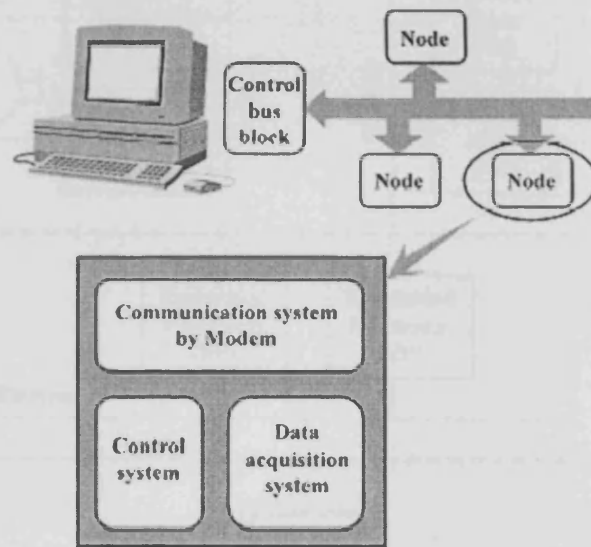


Figure 3.6: System architecture (De Frutos and Giron-Sierra, 2002).

Dassanayake et al (2001) proposed another layer between the distributed front end nodes and the PC making it a three layered system. This gives three levels for FDI namely component, machine, and system levels. Physical devices under control and monitoring were connected to Fieldbus Nodes (FN). An FN performs FDI at the component level by producing alarms when a signal value goes out of a set tolerance range. The tolerance range was provided to an FN by the corresponding upper layer Embedded Processor (EP). Several FNs were connected to an EP through a fieldbus.

An EP in the middle layer performed the machine level FDI on all the assets connected under it. It checked the consistency of an alarm and consistent alarms were passed on to the PC. They proposed to use a PC for Maintenance Information System (MIS) performing system level FDI. Several EPs were connected to a PC through the Ethernet. The MIS, on receiving an alarm, activated a direct high-speed data acquisition link to the faulty asset across the Ethernet and fieldbus routed through the EP. Features were

extracted from acquired data to identify symptoms. A neural network procedure was used to generate a recognized fault code. Figure 3.7 explains the proposed three layered architecture.

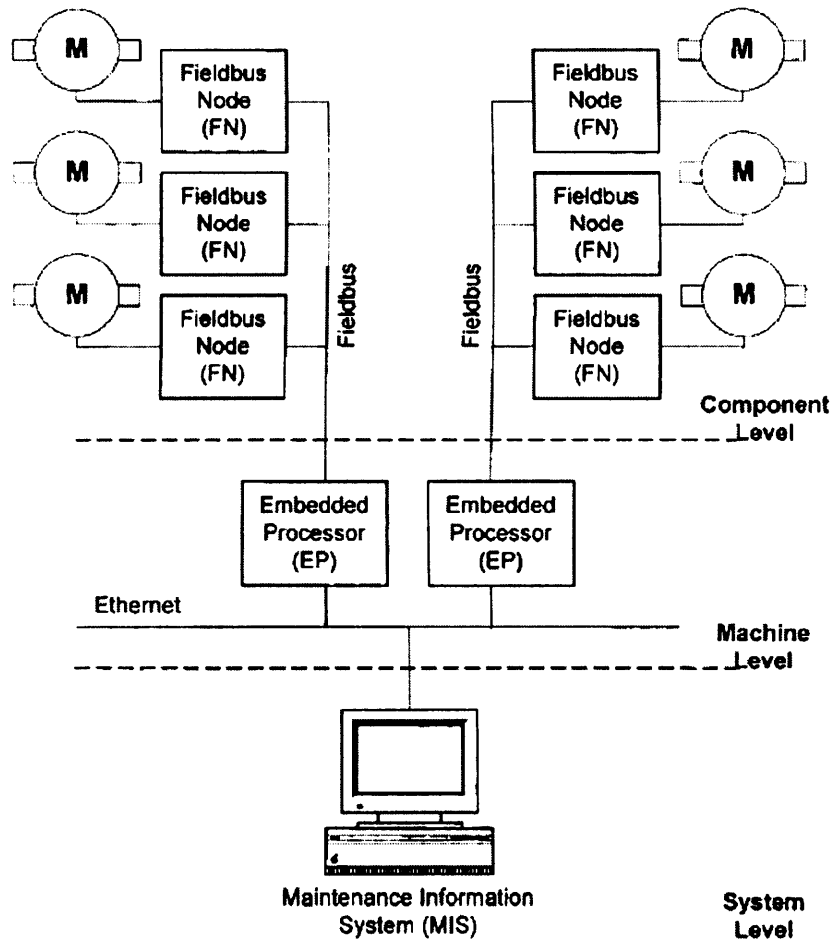


Figure 3.7: Proposed architecture for system-wide FDI (Dassanayake et al, 2001).

Fault detection and isolation for an automatic door was implemented on the proposed system. A digital signal processor (DSP) was used to control the motor responsible for moving the door and a microcontroller was used to provide its interface with fieldbus. The FN generated alarm when the motor drew over-current. A decentralized periphery fieldbus protocol was used to provide the alarm to EP. The EP calculated the controller effort index and the controller performance index by simple summation, subtraction and square operations. Upper and lower threshold values were applied to these indices for fault detection. Various system parameters were used as health indicating symptoms including time constant, damping, and peak response. A symptom vector was created from these symptoms and a self-organizing feature map (SOFM) neural network was trained accordingly to provide the FDI on MIS layer.

Installation of a monitoring system requires considerable effort. It included the mounting of sensors, their interfacing with the acquisition system, and cabling. Researchers have presented various ideas to make such an installation easier. Werneck and Abrantes (2004) proposed the use of live-line techniques for easy installation of power-distribution monitoring system. They described a quickly installable monitoring system for the temporary measurements of voltage and current in high-voltage power distribution systems. The portable system used live-line techniques and could be installed in the field in less than 10 minutes. The transducers provided the signals to an 8-bit microcontroller, 87C51, over a fiber optic link. The microcontroller calculated the voltage and current for three phases as well as the power factor and displayed it instantaneously. It also stored mean values every 5 minutes which remained available till the expiry of the system battery (typically ten days). Collected data from the installed system was either transferred to a notebook in the field or to a PC when the system was brought back to the office.

Valdastri et al (2004) explored the use of radio frequency for wireless data acquisition for monitoring systems. They described an implantable telemetry platform system for in vivo monitoring of physiological parameters. A microcontroller, rPIC 12F675F, interfaced with up to three transducers acquired the signals and transmitted them on a radio frequency. The signals were checked against threshold values and transmission was performed only for out of tolerance signal values. The telemetry transmission was obtained by using a carrier frequency of 433.92 MHz and an amplitude-shift keying (ASK) modulation. The microcontroller remained in low-power mode during normal signal values. This preservation of power and the small overall size (less than 1 cm³) made the system suitable for implantation in the human body. A transmission range of more than 5m was reported to be achievable from inside the human body. A wireless receiver connected the signal to a PC through a RS-232C serial link where it could be further processed or displayed. Physiological monitoring could thus be undertaken without restricting the patient's movements. The results were displayed on a graphical user interface developed in LabView. The system was implemented with pressure sensors and implanted in pigs. Its performance parameters, such as the transmission range and battery lifetime, were tested for gastric pressure monitoring of the pigs and met the design specifications.

Connecting all the modules of a distributed monitoring system using a digital communication bus saves clumsy cabling on the factory floor and is a cheaper alternative. Multiple modules may try to access this shared medium simultaneously causing conflicts and some measures are necessary to ensure the proper functionality under real-time constraints. Livani et al (1999) considered the use of CAN bus in dynamic, distributed, real-time systems. They considered it suitable for complex real-time applications because of its advanced built-in features. Higher level protocols are also available for CAN bus and were considered for distributed monitoring systems. They classified the activities on the bus as hard real-time, soft real-time, and non-real-time. They observed that fixed priority assignments are applied in the most common CAN based communication systems which are not very suitable for hard real-time demands. A flexible mechanism is therefore required using the CAN bus protocol at the lower layers of the network. Various methods for achieving the intended behaviour were considered and a hybrid scheduling mechanism was proposed. This mechanism combined the determinism of a Time Division Multiple Access (TDMA) method and the flexibility of dynamic Least Laxity First (LLF) resource scheduling. Each message was assigned a deadline and needed to be delivered within that time. The priority of a message was based on this deadline as well as its time in the queue. The priority of a message increased with the waiting time in this dynamic scheme. This hybrid scheduling mechanism achieved a higher resource utilization by reusing the redundant reserved times for non-critical communication. They aimed to exploit the inexpensive availability of 8-bit microcontrollers to implement the overall system and guaranteed up to 2754 hard real-time messages per second using their approach. An 8-bit microcontroller with 20MHz clock was expected to complete the required computations in 13 micro seconds with processing overhead limited to approximately 4% at 1 Mbps bus speed.

Bolic et al (2001) supported the use of microcontrollers over PC's for compact systems. They argued that although a PC is a good choice in most cases, there are applications where its power and resources are not required and smaller size and low cost solutions are more important. They proposed the use of 8-bit microcontrollers in such scenarios. They reported a microcontroller based distributed system for measurement and control applications consisting of one central node as master and multiple slave nodes. All the nodes were connected with an RS-485 serial bus. Slave nodes performed control tasks whereas the master node provided bus control and a user interface. Each slave node had

generalized control software whose parameters were defined by the user at run-time. It was stated that the same software suffices for all slave nodes and in all control conditions. The user interface consisted of four input keys and an alphanumeric display containing two 16-character lines. An options menu was displayed and the user selected various options for the settings of a particular slave node. The user chose a parameter value from the predefined list and was able to increment or decrement it before the final selection. The inter-node communication was based on a reduced OSI model consisting of physical, transport, and application layers. A detailed communication set up was explained where each node was assigned an address before connecting it to the system. The central node polls for all the 127 possible slave nodes on the system at regular intervals, detecting any new node available on the bus. Any new node can thus be connected to the system without disturbing other nodes. The user can then program the parameters for this new node and it becomes operational. The master node was implemented on AT89C55 and slave nodes on AT89C4051 microcontrollers.

Manders et al (2002) reported the implementation of a distributed measurement and control (DMC) application using components meeting the IEEE 1451 standard. It is a standard for smart transducers interface for sensors and applications that defines further components to accomplish various system tasks. IEEE 1451.1 specifies a Network Capable Application Processor (NCAP) which is an object-oriented information model representing the interface of an abstract transducer to a network. The standard IEEE 1451.2 defines a Smart Transducer Interface Module (STIM) that provides the plug and play capabilities at the transducer level. Manders et al presented an online model-based fault detection and isolation system for a multitank fluid system by implementing IEEE 1451 components. Six transducer nodes, each containing an STIM and an NCAP module, were defined in the system and communicated with each other over Ethernet. A STIM module was implemented on a microcontroller as suggested by the Microchip application note AN214 [Microchip web site, AN214]. An embedded Ethernet controller was used for NCAP as it provided the functionality as a built-in feature implemented through custom hardware. An off-the-shelf real-time embedded operating system (VxWorks) was used with a publish-subscribe mechanism implemented over IP/multicast. The fluid system consisted of three interconnected tanks with level, flow, and pressure sensors, control valves, and a fluid pump. One transducer node was dedicated to each tank and the main node subscribed to the data published by the other transducer nodes. The residuals

between the observed and the nominal behaviour were mapped into a symbolic form. The hypothesis generation algorithm, using temporal causal graphs (TCG), computed a set of possible fault candidates. The fault candidates were further refined using qualitative methods to reach a decision. The object oriented byte-compiled/interpreted language “Python” was used for software development. The experiments were devised in such a way that the dynamics of the model never exceeded first order behaviour. They observed that due to the various limitations on the network code and the data publishing rate, the effective sampling rate of the system was limited to one sample per second. Another problem affecting fault detection adversely was the system’s inability to time stamp the acquired data accurately. This reduced the system’s sensitivity towards faults and the experiments were devised with sufficiently large faults in the system to overcome these problems.

Lee and Hsiung (2004) considered the importance of software in embedded systems. They stated that embedded software now accounts for as much as 70% of total system functionality reducing overall cost and providing flexibility towards up-grading and ease of maintenance. Higher dependency on software also means more complicated software which may be difficult to synthesize and debug. They proposed the use of graphical modelling tools for embedded code generation and verification. They used Complex Choice Petri Net (CCPN) in their proposed synthesis and prototyping system because of its high expressiveness. Possible events were modelled in the form of Petri net places and transitions, providing a straight forward means of developing error-free codes. A 89C51 microcontroller based circuit was implemented for testing the generated code with an FPGA providing hardware emulation for various applications. The proposed system effectiveness was shown through two sample applications, namely, a vehicle parking management system and a motor speed control system. They also proposed the use of multiple threads in embedded software, rather than the commonly used single thread approach, as it preserves user conceivable concurrencies among the tasks. A real-time operating system would be required for scheduling multiple concurrent tasks in an embedded system.

Frankowiak et al (2005) provided a detailed review of developments in the important elements that make up a monitoring system. They considered sensor-based and non-sensor-based approaches and discussed the use of intelligence in this context. Various

monitoring methodologies were explained and evaluated on a cost and performance criteria. Effects of progress in technology were considered in this field and centralised and distributed system implementations were reviewed with PC, DSP, and microcontroller technologies. The role of the Internet in future monitoring was stated to be very important and remote performance monitoring was favoured.

3.7 EXAMPLES OF WEB-ENABLED MONITORING

Internet connectivity is widely available these days and its use is popular with technical as well as non-technical people. It provides an easy method of communication resulting in remote data access. The data can be processed remotely by powerful computers and the results are displayed via an easy to understand graphical user interface. This section looks at research related to web-enabled monitoring systems.

Bonastre et al (2001) reported the development of a distributed expert system, which they claimed to be the first one in analytical chemistry. The system consisted of four Control Nodes (CN) and a Programming and Supervision Node (PSN) communicating with each other on CAN bus. The PSN was implemented on a PC and provided an Internet connection to the system. The control nodes performed their relative control functions and informed the PSN about the change by updating the global variables. The PSN then informed other CNs about the changes. It also decided the frequency of analysis, compiled status reports and displayed them on the Internet as secure web pages. The system was tested on a wort fermentation process in a laboratory experimental plant. As the time gap between analyses in such applications run in hours, the system was claimed to give good results in real-time. The described system provides the monitoring results of various chemicals in the process and leaves the fault detection and diagnosis to the humans. A more useful system would provide some suggestions or guidance towards some specific faults or maintenance issues rather than just reporting the status of the chemicals.

Yang et al (2003, B) observed that most of the research work on Internet-based process control has resulted in small scale demonstrations mostly developed in Java. They concentrated on developing guidelines or systematic design methods for such systems as

little work has so far been done in this regard. They aimed to develop a methodology for the design of Internet-based control systems for process plants. They concentrated on adding an additional Internet control layer on top of the existing layers rather than engineering a total replacement. This approach was applied to a water tank system teaching rig to evaluate the methodology. The system was implemented using Java applets and Labview virtual instruments. They claimed that a server push mechanism was used with Internet Explorer to increase communication efficiency and to reduce server loading. As a matter of fact, Internet Explorer does not support server push technology [(Musciano and Kennedy, 2000), (Cunningham and Cunningham Inc web site)]. Apparently client-pull technology was embedded in the dynamic web pages to refresh the web page at regular pre-defined intervals. This may have created the impression of server push being used which is actually a different technology.

Bucci and Landi (2003) proposed a distributed architecture for industrial applications using three hierarchical communication levels: the fieldbus, the intranet, and the Internet. Remote measurement units (RMUs) formed the front end of the system that acquired the signal and provided it to a fieldbus server (FS) after necessary processing. Each RMU consisted of three modules each responsible for one task, i.e. signal acquisition, processing, and the fieldbus interface. The FS acted as master for communication with RMUs and obtained data from each of them sequentially. The RMUs did not communicate with each other although they were connected on the same bus. An RMU implementation was based around a 32-bit microcontroller from Hitachi with external memory. The board had a size of 220 x 110 mm² and costed about \$100. It was attached with a 20 x 4 lines display and a 16-key keyboard as local user interface. Several RMUs, or WMUs for wireless connection, connected with a fieldbus server constituted a measurement site. An FS handled data storage, analysis, display, report generation, and data sharing for a measurement site. Several measurement sites were interconnected using a LAN where personal computers provided the required processing power and management applications. The LAN had a measurement server performing advanced data logging, supervisory control and analysis. A Gateway computer was used to connect the LAN to the Internet providing the security. The system was designed with an aim to support dynamic web pages managed by an Apache server so that remote users could access the latest information. The fieldbus interface was based on the RS-485 protocol with a data transfer rate up to 38400 bps. The system performance was evaluated, for

power quality in an electrical distribution network and for management of a water distribution system, and was found to be well suited for such applications in terms of cost and performance.

Eady (2004) discussed the issue of TCP/IP stack implementation on microcontroller based systems. The resources in a microcontroller are very limited and the TCP/IP stack required for communicating on the Internet puts a heavy burden on these resources. Eady described the options for the microcontrollers in this regard and emphasized that a simplistic TCP/IP stack might suffice for small systems. The stack should be a modular one and only the modules required for a specific application should be included. He detailed the use of CMX-MicroNet, which is a TCP/IP stack designed for use with microcontrollers. It supports up to 127 UDP or TCP sockets. Its price for small system developers may be an issue, however, as it starts from \$5500.

Insam (2004) explored the development of fast Ethernet access from an 8-bit microcontroller stating that “it’s difficult enough to get a 10-Mbps Ethernet controller working anywhere near full speed when paired with a small microcontroller”. Getting the speed of 100-Mbps is far more difficult. He considered the use of FPGA for this reason but dropped the idea after some analysis. He supported the use of microcontroller in embedded systems over FPGA as its software development is more result-effective on a par-to-par comparative basis. The FPGAs were used in the system though to provide faster communication in block data transfers. Insam explained various differences in the 10 and 100 Mbps Ethernet standards but expressed satisfaction that there were few differences between writing the IP code for both systems. An 8951 microcontroller was used with the SMSC LAN91C111 Ethernet controller and ACEX EP1K50 family FPGA for the development.

Stipanicev and Marasovic (2003) reported the use of an 8-bit microcontroller as a web server providing dynamic web pages. They suggested that small systems do not need the full power of a desktop PC to display their results to a remote user on the Internet. Greenhouse monitoring and control was provided as an example of such systems. Various sensors in the greenhouse provide condition information and necessary control actions can be initiated by remote user by selecting suitable options on the displayed web page. Switching a “wetting” system on or off is an example for such control actions. A

DS80C390 microcontroller based embedded computer, TINI, was connected to the Internet using Point-to-Point protocol through a modem and hosted dynamic web pages. The TINI was connected to various sensors and actuators in the system on a one-wire network. The system was tested for up to 30 sensors and was reported to be successful. The TINI recorded the information provided by the sensors in its file system and served this information to the interested user. The web pages were constructed on the fly to provide latest information. The TINI has a multitasking operating system supporting multiple threads. It also has a rich library supporting network protocols from Java APIs. The program was therefore constructed as multiple servlets running in parallel. Data was archived in a local file, but memory restrictions apply to the embedded system. It was suggested that stored information should be pushed to some other computer as emails using Simple Mail Transfer Protocol (SMTP). System security was provided by enabling user ID and password mechanism.

Al-Habaibeh et al (2003) described the development of a diagnostic system for royal mail automatic sorting machines designated as Integrated Mail Processors (IMP). IMPs are complex electromechanical systems including enormous number of rollers, bearings, belts, gears, motors, electronic systems, etc. The Royal mail delivers about 82 million items of mail and parcel post every day. Accordingly, IMPs do an enormous amount of work and generate a lot of heat. A microcontroller based monitoring system was developed to check the generated heat and any abnormal patterns were detected for FDI. Infrared imagers were interfaced with microcontrollers connected to the Internet. PIC 16F877 microcontrollers were used in this application with the PICDEM.NET module from Microchip Ltd used to provide Internet connectivity. UDP/IP protocol was used to relay acquired data on Ethernet for processing by remote computers. The use of microcontrollers provided a low-cost acquisition system in this application.

Yang and Eagleson (2003) described a temperature control and monitoring application through the Internet. A remote user provided a desired value of temperature and a microcontroller based embedded system set the temperature in a tube accordingly. Heating and cooling were provided by a lamp and a fan respectively. The software was designed using unified modeling language (UML) classes and was implemented in C++ language. An HTTP server presented a web page to the remote user to input desired temperature. The dynamic web page also displayed the current temperature of the tube.

Server Side Includes (SSI) were used with HTML code to implement these features. A SMTP handler was also implemented and the system sent alarm emails for out-of-control situations.

A system on the Internet is vulnerable as it is accessible to hackers and intruders. Proper security is required against unauthorized users to minimize malicious attacks. Hackers may intrude into an Internet control system and change the settings for process controller causing undesirable effects. Axelson (2004) discussed network security issues for small embedded systems, which do not have enough processing power and memory resources to employ full-blown security encryption techniques. He stated that a firewall may be the first line of defence and an embedded system may be behind a firewall provided by the company LAN. Besides security reasons, this is often the case of network implementation in a company. The firewall provides security by hiding the local processors' IP addresses from the Internet and by allowing only the required services. Restricted access, based on username and password authentication, was urged by Axelson to provide further security, or in case a firewall was not available/suitable for the application. This kind of authentication can be implemented using simple HTML code with a HTTP POST request. This method does not encrypt the password though and anybody having access to the network traffic can see it. Use of some encryption technique was considered a better option therefore. Axelson stated that a server can also limit the number of tries from a single IP address in order to prevent a determined hacker trying different username and password combinations. Axelson stated that encryption used in basic authentication is the Base64 Content-Transfer-Encoding method described in RFC 1521. Digest authentication was considered more secure but more complex to implement. He also described how these techniques work and provided information about the support available for them in C and Java compilers for small embedded systems. Another security issue with these techniques is that password protection is applied on user identity only and not on the requested resource itself. Resource data encryption is required separately, if deemed necessary. He proposed the Advanced Encryption Standard (AES) as a suitable encryption method for small systems. Another recommended method is to use a firewall device with support for a Virtual Private Network (VPN). This method relieves small devices firmware from security issues. Another method of attack by some malicious users is to enter Server Side Include (SSI) directives in the authentication form fields, making the server do unwanted things. Axelson proposed disabling SSI directives in normal

HTML pages to prevent this kind of attack. The server would allow only secure SHTML pages for SSI directives in that case. Axelson observed Secure Socket Layer (SSL), used in online banking, as the most common security method. He considered SSL very secure but requiring resources beyond most of the small embedded systems. For wireless networks, Wi-Fi Protected Access (WPA) was considered more secure than Wired Equivalent Privacy (WEP).

3.8 COMMERCIAL SERVICES

This section provides some examples of commercially available systems providing process and condition monitoring as aids to improve process effectiveness. It shows that the technology is still not mature enough to be widely deployed as ready-made systems.

Keyif et al (2004) described a commercially available monitoring system to check the health of an electrical motor. Motor condition monitor (MCM) was claimed to be a result of 20 years of research and can detect developing faults in plants with motor-based machinery. It takes motor supply voltage and current as the only inputs and provides output as one of the five statuses. It can also provide information about the frequency contents of the acquired signals. The results are available on a display panel as well as on a serial port which may be interfaced with a computer. It detects signal pattern difference between existing and nominal signals where the nominal signal is obtained using a model. MCM makes this nominal model during its initial learning phase. It is available as an easy-to-install box of size 90x90x195mm. Keyif et al also provided some examples where this patented monitoring technology was successful in picking up developing faults.

Walchem Corporation's WebAlert is a remote monitoring device that web-enables already installed equipment (Walchem corporation web site). It has an embedded web server which allows it to function like a website. It can be connected with up to six 4~20mA and two digital signals providing data logging facilities. The data from these signals can be acquired and displayed on the website in real-time. It provides Internet connection via Ethernet or modem. To access its web page, a user has to first login at the Walchem corporation web site by entering his user ID and password. WebAlert then connects to the local Internet Service Provider (ISP) and logs onto the Internet. The user

then enters another set of user ID and password to access the data (Walchem corporation WebAlert web site). The web site also claims that it is capable of sending alarms as SMS messages to mobile phones but does not explain when and how.

Divan et al (2004) described a web-enabled near-real-time monitoring system for power quality and reliability. It collects power events and sends event data via the Internet to system database servers using a modem. Powerful computers at the central server take up much of the processing workload. The sensing unit was implemented on a Texas Instruments C54x DSP that communicated at 14400 bits/sec and had an HTTP/TCP/IP stack. The system was claimed to be designed and tested for scaling to over 50000 such sensors. The users are offered a variety of options to configure their displays according to their requirements. Java 2 platform enterprise edition was used and sensing unit software can be upgraded through the modem. Sensing units were manufactured on a commercial scale and over 1000 production monitors were reportedly deployed in US and Canada. The system recorded about 120000 events in the first 2 years of deployment. Divan et al claimed that this ultra-low-cost solution was probably the cheapest commercially available option with less than \$500 cost (SoftSwitching Technologies web site).

ProHelp® EPM is the real-time production and process monitoring system from Mattec Corporation (Mattec corporation web site). It is capable of monitoring up to 4096 machines and supporting hundreds of users simultaneously. It covers industries such as plastic injection molding, extrusion, blow molding, metal stamping, die casting, printing, painting, assembly, etc. It runs on Windows 2000 or higher and uses Microsoft SQL server 2000. ProHelp EPM provides email and voice alarms when a machine is either down or is out of specifications (Mattec corporation ProHelp web site). Mattec's ProStat® SPC/SQC, a real-time statistical process control software, is fully integrated in ProHelp EPM providing enhanced functionality and ease of use. Another version of software, ProHelp Millennium, runs on UnixWare, a PC-based version of Unix.

THE-MAN-A-ger© is another production monitoring system from Mattec. It is considered low-cost by the company as the base price is less than \$10,000 (Mattec corporation THE-MAN-A-ger web site). The user has to provide a PC with Windows 2000 and SQL server 2000 whose price is not included in the package. THE-MAN-A-ger can monitor cycle time, run-time, downtime and scrap levels for up to 64 machines in

real-time if several 'data concentrators' are used. A data concentrator is the machine interface unit that can be connected to a maximum of 12 machines.

Aspen Watch™ is a controller information system from Aspen Tech for performance monitoring and extracting useful information from large volumes of data (Aspentech web site). The complete data is stored in a database in an uncompressed way. It is therefore possible to reconstruct any controller action and events can be fully investigated. The data can be monitored remotely through a high speed connection such as ISDN. It also provides a graphical user interface to display trends and visualizations. It highlights the statistical performance of the controller over time and compares it with the best possible performance. It calculates the performance parameters on-line and generates reports which can be retrieved by the user on-demand. The software runs on the Windows NT operating system. An additional benefit up to 10% is claimed by using Aspen Watch.

Honeywell's Loop Scout tool collects configuration, event, and time-series operating data from the process plant and suggests maintenance and engineering actions to resolve the worst-performing loop (Honeywell process solutions web site). It helps increasing plant production rates and reduces the time to identify and address poorly performing control loops. The 'Loop scout overview' provides aggregate performance benchmarks for unit-wide or site-wide evaluation. The 'loop scout detail' provides individual loop metrics and diagnostics (Honeywell Loop Scout overview web site). Another service, Alarm Scout, automatically collects performance data on a system's alarms and events and stores it locally. An operator then performs a login at the alarm scout website and upload the data. The service analyzes the data and generates an alarm status report which is sent to key stakeholders in emails. The alarm scout service works only on the Honeywell systems such as GUS/APP node, PlantScape, and Experion PKS (Honeywell Alarm Scout web site).

Matrikon's ProcessDoc is a control loop performance assessment and monitoring tool that claims to provide improved overall operability and stability of process units, increased throughput, reduced maintenance costs and improved product quality without additional capital investments or IT infrastructure (Matrikon web site). It claims to achieve a 5% increase in plant performance through improved control and a 30% reduction in maintenance cost through a condition monitoring approach. It provides the continuous

online monitoring of processes in real-time. It claims to detect performance losses automatically and prepares reports accordingly. It also provides tools to diagnose the problem causing the performance loss and to fix the problem. Matrikon ProcessDoctor web site states the ProcessDoc uses proven technologies and mentions several success stories. The quoted examples include a \$100,000 increase in annual revenue for a major refinery, a 20% increase in run-time for a polymer producing plant, a 28% reduction of variance in the grinding area and 34% reduction in variance in flotation area in a large copper mining operation, and an annual cost saving of around \$400,000 in a large pulp and paper mill. The Matrikon clients list includes companies such as BP, Saudi Aramco, Bayer Polymers, GE Plastics, Mitsubishi Chemical, Newcrest Mining, NRG Energy, Alpac, etc.

Emerson's PlantWeb is the digital plant architecture that uses predictive intelligence to improve plant performance (Emerson Process Management, Plant Web web site). It enables the user to detect process and equipment problems even before they occur (Emerson Process Management, Results web site). It provides a networked approach using the Foundation fieldbus and employing standards at every level. It is engineered to efficiently gather and manage information from intelligent field devices. The details of the PlantWeb and its associated asset management software can be found in Hartley (2002).

3.9 SUMMARY

This review has shown that monitoring systems are still limited, mainly to the data acquisition systems that present the acquired system to human experts for decision making. The ready-made systems that provide complete monitoring solutions by reliably identifying the developing faults in the processes are almost non-existent. There is a huge requirement for a generic system that can be widely deployed in various process plants and industries. The system should be a low-cost one so that it can be deployed by small to medium enterprises (SMEs) as well as big companies. A modular design would be required for the generic system so that only the required components be included in the installed system. This would reduce the total system cost as well. The ease of upgrading the system with changing time is another desired feature in such a system.

SIGNAL ACQUISITION & ANALYSIS

The need for a generic, low-cost, and modular monitoring system was confirmed as a result of the review of existing monitoring systems and research in the previous chapter. This chapter presents the basis on which a low-cost compact monitoring system was developed. The motivation for selecting a distributed network of nodes over a centralized approach will also be reported. The chapter also details the signal acquisition requirements and those of the subsequent time and frequency domain analyses for such a monitoring system.

4.1 MONITORING SYSTEM OVERVIEW

In general, a monitoring system requires current information about the process to enable monitoring decisions to be made. This information must be obtained via sensors measuring various system variables which have usually been interfaced to the processing system through the use of cables. The cabling system can be an expensive part of a centralized processing system and can cause logistical problems in establishing the system. The prevalent industrial noise will also need to be addressed. In recent times the trend has been to replace the analogue signal wiring 'mess' with an organized digital communication bus (generally known as a fieldbus). All sensors communicate on the same bus with time-shared access. Sensor signals are converted into digital format close to the source before noise can significantly affect them. The author has selected Controller Area Network (CAN) bus as the digital communication medium between various nodes of the proposed distributed monitoring system partly due to its superior performance in noisy environments and also because of its availability in embedded microcontrollers.

In general, some digital electronics are required to convert an analogue sensor signal into digital format and then to communicate this information on the fieldbus. There will be advantages if the size of this conversion electronics can be minimized. For the system developed in this research, microcontrollers were identified as compact devices, sometimes called 'one-chip' computers, which integrate facilities and yet provide limited

processing power. In particular, the author has used low-cost 8-bit microcontrollers in a distributed monitoring system, thereby providing an additional benefit of front-end processing capabilities. Figure 4.1 summarizes the hardware architecture of the distributed monitoring system design. Thus a signal can be checked for basic characteristics at acquisition and an immediate decision on the normality of the signal is made possible. For example, a simple threshold check on signal amplitude can often be sufficient to distinguish normal and abnormal behavior. Information from several such microcontrollers is combined to reach robust conclusions when multiple signal information is required. This combined information formed the second processing layer in the developed distributed monitoring system. A hierarchy was accordingly evolved for processing layers where the front-end (acquisition) nodes (FEN) constitute the first or bottom layer. It is expected, from previous experiences in the IPMM group, that up to 80% faults may be detected at this level. A further 16% faults may be detected at second layer of hierarchy where information from various front-end nodes (FEN) is combined. At the third or top layer of hierarchy, specialized computers may be used to determine the remaining 4% of faults. It was decided that the user interface should be dealt with by a special node, which conveys user commands to other nodes, thus ensuring system synchronization in addition to presenting results to the user. This node was called the Synchronizing & User Interface Node (SUIN). Internet and mobile phone connectivity was also deemed as needing to be as part of the user interface in this research.

Figure 4.2 represents the hierarchical philosophy which can be summarized as follows: The approach taken in this study was to check the health of an acquired signal at the first hierarchical layer (FEN nodes). A quick normal/abnormal check is made and data is discarded for normal signals. Any detected abnormality is communicated to the second layer where the health status of all available signals are combined to robustly determine the cause of any detected abnormality. Detailed analysis may then be conducted by the third layer of the hierarchy (on server-side computers with much greater processing power) for cases where second layer processing proves inconclusive. This research concentrates on the development of the first and second layers and on consideration for data presentation to the third layer for processing when required. The processing required at the third layer is thus considered beyond the scope of this research and requires higher level software.

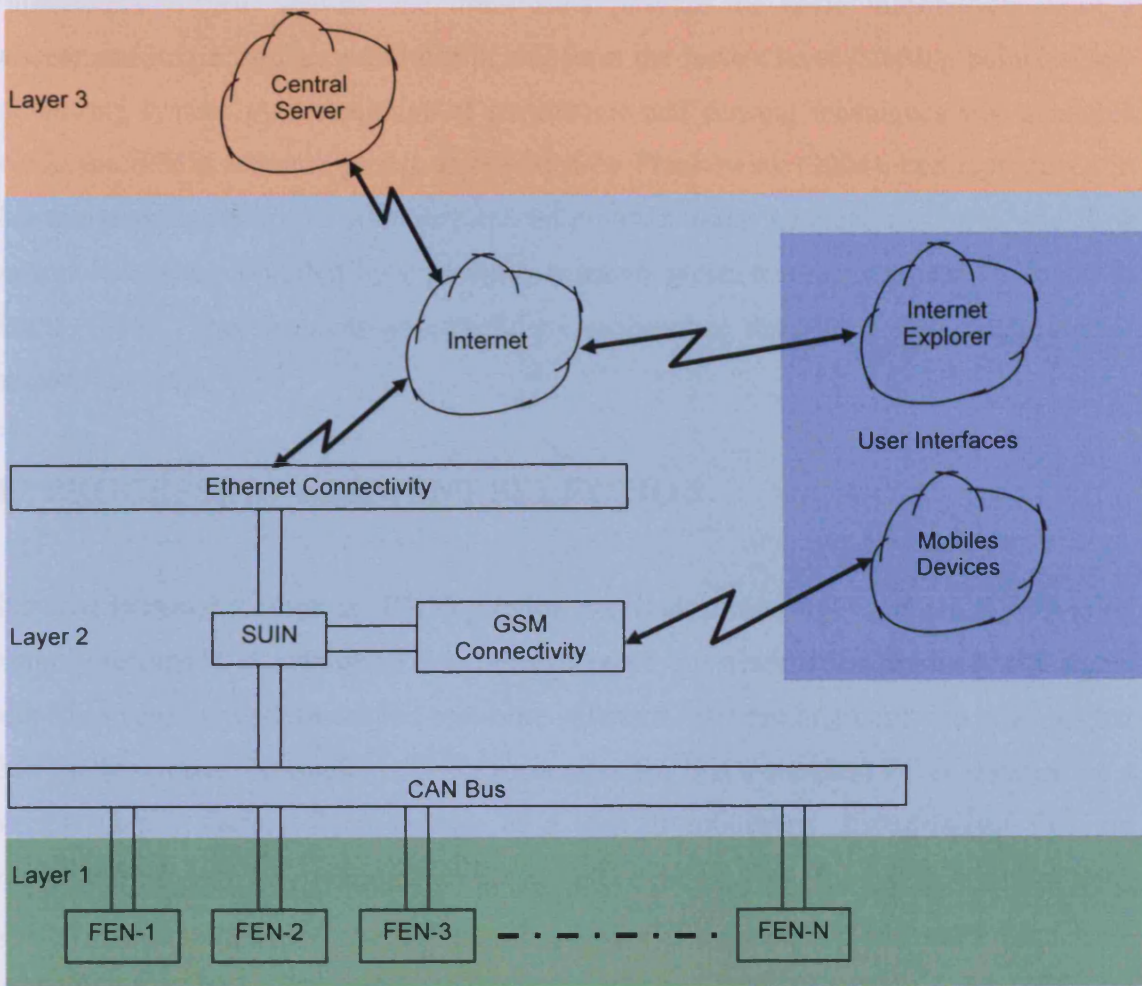


Figure 4.1: Hardware architecture of proposed distributed monitoring system.

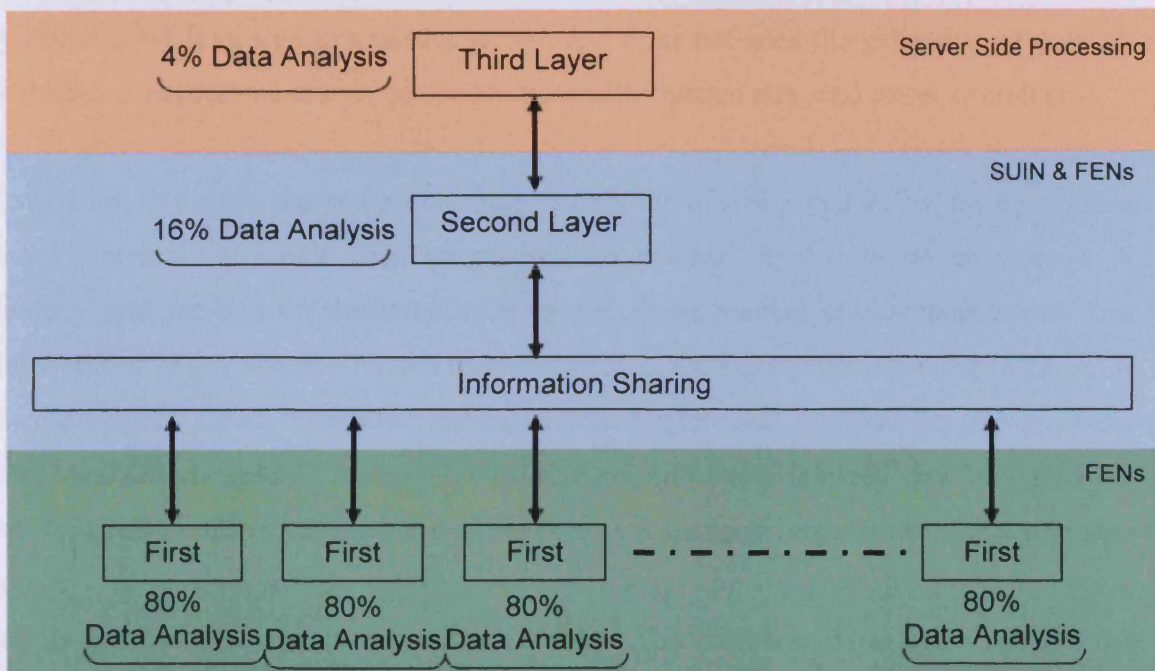


Figure 4.2: Computing hierarchy of proposed distributed monitoring system.

Fundamentally then, sensors and transducers provide the basic information about a process and its performance and health and form the lowest level (starting point) of any monitoring system. A compilation of transducers and sensing techniques was available within the IPMM research group, as provided by Frankowiak (2004), and reproduced in this thesis as Appendix A. Also detailed information about temperature, level, and flow sensors had been compiled by a previous research group member and can be found in Sharif (1999). These sources were usefully considered by the author when designing the proposed system.

4.2 PROCESSING ELEMENT SELECTION

Standard Personal Computers (PCs) provide excellent processing capabilities with good memory resources. A standard PC is not expensive but when it is combined with signal acquisition card(s) and associated real-time software, the resulting costs are much higher than the base price. Normally it is not recommended that a standard PC is installed on a process plant / factory floor because of a lack of robustness. Industrialised PCs are available for industrial environments but at higher prices. The physical footprint of a PC typically consisting of a Central Processing Unit (CPU), monitor, keyboard and mouse may also be an issue. The author therefore considered PCs not to be a good option for process monitoring applications and decided to use embedded systems. An embedded system can be dedicated to a particular task and does not need the generality of a PC. It will have a reduced number of components, smaller system size, and lower overall cost.

Having decided upon the embedded route, the choice of embedded PC's, microprocessor based systems and single chip microcontroller systems needed to be considered. A guiding principle was the desired need to minimize the number of components, and thus the footprint of any acquisition and front end node at the deployed monitoring function. A microprocessor-based embedded system required additional components for memory, peripheral devices, and I/O ports. The resulting circuit board is much smaller than a PC but it is still cumbersome to place these boards close to the sensor. A microcontroller contains all the peripherals, required for a small system, on a single chip and its use greatly reduces system size and associated cost. The compactness of size provided the opportunity to place the circuit board close to the sensor providing better signal

acquisition and analysis opportunity. The author therefore employed microcontrollers in this research.

Limited Silicon area on a single chip limits the possible features in a microcontroller. The amount of memory built into a microcontroller has increased in the last few years but is still very much restricted. Built-in peripheral devices, ports, and memory do not leave ample Silicon area to implement very powerful processing engines in microcontrollers; 8-bit processing engines are generally built in the microcontrollers. These 8-bit microcontrollers thus provide small low-cost circuits albeit at a fraction of a normal microprocessor's processing power. These microcontrollers are traditionally used in Input / Output (I/O) applications where a lot of I/O activity is supported with a little processing. The newer generation of microcontrollers, however, provides relatively higher processing capabilities because of improvements in architecture and clock speeds.

The acquisition of various process signals may give rise to a distributed system of microcontrollers communicating with each other and integrating information to form a holistic view of the process status. The author reviewed various available MCUs from various companies in order to select the most appropriate one for signal acquisition, processing, and onward communication of results. Microchip's PICTM (Peripheral Interface Controller) 18F458 MCU was selected because of its built-in ADC, digital I/O ports, memory, and various communication interfaces including CAN. In the following section 4.3, a general introduction to various PIC families along with a more specific insight to the capabilities of the PIC 18F458 microcontroller, is provided.

4.3 PIC MICROCONTROLLERS

Microchip Inc. have developed a large number of commercially available microcontrollers and its Peripheral Interface Controller (PIC) series offers a wide range of options to the design engineer [Microchip web site]. The PIC MCUs are available with permutations of maximum clock speed, internal memory size, instruction width, peripheral devices, interfacing protocols, etc. It is now possible to get an IC that has the optimum features for a particular application, yet being general-purpose so as not to

hinder the design views of the engineer. These features make PIC microcontrollers a very popular choice worldwide and Microchip is a leading supplier of 8-bit devices (Embedded Star web site). According to the yearly Gartner Dataquest rankings, Microchip was placed 20th in worldwide unit shipments in 1990 and rose steadily to number one by 2002. Table 4.1 shows some characteristics of various 8-bit PIC microcontroller families.

Family	PIC10x	PIC12x	PIC14x	PIC16x	PIC18x
Size (Pins)	6-8	8	28	14-80	18-84
Max Speed (MHz)	4-8	4-20	20	10-40	40-48
Program Memory (Bytes)	384-768	768-3584	7168	768-14336	0-131072
Data Memory (Bytes)	16-24	25-128	192	24-368	256-3968
Number of I/O Pins	4	6	20	6-53	16-72
ADC Resolution (Bits)	0-8	0-10	8	0, 8,10,12	10
Number of Timers	2	2-4	3	2-4	3-6
Serial I/O	None	None	None	USART, I ² C, SPI, USB	USART, I ² C, SPI, CAN, USB

Table 4.1: 8-bit PIC microcontroller families' characteristics [compiled from the Microchip web site]

4.3.1 PIC 18F458 Microcontroller

The author selected the PIC 18F458 MCU as the front-end node for the distributed monitoring system. This selection was based on the fact that PIC 18F458 contained the maximum of the features required for the proposed system. It was, at the time of selection, the best microcontroller available in the PIC series and provided digital and analogue signal acquisition capabilities and appropriate communication facilities. Both rising and falling edge detection is possible for individually selectable digital I/O pins and certain pins can generate interrupts on a change of voltage signal. The PIC 18F458 contains a built-in 10-bit successive approximation ADC with a maximum sampling rate of 30K samples per second. Up to 8 input channels can be connected to ADC under software control and reference voltages can be selected either internally or externally. Four timer modules are available to generate sampling rate, pulse width modulation (PWM) signal, time based interrupts etc. Other interrupt modes are also available, importantly including ones generated by CAN message reception and ADC conversion completion. The PIC 18F458 can work with clock frequencies up to 40 MHz and most of

its instructions operate consistently at four clocks per instruction cycle, resulting in a maximum execution speed of 10 million instructions per second (MIPS). A built-in hardware multiplier performs 8x8 operations in one instruction cycle and enhances the mathematical processing capabilities of this 8-bit microcontroller.

PIC 18F458 microcontrollers use Harvard architecture, where memory is divided distinctly into 'Program' and 'Data' memories. Coded programs of up to 16K instructions in length can be stored in the 32 KB built-in flash memory in this RISC (Reduced Instruction Set Computer) architecture. The program memory can be written to using a low voltage InCircuit Serial Program™ (ICSP™) option as well as under the program being executed inside the microcontroller. The self programming capability provides the possibility of software upgrading (eventually via the Internet in the developed system) and is very useful. The 1536 bytes RAM and 256 bytes EEPROM constitute the built-in data memory and are used to store program variables and long term temporary variables, such as configuration settings, respectively. In addition, the PIC 18F458 contains a number of special function registers (SFRs) which deal with various peripheral devices and interrupts reducing the need for RAM storage space for variables. The RAM is divided in various banks and bank switching is required to access a certain bank. This limitation is however softened with an 'Access RAM' area that is accessible irrespective of the current bank selection.

The microcontroller provides several options for serial communication. These include an addressable Universal Synchronous Asynchronous Receiver Transmitter (USART) (that can be used to implement RS-232 standard for communication with a PC COM port), a built-in Master Synchronous Serial Port (MSSP) module (that provides Serial Peripheral Interface (SPI™) and Inter-Integrated Circuit (I²C™) protocols) and a built-in Controller Area Network (CAN) controller (used in this research for inter-node communications). The Microchip schematic block diagram for a PIC 18F458 microcontroller is provided in appendix B for reference.

The author used assembly language when developing software programs for the microcontroller. Microchip's MPLAB software was used as an Integrated Development Environment (IDE). MPLAB integrates the editor, assembler, linker, simulator, and microcontroller programmer facilities. It provided debugging support with breakpoints,

software watches, memory maps, single-step and animate execution modes. The MPLAB In-Circuit Debugger 2 (ICD2) provided emulation support for testing the software on the target hardware circuits. A PICDEM 2 PLUS demo board was used with the ICD2 for all the initial development and testing of acquisition and analysis routines. This board provided digital and analogue input ports, input switches, LEDs, a buzzer, and a 2 line LCD display.

4.4 SIGNAL ACQUISITION

As stated, PIC 18F458 microcontrollers were used in this research to acquire process signals and as the heart of a general purpose node interfaceable with several types of sensors and transducers. The usage, as an acquisition tool, for discrete and analogue signals is explained in the following sections.

4.4.1 One Analogue Signal per MCU

As recounted in the monitoring system overview (section 4.1), the use of microcontrollers results in compact circuit boards for the front-end node (FEN) signal acquisitions. The compactness provides the opportunity to locate the circuit board close to the source. Indeed, the current generation of 8-bit microcontrollers when implemented with surface mount technology is so small that their circuit boards may be placed inside the sensor assembly. The author therefore determined that each microcontroller should acquire only one analogue signal in the developed FEN node, although it is capable of acquiring several signals simultaneously. The sensors used in this research did not contain MCUs in them and (non surface mount) FENs were located close to them. The further reduction in physical size of an embedded MCU (and the elimination of the requirement of converting the physical parameter into 4~20mA format, for example) was outside the current remit and is discussed in chapter 10.

4.4.2 Digital Signal Acquisition

Digital signals provide information about discrete events in the process such as a switch on/off or the start/end of a batch process. This information is generated by transducers typically as different voltage or current formats (in different applications) and appropriate signal conditioning is required to make them TTL compatible (0 to 5 volts) before their

interfacing with the MCU. The change in signal can be detected by polling the input pin or by generating an interrupt. The selected MCU handled interrupts on both rising and falling edges. A global flag enables the MCU interrupt system and individual interrupts can be masked or unmasked as required by the situation. High and low priorities can be assigned to various interrupt sources according to their nature and urgency. The PIC 18F458 supports only one Interrupt Service Routine (ISR) for each priority level and the programmer has to check various flags to ascertain the cause of an interrupt. This adds a burden for the software developer and makes interrupt responses slower than systems with more advanced interrupt handling capabilities.

Often it is necessary to detect the time between two monitored process events. With the PIC MCU system, this was achieved by starting a timer at the first event occurrence and stopping it on detection of the second event. A pre-set time was specified (take unit of 0.1 seconds) and the timer count automatically incremented. The timer count multiplied by the pre-set time period gave the total time elapsed between the two events. In alarm setting scenarios, the timer count was checked against a time-out value, specified in the ISR. If a time-out was detected a conclusion was made that the second event failed to occur. The MCU was operated at its maximum clock frequency of 40MHz and the interrupt rate was slowed down via 16 bit timer register initializations. These lead to an interrupt every 6.5536 millisecond. The interrupt rate could be further decreased by assigning various values for prescalers and postscalers with the available timers, which divide the input/output timer frequency by a scalar value.

The acquisition of pulse train type signals was possible simply by accumulating the number of rising or falling edges within a predefined duration. The MCU timer modules were used in their counter mode for such acquisition and incremented on every edge detection on their respective input pin. Often the accumulated number of edges were recorded over a 1 second interval and provided frequency value. Shorter accumulation periods were also used as appropriate.

Pulse Width Modulation (PWM) is another commonly used technique in sensors and transducers where the pulse rate of the signal remains constant but the duty cycle changes between a minimum and maximum according to the sensed information. There is no direct PWM input port available in the PIC 18F458. However, the basic time period of

the signal can be calculated by measuring the time between two successive rising (or falling) edges. The pulse width times for 0% and 100% duty cycle signals were also measured by detecting times between a rising edge and the following falling edge (or vice versa). The timer module was then initialized in such a way that the need for floating point division was eliminated, making the system simpler and faster. This method provided a quick detection of the duty cycle with resolution of 1%.

4.4.3 Analogue Signal Acquisition

Most signals in the process industry, typically representing flow, pressure, level, or temperature measurements are analogue in nature. These signals have to be converted into their digital equivalents before processing by a computer or microcontroller. Analogue to Digital Converters (ADC) provide this transformation. Analogue to digital conversion may be viewed conceptually as a three-stage process: sampling, quantisation, and coding as shown in figure 4.3. Full signal information $x_a(t)$ can be regenerated from samples $x(n)$ according to Shannon's theorem if the sampling frequency is at least double the highest signal frequency. The difference between the unquantised sample $x(n)$ in figure 4.3 and the quantised output $x_q(n)$ is called the quantisation error and is irreversible. The precision of an ADC therefore depends on the number of quantisation levels. Further details can be found in Proakis & Manolakis (1996).

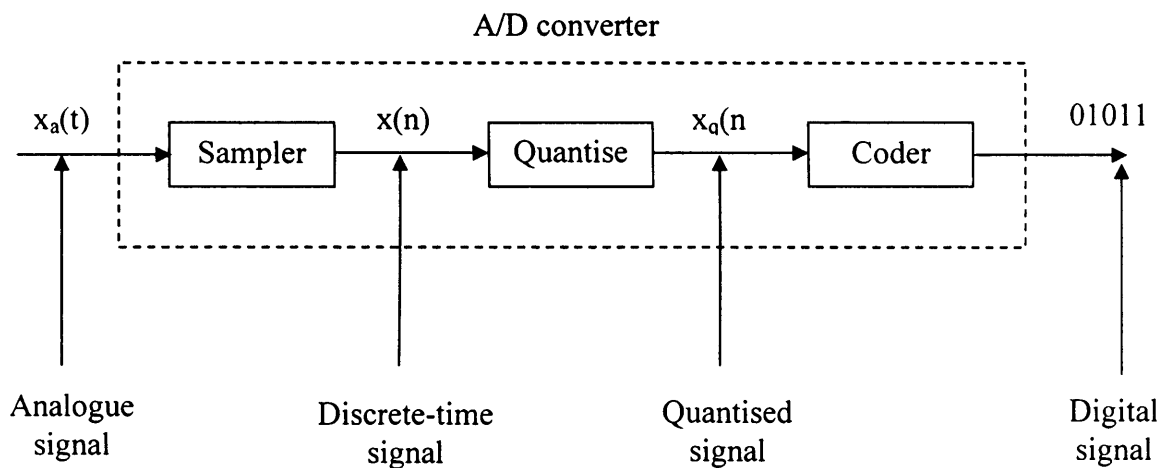


Figure 4.3: Basic parts of an ADC (Proakis and Manolakis, 1996)

The PIC 18F458 microcontroller contains a 10-bit 5 volt range ADC, that works on the principle of successive approximations. One quantisation step is therefore equal to

$5000/2^{10} \cong 4.88$ mV. The ADC accepts unipolar inputs within the range of 0 to 5 volts. A Sample & Hold (S&H) circuit freezes the input at the sampling instant. The frozen sample is then converted to a 10-bit binary number. The maximum throughput of PIC 18F458 ADC is approximately 30K samples per second. Eight MCU pins can be configured as analogue inputs and can be connected to the S&H circuitry under software control. Eight analogue signals can therefore be acquired sequentially.

It is important to sample the analogue signal at consistent intervals. A timer interrupt was thus used to trigger the analogue to digital conversion. The 10-bit ADC result was automatically stored in a combination of the two 8-bit Special Function Registers (SFR) from where the ISR moved it to a data memory buffer. The successive results were stored in contiguous memory locations and were available for onward processing.

4.4.4 Eight bit ADC Results

The selected MCU acquires analogue signals with 10-bit resolution but its processing engine works on 8-bit numbers. Processing of 10-bit numbers effectively needs 16-bit calculations requiring longer code and processing times. An easier approach can be to use only the most significant 8-bits data from ADC result. The author tested this approach whereby the most significant 8-bits only were used. The remaining two bits were ignored, thereby reducing the software overhead. It was observed that the acquired signals were less clear with reduced resolution but remained sufficient for the intended rough estimation of signal health. It was still possible to differentiate between signals from normal and abnormal process states. The findings from previous studies for PIC based analogue signal acquisition systems had also indicated the same results (Ahsan 2002, Amer 2002).

As the approach taken in this study is to discard normal signal data, a first layer check can be implemented based on the limited resolution 8-bit data and an initial decision of normal/abnormal status can be made within a front-end node. Information from several front-end nodes is combined at the second layer of hierarchy to ascertain the likely cause of a detected abnormality. For cases where the second hierarchical layer is unable to reach a conclusion, the third layer server-side of hierarchy may be instigated and all

nodes set to transmit data to the third layer. Each FEN would not be processing the acquired data in such cases and would simply forward the acquired signal data.

4.4.5 Data Storage

Data resulting from acquired signals is further processed to find out the fault symptoms and features hidden within it. Data acquired from digital sources is easily stored and requires only small structures in data RAM. The storage of analogue data, on the other hand, requires a larger space. The first choice for data storage in the PIC 18F458 was its internal RAM. The microcontroller has 1536 bytes of RAM and the storage space available for signal data storage is therefore very limited. One approach is to process the data quickly so that it can be disposed off before being over-written by new data. Discarding the data quickly without resource to any other storage media was the approach taken in this research. Also the decision to store only the 8 most significant bits of any ADC signals effectively reduced the data memory requirement by 50%. It was therefore possible to store signal data for twice the duration that was possible with 10-bit data. The MCU data memory is divided and bank switching is required for direct memory access. The author therefore used indirect memory addressing to manage data storage. This resulted in efficient data storage coding via the use of the three available 12-bit pointers in the MCU with pre- and post- increment, post-decrement, and base plus index options for faster execution.

For circumstances where the PIC's built-in RAM is not sufficient, other types of built-in memory can be used. The MCU contains 256 bytes of eeprom memory, which has the primary purpose of storing long time temporary variables such as configuration settings but can also be written to under software control. Ahsan (2002) evaluated eeprom as storage memory for a PIC 16F877 MCU and found it to be implementable but slower than RAM. Ahsan (2002) also deemed it feasible to store data in flash program memory as the MCU can write to its own program memory under software control. The program memory consists of 16384 words and can provide reasonable data storage in cases where ample space is left unused by the program code. Typical cell endurance of 1M cycles for eeprom and 100K for flash memory (PIC 18FXX8 Data Sheet, 2001) makes it possible to store data in these memories which are actually not designed for data storage. These options provide additional storage space inside the microcontroller without using external memory. Adding external RAM increases the system size and cost and was therefore

avoided in this research, but is possible and its details for PIC based monitoring are provided by Amer (2002).

4.5 SIGNAL ANALYSIS

Analysis routines were developed in both time and frequency domains so that the hidden fault symptoms in signals can be located. Various time domain methods for monitoring applications were evaluated for 8-bit microcontrollers in light of their limited processing capabilities. The author developed a new technique for frequency analysis which was suitable for 8-bit MCUs because of its lower mathematical complexity.

4.5.1 Time Domain Analysis

Analogue signals require more processing than digital ones. Time domain analysis for an analogue signal can be divided in two types. In the first type, every new sample of the signal is analysed as soon as it is acquired. In the second type, several samples are stored in a buffer before they are processed together. This kind of processing may be repeated in a real-time system on every new sample obtained after the first calculation. This therefore becomes a moving window calculation approach.

A sample from an analogue signal can be checked as soon as it is acquired. Its value can be matched with some expected value to determine the deviation of the physical signal. Such deviations may be used to indicate a non-nominal status when they are outside defined upper and lower threshold values. Threshold crossing on the positive/negative side usually indicates different faults and should be used to establish fault isolation. The threshold levels of course need to be ascertained, based on process knowledge and history. In the current research, the FEN needs to acquire data and present it to the developer for detailed analysis during a system study. Possible or expected faults may be introduced in the process and their impact on the signal recorded and analysed so that fault resolution can be achieved. The signals for the monitoring system development were initially generated from power supplies and later on via an analogue output interface card in a computer. Various signals were generated in Matlab and output using a signal generation card to test the analysis routines. The author did not regard isolated instances of threshold crossing as evidence of abnormality in this research. A single threshold

crossing can be a response to external noise rather than a persistent fault of the type being identified here. The number of samples to be included depends on the application sensitivity and the selected threshold value, and is therefore application dependant. A code variable is used to count the number of samples showing persistent behaviour and is compared to a user configurable preset value. Any persistent abnormality is reported to the second layer of hierarchy where other evidence of abnormality can also be taken into account before raising an alarm.

Thresholds can also be applied on processed data such as mean value or running sum. The author developed code to calculate the sum, running sum, mean value, variance, and trend detection on a predefined number of samples for which calculation was attempted. Acquired data was temporarily stored in a buffer for such computations and was overwritten by new data once the calculation results were found to be within threshold limits. A flag was set in cases of abnormality detection and this status was communicated to the second hierarchical layer. The moving average of the acquired data was also calculated over a predefined number of samples.

The PIC 18F458 microcontroller was operated at its maximum speed of 40 MHz. With its pipelined architecture, this results in 0.1 μ second time per instruction cycle. The codes written for the analysis were checked for timings, based on the instruction cycles they needed to execute. The achieved timings for various time domain analysis techniques are shown in table 4.2.

4.5.2 Frequency Domain Analysis

Time series data analysis provides a useful insight into the process health status yet several aspects cannot be covered with time series analysis alone. There may be certain scenarios where a frequency domain analysis of a signal may be more fruitful than the time domain analysis. The presence or absence of a particular frequency component may indicate a fault in the system. The power content of these frequency components may be checked against predefined thresholds to generate fault symptoms. The Fast Fourier Transform (FFT) is a widely employed technique for frequency analysis but is generally considered too computationally expensive for 8-bit microcontrollers. Microchip provide a FFT method for the 8-bit PIC17C42 microcontroller (Palacherla, 1997). Lacoste (1998)

Feature	Input	Output	Time
Summation	N 8-bit samples ($N \leq 255$)	16-bit	$2+7*N$ Instruction cycles $3.7\mu s$ for $N=5$
Running sum	8-bit sample	24-bit	6 Instruction cycles fixed, $0.6\mu s$
Mean value	N 8-bit samples ($N \leq 255$)	8-bit quotient, 8-bit remainder	$217+7*N$ Instruction cycles $25.2\mu s$ for $N=5$ ($3.7\mu s$ for sum+ $21.5\mu s$ for division)
Variance	N 8-bit samples ($N \leq 255$) 8-bit Mean value	24-bit	$10+22*N$ Instruction cycles $12\mu s$ for $N=5$
Window sum	8-bit sample	16-bit	10 Instruction cycles, $1\mu s$
Trends	8-bit sample	Flag	18 Instruction cycles maximum $1.8\mu s$ maximum
Moving average	16-bit window sum Number of samples	8-bit	227 Instruction cycles fixed $22.7\mu s$

Table 4.2: Achieved timings for various time domain analysis techniques

implemented FFT on a PIC 17C756 microcontroller but its resolution was very low (64Hz) which had some applicability to audio applications. The windowing required before applying FFT is also considered to be a computationally expensive task. Microchip's new 16-bit microcontrollers, called dsPICs, have a software library for windowing and FFT routines but are limited to 256-points which may still not provide satisfactory results.

Another way to detect the presence of a certain frequency component in a signal is to use a narrow bandpass filter with the pass band centred at the target frequency. Bandpass filters can be implemented as analogue or digital filters. Microchip provides an application note for implementing Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) digital filters using its 18x series of PIC microcontrollers (Ramu, 2002). On investigation however it was observed that the given implementation is resource intensive especially concerning the use of pointers for indirect memory addressing. All three available pointers were used in the implementation and the monitoring system, proposed in this research, would require pointers for communication tasks. Another limitation for implementing digital filters was their memory requirements, especially for FIR filters.

The author has therefore developed a sweeping filter technique, for frequency analysis using a programmable analogue filter (Ahsan et al, 2004). Figure 4.4 shows the block diagram of the sweeping filter system. Signal acquisition applications generally have an anti-aliasing filter at the input stage which was replaced with a programmable filter in this technique. The signal was provided to a precision programmable analogue filter controlled by the microcontroller in bandpass filter mode. The microcontroller swept the range of frequencies of interest, band by band, and determined signal strength in each band. It acquired the filtered signal for one time-period for maximum frequency in a particular band. It determined the maximum and minimum amplitude values in the acquired data and thus calculated the peak to peak amplitude difference. This amplitude difference was taken as a measure of signal strength for the frequency component in that band. The width of the band thus became the resolution of the frequency analysis. The microcontroller then shifted the programmable filter settings to the next band centre-frequency and repeated the process. The entire frequency range of interest was swept in this way generating a total profile of the signal.

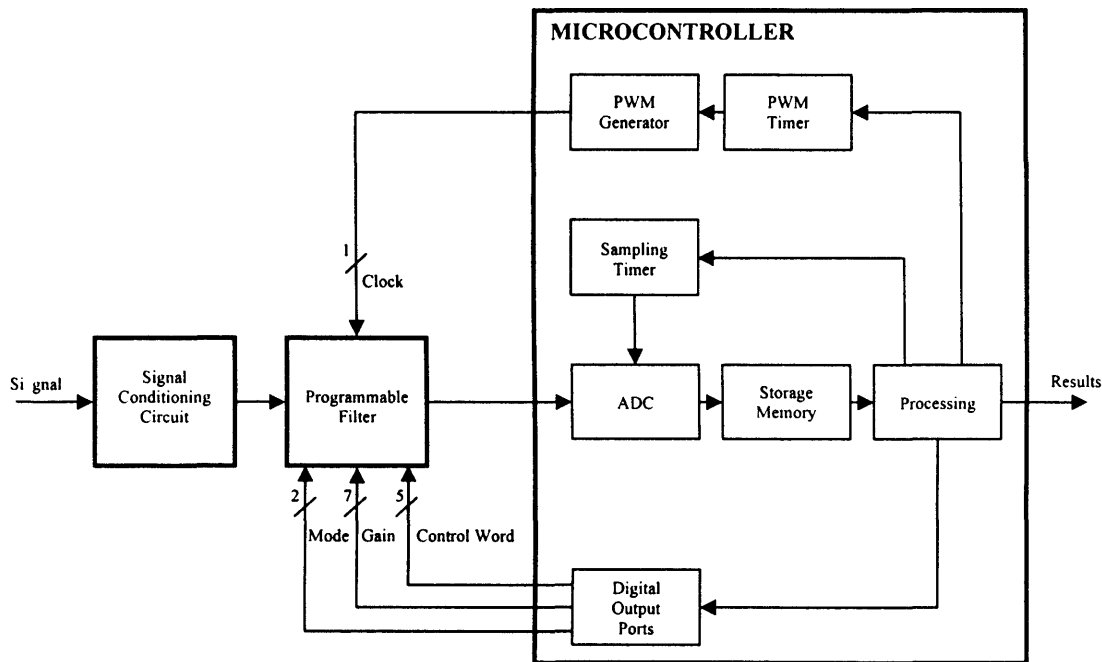


Figure 4.4: Block diagram of sweeping filter system.

Maxim's MAX264 precision programmable analogue filter IC was used in this research (MAX263-MAX268 Data sheet). It contained two second order filters, configurable as low-pass, high-pass, band-pass, or notch filters individually. These could also be cascaded to provide 4th order filtering. They were controlled by the same programmable gain, mode, and frequency control inputs, as shown in figure 4.5.

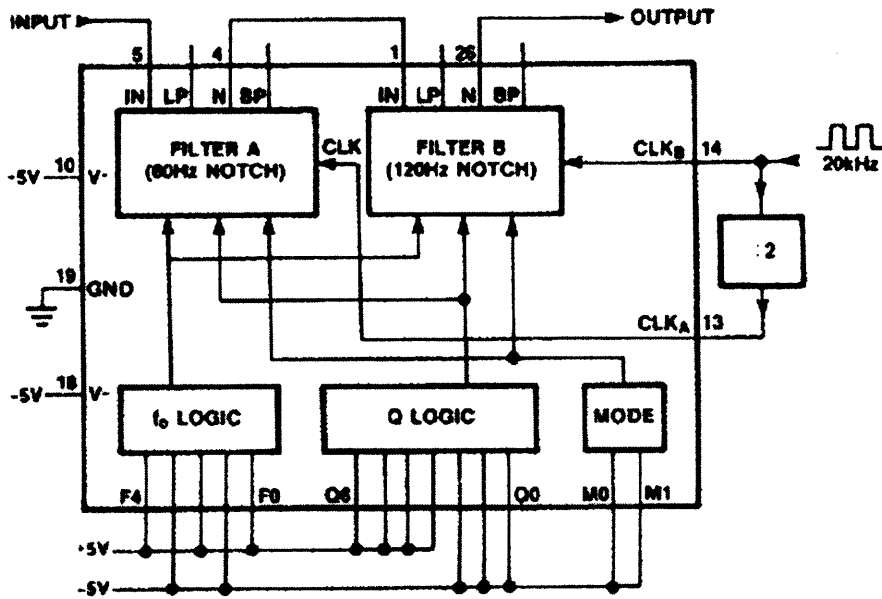


Figure 4.5: MAX264 filter block diagram [MAX263-MAX268 Data sheet]

The actual cut-off frequency for a bandpass filter is a function of clock rate, frequency control word, and the operating mode of the filter. The MCU controlled all these parameters and a resolution of 1 Hz bandwidth was achieved for the normal frequency range of interest. The PWM module in the microcontroller provided accurate clock rates to the filter IC reducing load on the 8-bit processing engine. The actual filter response deviates from the ideal one, especially for lower gain and input-clock/cut-off frequency ratio, but the deviation being predictable was eliminated.

A range of signals were generated to test the sweeping filter approach. A 100mV peak to peak amplitude sine wave of 20Hz frequency was used as an input signal and figure 4.6 depicts the sweeping filter output. The calculated difference (maximum - minimum) provided the relative strengths (peak to peak) of various frequency components in the signal. The filter was configured for 1Hz bandwidth with a quality factor (Q) of 16. The

input signal was correctly analysed and a peak at 20Hz frequency can be seen in figure 4.6. The neighbouring frequency bands showed relatively higher strengths because of the filter band but were lower than the 20Hz principal component. A peak strength value of 60 was achieved in this test but a higher value may be desired to achieve better decision making. That can be achieved by using higher amplitude input signals. Effects of input signal amplitude on the acquired peak value can be seen in figure 4.7 where wider peaks are visible due to increased strengths in all components. Signal amplitude beyond a certain voltage caused filter saturation and peak suppression in the output was observed for 500mV input as shown in figure 4.7. The maximum possible number attainable with 8-bit computations is 255 theoretically but practical constraints reduced the achievable upper limit.

Another factor affecting sweeping filter performance was filter quality factor (Q) which is the ratio of centre-frequency to bandwidth and can be programmed from 1 to 64 for a MAX264 filter IC in 128-steps (MAX263-MAX268 Data sheet). Figure 4.8 shows the effect of Q on achieved strength value with a 100mV sine wave input. Larger separation between principal and neighbouring components can be seen with increasing Q. This provided better decision making opportunities by providing larger range for threshold placing. Increased Q value caused filter saturation at lower input voltages and actual parameter selection had to be a compromise.

Square wave inputs were provided to the sweeping filter in order to observe its harmonics detection behaviour. Figure 4.9 shows results for a 10Hz square wave input signal with 100mV amplitude. The microcontroller successfully detected the fundamental frequency of the periodic waveform as well as the expected harmonics at 30Hz. The harmonics strength was lower than the fundamental component, as expected. This showed the sweeping filter's capability to isolate multiple frequencies present in a signal. Care was however required about the minimum separation between frequency components. Band overlapping may occur for two close-by components enhancing the total signal strength for components between them. Figure 4.10 shows this limitation where input signal contained 20 and 24 Hz components at 100mV amplitude. The microcontroller showed high strength values for both components but the in-between components showed false strengths.

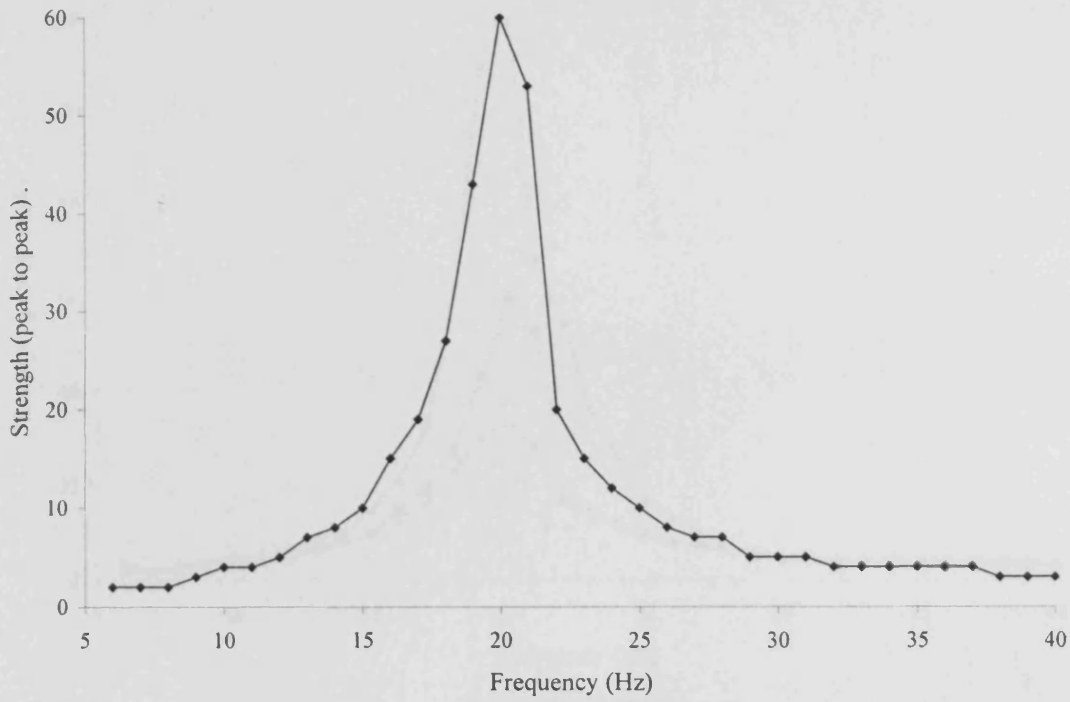


Figure 4.6: Detection of 100mV 20Hz sine wave

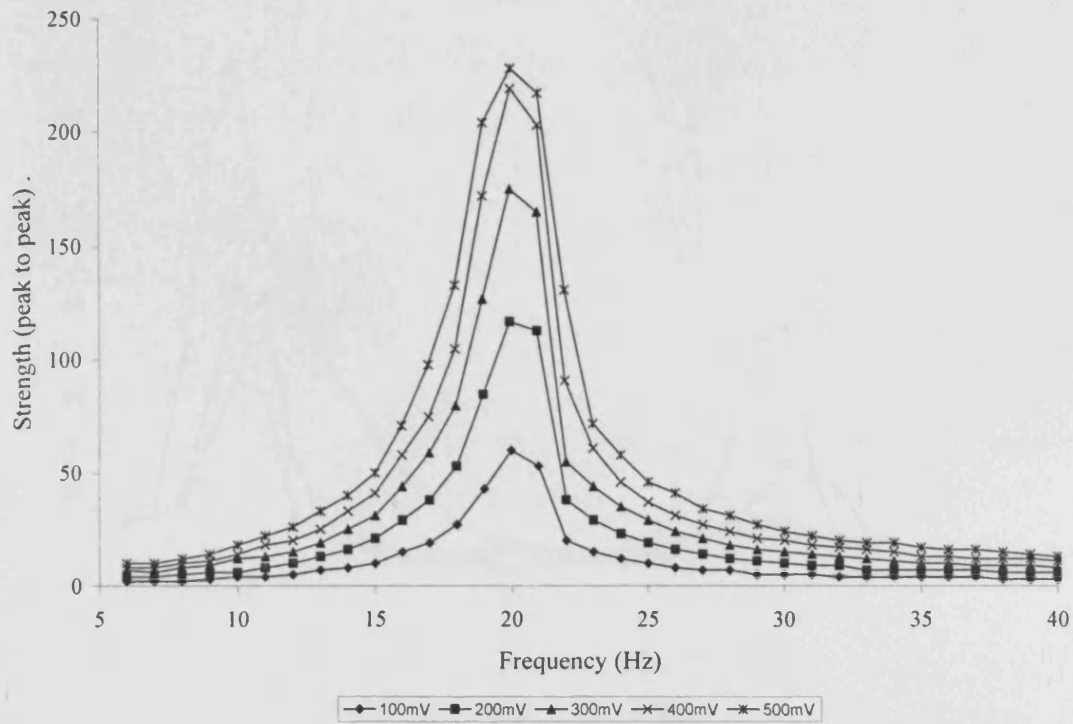


Figure 4.7: Higher amplitude sine wave detection

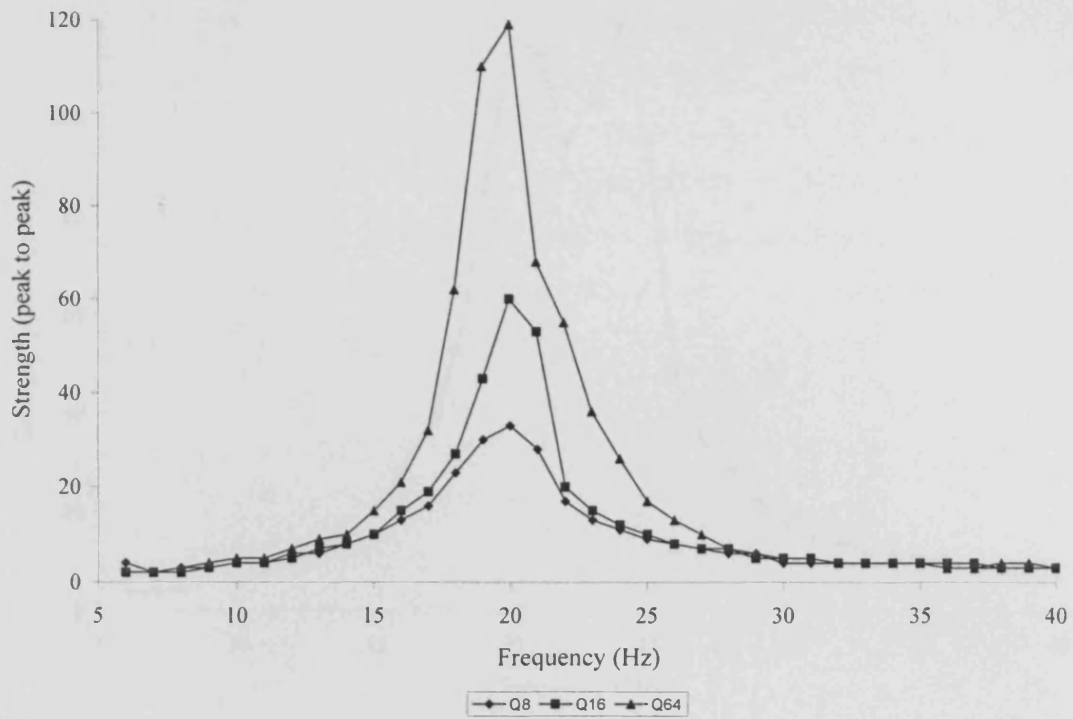


Figure 4.8: Detection of 100mV 20Hz sine wave with Q values of 8, 16, and 64

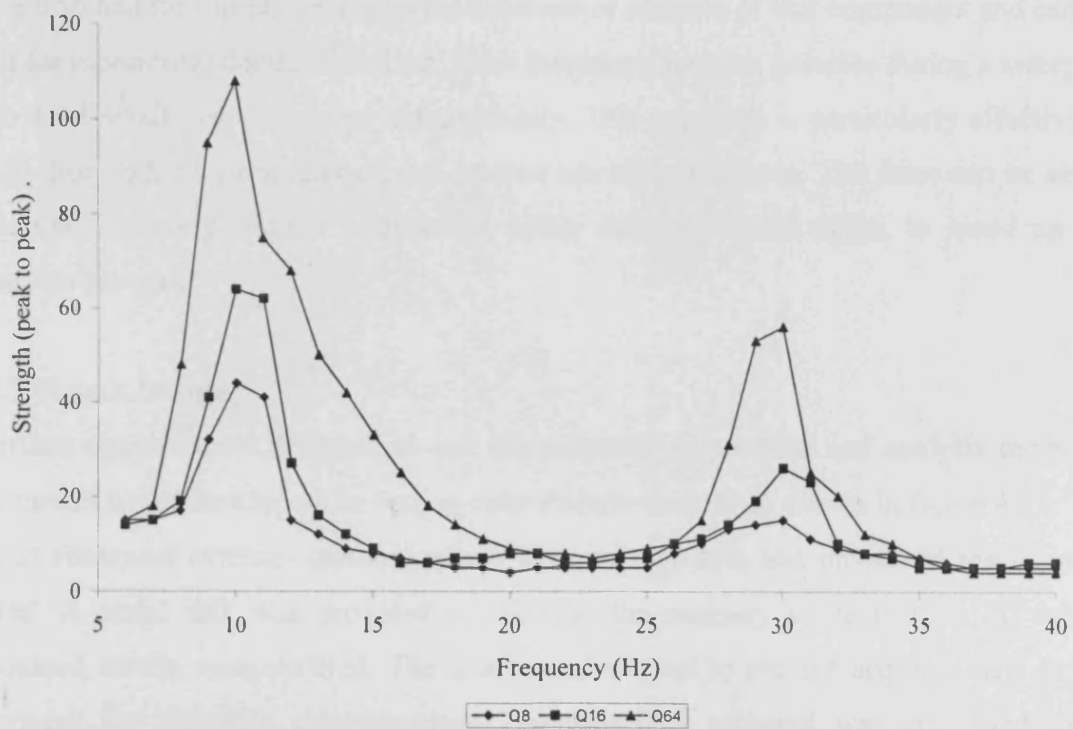


Figure 4.9: Harmonic detection for Q 8, 16, and 64

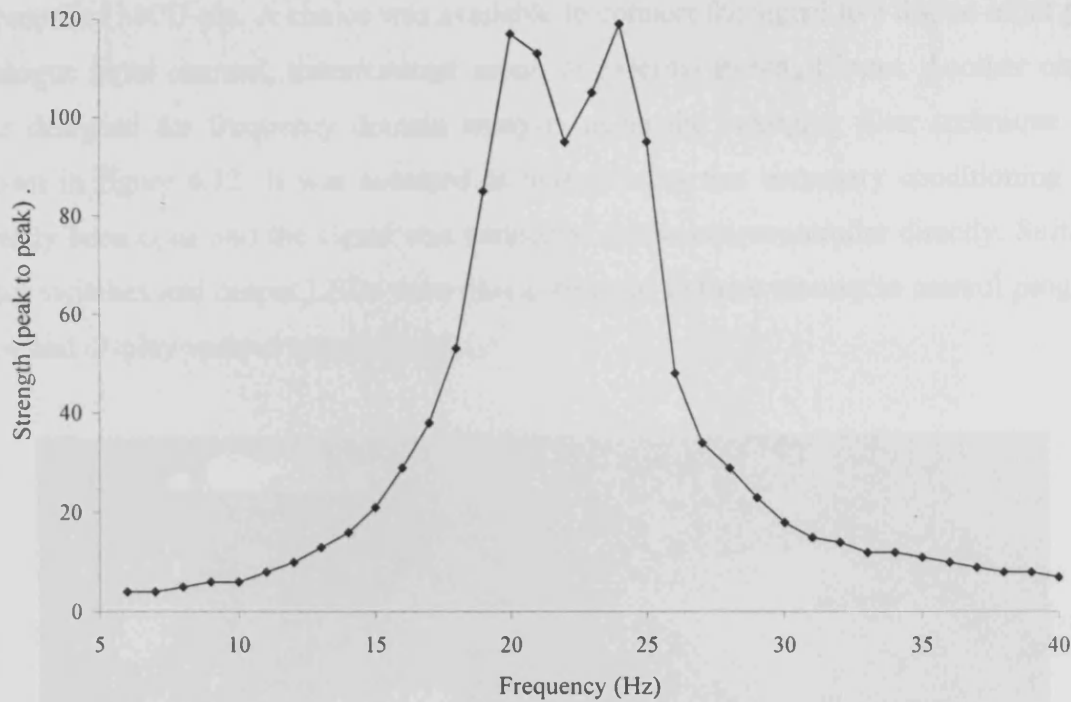


Figure 4.10: Multiple frequency detection

The detected strength of a particular frequency component, when compared with a predetermined threshold, determine the presence or absence of that component and can be used for monitoring decision making. Such detections are also possible during a sweep as each band results are calculated independently. This approach is particularly effective in applications where the frequencies of interest are already known. The filter can be set to sweep only those particular frequencies, rather than the whole range, to speed up the detection process.

4.5.3 Circuit Design

Interface circuits were designed to test the proposed acquisition and analysis methods. The circuit board developed for testing time domain analysis is shown in figure 4.11. The circuit contained external memory where accumulated data and processed results were stored. A serial link was provided to transmit the memory contents to a PC where processed results were verified. The link was also used to present acquired data to the developer for threshold determination. The data thus gathered was also used for a frequency domain analysis on the computer before the sweeping filter technique was developed. A CAN bus interface was also provided on the PCB. In a later generalized design, the input signal was connected to a set of jumpers that connected the signal to the

appropriate MCU pin. A choice was available to connect the signal to a digital input port, analogue input channel, timer/counter input, or external interrupt input. Another circuit was designed for frequency domain analysis using the sweeping filter technique and shown in figure 4.12. It was assumed in both circuits that necessary conditioning had already been done and the signal was connected to the microcontroller directly. Suitable input switches and output LEDs were also connected to these circuits to control program flow and display various tests outputs.

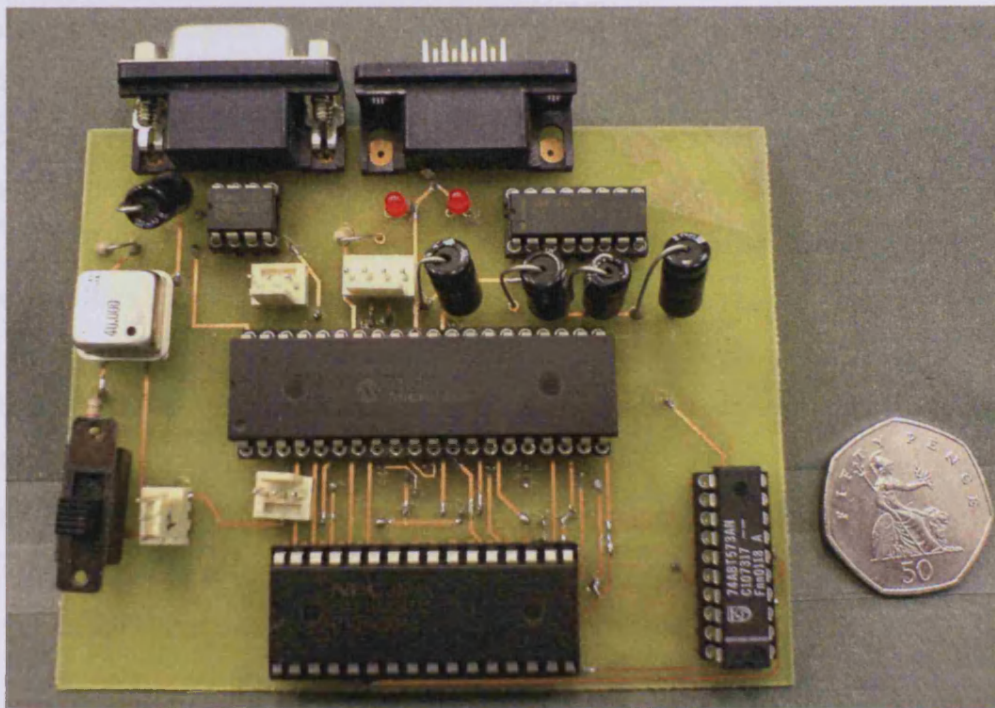


Figure 4.11: Time based analysis circuit

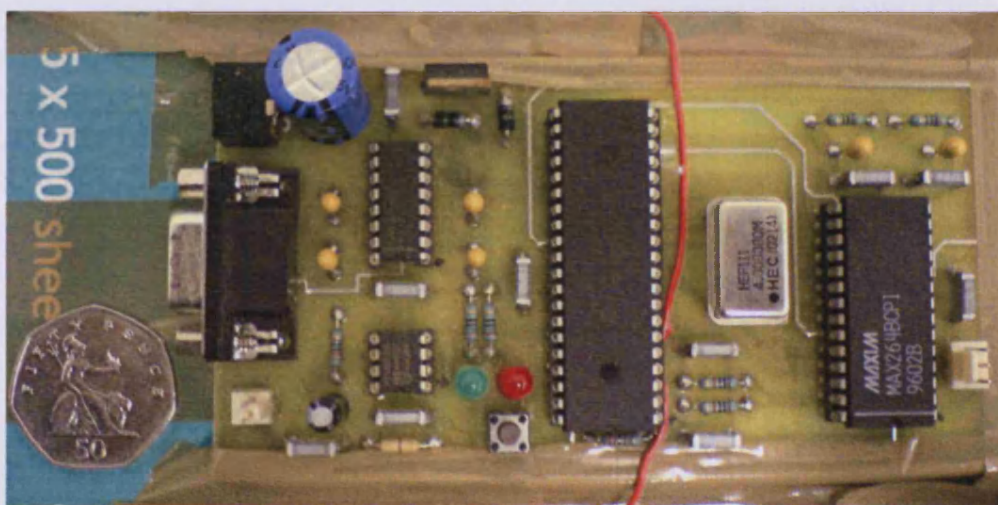


Figure 4.12: Frequency based analysis circuit

4.6 SUMMARY

The need for a low-cost generic monitoring system was identified via the literature review in chapter 3. A distributed and hierarchical monitoring system was proposed in this chapter as a generic solution. Cost-effectiveness is attained by using 8-bit microcontrollers. One MCU was dedicated to each process signal, thus placing it close to the sensor (or ideally inside the sensor assembly). A general overview of such a system was provided in this chapter along with signal acquisition and analysis details. The limited resources of the 8-bit MCUs constrained the signal analysis levels and simple computations on sampled 8-bit A/D data were used. Small subroutines were seen to be effective for time domain analysis but problems were faced in attempting frequency domain analysis. A novel sweeping filter technique was therefore developed and good results were achieved. It is believed that programmable filter chips have not previously been used before for industrial signal frequency analysis in collaboration with microcontrollers and this aspect provided innovation in this research.

The feasibility of a monitoring system based on 8-bit microcontrollers was established in this chapter. This provided the base or first layer of hierarchy in the overall monitoring system. Several first layer nodes were connected with each other and with a synchronising and user interface node to establish the second layer of this hierarchy. The details of node connectivity to form a working system are provided in the next chapter.

DISTRIBUTED MONITORING SYSTEM

A brief introduction to the concept of a distributed monitoring system with hierarchical layers was provided in chapter 4 along with details of the signal acquisition and analysis by front-end nodes. Individual results emerge from all first level FENs in the hierarchy of the distributed monitoring system. These results are combined at the second computational layer of the hierarchy to form a holistic view of the process at any point in time. A communication medium, CAN bus, was provided between the FENs and other network devices for information exchange. One node on the CAN bus was used to provide the user interface. This node can accept a user command and communicate it to the FENs. It provided synchronization of the system and was called the Synchronization & User Interface Node (SUIN). It also presented the monitoring decision results to a user, who may be at a remote location. This chapter details how the system was set up for individual FENs to communicate with each other and with SUIN. It also introduces the integration of results and possible decision making processes.

The SUIN was developed using an 8-bit microcontroller (for similar reasons to those explained for FENs in chapter 4). The SUIN forms the second layer in the hardware hierarchy as shown in figure 5.1. The computational hierarchy second layer may however be between the SUIN and FENs in this development to allow flexibility in the way they collaborate to reach a conclusion for a particular application. Implementation details are provided in this chapter. The SUIN was developed to communicate with the user(s) over the Internet. Multiple application layer protocols, such as Telnet or HTTP, are used for this purpose and run on top of the TCP/IP stack. A brief introduction to these protocols is provided in section 5.2.3. The developed code can serve multiple remote users simultaneously. The author believes that no real-time process monitoring system has previously been reported based on a 8-bit microcontrollers' decision making powers. The reported monitoring system is an attempt to find as many as possible faults at the first and second layers (implemented solely on 8-bit microcontrollers) although it allows the use of a server-side PC or other high power processor at the top third layer which can then be dedicated to specialized and high level processing.

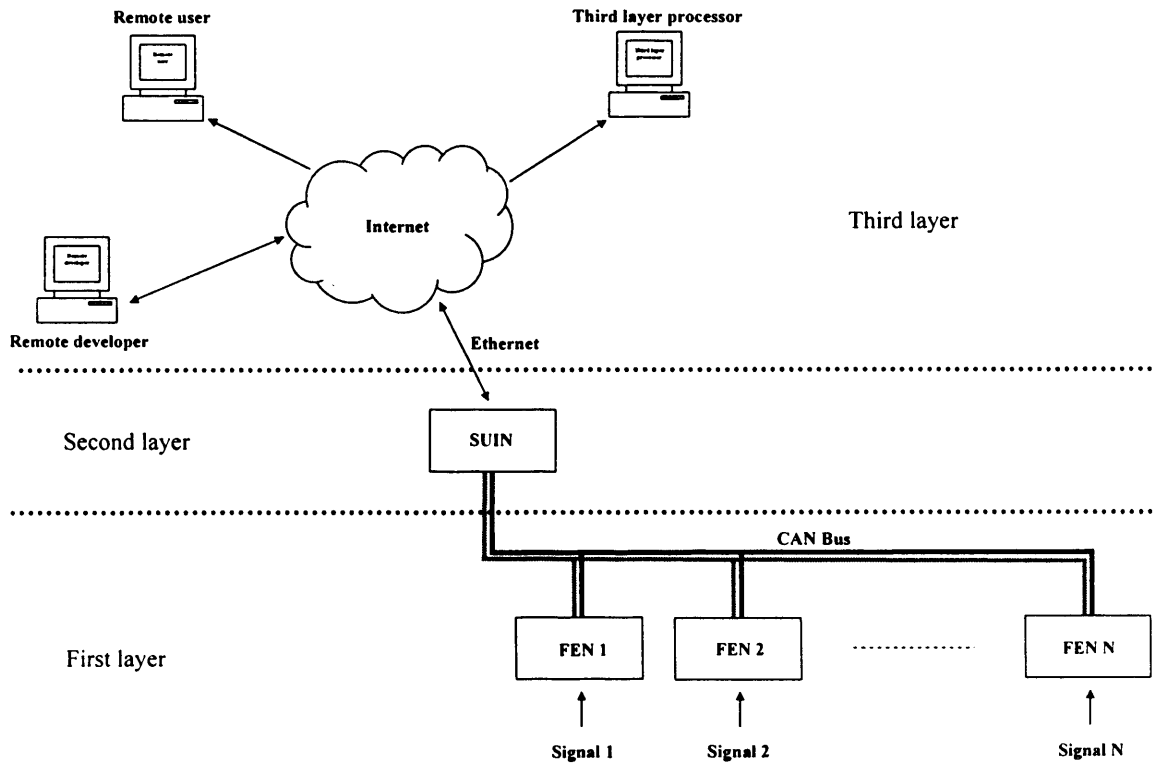


Figure 5.1: The developed monitoring system with hardware layers

5.1 FIELDBUS

Fieldbus is the general name for any shared digital communication medium used to connect various instrumentation in the field. There is no general consensus about what a fieldbus specifically means but it has to be an all-digital communication network. Various protocols for fieldbus are available such as Profibus, DeviceNet, Foundation fieldbus and CANOpen. A fieldbus protocol specifies the physical medium for communication and the associated messaging in a data link layer. It also specifies some higher level protocols required for applications using it. Controller Area Network (CAN) bus is a widely deployed communication medium used in noisy environments (automotive, industrial) that defines physical and data link layers. Higher layer protocols are added to it to form fieldbuses such as CANOpen and DeviceNet. The CAN bus was selected to connect network nodes in this research.

5.1.1 Controller Area Network

The Controller Area Network (CAN) bus communicates digital messages over a differential pair of wires. It was initially designed by Bosch for motor cars where

electronic devices are spread all over the body (Robert Bosch web site). This causes a complex wiring loom and CAN is an effective protocol to reduce this clumsiness. Another important aspect in cars is the noisy environment where ignition switching, generator, spark plugs, etc. produce a lot of noise. The CAN protocol was designed in order to work reliably in such severe conditions. The industrial environment is traditionally very noisy and CAN's noise immunity gives additional benefit in this environment. CAN has developed into a mature industrial standard over the years and was internationally standardized in 1993 as ISO 11898 for serial data communication (CAN in Automation: Home, web site). Several standards are available for various CAN variants such as Fault Tolerant CAN and Time Triggered CAN.

A reduction in wiring complexity is a major advantage of CAN which works on two-wire balanced system with CAN High (CANH) and CAN Low (CANL) wires. A logic 0 bit is transmitted on the bus as 'dominant' bit where the CAN high (CANH) wire goes to +3.5V and the CAN low (CANL) wire goes to +1.5V. A logic 1 bit is transmitted as a 'recessive' bit with both wires at +2.5V level. The ISO standard specifies twisted pair wires but other physical media, like radio and optical links, have also been used successfully. CAN provides a maximum throughput of 1Mbps at a distance up to 40 meters (130 ft). Longer cable lengths are possible at reduced data rates, such as 1km at 50 Kbps (CAN in Automation: CAN Dictionary, web site). Figure 5.2 shows the wiring connections between CAN nodes. The bus works in 'logic AND' i.e. a dominant bit overwrites a recessive bit. Two nodes may try to transmit a recessive and a dominant bit at the same time and the discrepancy is used for bus arbitration.

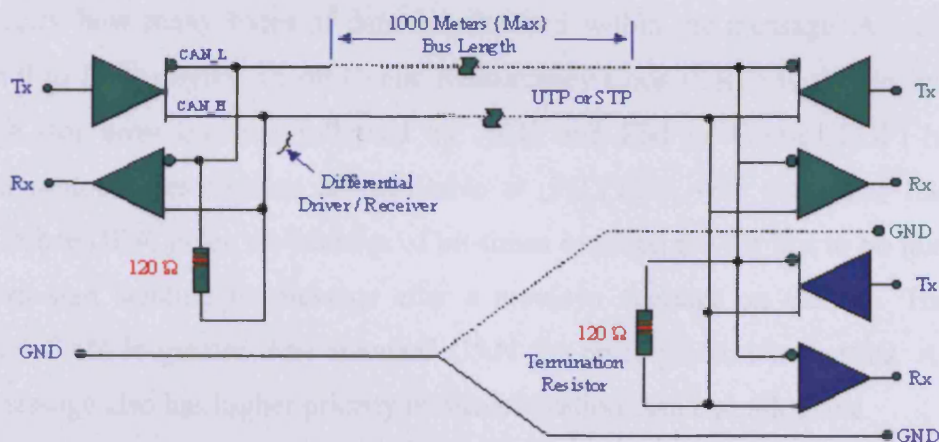
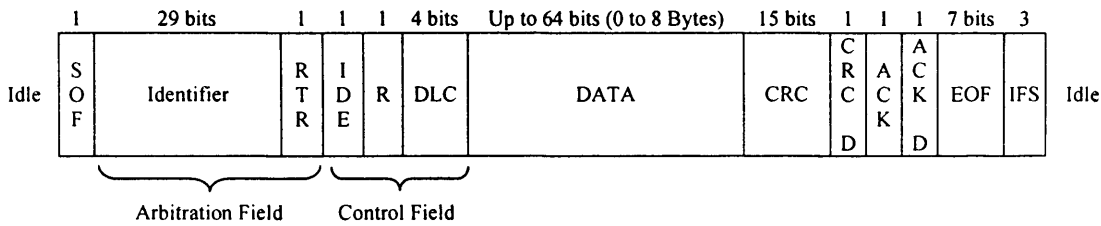


Figure 5.2: CAN Bus (Leroy Davis, web site)

CAN defines message types, arbitration rules for bus access, and methods for fault detection and confinement. It is a broadcast type bus and every node transmits on the same line. The messages do not contain any explicit address and are said to be content-addressed. All nodes of the bus receive all the messages and pick up the related message using local filters. Bus arbitration is based on message identifiers where lower identifiers have higher priority on the bus. Each transmitter starts transmitting its message when it finds the bus is in idle state. Several transmitters may start transmission at the same time. Each node monitors the bus state while transmitting and aborts on finding a discrepancy between sent voltage level and the actual voltage level on bus. It then starts receiving the incoming message transmitted by some other node. Bus arbitration is thus undertaken with a 'Carrier Sense Multiple Access with Collision Avoidance' (CSMA/CA) access control mechanism.

The original CAN standard contained an 11-bit identifier as an arbitration field. Later on, this field was extended to 29 bits on customer demand. A reserved bit in the control field differentiates the two CAN standards. The 11-bit standard is now known as CAN 2.0A and the 29-bit standard is called CAN 2.0B or 'extended CAN' (Kvaser, web site). Up to 8 data bytes can be attached with an identifier in a CAN message.

A data message on a CAN bus contains several fields in order to accomplish bus arbitration, synchronization, and information transmission. Figure 5.3 shows the sequential flow of the bits in an extended CAN data message. The message starts with a Start of Frame (SOF) bit followed by a 11 or 29 bit identifier field. Control bits in the message indicate which type of identifier is being used and the Data Length Code (DLC) bits specify how many bytes of data are attached within the message. A message may contain 0 to 8 data bytes. 15-bit Cyclic Redundancy Code (CRC) is also included in the message (for error control) followed by ACK and End of Frame (EOF) bits. More detailed protocol descriptions are available at (PHYTEC, web site). The Intermission Frame Space (IFS) gives the number of bit-times required for the bus to be idle before a node can start sending its message after a previous message on the bus. The IFS for extended CAN is greater than standard CAN for error protection reasons. A standard CAN message also has higher priority in bus arbitration than extended one.



SOF	Start of frame	
RTR	Remote transmission request	Dominant for standard Recessive for remote
IDE	Identifier extension	Dominant for standard Recessive for extended
R	Reserved	
DLC	Data length code	
CRC	Cyclic redundancy code	
CRC D	CRC delimiter	Always recessive
ACK	Acknowledge	Dominant for ok Recessive for error
ACK D	ACK delimiter	Always recessive
EOF	End of frame	Always recessive
IFS	Intermission frame space	Always recessive

Figure 5.3: Extended CAN data message bits

In summary, CAN protocols provide excellent error handling. A Cyclic Redundancy Check (CRC) is added with each frame and frame formation and acknowledged errors are checked. The transmitter checks the bus status during transmission and any discrepancy is detected immediately. Bit stuffing is used if five consecutively transmitted bits have the same logic level. The stuffed bit has a level complimentary to the transmitted ones and is removed by the receiver. Bit stuffing provides edge synchronization and avoids excessive dc components on the bus but prolongs the transmission time. The worst-case transmission time of an 8-byte frame with an 11-bit identifier is 134 bit times, i.e. 134 microseconds at 1Mbits/sec baud rate (Leroy Davis, web site). After transmission of an erroneous message that has been aborted, the sender automatically re-attempts transmission. The CAN protocol also provides a statistically-based mechanism to distinguish sporadic errors from permanent errors and local failures at a node. A faulty node can switch itself off and does not then negatively affect the whole system (CAN in Automation: CAN protocol, 2005).

As stated, the author selected CAN bus for inter-node communication in this research because of its superior performance in noisy environments. A number of aspects were considered and the selection reasons are summarised below:

- Only an unshielded twisted pair of wires is required reducing cabling cost.
- Differential communication provides immunity to noise.
- Multiple error checking systems make it robust.
- Filters available in the CAN protocol discard un-related network traffic and the processor is disturbed only when a message of its interest is received.
- Reasonable data rates are possible at reasonable bus lengths.
- CAN is a time tested and widely accepted proven protocol.
- CAN enabled devices are already in successful use in industry.
- No upper limit is imposed on number of possible nodes on the bus.
- CAN controller hardware is available inside 8-bit MCUs reducing circuit size, cost, and interface load on the processing engine.

A maximum possible data rate of 1 Mbps is permissible under CAN protocols for bus lengths up to 40m. This may be sufficient for many monitoring applications but a 125 Kbps data rate was selected to allow longer bus lengths and a more generally applicable system. The reduction in bus speed was eventually compensated in the developed design by reducing the network load by taking first level decisions at acquisition nodes. Other measures for reducing network traffic are explained later.

5.1.2 CAN in PIC 18F458

The PIC 18F458 has a built-in CAN controller and only an external transceiver is required to connect the MCU with the CAN bus. The built-in CAN controller has three transmit and two receive buffers. A message from a higher priority buffer is sent on the bus before a lower priority one. It is thus possible to send an urgent message in front of queued up messages. Figure 5.4 shows a simplified diagram for CAN protocol implementation in the PIC 18F458.

The CAN engine receives all transmitted messages and filters the messages' identifiers in order to select the messages to store in the receive buffers. Two acceptance filters are available for receive buffer 0 (RXB0) and four for receive buffer 1 (RXB1). It is possible to check only specific bits in the identifier and an acceptance mask is available for each receive buffer. A node can therefore filter out the messages of interest to it from all the bus traffic. The processor is thus not disturbed by excessive network traffic. The PIC

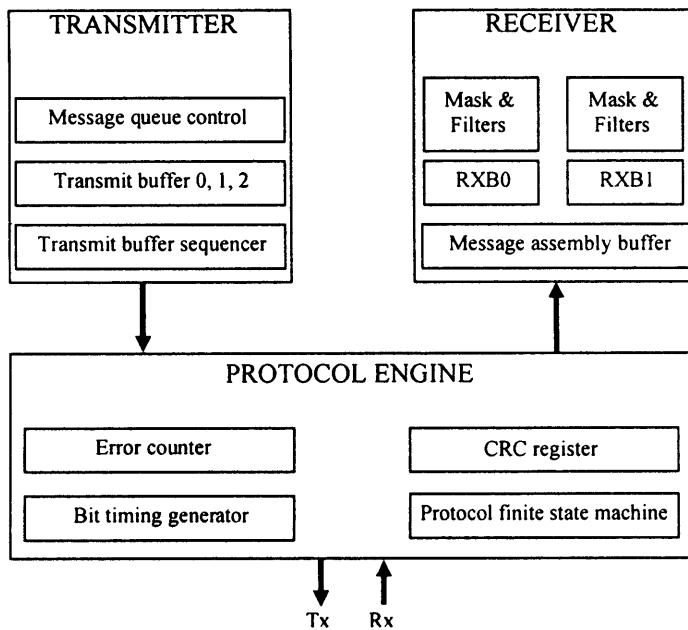


Figure 5.4: CAN protocol implementation in PIC 18F458

18F458 can transmit and receive messages with both standard and extended identifiers and can be configured to use only one of them if required. It can also be configured to receive all messages or only valid messages. The MCU has the overflow reception capability and a message intended for RXB0 can be stored in RXB1 when RXB0 is already full. A received message is thus protected from overwriting providing enough free time for the MCU to shift the first received message to RAM.

5.2 SYNCHRONIZATION AND USER INTERFACE

In the developed monitoring system the Synchronization and User Interface Node (SUIN) was central to the distributed system. The SUIN takes a user's command and communicates it to networked acquisition nodes (FENs) ensuring that the whole system works in unison. It is also responsible for displaying monitoring results to users in real time. The SUIN was designed to provide access to a remote user and/or developer via the Internet. Its connection to FENs via the CAN bus was also used to obtain process signal information from the FENs. Thus, the SUIN was not directly interfaced with sensors and did not require a microcontroller with built-in signal acquisition hardware. The microcontroller selected for the SUIN needed to support CAN and have the ability to handle a burst of messages without any message being over-written by the next received

message. This was required because all FENs would communicate with it asynchronously and may try to send messages at the same time. This scenario was considered to be highly likely in case of an abrupt fault occurrence when several FENs would detect abnormal conditions at around the same time.

It was decided that only limited number of output signal interfaces would be required for local annunciation and that the primary user interface was a remote one via the Internet. The SUIN microcontroller needed to have the capability to interact via the Internet through either an Ethernet or telephone modem connection. Another important consideration for this node was its memory requirements. While dealing with several FENs and remote users simultaneously, the microcontroller would almost certainly require a larger memory than typically built into the current generation of microcontrollers. The MCU architecture should therefore support external memory without significant software overheads. The author selected a Dallas Semiconductor 8-bit microcontroller (DS80C390) to provide the SUIN functionality. The details of this 8-bit microcontroller and reasons for its selection are provided in the following section 5.2.1.

5.2.1 80C390 Microcontroller as SUIN

Dallas Semiconductor's 80C390 is an 8-bit microcontroller that provides an extension of the very popular Intel 8051 family of microcontrollers (Dallas Semiconductor, web site). It improves on the 8051 MCU's 12 clocks per instruction cycle to 4 clocks per instruction cycle and can work with clock frequencies up to 40 MHz. A single cycle instruction can therefore be executed in 0.1 microseconds and the MCU can work at up to 10 MIPS.

The MCU contains two built-in CAN controllers with dedicated memory for multiple Message Centres (MC). A CAN controller in the MCU can be configured to work with standard or extended CAN. Message Centres (MCs) are used for message transmission and reception by each controller. A controller has 15 MCs and 14 of them can be configured either for transmission or reception. The fifteenth MC has different hardware and is designated as a First-In First-Out (FIFO) buffered receive-only MC to help prevent data overruns. 512 bytes of RAM is reserved for these message centres. This memory is in addition to the built-in SRAM and scratch pad memory. The author has utilized the high number of MCs to receive any bursts of messages without any message loss. One message centre was dedicated to each FEN and simultaneous messages from different

FENs were received in different MCs. This method ensured that no unread message could be over-written by another message. Separate handshaking and timing control for each FEN ensured proper message reception. This capability was identified as an essential one for SUIN functionality and played an important role in the selection of this microcontroller.

The Dallas Semiconductor 80C390 MCU contains 4 KB built-in SRAM and 256 bytes of RAM as scratch pad area in addition to the memory dedicated to the CAN message centres. The MCU architecture also supports easy interfacing to external memory and a 22-bit program counter can access up to 8 MB memory (4MB for program and 4MB for data). Two 8-bit ports are dedicated to the external memory interface. The memory arrangements thus met the identified SUIN requirements.

Other attractive features included a hardware math accelerator providing fast execution of 32- and 16-bit multiply and divide operations. The accelerator output from the multiply and divide operation is automatically added to a 40-bit accumulator providing multiply-and-accumulate (MAC) and divide-and-accumulate functions which are useful in DSP operations. The math accelerator also provides a normalize function that converts 4-bytes unsigned binary integers into floating point format. The MCU also contains dual data pointers with increment/decrement features to speed block data memory moves. These features were found useful later on when the author developed Java codes for the SUIN. Other useful MCU features include four 8-bit I/O ports (other than the two used for external memory), two serial ports, three 16-bit timers/counters and support to 16 interrupt sources including 6 external sources. Appendix B shows the block diagram of 80C390 microcontroller.

Easy Internet access had been identified as another important requirement for the SUIN. Popular methods of interfacing a processor with the Internet either use an Ethernet controller or a telephone modem with dial-up connection. The author was unable to find any microcontroller with any of these two options at the time of the SUIN MCU selection, although microcontrollers with built-in Ethernet controller circuitry emerged soon afterwards. An example of such microcontrollers is 80C400 MCU where Dallas Semiconductor replaced one CAN controller of 80C390 MCU with an Ethernet controller. The 80C390 MCU however contained ample resources to communicate on the

Internet. A commercially available embedded system, Tiny InterNet Interface (TINI), that included an Ethernet controller interfaced to a 80C390 microcontroller was sourced and selected for the SUIN platform in this research.

5.2.2 Tiny InterNet Interface (TINI)

TINI is a platform developed by Dallas Semiconductor that consists of a TINI stick and a TINI socket (Eisenreich and Demuth, 2003). The stick contains the 80C390 microcontroller with an Ethernet controller, memory, and real-time clock interfaced to it. It is available as a small PCB in shape of a 72-pin module (similar to a 72-pin Single Inline Memory Module (SIMM)). The TINI socket holds this stick and provides connectors so that it can communicate with the outside world. Revision D of the TINI stick provided 1MB RAM and 512 KB flash memory and worked with a 36.864 MHz clock. A LAN91C96 Ethernet controller was used to interface the MCU with Ethernet. This has a 10-Base-T protocol and a maximum connection speed of 10 Mbps. The complete TINI hardware set was purchased for £67. Table 5.1 provides a comparison of TINI with some other Internet enabled embedded devices (commercially available at the time of its selection). It can be seen from the table that although Internet access was not rare for embedded devices the provision of CAN with Internet access was not generally available at that time.

The purchased TINI contained a loader program in its flash memory which was used to upload other software. Dallas Semiconductor provided a PC program, JavaKit, to communicate with the loader through a serial port. A provided file, `tini.tbin`, contained the basic firmware and was uploaded in the stick memory using JavaKit. This firmware provided boot-up code to the TINI and included Java Virtual Machine (JVM) and Application Program Interface (API). The JVM provided access to core Java packages such as `java.lang`, `java.io`, and `java.net`. and APIs were used in application programs developed for process monitoring. The firmware supported multi-users access and multi-tasking was possible with programs having their own threads.

A Unix-like shell, Slush, was installed in the TINI memory. This provided a command prompt environment where user could enter commands. A user has to login, with a username and password, to access the system resources Telnet, TTY, or FTP servers. It was possible to assign permissions and privileges to different users, as with Unix.

Device	Processor	Memory Flash/RAM	Internet protocols	Serial ports	Preferred language	Network connection	Price
EtherNut	Atmel Atmega 103	128K/32K	TCP/IP HTTP	RS232	C	10base-T	\$125
Net186	AMD AM186-EX	512K/512K	TCP/IP HTTP	RS232	C Assembly	10base-T	\$420
OT731	Microchip PIC 16F877	128K/368B	TCP/IP UDP, PPP	RS232 RJ11	Assembly	2400 baud Modem	\$299
Picoweb	Atmel AT90S8515	8K/512K	TCP/IP HTTP	RS232	Assembly	10base-T	\$149
Rabbit TCP/IP	Rabbit microprocessor	512K/128K	TCP/IP, HTTP SMTP, FTP	RS232 RS485	C	10base-T	\$199
Siteplayer	Philips 8051	48K/768B	TCP/IP HTTP	-	SiteObjects	10base-T	\$99
Snijder EJC	ARM7TDMI	8MB/8MB	TCP/IP, HTTP SMTP, FTP TELNET, POP3	RS232 RS485 I ² C TTL	Java	10base-T	?
SX Evak Kit	Scenix SX52BD	32K/?	TCP/IP, HTTP SMTP, DHCP	RS232	C	10base-T	\$199
TINI	Dallas Semiconductor 80C390	512K/512K	TCP/IP, FTP, TELNET, DHCP, HTTP, SMTP	RS232 CAN I ² C	Java	10 base-T	\$85

Table 5.1: Comparison of TINI with other Internet enabled embedded devices (compiled from Eisenreich and Demuth, 2003).

Java was the preferred language for TINI code development. Complete Java is not supported for the TINI platform and resource intensive features such as serialization and reflections are not fully implemented. Numerous useful features were however available including multi-threaded programs and network support. The codes in this research work were written with Standard Edition of Java version 2 (J2SE) compiler, using only the features implementable on TINI. The resulting Java bytecodes were converted to TINI executable code using TINICConverter software. The converted code was uploaded to the TINI and was stored in its file system. The file system provided an organized manner of storage and supported separate sub-folders for each user.

5.2.3 Brief Introduction to Protocols

Communication, especially on the Internet, uses a large number of protocols to implement various facilities with various options. This has resulted in a large number of protocols each having its acronym. This section provides a brief introduction to various protocol acronyms that were used in this research. CAN has already been explained and is not covered in this section. Table 5.2 provides an introduction, including the port numbers used by certain servers. A port is a 16-bit number typically associated with a particular application layer service (Eisenreich and Demuth, 2003). The table also includes some acronyms used in mobile communications used in this research. Further details of Internet communication and protocols can be found in books such as Stallings (2004) and Tanenbaum & Steen (2002). Global networks use very complex technologies and a layered architecture is used for them. Tanenbaum (1996) provides detail of seven layers used in International Standards Organization's Open System Interconnection (OSI) model for networks. The Internet is practically working on a TCP/IP protocol suite which covers transport and network layers of OSI model. Application programs use the reliable communication service provided by the TCP/IP suite. A reduced set of the OSI model is also available for devices with low memory resources. This model covers the two lower layers and an application layer (Frankowiak, 2004). The author connected the SUIN to the Ethernet using a TCP/IP suite so that standard application layer programs could be used. FEN codes were developed using the reduced OSI model as they only have the memory built-in the MCUs. The robust CAN bus covers the two lower layers in the implementation and application layer on top of it takes care of the remaining issues.

ACRONYM	DESCRIPTION
Ethernet	Ethernet refers to the family of LAN products covered by IEEE 802.3 standard and is the most popular standard for LAN (Cisco, web site).
FTP	File Transfer Protocol manages uploading and downloading of files. FTP server listens on port 21 for client requests (Bentham, 2000).
GPRS	General Packet Radio Service provides always-on access to network and is suitable for non real-time Internet usage (Rappaport, 2002).
GSM	Global System for Mobile communication is a universal digital cellular system with modern network features extended to mobile users (Rappaport, 2002).
HTML	Hyper Text Markup Language contains predefined mark up tags that tells web browser how to display the web page (W3schools: HTML, web site).
HTTP	Hyper Text Transfer Protocol provides a web page in response to a browser request. HTTP server listens on port 80 for client requests (Bentham, 2000).
HTTPS	Hyper Text Transmission Protocol Secure provides secure web pages with encryption. HTTPS server listens on port 443 for client requests (Apple computers, web site).
IP	Internet Protocol delivers packets obtained from TCP to intended destination through any available path (W3schools: TCP/IP, web site).
SIM	Subscriber Identity Module is a memory device that stores user identification number and other user-related information for GSM (Rappaport, 2002).
SMS	Short Messaging Service sends alphanumeric pages of up to 160 characters between users in real-time (Rappaport, 2002).
SMTP	Simple Mail Transfer Protocol handles emails. SMTP server listens on port 25 for client requests (Bentham, 2000)
TCP	Transport Control Protocol is used for communication between applications running on different computers connected through Internet. It sets up full duplex communication that continues until one of the applications puts an end to it. It breaks data into packets and hand them over to IP for transmission (W3schools: TCP/IP, web site).
TELNET	TELEphone NETworking provides remote access for program execution. TELNET server listens on port 23 for client requests (Bentham, 2000)
TTY	TeleTYpewriter provides access for program execution through serial port and was developed for text telephone services (NOAA, web site).
XML	eXtensible Markup Language contains user defined mark up tags and compliments HTML (W3schools: XML, web site).

Table 5.2: Protocols and acronyms

5.2.4 Human Interface

TINI provides three servers for user access to its resources namely Telnet, TTY, and FTP. The author added a fourth one as an HTTP server. Telnet is an application layer protocol

on top of TCP/IP stack designed to provide a remote user with a command line environment. A remote user can access the TINI file system from his/her PC using the password protected Telnet server. A program on a PC, usually Telnet.exe, presents a logged in user with command line prompt. The user can then start/stop the execution of any program on the TINI and can also use Slush commands (if permissions set). Telnet is a powerful tool and access to this was restricted to the system developer/manager only. The TTY server was shutdown (using a Slush command) since this is designed for serial connection to a local PC rather than a remote access.

The FTP server can also be invoked by Internet Explorer software and files can be downloaded from the TINI in a more user friendly way. Internet Explorer however does not upload files to the TINI. A user cannot start/stop the programs on the TINI using FTP but he/she can still replace good code with bad code either maliciously or by mistake. The author therefore did not set such permission for ordinary users and reserved it for system developer/manager access only. A policy to enable the FTP server only when required was implemented.

The author did not consider the provided servers as safe media for presenting monitoring results to ordinary users. He therefore developed a HTTP server to host web pages showing the latest process status. Any user can access the web pages using a standard web browser, such as Internet Explorer or Netscape Navigator. No username or password is required to access such web pages and any interested person can see the results. Secure web pages that require user login and encryption use HTTPS protocol. The author considered this protocol too heavy for an 8-bit microcontroller already loaded with CAN communication and decision making. He recommended a small dynamic web page containing only the necessary information in coloured text and backgrounds. Use of graphics cause more data traffic and was therefore avoided. The SUIN code, developed in Java, generated HTML description for dynamic web page on-the-fly according to various process variables' status. Self-updating web pages were developed that refreshed themselves regularly ensuring that the user gets the latest information. The refresh rate (typically 10 seconds) of the page was included in the generated HTML code. Colours in the web page were used to grab user attention and background colours differentiated in normal and faulty conditions as well as under-control faults causing performance deterioration.



5.2.5 Mobile Phone Interface

A monitoring alarm is supposed to be sent to appropriate personnel for immediate action. The SUIN does update the web page code immediately on detection of an alarm condition. The user, however, cannot see the updated web page until it is refreshed because web browsers, such as Internet Explorer, do not allow any unrequested data to be sent to them. They work on 'pull' technology and discard 'push' items. The web page refresh rate should thus be sufficiently fast. A fast refresh rate however puts burden on SUIN resources and a compromise is required. The SUIN can push information to computers over the Internet but specialized software would then be required on the viewers' PCs. The aim of this research was to use generally available software with no need for proprietary software. A user can thus check the process status from anywhere in the world on any computer. Another limitation with web pages is that a user may not be close to a computer at the time of alarm generation. The monitoring system should thus have an alternate means to push alarm information to concerned personnel and mobile phone technology was selected for that purpose.

Mobile phones can now be considered as widely used devices supporting text, image, and voice communication with Internet access on GPRS and WAP. The monitoring system was connected to a mobile phone network using a Machine-to-Mobile (M2M) engine based on Sony Ericsson GR47 mobile device (that provided voice, SMS, MMS, and GPRS facilities with a SIM interface) in the developed system. An 8-bit microcontroller was connected to the M2M engine using serial communication and AT (attention) modem commands extended by Sony-Ericsson for its mobile phones. Complete functionality of a mobile phone can be controlled with the extended commands. The function used in this research was to send SMS messages only. A PIC 18F458 MCU was dedicated to deal with mobile communication. The PIC MCU contained several predefined text messages with destination phone numbers. It received information from the SUIN to send a specific message and acted accordingly.

5.3 FAULT DETECTION AND ISOLATION

The monitoring problem was divided into clear and simple logical decisions in order to reduce the computational overheads. This required a clear understanding of the process in

terms of any inter-relations of its signals. To achieve this, “normal” data was gathered from a healthy process under optimum conditions, and presented to the developer for analysis. Various faults were then intentionally introduced into the process and the resulting raw data was again captured. Fault finding procedures were then formulated based on this experimental gained process knowledge and were then embedded in the network nodes. This approach eliminated the requirement for any mathematical models of the process.

All process parameters of interest were individually acquired by a FEN mounted close to the sensor. Each FEN was programmed to apply the most suitable analysis method to its acquired signal such that it can then classify subsequent behavior as normal or abnormal. The various nodes combine the results from their fault finding procedures and can make process-related monitoring decisions at the first levels for the majority of cases. Any data determined as “normal” was not presented for off-line processing, unless the system was specifically put in data acquisition mode. This eliminated the need for data storage media.

The abnormality checks in each FEN could be applied to raw data, its calculated running sum, or its calculated running mean value for example. Thresholds were defined for each signal and signals were continuously compared, as appropriate, to either or both the upper and lower bounds. Out of bounds results indicated either a fault or a disturbance. Disturbances were generally expected to be of short duration and were noted but ultimately ignored by the FEN. The longer term abnormalities were reported by the detecting FEN to appropriate other FENs through a CAN message. Any FEN receiving such a message checked its own status and forwarded the combined information as an alarm message to the SUIN. The SUIN used this combined status information to isolate the fault cause according to the knowledge rules provided at the time of installation. The SUIN updates the process status in its results file as soon as a change is detected. The interested user can check the monitoring results on the Internet at any time. If a specific combination of signal conditions occurs, which was not considered during the system study the system will not be able to detect the fault cause and such a condition will be reported to the user. An engineer can then perform detailed analysis on data obtained by putting the monitoring system in data acquisition mode. The resulting newly gathered knowledge will then be integrated in the existing code so that the system can automatically deal with similar situations in the future.

5.4 CAN BUS MESSAGES

The monitoring system design consists of a SUIN and a number of FENs communicating through CAN bus connections and protocols. Each FEN performs its duty according to the signal it acquires. The SUIN is used to start/stop the monitoring process by conveying user commands to the FENs. It also generates a holistic view of the process being monitored, by gathering information from FENs, and provides it to the user via the Internet. The SUIN determines the number of active FENs in the system at boot-up and keeps an eye on new arrivals or departures of FENs. It thus provides plug & play capabilities to the system enabling it to function in changing situations. All nodes of this distributed system send CAN messages to each other to share information. The devised message structure for such communication used the extended CAN protocol with a 29-bit identifier. 4 bits were used for source node identification and another 4 for destination node identification. Their placement in the identifier is shown by the S and D bits respectively in figure 5.5. Further, the structure reserved 8 bits for message types, shown as T bits in figure 5.5, to allow up to 256 different message types. The 15 message types used in the monitoring system development to date are listed in Table 5.3. The details of these message types and their roles are described in the following subsections. In the 29-bit identifier eight additional bits indicated message sub-types (shown as M bits in figure 5.5). The sub-types were used to indicate message number during bulk data transfers. A missed number in this field indicated a message loss to the receiving node which would then generate a re-send request. The five remaining bits in the identifier were unused and thus available for future system enhancements. The unused bits were assigned logic 0 in the implementation.

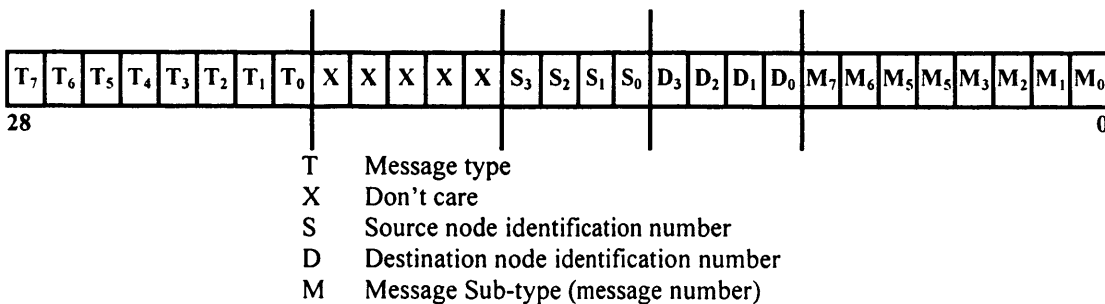


Figure 5.5: CAN message identifier

Message Type	Code	Originator	Receiver	Purpose
START	01h	SUIN	All FENs	Start monitoring
ACK	03h	FEN	SUIN	Acknowledge START
ERR	07h	FEN	SUIN	Abnormal signal detected
OK	08h	FEN	SUIN	Normal signal detected
EVENT	09h	FEN	FEN	Event detected
REBOOT	02h	FEN	SUIN	FEN powered-up
WELCOME	04h	SUIN	FEN	New FEN start monitoring
ACQ	06h	SUIN	All FENs	Start data acquisition
DATA	0Ah	FEN	SUIN	8 bytes of data
NEXT	0Bh	SUIN	FEN	Send next data
UPDATE	05h	SUIN	FEN	Update software
CODE	0Ch	SUIN	FEN	8 bytes of code
ACK_CODE	0Dh	FEN	SUIN	Acknowledge CODE
RESEND	0Eh	FEN	SUIN	Missing code request
RESENT	0Fh	SUIN	FEN	Missing code resent

Table 5.3: List of used messages types

5.4.1 Node Identification

Network nodes work asynchronously on independent clocks and send messages to each other for collaboration. Messages from each node compete with each other for bus access according to the identifier in the message header. Each node therefore must use unique identifiers that do not exactly match with any message generated by any other node on the network. The author therefore assigned a unique identification number to each node with this being used as the source node identification field in all message identifiers it generates. Four bits were reserved for the node identification number. This puts an upper limit of 16 nodes on the CAN bus, although the actual implementation allowed only 13 nodes. The actual implementation limit is due to the availability of only 14 similar message centres in the 80C390 MCU. The first MC was used for receiving broadcast messages as will be explained in next paragraph. The second MC was used for transmitting SUIN messages, and the remaining 12 MCs were dedicated to receiving messages from 12 FENs. This made total of 13 nodes (1 SUIN and 12 FENs).

Each generated message contained the intended 4-bit destination node identification number. Any message transmitted on the CAN bus can essentially be received by all

other nodes. The nodes' receive filters were however set in such a way that they only received messages when the destination node identification number matched either their identification number or was zero. Any message sent to destination 0 was accepted by all nodes and thus acted as a broadcast message. This scheme reduced the load on receiving nodes' processing engines as they processed only the messages intended for them. Any node can seek information from other nodes in this scheme even if the node with information was not initially set up to serve the seeking node. Having decided on this coding arrangement there would have then only been 3 bits remaining if the standard 11-bit CAN identifier was used. The need to use the extended CAN protocol (with 29-bit identifier field) was thus established.

To summarise, the following example confirms the use of the source and destination identifiers. If node 1 sends a message to Node 2, the node identification field would contain $S_3S_2S_1S_0 = 0001$ and $D_3D_2D_1D_0 = 0010$ bit. In general terms, with X depicting unknown bits for message type and sub-type and unused bits set as 0, the resulting message would contain the identifier described in figure 5.6. Alternatively if node 1 was to send a broadcast message then destination bits $D_3D_2D_1D_0 = 0000$ would be set. The resulting identifier (keeping the above given style) would then be as described in figure 5.7.

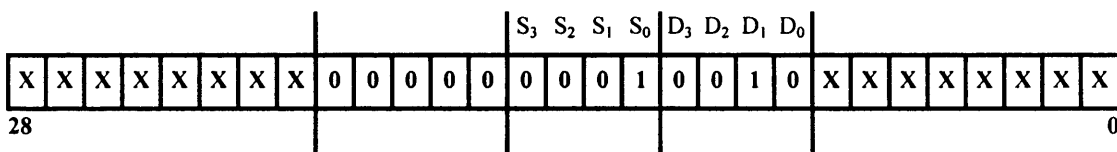


Figure 5.6: Message from node 1 to node 2

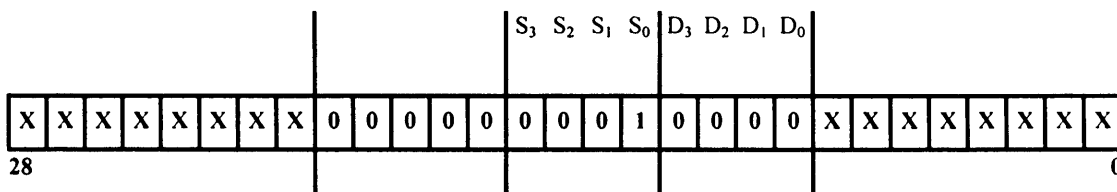


Figure 5.7: Broadcast message from node 1

5.4.2 Message Priority

A message is transmitted on CAN bus with the most significant bit of the identifier first. The bus arbitration is based on the contents of the identifier with logic 0 having a priority over logic 1 bit. The message with least number in message type, T bits in figure 5.5, therefore has the highest priority in the developed scheme. This scheme ensures that an important message has higher priority than others irrespective of its originating node. The author decided to assign higher priority to source node, S bits, than destination node, D bits, as can be seen in figure 5.5. A source node with a lower identification number has higher priority in this scheme. Identification number 0 is not assigned to any node because it is used for broadcast messages. The next highest priority identification number, 1, was assigned to the SUIN as it synchronizes the distributed system and should be able to interrupt other nodes. The remaining identification numbers were assigned to various FENs. Message sub-type has the lowest priority in the identifier and will never practically affect bus arbitration. Figure 5.8 provides some example identifiers showing message priority resolution. Higher priority identifiers are shown earlier than lower priority ones.

5.4.3 Power-up Sequence

Figure 5.9 shows the CAN message sequence at system power-up. Each FEN waits to receive a START message from the SUIN whereas the SUIN waits for a user to login using telnet and to issue a command. Automatic execution of the monitoring program at power-up is also possible. The SUIN broadcasts the START message on the CAN bus. A receiving FEN immediately starts its monitoring task and also sends an ACK message back to the SUIN. All ACK messages from FENs compete for bus access and would be received by the SUIN one by one in quick succession. The SUIN assigns different message centres for receiving messages from different sources and the messages do not overwrite each other even if not dealt with quickly. The source identification number bits are used to filter and store received messages in the appropriate MC. The SUIN checks the number of received ACK messages and thus determines the number of active FENs in the system. It also checks the sources of these messages and hence knows which monitoring inputs are available in a holistic scenario.

If the user does not start the monitoring program on the SUIN within a specified time, the FENs do not get the START message within their time-out period and assume the system to be already functional. They then send a REBOOT message to the SUIN. This provides

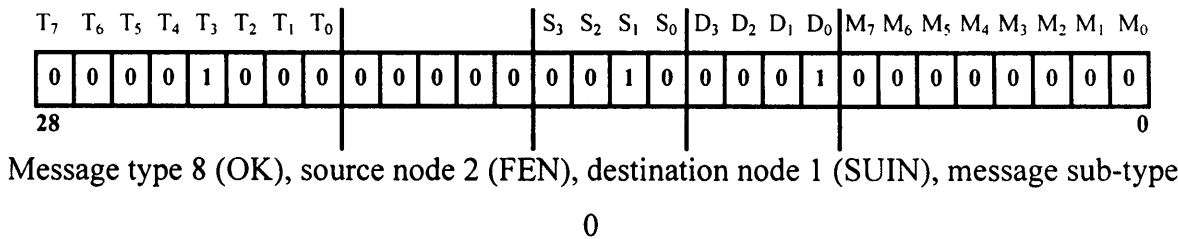
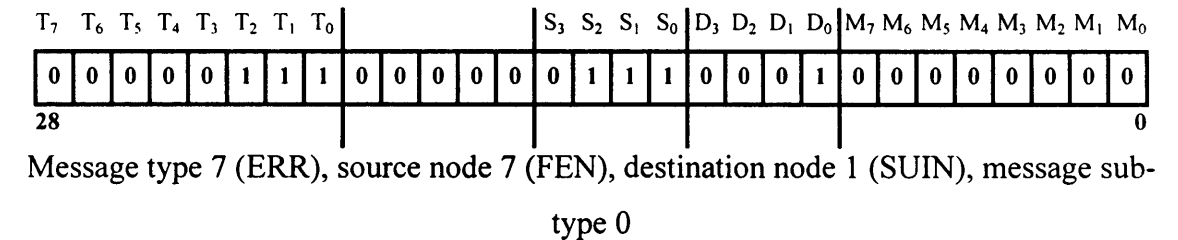
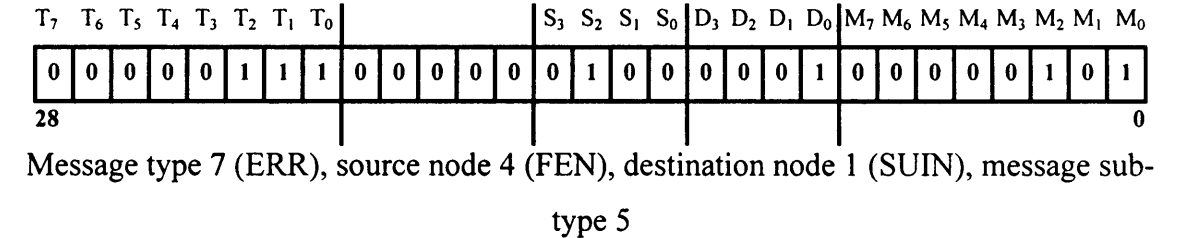
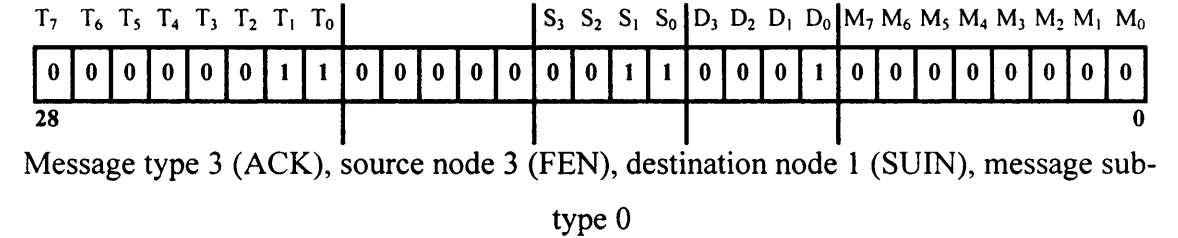
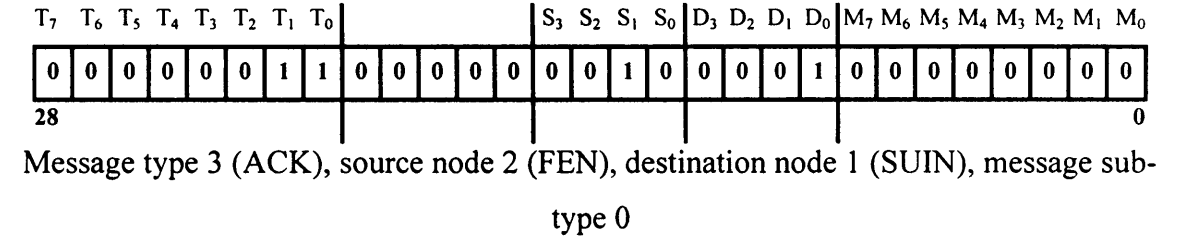
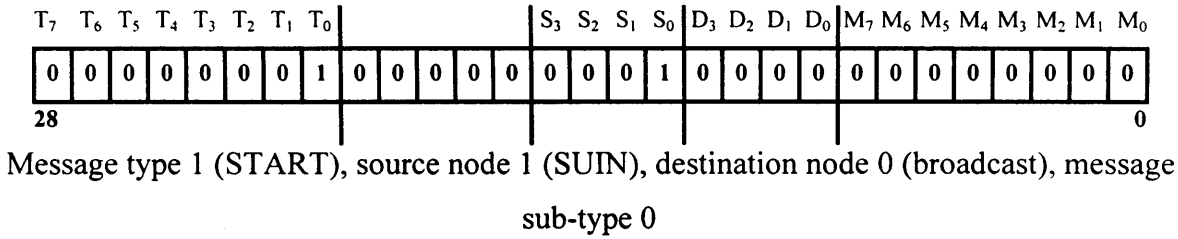


Figure 5.8: Some example identifiers of CAN messages

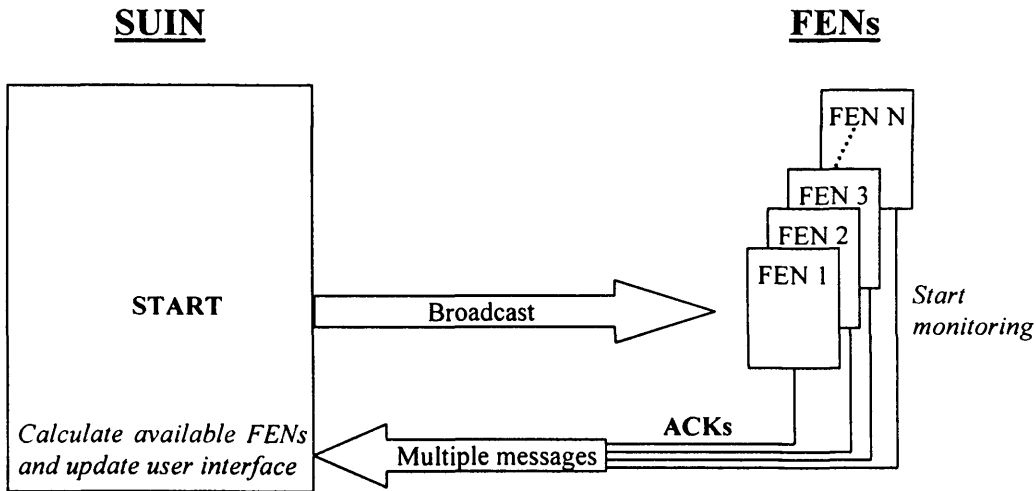


Figure 5.9: CAN message sequence at system power-up

plug & play capability to the system as will be explained in section 5.5. The FENs then remain in an infinite loop listening on the CAN bus for a message from the SUIN. Figure 5.10 shows the messaging when a monitoring program is executed on SUIN after FEN time-out period.

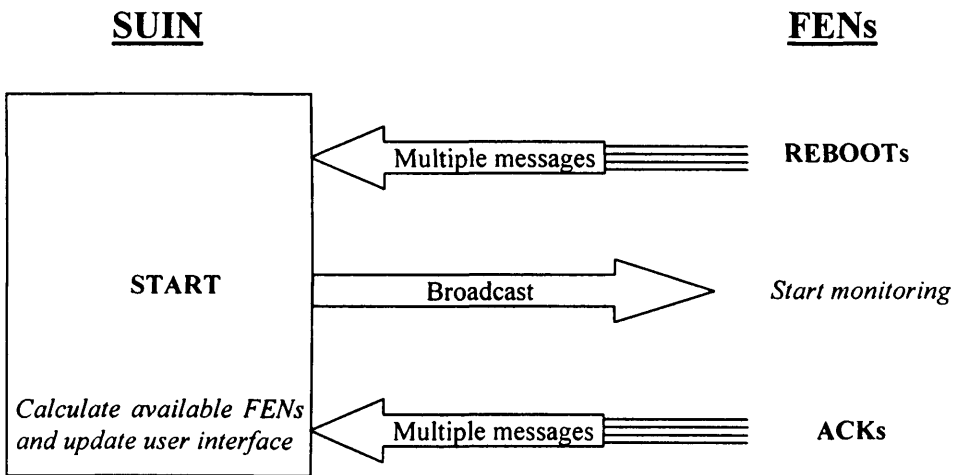


Figure 5.10: CAN message sequence after FEN time-out

5.4.4 Process Monitoring

An FEN determines whether the behaviour of its process signal input is normal or not. Accordingly, it sends either an OK or an ERR message to the SUIN. The SUIN receives all such messages from all FENs and evaluates the whole process health status. The SUIN is implemented on an 8-bit microcontroller and performing that many tasks in real-time is

difficult. The monitoring system therefore works on a peer to peer network paradigm and the FENs can send messages to each other when an abnormality is detected. FEN groups resolve the problem and provide the results to the SUIN which only then combines the information from these groups. This scheme reduces the load on the SUIN as it acts as a synchronizer only rather than a master. Figure 5.11 shows an example where a FEN detected an error which was later removed and normal signal was detected again. Node 2, for example, would send ERR message to SUIN as 00000111-00000-0010-0001-00000000 and OK message as 00001000-00000-0010-0001-00000000.

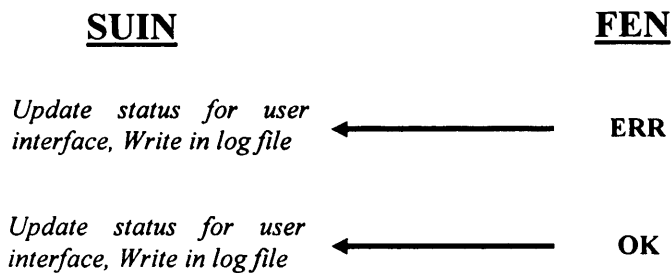


Figure 5.11: Messages during process monitoring

5.5 PLUG & PLAY

A plug-and-play facility was designed into the monitoring system. It can cope with a varying number of FENs being available for process monitoring. The SUIN takes care of changing node availability as certain FENs are switched on or off (or become inoperative) during process monitoring. A new FEN may be added to the already running monitoring system. The new FEN would wait for a START message initially and will not get it because the SUIN is not aware of its existence at this stage. The FEN would then send a REBOOT message to the SUIN, which then responds with a WELCOME message. A WELCOME message acts like a START message but is sent only to the newly connected FEN rather than as a broadcast message. The FEN acts on this message in the same way as to a START message and starts process monitoring. It also sends back an ACK message which completes the plug & play addition of the new FEN to the system. Figure 5.12 summarises the messaging sequence in this case. Consider a case when node 4, for example, is switched on and sends REBOOT message. SUIN will respond it with WELCOME message with identifier 00000100-00000-0001-0100-00000000.

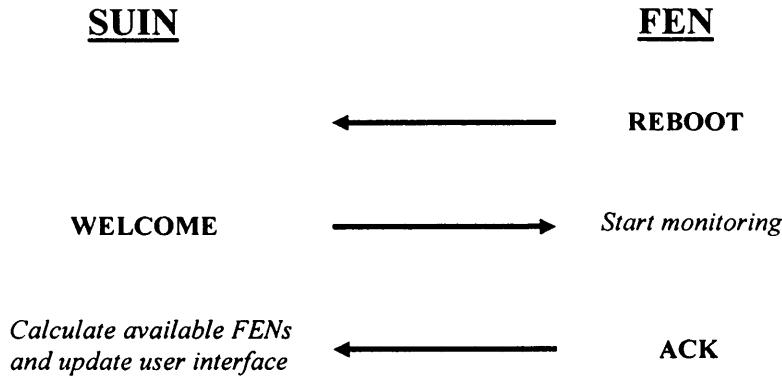


Figure 5.12: Message sequence when a new FEN boots-up

The SUIN also keeps track of available FENs through the messages they send to it. The absence of any message for a predefined time-out period is taken as a node's unavailability. It is therefore removed from the plug & play information and is not considered a part of the system any longer. The system then adapts itself to work with the remaining resources. An active FEN thus has to send a message to the SUIN within the required time. It actually re-sends the last process status (OK or ERR) message according to its acquired signal's health status.

The plug-and-play facility also helps when a node turns faulty as the SUIN can then adopt a reduced functionality model comprising the remaining nodes only. This is useful in processes where some optional plant components are switched off when not required. Such process changes do not require modifications to the monitoring system which can be set to detect modules that are switched on as the plant requirements change.

5.6 SOFTWARE MODELS FOR FENs AND SUIN

The proposed system can work in three different modes, namely the data acquisition mode, monitoring mode, and software update mode. The system presents raw data to the remote developer for analysis in the data acquisition mode. It provides remote users with monitoring results and process health in the monitoring mode. The software update mode is used to remotely upgrade the software on the nodes.

5.6.1 Data Acquisition Mode

The remote user starts the data acquisition program on the SUIN to gather raw signal data obtained by the FENs. The SUIN broadcasts an ACQ command to the FENs. Each FEN then starts signal acquisition and stores data in a circular buffer in its memory. A CAN message can take up to 8 data bytes and the FEN sends the DATA message to the SUIN once 8 bytes are gathered. The FEN includes the message number in the identifier so that the SUIN can arrange the received data correctly. The FEN waits for the NEXT message from SUIN before sending the next DATA message. This ensures that no data is overwritten in the SUIN. Figure 5.13 shows the FEN software model with the data acquisition mode elaborated. The corresponding software model for the SUIN is shown in figure 5.14. The SUIN dedicates a separate MC for receiving messages from each FEN and stores the acquired data in separate software buffers for each FEN. Data from a buffer is transferred to the file system when the buffer size reaches a certain limit. The resulting files are transferred to remote users over the Internet separately. Figure 5.15 shows the model for this software.

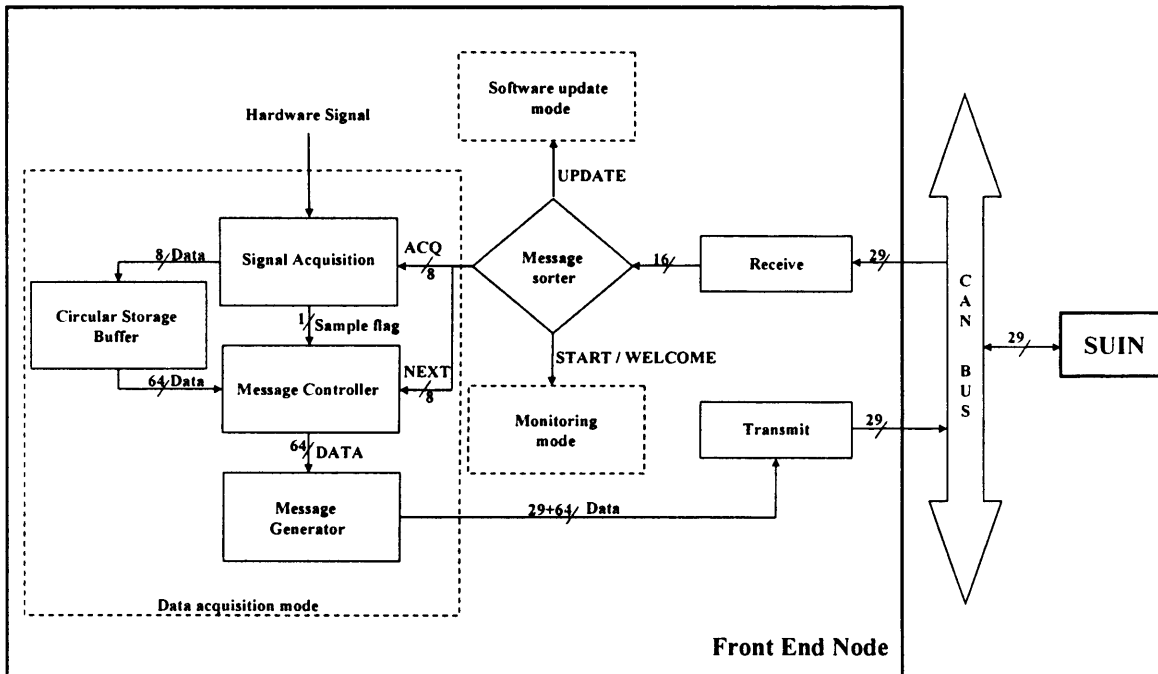


Figure 5.13: FEN software model with data acquisition mode elaborated

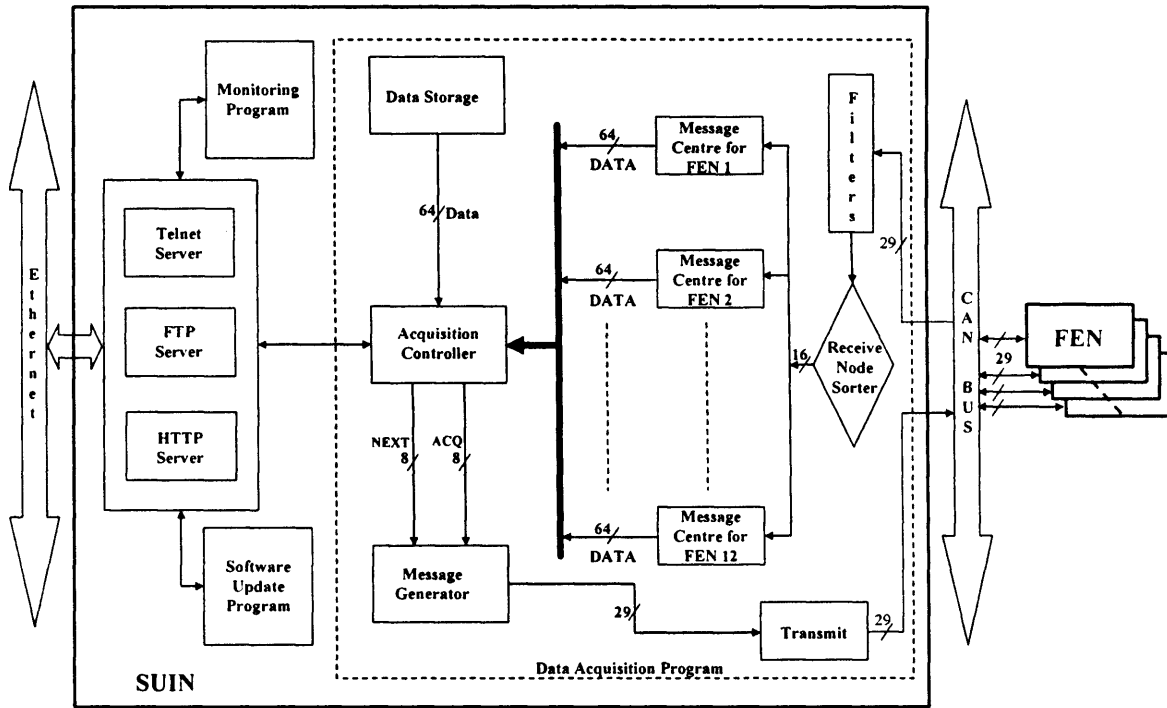


Figure 5.14: SUIIN software model with data acquisition mode elaborated

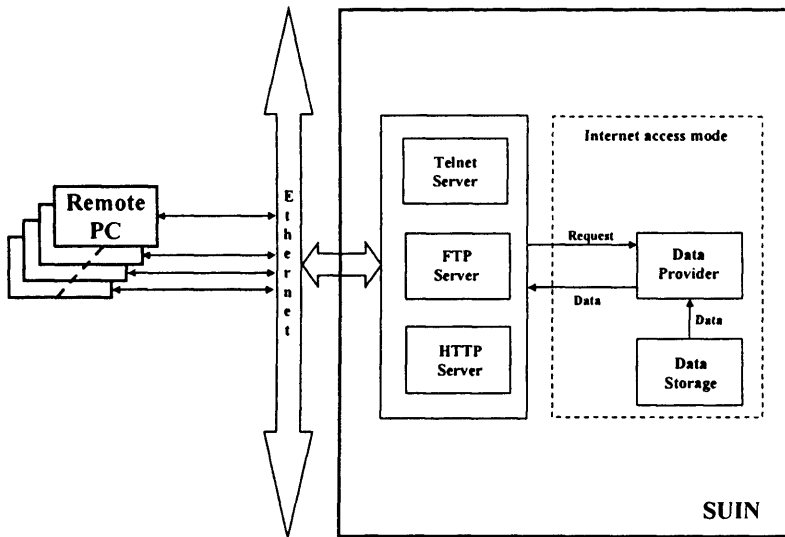


Figure 5.15: SUIIN software model for Internet access to data

5.6.2 Monitoring Mode

The remote developer analyses the data acquired in the data acquisition mode and finalizes the threshold values and the required processing method for each FEN. The

developer then configures the system accordingly and starts the program in the monitoring mode. Figure 5.16 shows the software model for the FEN with its monitoring mode elaborated. Figure 5.17 shows the corresponding model for the SUIN. The SUIN starts the monitoring process after checking the FENs availability. All FENs acquire their respective signals and apply processing to them. A FEN decides about its acquired signal's health and generates an 'ok' or 'error' result. The result is communicated to the SUIN. The FEN keeps on checking the signal status and sends messages as soon as it detects a status change. The SUIN gets such messages from all FENs and has a holistic view of the complete process. FENs can also send EVENT messages to each other such that a group of FENs can produce a partial view of the total picture and provide collective results to the SUIN. This FEN collaboration reduces the computational load on the SUIN which also takes care of the user interface. Source and destination identification numbers are used in message identifiers to implement this FEN to FEN communication. On detecting an event or error a FEN can send its data/status to another (predefined) FEN. The receiving FEN then combines its own information with the received information and sends it to the SUIN. Each OK or ERR message contains a code in its message sub-type part of an identifier in such cases which is decoded by the SUIN.

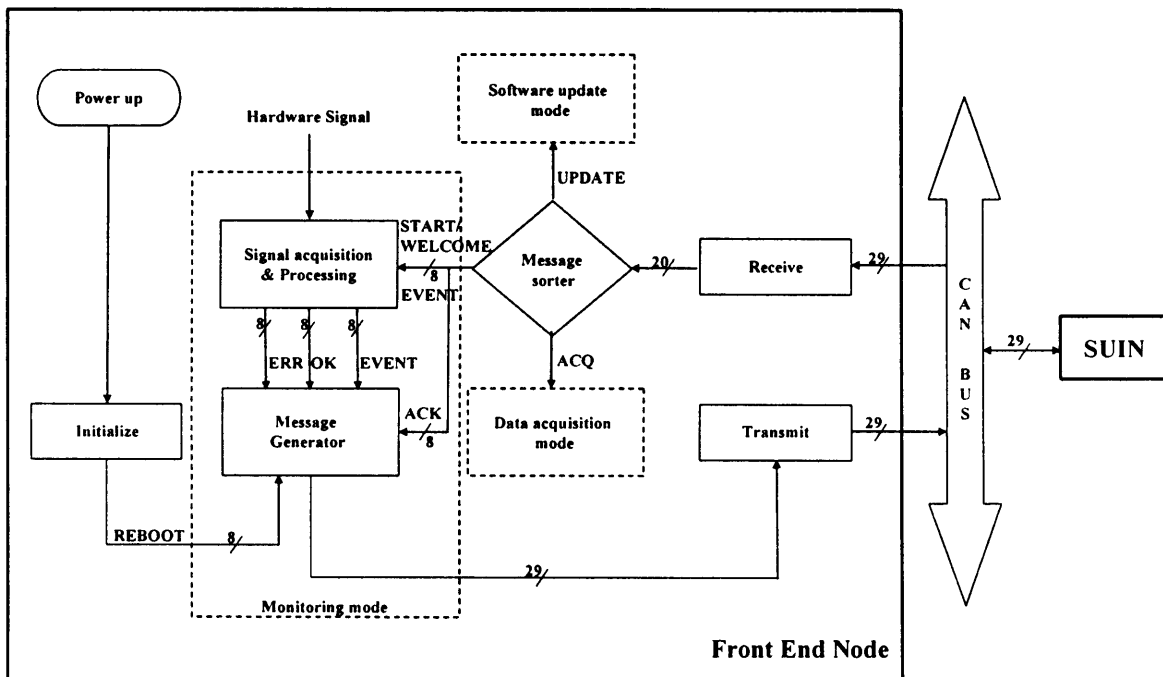


Figure 5.16: FEN software model with monitoring mode elaborated

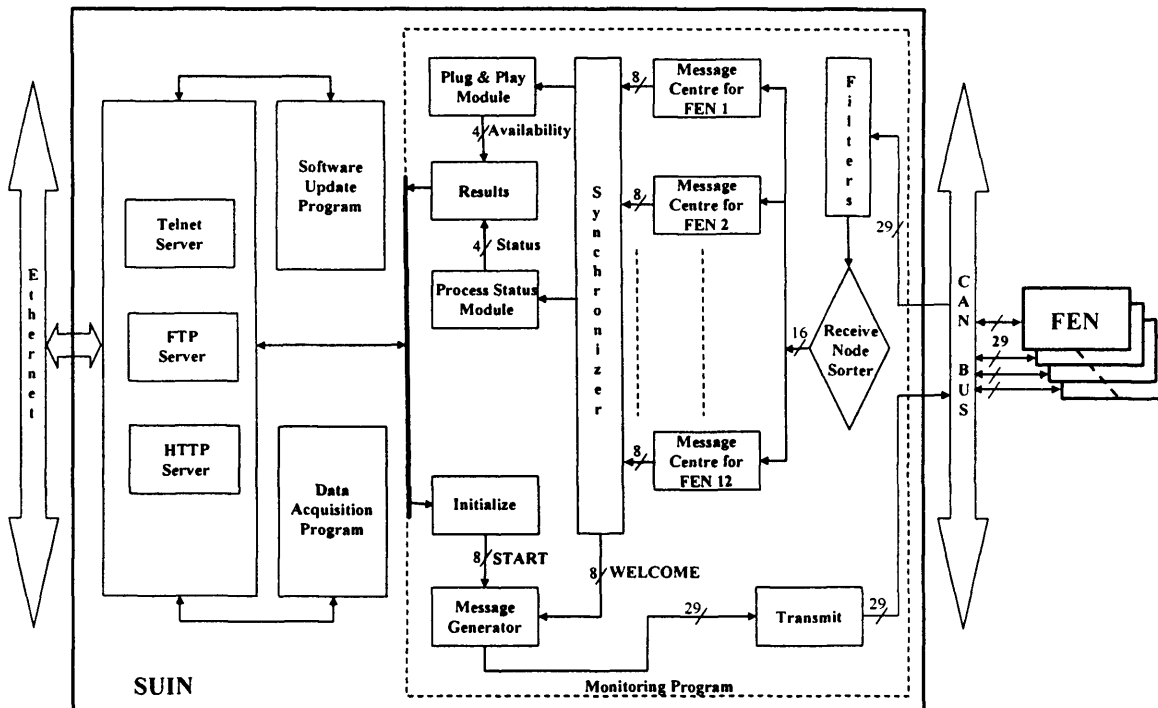


Figure 5.17: SUIN software model with monitoring mode elaborated

5.6.3 Software Update Mode

The proposed monitoring system has the capability to upload newer versions of code from the Internet under remote developer control. The developer can upload new SUIN and FEN codes to the SUIN using FTP protocol. The code for the SUIN is in the form of an executable program file that can be started like any other program. The new code for each FEN is delivered to the SUIN as a data file stored in the SUIN file system. The remote developer will start a program on the SUIN that will upload new code to each FEN using CAN messages. Figure 5.18 shows the FEN software model with its software update mode elaborated. The corresponding figure for the SUIN is shown in figure 5.19. The program sends an UPDATE message containing the FEN memory address to which the code should be placed along with the total length of code being sent. The program sends sequential CODE messages to the destination FEN containing 8 bytes of code. Instruction opcodes for PIC 18F458 are 16-bit wide and 2 bytes are required to transfer one instruction code. One CAN message can therefore takes code for 4 instructions. Message sequence numbers are used, in message sub-type field, so that the received code can be correctly arranged in the FEN. Any missing message can therefore be detected and a RESEND message is generated. The SUIN then sends the missing message again. Each

message is acknowledged by the receiving FEN by sending an ACK_CODE message containing the received message number in the message sub-type field. The SUIN re-sends a message for which it does not receive an acknowledging message. Re-sent messages use a different message type to originally sent messages and FENs can therefore detect a message received twice. The PIC 18F458 has the capability to write to its own flash program memory under its own software control. A FEN uses this capability and updates its code during run-time.

The code is written to program memory only when the whole promised code is received from the SUIN. The PIC 18F458 has more program memory than its internal data memory where the new code is stored initially. It means that the complete code may not be transferred in one go and several sets of code transfer may have to be executed. This is achieved by using the starting address and code length in each set and full code is transferred eventually. The boot-up code and CAN communication codes are not updated remotely. This is a precautionary measure against communication or power failure where a node may not be able to receive or program complete code in flash memory. A node should be able to function correctly on next boot-up and new code may be uploaded to it again.

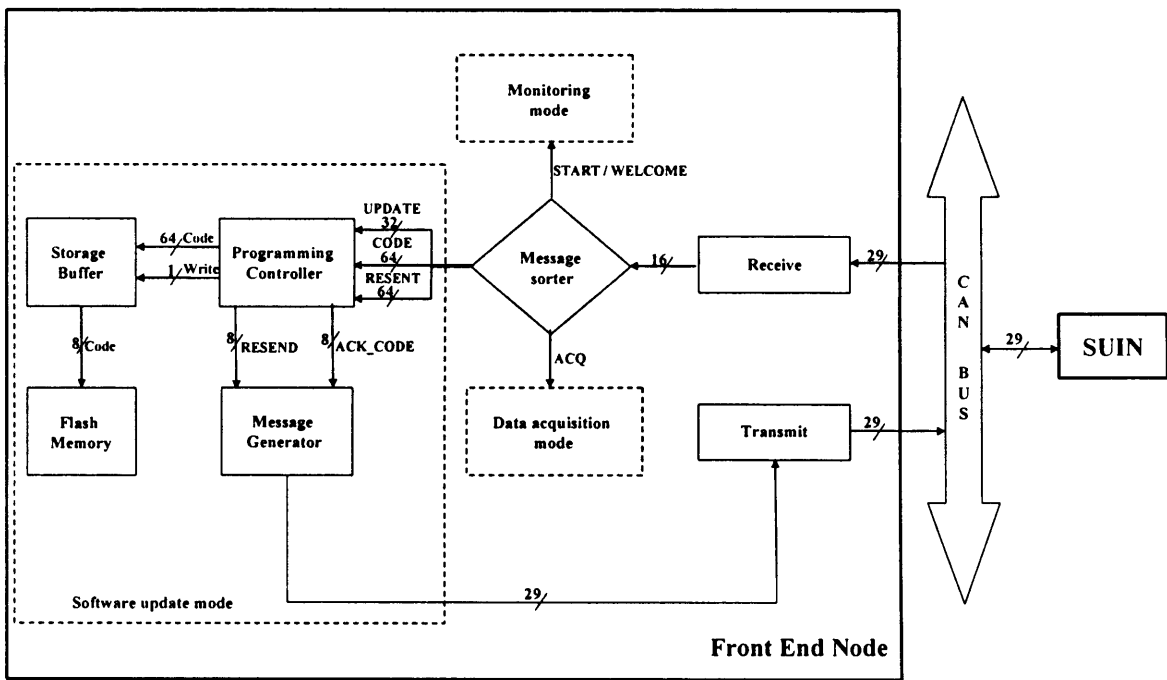


Figure 5.18: FEN software model with software update mode elaborated

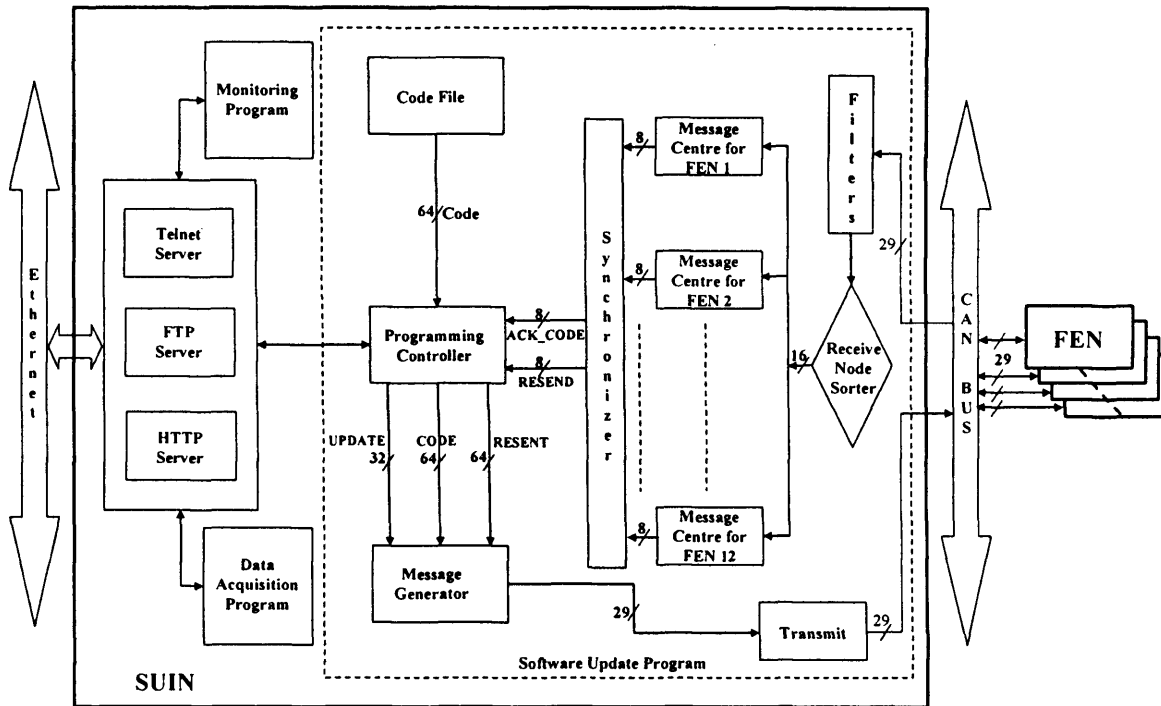


Figure 5.19: SUIN software model with software update mode elaborated

5.7 NETWORK TRAFFIC REDUCTION

Excessive traffic on communication lines in a distributed network causes bottlenecks and severe performance degradation. It was therefore essential to minimize network traffic ensuring smooth functionality. Periods of transmission of raw data caused maximum traffic on the network. The total throughput of the communication channels thus put an upper limit on the achievable FEN sampling rates in data acquisition mode. Accordingly, the author controlled the network traffic volume by carefully selecting the signals' sampling rates. Reliability was the most important aspect in the software update mode and the author ensured the secure delivery of every message using CAN bus error control mechanisms topped with acknowledgments and re-send requests. The software update mode did not require any real-time response from the network and posed no traffic problems.

The monitoring mode presented the most complicated situations where any event could occur at any time. It was not possible to accurately predict the network response and traffic in this mode. A fault detected by multiple FENs simultaneously could result in

excessive traffic with congestion, delays, and missed events. The author used interrupts with very short ISRs for quick event handling. The timer interrupt was used to generate accurate sampling times and also it initiated any required analogue to digital conversion process. Completion of the conversion process generated another interrupt and its ISR read and stored the results. The processor utilized the conversion time for performing other tasks. CAN message reception generated another interrupt in the FEN that transferred the message contents to a 16-message storage buffer, to be processed later in a subroutine. The PIC 18F458 MCU had only two interrupt service vectors and the author assigned higher priority interrupt vector to CAN so that no messages are lost. All other interrupts were serviced through the low priority vector where various flags ascertained the interrupt source. Computations were done in subroutines in the time between the interrupts. Pseudo-codes for the interrupts and associated subroutines are given in appendix C. Other measures employed in this research to reduce network traffic are given below.

- Processed results, rather than raw data, were transmitted on CAN bus.
- FENs collaborated with each other and the combined information was sent to the SUIN. This avoided bottlenecks that would be caused by sending everything to one node.
- In case a node observes an abnormal situation, it is highly likely that it will detect the same on next acquisition sample. A node was restricted from sending the same message again unless a certain predefined time was elapsed.
- Short messages were used with the majority having no data bytes attached. Information was included in message identifiers as message type and sub-types. For example, ERR messages included a code in message sub-type combining information from more than one FEN.
- Routine status messages from a FEN were used to detect its continuous availability and no special messaging was used.
- A FEN sending no message within time-out period was automatically considered unavailable and no messages were generated for it.
- Urgent messages were assigned higher priority to ensure their quick delivery.
- Destination identification was included in message header so that unconcerned nodes were not disturbed.

5.8 FAULT REPORTING BY SMS

Monitoring results in this research were primarily displayed on a dynamic web page for any remote user. An inattentive or otherwise occupied user may miss a monitoring alarm by not paying attention to the web site all the time. Mobile phone connectivity was added to the monitoring system to communicate with remote users. The monitoring system could thus distribute SMS messages to important remote users as real-time alarms. The message receiving tone from the mobile phone would alert the identified receiver who could then take action necessary for fault mitigation. Once alerted, he can also check the full monitoring results by accessing the dynamic web page again through the mobile phone. SMS messages were generated only on the first time detection of a fault. Persistent faults were not reported on mobile phones repeatedly as that would annoy the receiver besides escalating phone bills.

Figure 5.20 outlines the hardware arrangements in the distributed monitoring system for SMS generation. A Sony Ericsson GR47 is a GSM/GPRS radio device that provides connectivity to mobile phone network through a SIM and is optimized for M2M communications (Sony Ericsson web site: GR47/GR48). Comtech's μ WEB LITE module hosted GR47 and provided a serial connection for other devices to communicate with it (Sony Ericsson web site: Where to buy). The author connected the PIC 18F458 MCU to GR47 using this serial connection and communicated modem AT (ATtention) commands to it. The MCU was thus in a position to initiate and receive mobile communications and had access to Sony Ericsson's extended AT command set (Developer Guidelines, 2005) for modern features such as SMS, MMS, and Internet access. The monitoring system did not require incoming messages and phone calls and all incoming services were barred. A shareware software was used to convert error messages (ASCII characters) into Protocol Data Unit (PDU) format for SMS (USB Developer web site, 2005) and the resulting PDU codes were stored in the PIC MCU along with predefined phone numbers.

Mobile phone connectivity was added to the monitoring system towards the end of the reported research and was not fully integrated into the system. SMS messaging was not used with the research reported in chapter 6, 7, and 8. It was used only for air flow process monitoring reported in chapter 9 and was tested as an add-on feature. Parallel port was used for communication between the SUIN and the new node (the PIC

microcontroller responsible for SMS generation) for initial development. The new node will communicate on the CAN bus after its full integrated in the system. Node identification number 14, previously unused, would be assigned to the new node. One message centre in the SUIN was previously left unused and would receive messages from this new node.

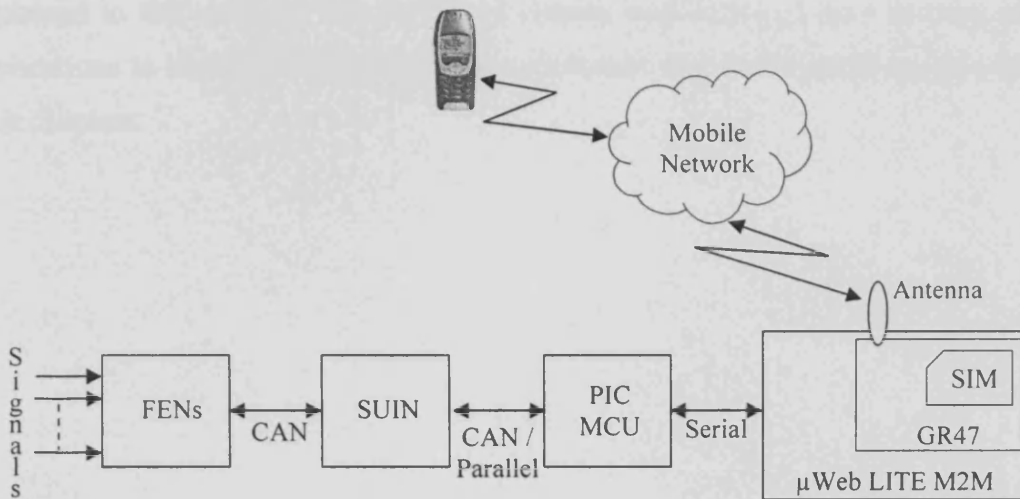


Figure 5.20: Hardware arrangements for SMS generation.

5.9 SUMMARY

Details of the first and second hierarchy layers of an 8-bit microcontroller based distributed monitoring system were provided in this chapter. The nodes of the system communicated using a CAN bus and via the Ethernet (Internet) to either the remote system developer or to monitoring system users. The capability to send urgent messages to remote users on their mobile phones was also provided. The first and second hierarchy layers were implemented solely on 8-bit microcontrollers and it is believed to be the first reported microcontroller-based system providing monitoring results from sensors to remote users. It provided specific monitoring results based on time and frequency analyses on discrete and continuous signals and was considered to be unique.

The messages communicated between various system nodes were provided in detail to explain the operation of the features of the deployed system including; plug & play, fault detection, and isolation. The microcontrollers were not able to simultaneously provide all

the required features because of their limited resources and the system operation was divided into three separate modes. Software models for these modes were described in this chapter for both the SUIN and the FENs. Measures enabling real-time operation by reducing network traffic were also detailed.

The functional aspects of the hierarchical and distributed monitoring system were explained in this chapter. The developed system was deployed on a number of process applications to evaluate its performance in real-time and their details are provided in the next chapters.

PIPE BLOCKAGE DETECTION

A distributed monitoring system was implemented according to the methodology described in earlier chapters. This chapter describes details of the application and testing of the monitoring system for detecting pipe blockages in a laboratory-based process rig. In summary, process signals were acquired via the FENs and were communicated over Internet. During system development, the signals were analysed for fault symptoms to determine the appropriate processing method(s) for the FENs. Threshold levels were also determined for each signal and were then programmed into the respective FENs. Subsequently, the FENs provided their results to the SUIN where their combinations were then automatically processed to identify and confirm a fault condition. The holistic information available to the SUIN was also used to determine the extent of any pipe blockage. Also, as an example of a remote user interface, the monitoring results were made available via dynamic web pages. These presented the extent of a fault as a low, medium, or high level. The example application and the results obtained are presented in more detail in the following sections.

6.1 BYTRONIC PROCESS RIG

A laboratory test rig from Bytronic was used to emulate a batch process in this research. Figure 6.1 shows the Bytronic test rig and highlights its salient features. The rig contained a lower level sump and an upper level storage tank. A centrifugal pump transferred water from the sump to the tank and its speed was controlled from a PC. The sensors signals that were used were a water level sensor (tank mounted) and a flow sensor (mounted in the connecting pipe). Manual valves in the connecting pipe were used to emulate faults such as partial blockages and leakages. A solenoid valve, controlled by the PC, was provided to empty the tank water back into the sump. A manual valve was also provided for the same purpose and could emulate leakage from the tank.

The test rig thus provided a variety of physical signal types in voltage, current, and pulsed signal formats (as summarised in table 6.1). It was therefore ideal to enable the evaluation of FEN suitability for interfacing with various kinds of physical signals. All sensor and

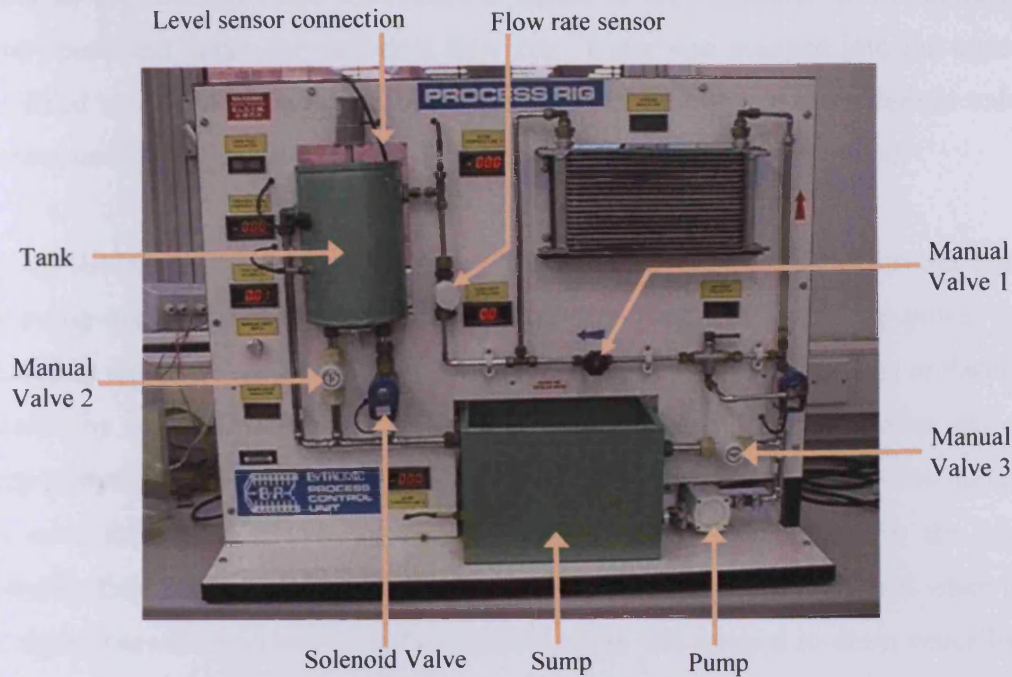


Figure 6.1: Bytronic process rig

Process variable	Description	Level
Pump power	Analogue	0 to 10 V
Solenoid valve control	Digital	0V / 24V
Water flow rate	Pulse rate	0V / 15 V
Water level	Current loop	4 to 20mA

Table 6.1: Signal description for process variables

control signals for the rig were connected to a Siemens ET200M distributed I/O system (SIMATIC, 1998). The distributed I/O system had been added to the standard Bytronic test rig in previous studies and provided a convenient link between the test rig and the controlling PC. Profibus was used as the communication medium between the PC and the ET200M and batch type control was programmed via a LabWindows interface.

6.2 BATCH PROCESS

Some of the previous work with the modified Bytronic rig is reported by Hopkins (2001). He developed a PC application emulating a chemical batch process where water was transferred from lower level sump to higher level tank under feedback control. The level

sensor in the tank provided the feedback signal to the controller which controlled the pump speed and hence the delivered flow rate. Water was pumped into the tank until it was filled to its full capacity and then was emptied by opening the solenoid valve. The operator could specify the desired number of batch cycles to be completed.

The application developed by Hopkins (2001) used a very simple controller algorithm. The pump operated at full power until the tank was 70% full. The pump power was then reduced in steps at every 10% increment of tank fill. This controller was replaced in this research by one making better use of the feedback signal. The controller changed the pump power dynamically to fill the tank according to a certain pattern. 70% pump power was used for the first 4 seconds, enabling flow to be established in the pipe. The controller then started working in feedback mode. The pump was stopped when the tank full signal became true and then the solenoid valve was opened to drain water back into the sump. Figure 6.2 explains the signals used in the batch process for both control and monitoring purposes.

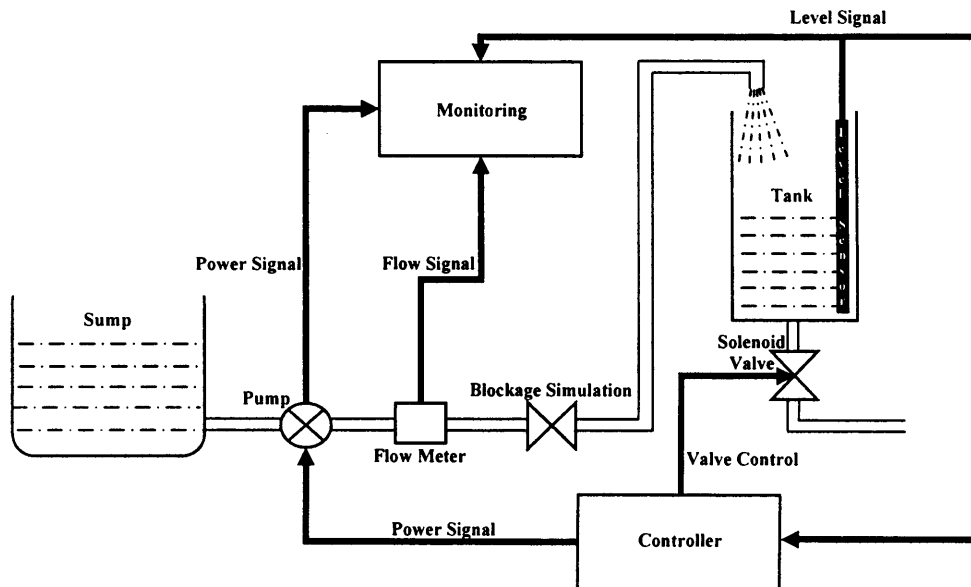


Figure 6.2: Control and monitoring signals in batch process

6.3 PROCESS MONITORING

It was decided to deploy the distributed monitoring system to locate pipe blockage faults during the existing batch cycles and control regimes (as described above). Signals were acquired directly from the existing sensors and control elements. They were simply

conditioned to limit them to a 0 - 5 volt range as required for MCU interfacing. The water level signal was also converted into this range from its original 4-20mA current loop. Each conditioned signal was connected to a FEN; three FENs acquired pump power, flow rate and tank level signals. The coordinating SUIN was connected to the Ethernet system provided at Cardiff University. It was therefore possible to monitor the process application and transmit the results to a remote user. Unique identification numbers were assigned to all monitoring nodes and these were used to specify source or destination nodes in the CAN messages. Table 6.2 summarizes the assignments used.

Node ID #	Node type	Acquired signal	Description	Level
1	SUIN	-	-	-
2	FEN	Pump power	Analogue voltage	0 to 5 V
3	FEN	Water flow rate	Pulse rate	0V / 5 V
4	FEN	Water level	Analogue voltage	0 to 5 V

Table 6.2: Monitoring node identification numbers

Table 6.3 provides the details of actually acquired signals and the conditioning applied to them, the levels being determined and set after some initial testing of the process control system.

Process variable	Signal range	Further conditioning	Resulting signal range
Pump power	0.7 to 5 V	0.7 volts subtracted by diode	0 to 4.3 V
Water flow rate	0 to 2.2 V	Signal amplified with gain 2 by op-amp	0 to 4.4 V
Water level	1.6 to 2.4 V	1.6 volts subtracted by op-amp summer. Remaining signal amplified with gain 6 by another op-amp	0 to 4.5 V

Table 6.3: Further signal conditioning for MCU interfacing

The process was run and monitored over a number of batch cycles in normal (fault-free) conditions. The monitoring system was operated, during this phase, in data acquisition mode to capture full data sets for all monitored signals. Figure 6.3 shows typical normal condition signals for (a) pump power, (b) tank water level, and (c) pipe flow-rate for a batch cycle. For brevity these signals will be referred to as 'Power', 'Level', and 'Flow'

respectively in the rest of the thesis. In each case in figure 6.3 the Y-axis shows the actual values (8-bit A-to-D results) acquired by respective FENs. No further computations were done to the acquired numbers (to convert to units or percentages, for example) as it was easier for FEN application programs to deal with raw 8-bit numbers. Figure 6.3(d) combines all these signals on a single plot for comparison.

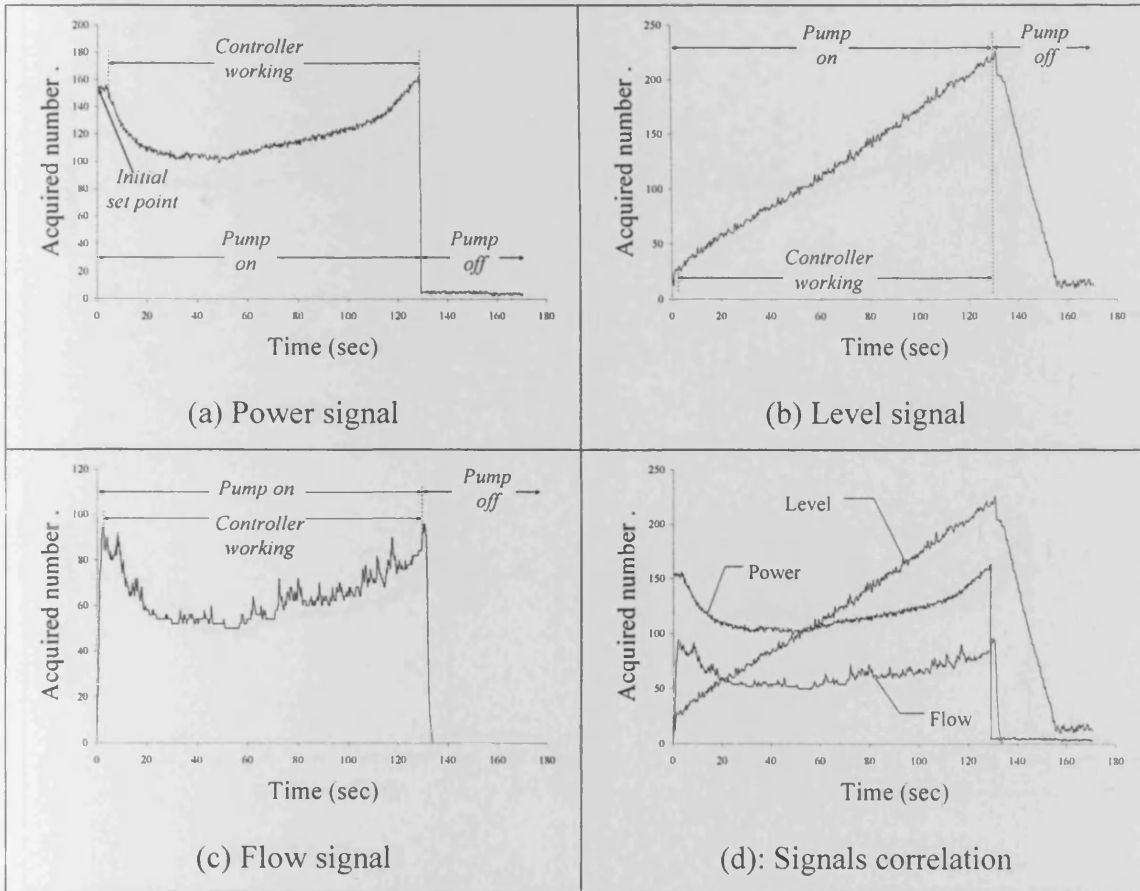


Figure 6.3: Typical normal condition signals

6.4 FAULT SIMULATION

Partial pipe blockage was used as a test fault. A rotary manual valve (manual valve 1 in figure 6.1) was partially closed to various degrees to simulate partial blockage faults. Figure 6.4 shows the results obtained for 5 levels of simulated blockages. The blockage percentages assume a linear characteristic for the rotary valve. The tests were repeated several times to confirm the consistency of the results. The partial pipe blockages caused

disruption to the water flow and reduced the tank fill rate, as can be seen from the increasing cycle times. The controller attempted to compensate by applying increased pump power (to maintain the required water level in the tank at any given time). Thus the pump power increased gradually with blockage severity, again as shown in figure 6.4.

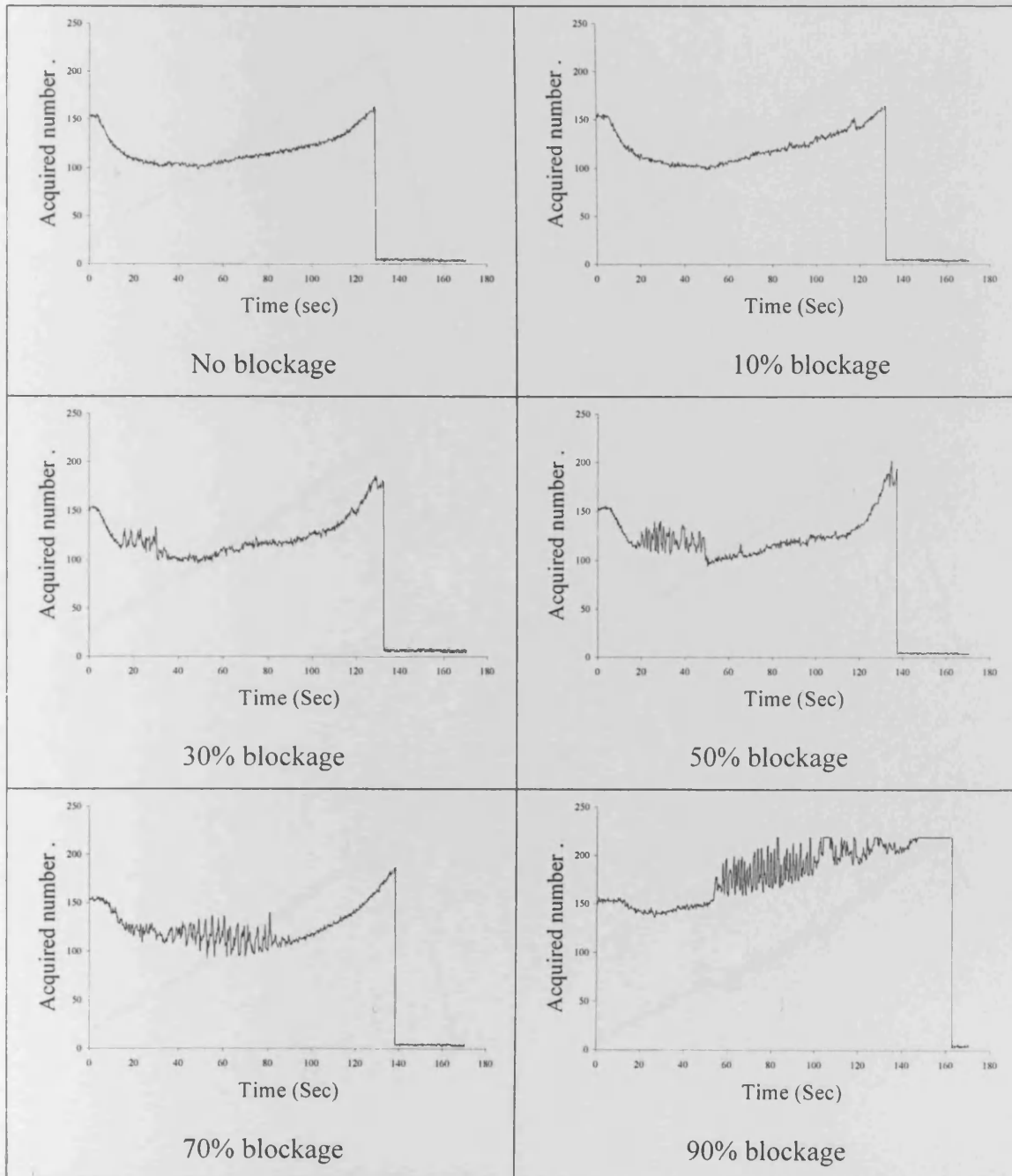


Figure 6.4: Typical control signals (power) for partial blockages

Figure 6.5 presents the tank level results. The observed insensitivity of this signal to partial blockages is due to the compensating controller efforts. The increased overall cycle time is, however, apparent. It was deemed that detection of increased controller activity provided a mechanism for fault detection in this example application.

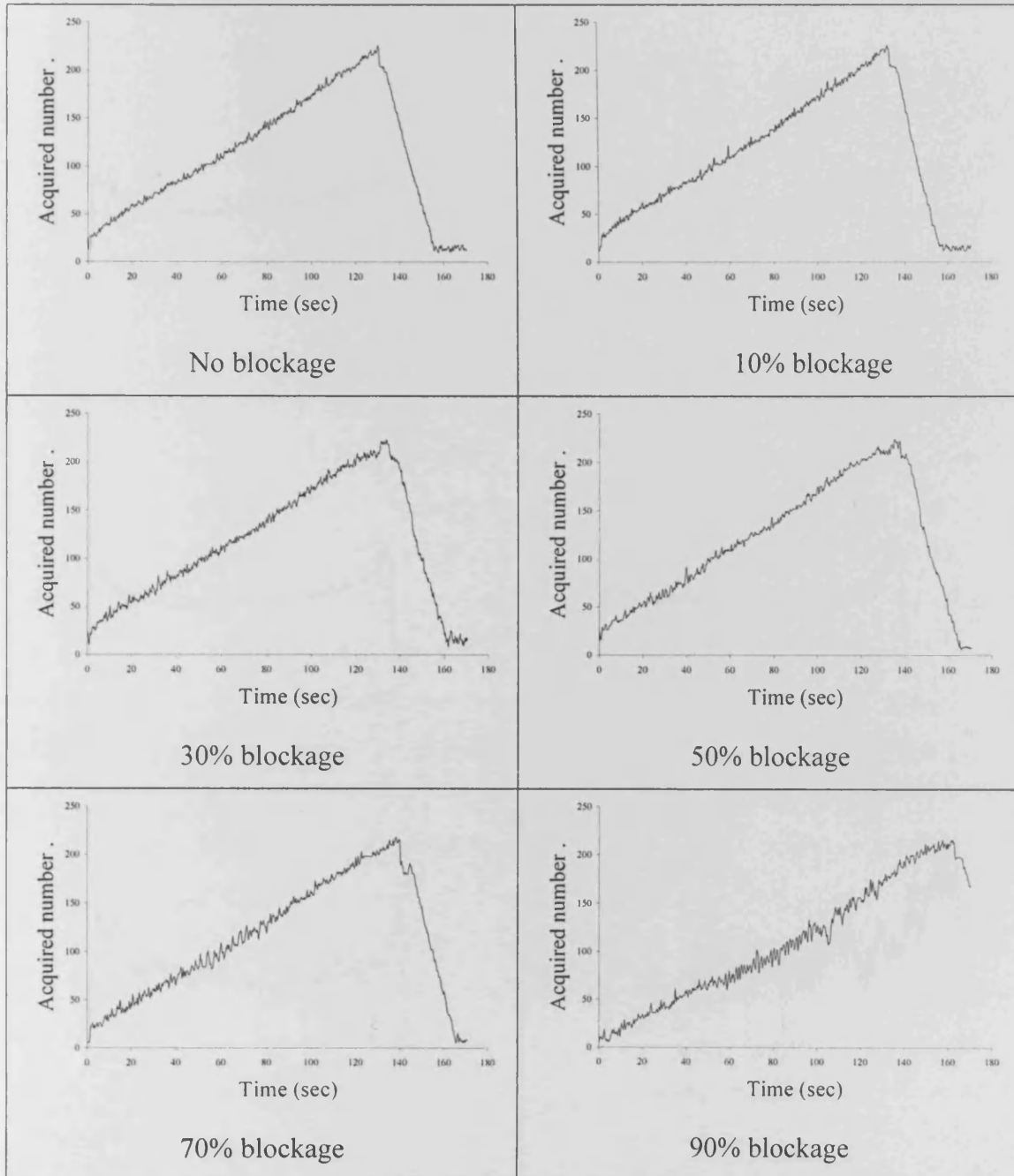


Figure 6.5: Typical tank level signals for partial blockages

The third acquired and monitored signal, the flow rate signal, proved to be of more interest than expected. Prior to testing, a gradual reduction in flow rate was expected with increased blockage levels, and indeed this was observed for pipe blockages up to 50%. However, for larger blockages, the acquired data typically showed both increased magnitude and fluctuations, as shown in Figure 6.6.

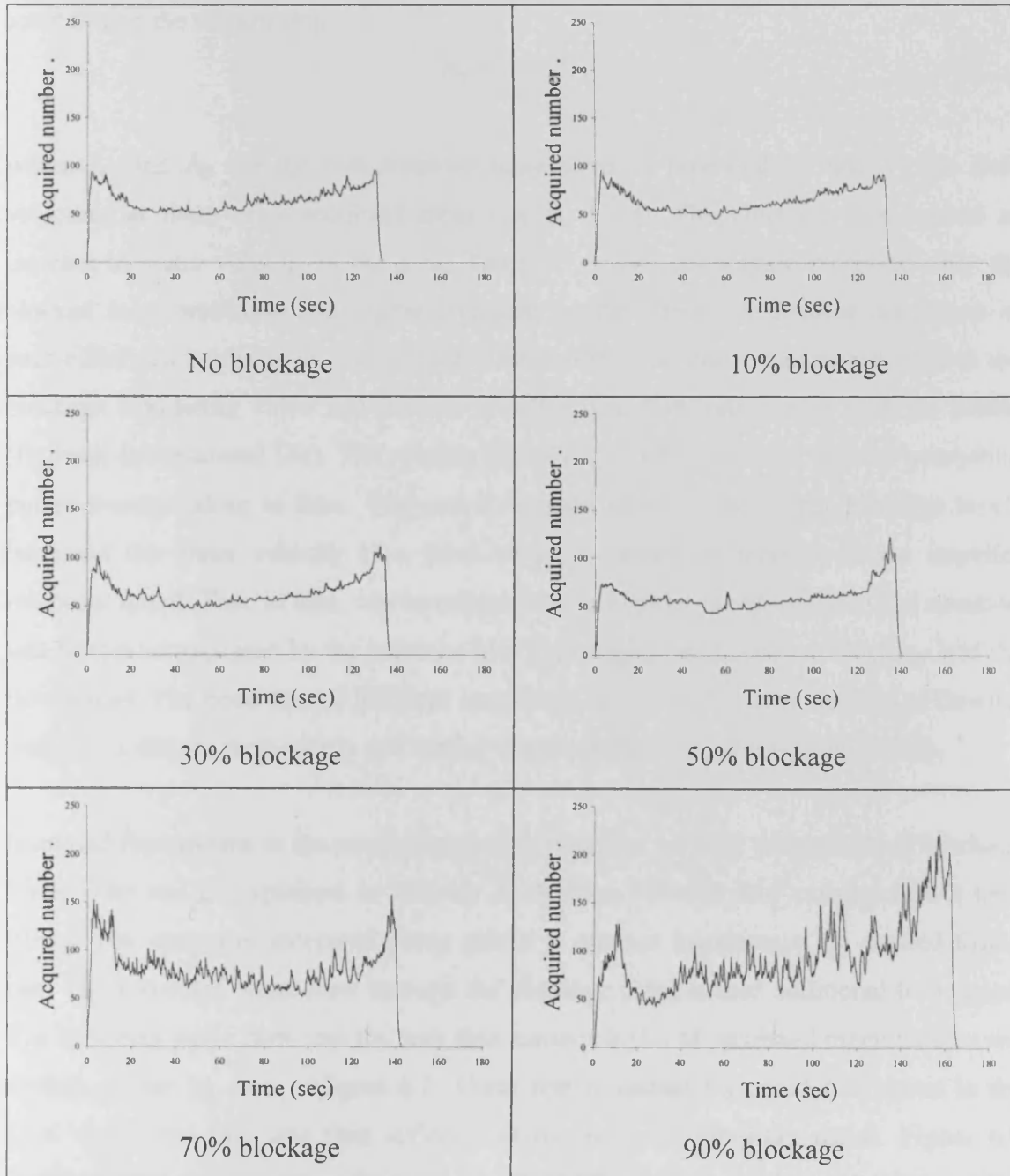


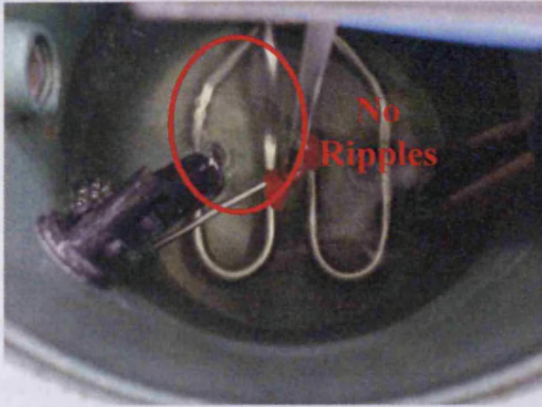
Figure 6.6: Typical flow rate signals for partial blockages

The proposed explanation for this behaviour is as follows: A pipe blockage reduces its effective cross-sectional area. The pump operating at a given power level will thus attempt to deliver the same flow of water through the pipe. Any reduction in water volume accumulating in the tank was compensated by increased pump power. Thus the same volume of water passed through the pipe irrespective of the blockage. The change in cross-sectional area of pipe however affected the velocity of incompressible fluid (water) according to the relationship

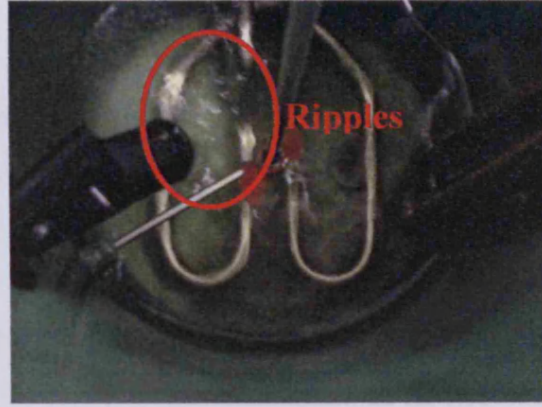
$$A_1V_1 = A_2V_2$$

where A_1 and A_2 are the two cross-sectional areas of pipe and V_1 and V_2 are fluid velocities at these cross-sectional areas (Tullis, 1989). The blockage thus caused an increase in water velocity in the pipe. The pipe cross-section again increased after the blocked area, providing convergent-divergent nozzle effects. A detailed discussion of such effect can be found in Ward-Smith (1980). The flow sensor was placed close to the blockage simulating valve and was an impeller type flow rate sensor with six blades (Bytronic International Ltd). The rotating blades cut an infra-red beam thereby generating pulses corresponding to flow. The con-div nozzle effect at the higher blockage levels increased the water velocity to a level where it caused an increase in the impeller rotational speed. This, in turn, was communicated to FEN as increased flow. The situation was further complicated by the presence of a right angled bend between blockage and the flow sensor. The bend caused different centrifugal forces on different sections of flowing water according to their speeds and further disturbed the flow (Ward-Smith, 1980).

Increased fluctuations in the power signal were observed with the progression of blockage levels. This can be explained as follows. A blockage reduced flow causing slower tank filling. The controller increased pump power to attempt to maintain the desired filling rate. The increased water-flow through the blockage point caused additional turbulence. The incoming water flow into the tank then caused ripples of increased magnitude on the surface, as can be seen in figure 6.7. These ripples caused increased fluctuation in the level signal and this was then reflected in the power (controller) signal. Figure 6.8 confirms this behaviour when the power, level, and flow signal trends are shown together. The effect of flow turbulence can be seen to be increasing with increased blockage levels.



(a) Normal



(b) 50% blockage

Figure 6.7: Ripples effect of blockage

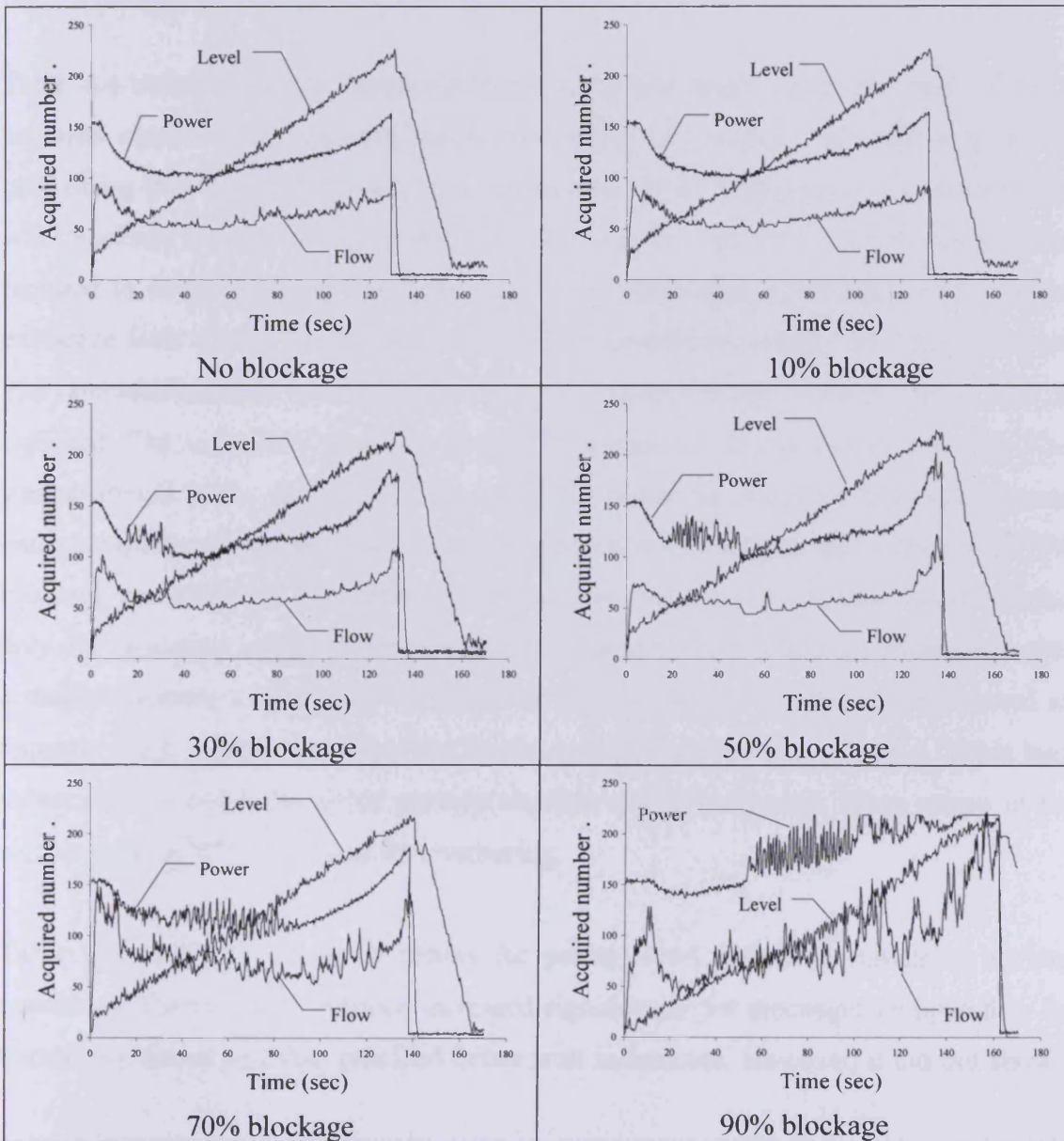


Figure 6.8: Turbulence effect of blockage

6.5 MONITORING DECISIONS

The next stage of research with the process rig application was to analyse the results obtained and to develop fault detection algorithms for deployment on the distributed monitoring system. The observable differences were not easily translated because of varying control strategies during a batch cycle and increased signal fluctuations under fault conditions. Further, relatively simple algorithms were required to be consummate with the numerical manipulation capabilities of the PIC MCUs. Simple statistical methods such as mean, variance, and running sum were considered therefore. Fault symptoms obtained with these techniques are described below with running sum found as the most suitable technique.

Table 6.4 summarizes the variations in the computed mean values for each of the 3 acquired signals with increasing blockage severity. The means were calculated on the tank filling part of the cycle only. The mean value of the pump power signal increased with blockage severity. However the sensitivity was low and very strict thresholds were required to detect the marginal differences. Since such strict thresholds could generate excessive false alarms the mean-power was not considered suitable for fault detection. The calculated mean-level effectively remained constant for pipe blockages up to 50%, as expected. The mean-level however sharply decreased for higher blockage levels. This was attributed to the delay in batch completion caused by excessive blockage. Normal batch completion times had been observed to be up to 133 seconds and a batch with 70% blockage completed in 138 seconds, for example. Mean-level was thus found suitable only for detecting severe blockages causing out-of-control situation. Mean-flow also remained virtually constant for blockages up to 50%. Higher blockage levels caused an apparent sharp increase in mean-flow for the reason discussed in section 6.4. These high values did not depict the actual process situation and were ignored. Mean values of the signals were therefore not used for monitoring.

Table 6.5 shows the variance results for power, level and flow signals in various conditions. Power signal variance increased significantly for blockages compared to for normal conditions and thus provided better fault indications. However, it did not show a

gradual rise with increased blockages and was not useful for identifying fault extents. Level and flow variances also provided roughly the same results.

Blockage	Mean power	Mean level	Mean flow
0 % (normal case)	119	120	63
10%	121	119	63
30 %	124	121	62
50 %	125	123	59
70 %	126	115	79
90 %	181	106	93

Table 6.4: Mean values of acquired signals (no units used for ADC output numbers)

Blockage	Variance power	Variance level	Variance flow
0 % (normal case)	252	3123	128
10%	281	3010	138
30 %	409	3183	171
50 %	403	3459	169
70 %	371	3354	388
90 %	819	3801	1397

Table 6.5: Variance values of acquired signals (no units used for ADC output numbers)

Another problem with mean and variance was the requirement for the entire signal acquisition before the start of calculations. It was therefore not possible to generate an alarm during a batch. An alternate approach of calculating a running sum for each signal was therefore investigated. The running sum of a signal was attained by adding every new sample value (number from the ADC) to the sum of all previous sample values in a batch. Analysis of the determined running sum profiles showed better fault detection features and was adopted in this research.

Any increase in pump power, for example, affected every sample and its cumulative effect became visible much earlier than with other techniques. The difference in the corresponding values under different fault conditions became distinguishable quickly because of the involvement of large numbers. A gradual increase in difference of running

sum values for normal and 40% blockage cases is shown in figure 6.9 where, for example, a difference of approximately 1000 (22133 for normal and 23134 for blockage case) was observed at 30 seconds into the batch cycle time. Figure 6.10 combines results from separate tests with various blockage extents and describes the power running sum values attained at 30 seconds batch cycle time. Values attained at 30 seconds batch cycle time in figure 6.9 are shown in figure 6.10 for reference. Figure 6.11 provides similar plots for flow and level running sums.

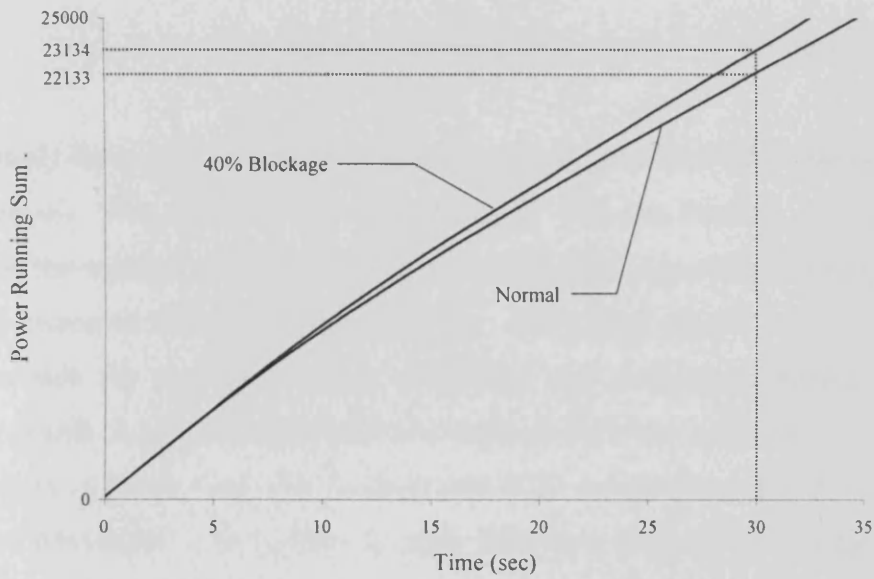


Figure 6.9: Power running sum values for normal and 40% blockage cases

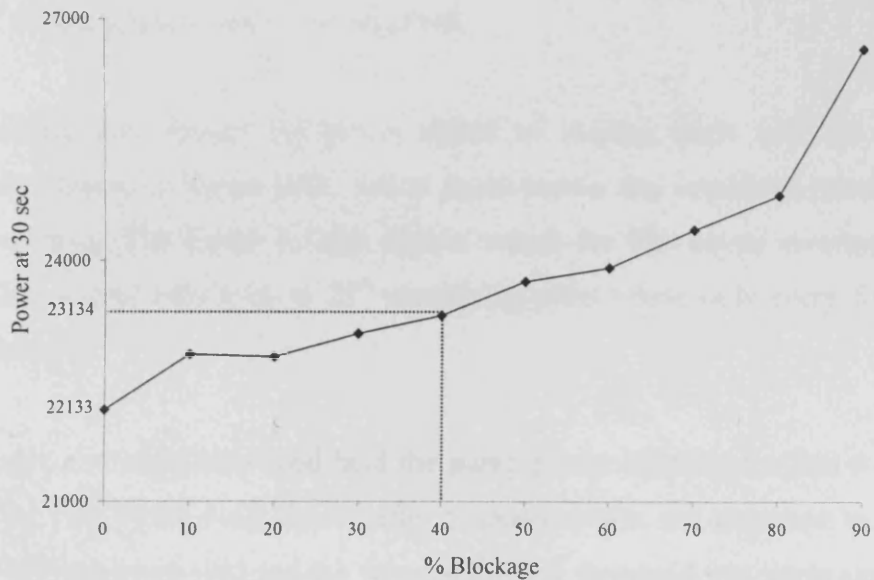


Figure 6.10: Power running sum values at 30 sec time

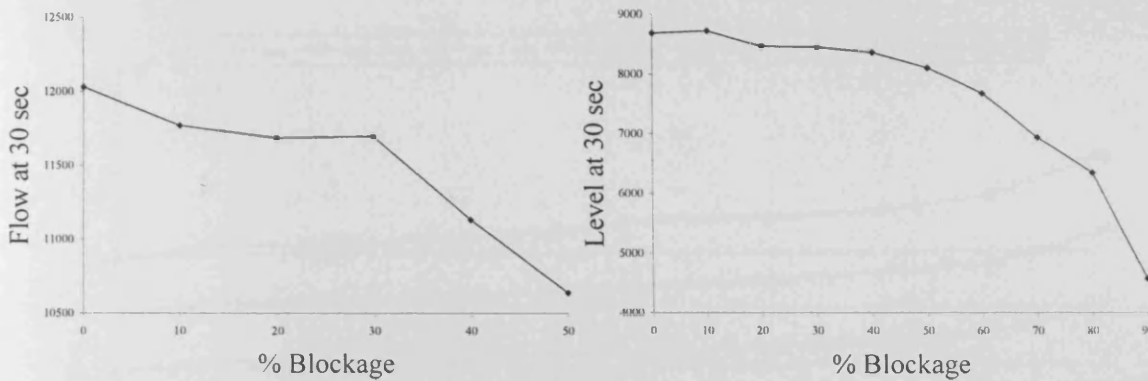
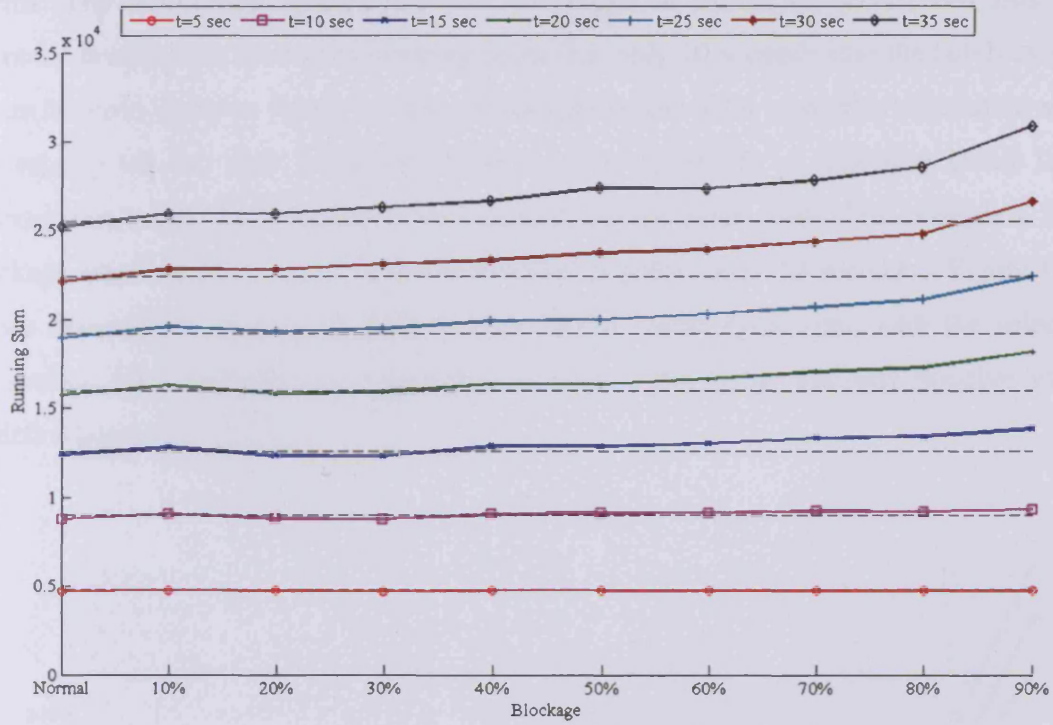


Figure 6.11: Flow and level running sum values at 30 sec time

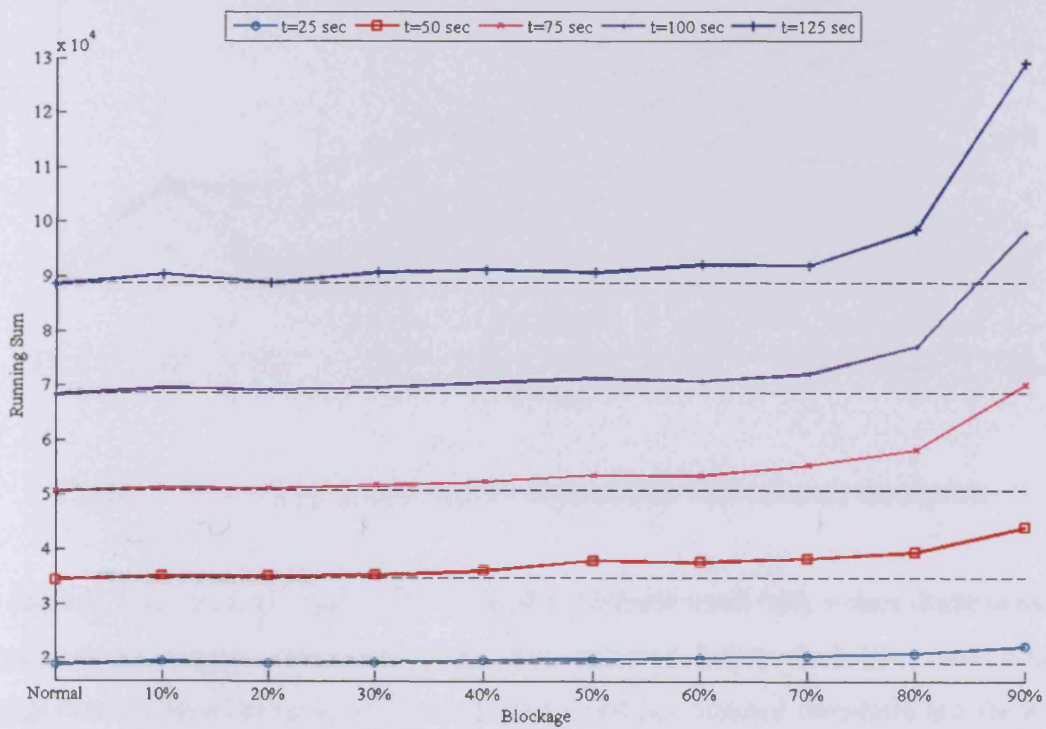
Results initially based on running sums at every sample generated false alarms because of the fluctuations. The method was thus modified and the running sum totals were compared to the selected thresholds every 5 seconds. This approach minimised the false alarms and increased the monitoring reliability. The author monitored running sums at every 5 seconds for various blockage conditions and established thresholds at these monitoring points. A separate threshold was required for every monitoring point because of non-linearity in the process. This led to a total of 27 monitoring points for each of the 3 signals. The thresholds were defined in each FEN as a look-up table. Also to prevent overflow errors 24 bits (3 bytes in memory) were assigned for running sum storage. Each threshold value was also stored in 24 bits format and for the specific batch tests. A total of 81 bytes of data eeprom memory was required.

Typical running sum values for power signal in various cases and the established threshold are shown in figure 6.12, which again shows the combined results obtained from various tests. The figure 6.12(a) depicts values for first seven monitoring points. Figure 6.12(b) shows values up to 25th monitoring point where only every fifth value is shown for brevity.

Since the controller algorithm used held the pump power constant for first 4 seconds of any batch the first monitoring check (after 5 seconds) was not expected to detect any blockage. This was confirmed and the value of the first threshold was set to signal an 'ok' result in all cases. For example, the threshold at 10 seconds was selected to be a value of 9000, this being approximately 200 above the nominal fault-free running sum value. This equated to a tolerance of roughly 2% for pump power. This was chosen to minimize false



(a) Typical running sums and thresholds for first 7 monitoring points



(b) Typical running sums and thresholds for up to 25th monitoring point

Figure 6.12: Threshold determination for power FEN

alarms. The monitoring system detected blockages in excess of 40% (with this 2% tolerance level) at the second monitoring point (i.e. only 10 seconds into the batch cycle), as can be seen again in figure 6.12(a). Blockages below 40% were also detectable with this regime but not with sufficient reliability. The reliability of detecting lower level blockages increased with threshold tests further into the batch cycle. For example a 30% blockage was reliably detected at sixth monitoring point (after 30 seconds). Figure 6.13 shows detectability of various fault extents against batch cycle time with the selected thresholds. All thresholds are normalised to zero in the figure and any positive value depicts blockage detection.

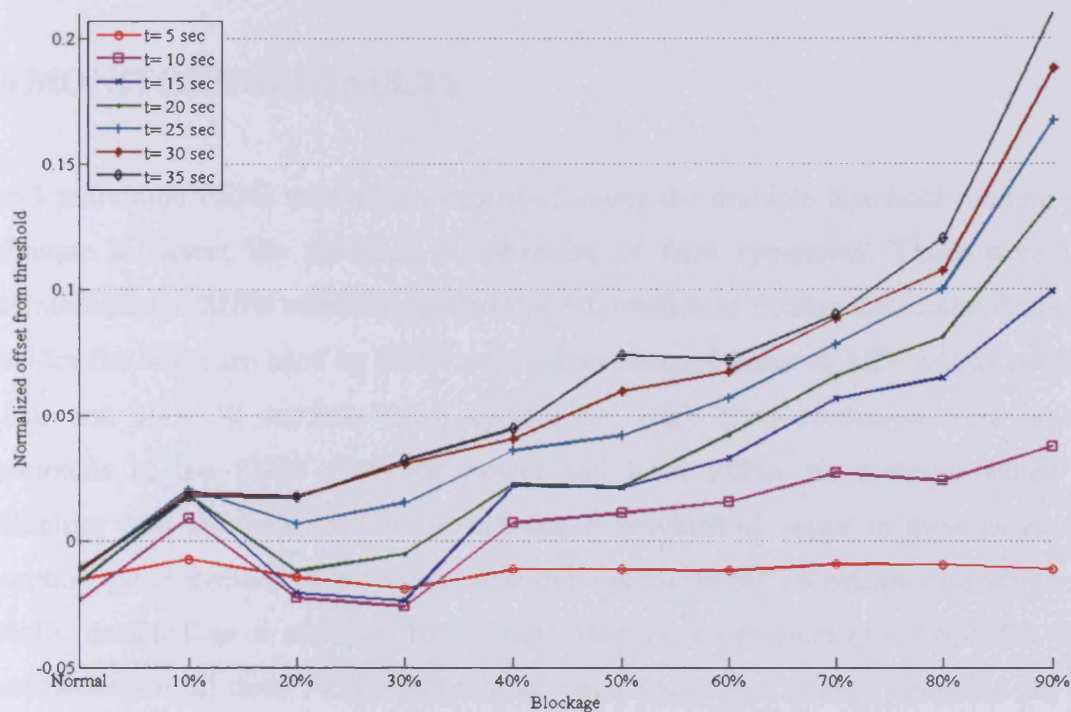


Figure 6.13: Blockage extent and its detectability with selected thresholds

The running sum for level signal FEN showed a different trend with values decreasing for progressive blockages. Thresholds were thus selected below fault-free case values. Typical running sums in various blockage levels and the selected threshold are shown in figure 6.14. The first threshold (5 seconds) was very conservatively selected and could only distinguish the extreme case of 90% blockage. Blockages in excess of 60% were detected after 10 seconds (2nd monitoring point). A 40% blockage became detectable by

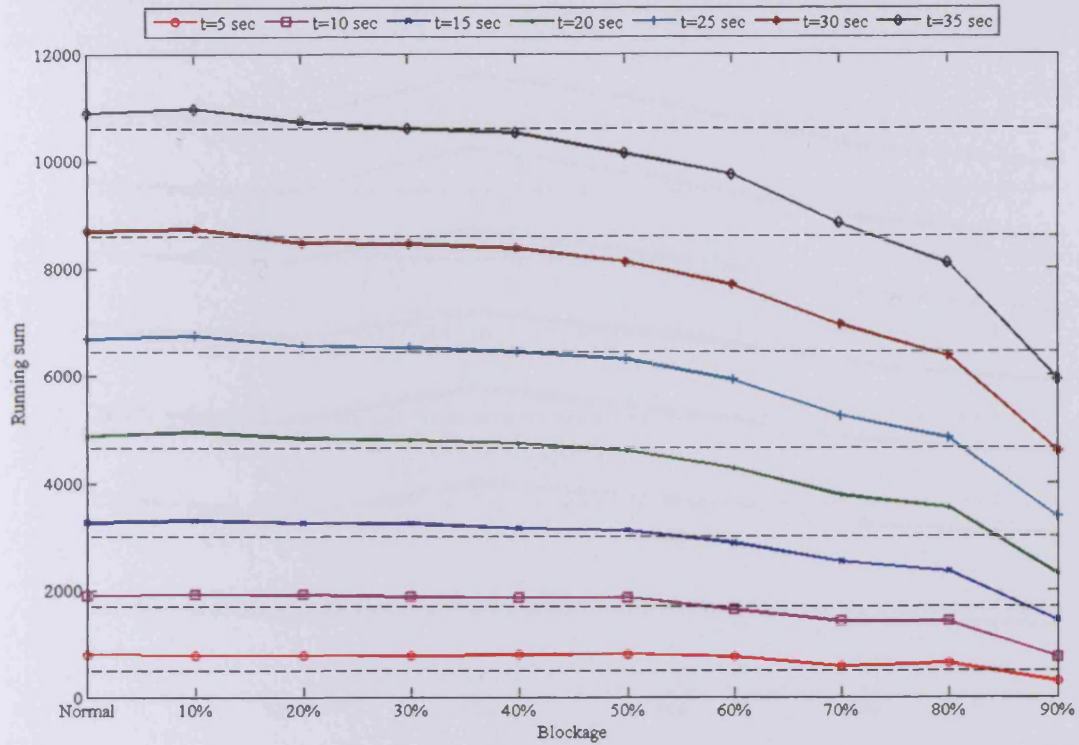
the threshold trigger at 25 seconds. A 20% blockage was generally not detectable before 40 seconds into the batch cycle.

A blockage typically affected water flow as soon as the batch started and the running sum for the flow signal FEN successfully detected 40, 50, and 90 percent blockages at the first monitoring point. There were difficulties observed with this FEN in that flow rates falsely showed high values when the blockage exceeded 50%. Reliable data was available for blockages only up to this level and thresholds were set accordingly. Typical running sums and selected thresholds are shown in figure 6.15 for level FEN for blockage levels up to 50%.

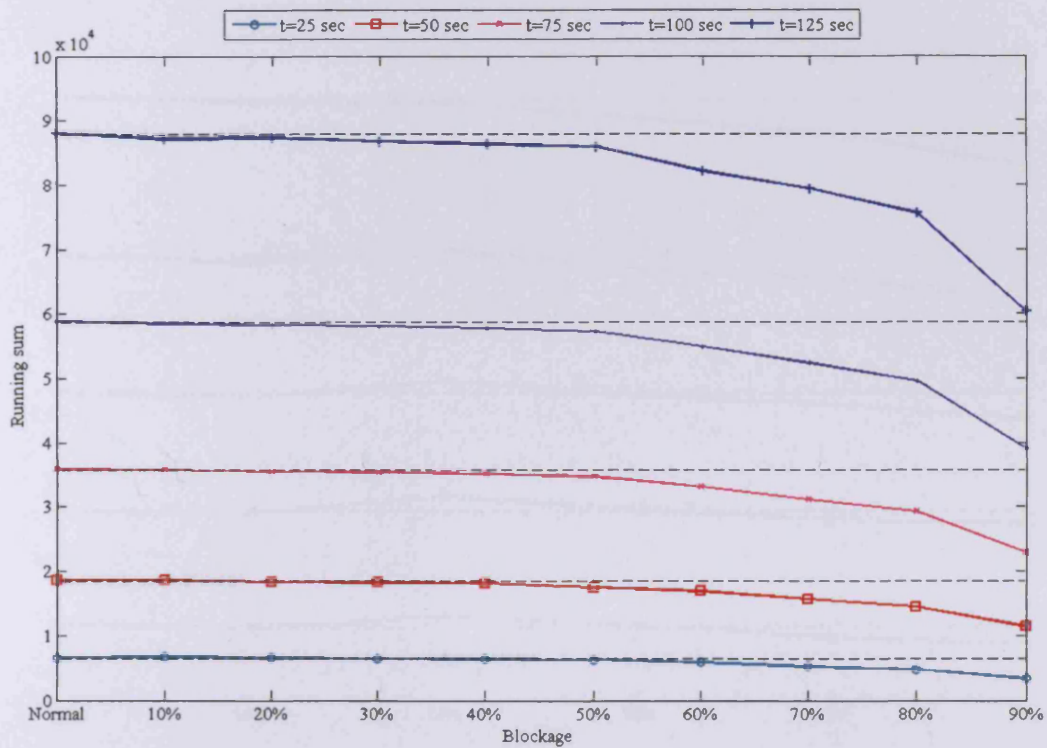
6.6 MONITORING RESULTS

The 3 individual FENs were thus established, using the multiple threshold running sum technique to detect the presence or otherwise of fault symptoms. These were then communicated to SUIN which integrated this information to finalise the results. Table 6.6 provides the rule base used by SUIN which discriminated between FEN results obtained before and after 30 seconds batch cycle time. High level blockages were quickly discernable at the SUIN from the power and level FENs. As previous stated the difficulties with the flow FEN were such that it provided ok signal in these cases. The exception to this decision rule (error | error |ok) was for the 90% blockage case which was initially detected as a medium level fault. Medium level blockages (40-50%) were identified when all three FENs indicated an error. Quick and reliable detection for low level blockages (up to 30%) proved to be more difficult because of the dynamic nature of the process. It took the signals at least 30 seconds to differentiate between low blockage cases and normal behaviour. Blockages detected after 30 seconds of batch start were therefore considered low level.

An interesting case was the situation where power and flow FENs indicated errors but level FEN decision was ok. The controller put extra effort to maintain level in such cases. It can be taken as the first indication of a low level blockage and also as the initial

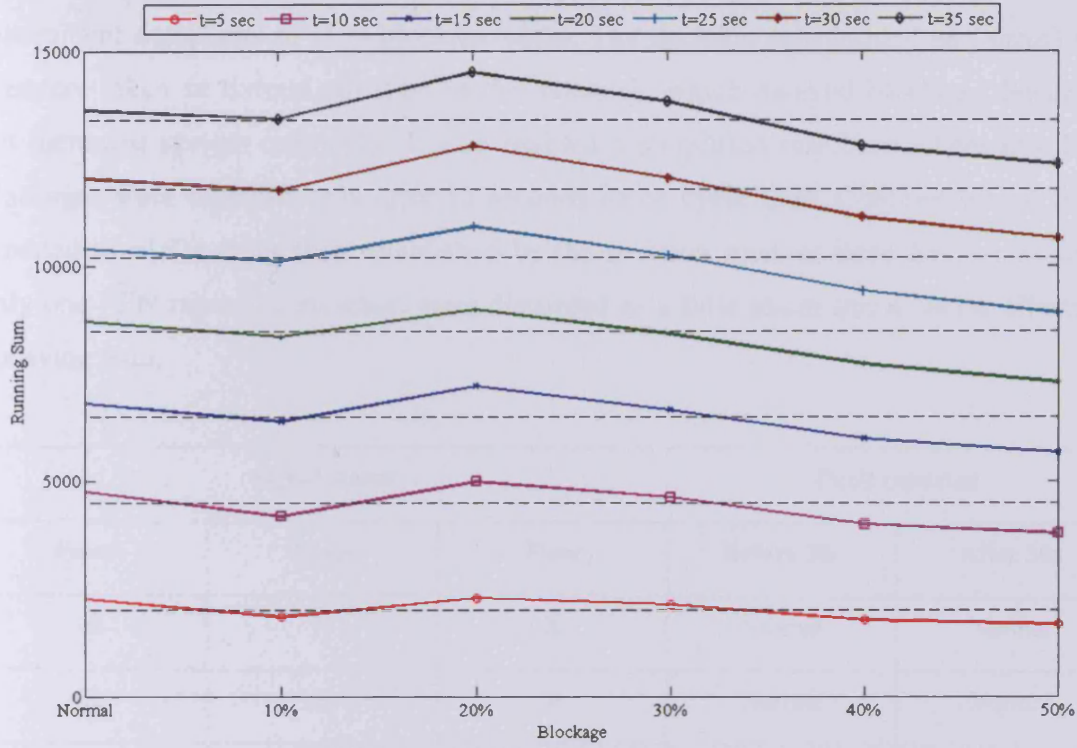


(a) Typical running sums and thresholds for first 7 monitoring points

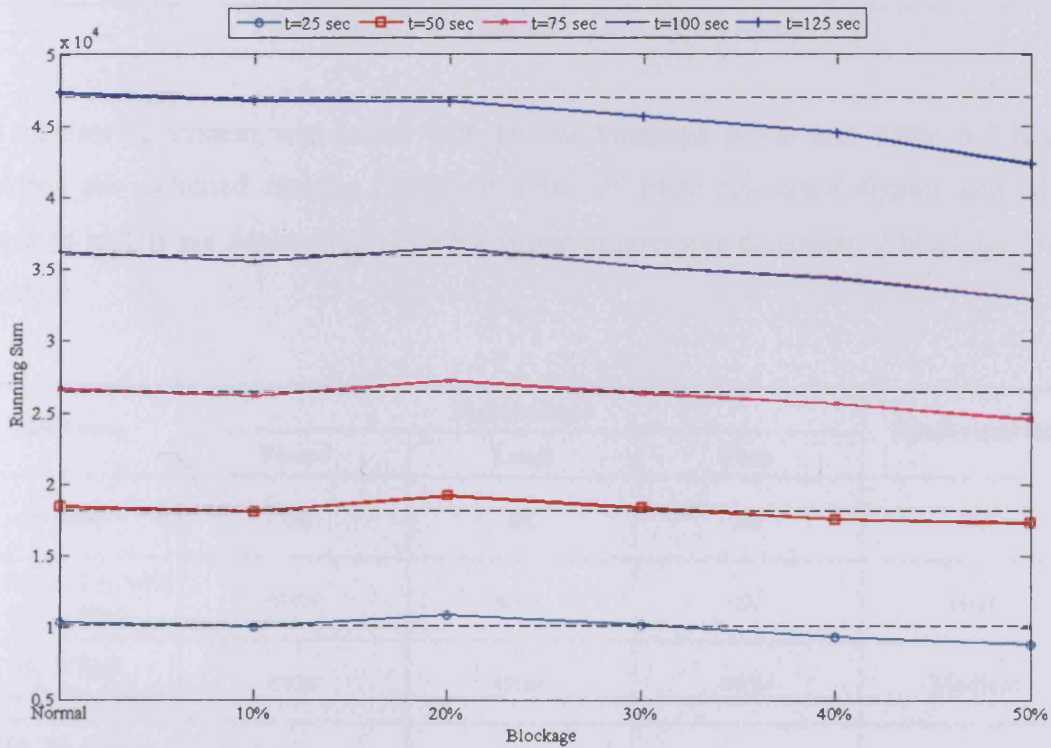


(b) Typical running sums and thresholds for up to 25th monitoring point

Figure 6.14: Threshold determination for level FEN



(a) Typical running sums and thresholds for first 7 monitoring points



(b) Typical running sums and thresholds for up to 25th monitoring point

Figure 6.15: Threshold determination for flow FEN

detection of medium level faults. This condition was however observed to be causing intermittent detections in 10% blockage cases. The decision rule (error | ok | error) was therefore taken as normal situation in this research, which delayed blockage detections but increased system reliability. It also enabled a simplified rule base where low level blockages were detected only after 30 seconds batch cycle time. Combination of faults reported to SUIN other than established by the decision rules of table 6.6, for example only one FEN reporting an error, were discarded as a false alarm due to noise effects or out-lying data.

Signal status			Fault reported	
Power	Level	Flow	Before 30s	After 30s
ok	X	X	Normal	Normal
X	ok	X	Normal	Normal
error	error	error	Medium	Low
error	error	ok	High	Low

Table 6.6: SUIN rule base (X = Don't care)

The monitoring system was tested with known blockage levels and Table 6.7 briefly describes the obtained results. Complete detail of FEN processed signals and SUIN integrated results are depicted in table 6.8 where progressive detection of blockage levels is evident.

Fault case	Signal status			Fault reported
	Power	Level	Flow	
Normal	ok	ok	ok	Nil
60, 70, 80% & 90% (after 10s)	error	error	ok	High
40, 50% & 90% (before 10s)	error	error	error	Medium
10, 20, 30% (after 30s)	error	error	ok/error	Low

Table 6.7: SUIN monitoring decisions

Monitoring Point & Time	Thresholds for Power Level Flow	Blockage Levels (percentage)										
		0	10	20	30	40	50	60	70	80	90	
1. 5 sec	4800	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok
	500	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Error
	2000	Ok	Error	Ok	Ok	Error	Ok	Ok	Ok	Ok	Ok	Error
2. 10 sec	9000	Ok	Error	Ok	Ok	Error	Error	Error	Error	Error	Error	Error
	1700	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Error
	4500	Ok	Error	Ok	Ok	Error	Error	Ok	Ok	Ok	Ok	Error
3. 15 sec	12600	Ok	Error	Ok	Ok	Error	Error	Error	Error	Error	Error	Error
	3000	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Error
	6500	Ok	Error	Ok	Ok	Error	Error	Ok	Ok	Ok	Ok	Error
4. 20 sec	16000	Ok	Error	Ok	Ok	Error	Error	Error	Error	Error	Error	Error
	4650	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Error
	8500	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
5. 25 sec	19200	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	6450	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Error
	10200	Ok	Error	Ok	Ok	Error	Error	Ok	Ok	Ok	Ok	Error
6. 30 sec	22400	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	8600	Ok	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error
	11800	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
7. 35 sec	25500	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	10600	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Ok	Error
	13400	Ok	Ok	Ok	Ok	Error	Error	Ok	Ok	Ok	Ok	Error
8. 40 sec	28600	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	13100	Ok	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error
	15000	Ok	Error	Ok	Ok	Error	Error	Ok	Ok	Ok	Ok	Error
9. 45 sec	31700	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	15700	Ok	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error
	16600	Ok	Error	Ok	Ok	Error	Error	Ok	Ok	Ok	Ok	Error
10. 50 sec	34800	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	18500	Ok	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error
	18150	Ok	Error	Ok	Ok	Error	Error	Ok	Ok	Ok	Ok	Error
11. 55 sec	37850	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	21500	Ok	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error
	19750	Ok	Error	Ok	Ok	Error	Error	Ok	Ok	Ok	Ok	Error
12. 60 sec	40900	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	24700	Ok	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error
	21300	Ok	Error	Ok	Ok	Error	Error	Ok	Ok	Ok	Ok	Error
13. 65 sec	44200	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	28150	Ok	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error
	23050	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
14. 70 sec	47600	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	32000	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	24600	Ok	Error	Ok	Ok	Error	Error	Ok	Ok	Ok	Ok	Error
15. 75 sec	50900	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	35700	Ok	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error
	26450	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
16. 80 sec	54350	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	39850	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	28450	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
17. 85 sec	57800	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	44200	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	30300	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
18. 90 sec	61350	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	48800	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	32200	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
19. 95 sec	64900	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	53600	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	34000	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
20. 100 sec	68600	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	58700	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	36000	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
21. 105 sec	72350	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	64050	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	38000	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
22. 110 sec	76200	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	69500	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	40100	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
23. 115 sec	80200	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	75500	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	42400	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
24. 120 sec	84350	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	81500	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	44740	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
25. 125 sec	88800	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	87800	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	47000	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
26. 130 sec	92800	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	94200	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	49500	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error
27. 135 sec	92950	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	100500	Ok	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
	50500	Ok	Error	Ok	Error	Error	Error	Ok	Ok	Ok	Ok	Error





SUIN colour codes: Normal  Low  Medium  High 

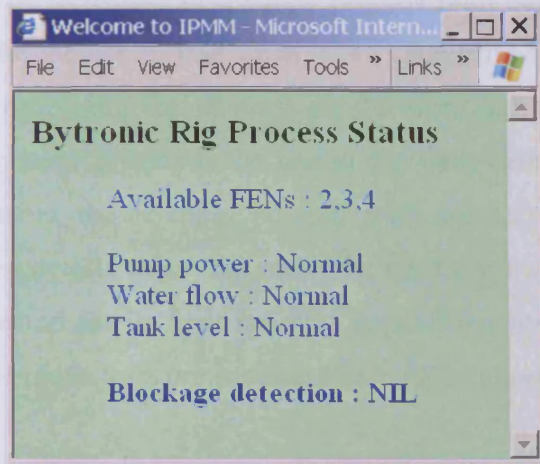
Table 6.8: Blockage level detection with FEN processing and SUIN integration

In this fashion the SUIN integrated the 3 FEN information and provided an overall monitoring result. At this stage of application testing the plug-and-play concepts were not introduced. The deployed monitoring system for the Bytronic process rig thus assumed all 3 nodes to always be available. Normal, low-level fault, medium-level fault, or high-level fault statuses were reported to remote users using developed dynamic web pages. These were established such that any user could access SUIN home page by entering its IP address or web page address onto a standard web browser, such as Internet Explorer. The monitoring results web page was set to refresh every 10 seconds. Also it was designed to be a small web page containing only text. This was used to reduce the Internet traffic load on the SUIN and to be commensurate with the limited memory of the SUIN. Another benefit of the small web page was that it was accessible from mobile phones and PDAs. Examples of the tested web pages and a brief insight into the layout design follows.

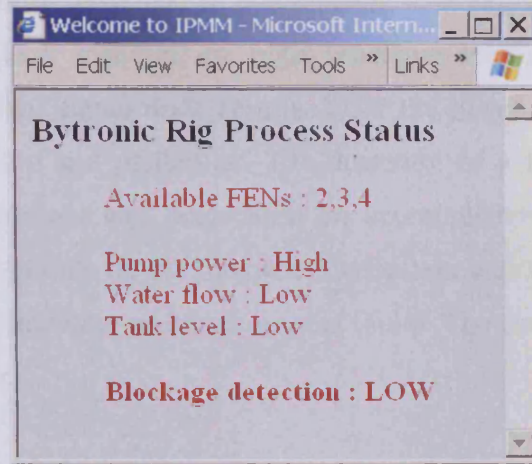
A user will typically not be continuously viewing the web page to locate a process fault. The web page was thus designed to grab an indifferent user's attention with colours used for this purpose. A light green background colour was used to indicate normal conditions. The SUIN changed the web page background colour to light grey on detecting a low level blockage. The foreground colours were also changed to maintain ease of reading. Light blue and red background colours were used to show medium and high level blockages respectively. Figure 6.16 shows the web page appearance for each of the four possible statuses.

6.7 SUMMARY

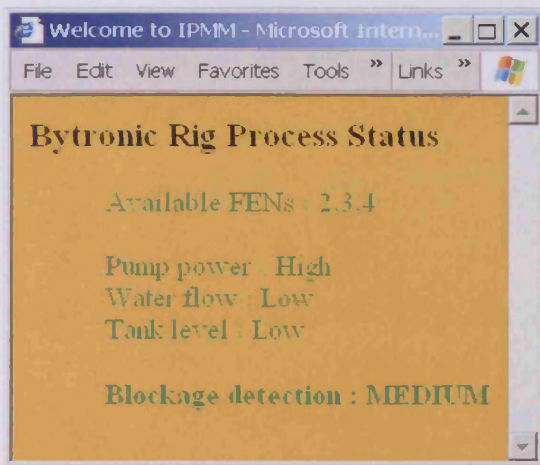
The proposed monitoring system was deployed on the process test rig as a first application and to test and develop and prove the concept. Various degrees of pipe blockage fault were simulated by closing a manual valve. It was noted that blockages caused disturbances in the water flow, thereby affecting flow and level sensor readings. The process controller was programmed to adjust pump power in order to attempt to restore desired water level in the tank and thus the simulated faults were also detectable via the pump power signal. Three FENs monitored power, level, and flow signals and located the fault by matching the processed signal with carefully determined multiple (in



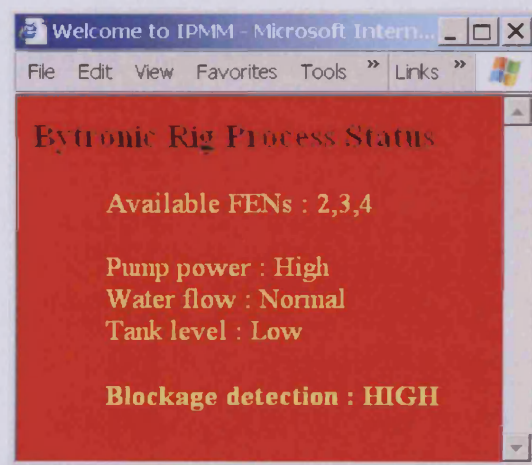
(a) Normal



(b) Low level blockage



(c) Medium level blockage



(d) High level blockage

Figure 6.16: Monitoring results on web page

time) thresholds. The first layer decisions by the FENs were integrated by the SUIN operating at second hierarchy layer. The combined information proved useful in classifying the fault extent as being low, medium, or high level. The first layer FEN nodes were proved as suitable data acquisition devices and of being capable of providing simple first layer processing. In isolation each FEN was able to discriminate between normal and faulty operation. The second layer computations, based on first layer results, provided integrated and robust information about process health. The results were provided on Internet as a dynamic web page and were open for viewing to all interested users. The web page was kept up-to-date via its auto-refresh feature. The web page was also accessible to mobile Internet users.

The detection of a single process fault has been described in this chapter. The extent of the detected fault was also categorised (low, medium, or high blockage level) by combining the ok/error results from individual signal node results. Thus the developed system proved to be useful for fault detection and prediction. The detection of a fault using the resource limited 8-bit microcontrollers was considered an accomplishment, especially when the real-time results were available on the Internet. The system was then tested for its fault isolation capabilities by simulating multiple process faults. The details of these tests are provided in the next chapter.

MULTIPLE FAULTS ISOLATION

The monitoring system was next deployed on the Bytronic rig in order to evaluate its performance in isolating different faults. A batch process was monitored for two types of faults, namely leakage and blockage. The blockage simulation was as described in the previous chapter. Leakage from the tank was added as the second process fault. Data acquired and processed by the hierarchical monitoring system was used to differentiate between the two faults. Another application was developed to evaluate the monitoring system's performance for multi-loop processes. In this mode, the process rig ran a continuous process and a batch process simultaneously. This chapter details these two applications and presents the achieved results.

7.1 FAULT ISOLATION

The monitoring of a batch process with pipe blockage faults, emulated on the Bytronic process rig, was explained in the previous chapter and was used as the first single fault detection application. A leakage in the tank was then introduced in the same process as a second possible fault. Figure 7.1 shows the schematic process diagram, the leakage and blockage arrangements along with the control and monitoring signals. A manual valve, labelled 'leakage simulation' in figure 7.1 was used to produce leakage from the tank. The valve was not easily set and large volume changes resulted from slight alterations. The leakage rate quickly became larger than the incoming water flow rate, this resulting in uncontrollable situations. It was felt that a leakage fault in a real industrial process would realistically only provide a small trickle from leakage point and not a full liquid flow. The manual valve was therefore opened only slightly causing a small trickle which was barely detectable. It was not possible to repeat the exact leakage by repositioning of manual valve setting. It was also not possible to reliably ascertain the fault level in any exact quantitative way. The extent of leakage was not investigated and only fault presence detection was tested.

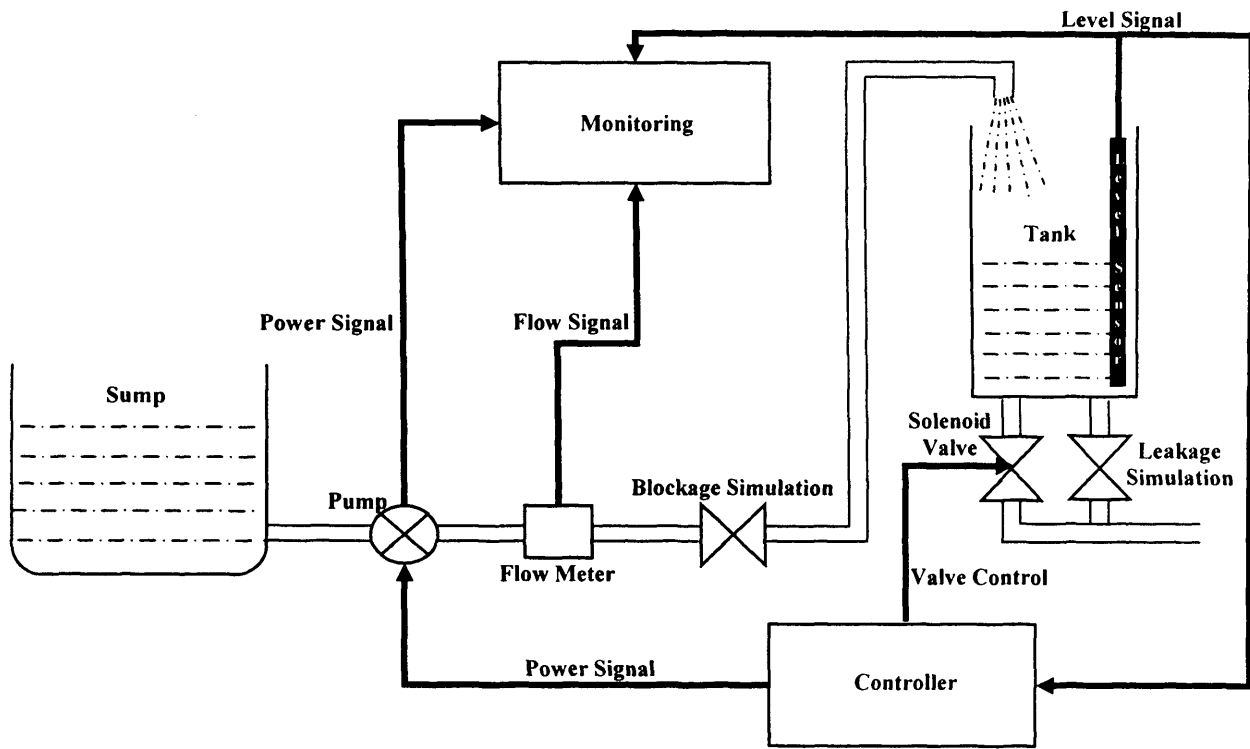


Figure 7.1: Leakage and blockage faults arrangement in batch process

The monitoring system was deployed as before and the 3 signals showed the same characteristics for normal conditions. During the system study stage the nodes were set to acquire full raw data and only the leakage fault was introduced to the system.

7.1.1 Leakage Fault Simulation

A small leakage was introduced in the water tank and the monitoring signals were acquired. The water level in the tank had an expected slower rise than normal because of the leakage during the first four seconds of the batch cycle (with the controller set to deliver constant pump power). Then the controller sensed the actual level signal and started generating appropriate power control. The lower than normal feedback signal resulted in a higher than normal power signal. The detection of this increased controller action again provided the opportunity for fault detection in an otherwise controlled process. The increased pump power actually achieved and maintained the desired rate of tank filling by, in effect, compensating for flow losses due to the leakage. Figure 7.2 shows the combined plot for the three monitored signals.

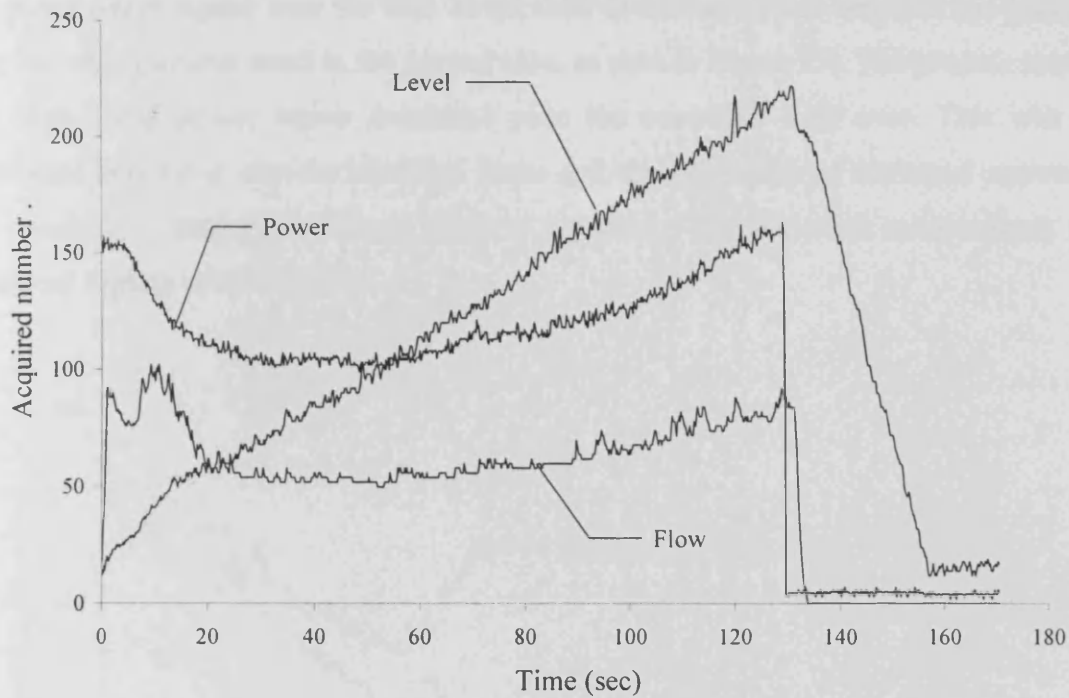
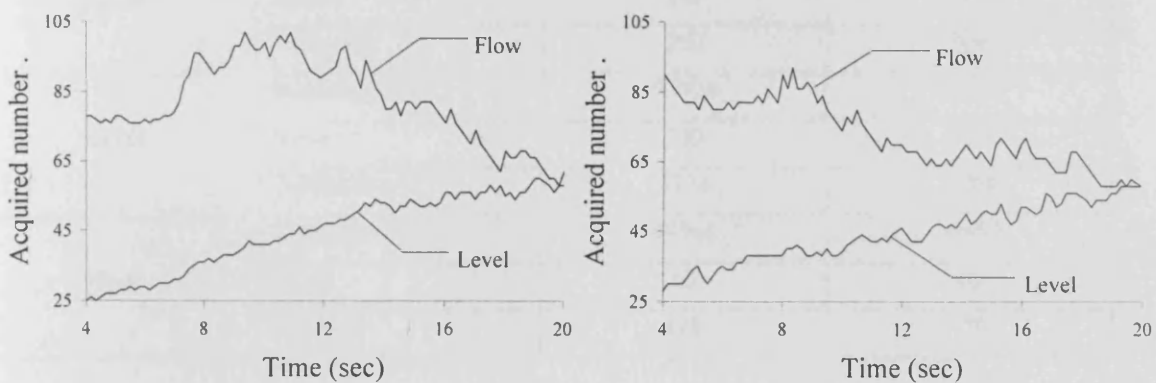


Figure 7.2: Signals for leakage fault only

The power signal pattern with the leakage fault present is overall quite similar to the normal case but did have some important differences. A second peak in flow was clearly visible after the starting transient. The effect of the leakage on the flow and level signals can be more clearly seen in figure 7.3. Figure 7.3a shows the level and flow signals at the start of batch for the leakage fault case. Figure 7.3b, for comparison, shows the normal signals. The controller (in operational mode) sensed the decreased level and increased the pump power, and hence the flow in response. A delay was observed between the start of controller action and the flow response. The increase in flow also caused a sharp increase in the level signal.



(a) Leakage fault

(b) Normal case

Figure 7.3: Enlarged flow and level signals

The pump power signal over the first 20 seconds of the batch was larger in the leakage case, but with a similar trend to the normal case, as seen in Figure 7.4. The process started with high pump power which decreased once the controller took over. This was an established behaviour also for blockage faults and the evaluation of statistical measures was required to distinguish effects. Table 7.1 provides the statistical measurement for monitored signals in both cases.

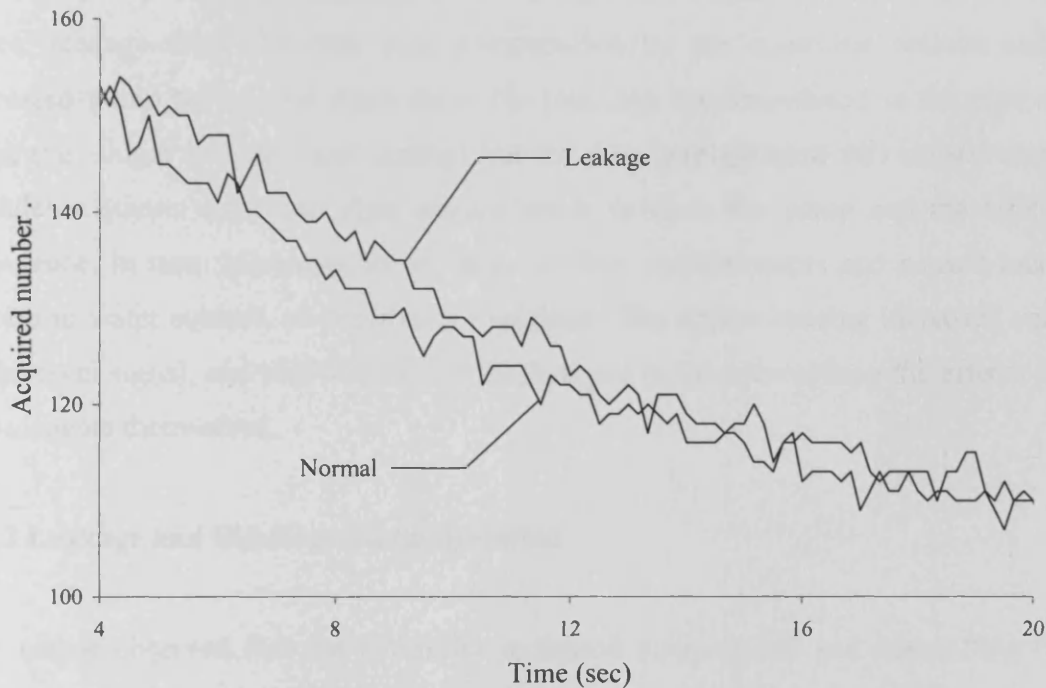


Figure 7.4: Power signal in normal and leakage conditions

Signal	Statistic	Normal	Leakage
Power	Running sum	92535	94563
	Mean	119	121
	Variance	252	304
Level	Running sum	93924	93498
	Mean	120	120
	Variance	3124	3194
Flow	Running sum	49368	50682
	Mean	63	65
	Variance	128	159

Table 7.1: Statistical measurement for monitored signals

The increased controller effort under leakage conditions was visible from the statistical measures for the power signal. Both its mean and running sum showed increased power usage, as expected. The knock-on effect of increased flow was verified by increased mean and running sum values for flow signal. The level signal measures were almost the same as normal. This was due to the controller effectively achieved its function of maintaining a correct water level in the tank.

There was also increased variance in all the signals. This can be explained as follows. Water leakage from the tank was compensated by the controller actions and thus increased pump power and water flow. No blockage was introduced in the pipe at that stage (i.e. single leakage fault testing) but the pipe configuration still caused increased turbulence (there were four right angled bends between the pump and the tank). The turbulence, in turn, increased the variance in flow measurements and caused increased ripples in water surface, as previously explained. The ripples causing increased variance in the level signal, and this was the feedback signal in the control loop the effects tended to propagate themselves.

7.1.2 Leakage and Blockage Faults Isolation

The author observed that the controller increased pump power and hence flow rate in response to both leakage and blockage faults. Blockage faults produced resistance to flow and hence the decreased flow signal values were of prime consideration in blockage detection. Conversely the lack of a reduction in the flow signals could isolate the leakage fault.

The small leakage introduced caused a small loss in water volume and the controller quickly recovered the situation and only a small increase of flow was evident. Indeed, it was very difficult to distinguish this flow increase from the flow in the normal case. A different approach for leakage fault detection was therefore developed. The initial 4 seconds when the controller was set at a fixed level was utilized. During this time (with no blockage fault present) the usual volume of water reached the tank. A portion of this volume was lost due to leakage and the level signal showed lower values. Individual samples did not provide any reliable fault determination and resort was again made to running sums. The running sum of the level signal was matched with the threshold at 5

seconds batch time (as for blockage detection). The controller started working in feedback mode after 4 seconds but inertia and delays in the system meant that the level signal was unaffected at this first monitoring point at 5 seconds batch time. The system showed almost the same water level for normal and low to medium blockage cases. A lower running sum of the level signal thus indicated the presence of either the leakage fault or high level blockage fault. A threshold value, based on the acquired experimental data, was still established in order to initiate a fault indication.

The level monitoring FEN, on detecting the fault, generated an ‘event’ message and sent it to the flow monitoring FEN. The second layer of computational hierarchy was thus involved in isolating a leakage fault from a high blockage one. The flow monitoring FEN compared its own running sum with a threshold on receiving the event message. It combined its own information with the received information and sent it to SUIN as an error message containing the correct fault situation. The SUIN was thus able to convey the proper fault situation to remote users. Table 7.2 provides running sum values for level and flow signals at first monitoring point for fault isolation. The author established 735 and 2500 as threshold values for level and flow FENs respectively for leakage detection. Figure 7.5 elaborates fault isolation procedure carried at the first monitoring point (5 seconds batch cycle time).

Condition	Level	Flow
Normal	796	2266
Leakage	672	2294
10% blockage	775	2036
20% blockage	778	2278
30% blockage	766	2130
40% blockage	785	1774
50% blockage	796	1666
60% blockage	744	3168
70% blockage	572	3094
80% blockage	628	2892
90% blockage	291	1614

Table 7.2: Running sums for fault isolation

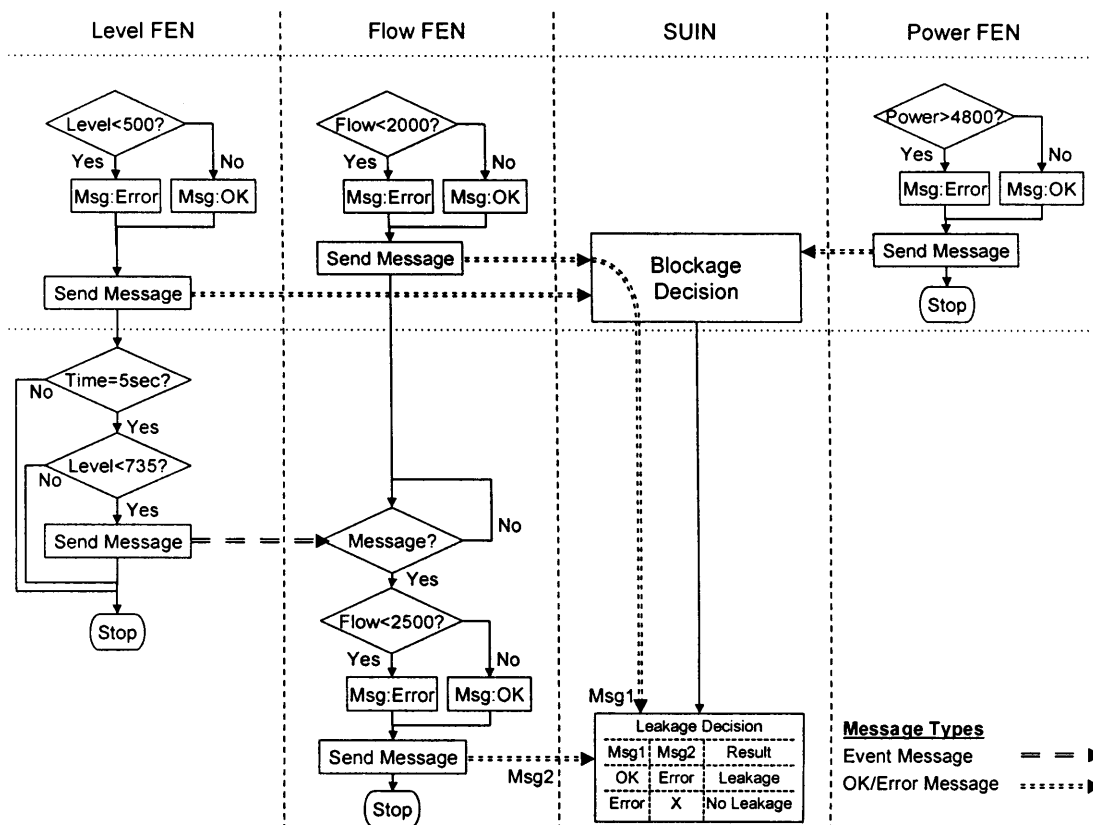


Figure 7.5: Fault isolation processing

The leakage FEN generated an event message to the flow FEN on detecting the running sum to be below threshold (735). The flow FEN, in response, generated an ‘integrated’ error message to the SUIN after checking its own threshold for leakage (2500). The SUIN then combined the information from the two flow FEN messages to decide about the leakage fault. This procedure provided blockage fault indication as well, which was not used and the previously reported method was adopted. It was interesting to note that only one threshold value was used in level and flow FENs each to detect leakage as opposed to 27 values for blockage detection. The power monitoring FEN played no role in leakage detection and isolation.

Running sum values at other times were also considered for leakage fault detection and isolation. Level and flow running sum values calculated at 6 and 7 seconds provided wider range for thresholds but with delayed results. Further delay in decision making made the situation worse as the controller effects complicated the proceedings. 5 seconds

batch cycle time was thus finalised providing early results and ease of programming, as it matched timings with blockage detection procedure.

Only a small change in FEN software was required for the additional functionality using the same hardware. Modularity of the developed code ensured that only a small subroutine for processing the acquired samples was amended minimizing the programmer effort. Changes required in Java code for the SUIN were also minimal and only the results integration part and web page text strings were amended. Easy code upgrading was previously identified as one of the essential requirements for a generalised monitoring system and the developed system exhibited this capability in this example application.

7.2 MULTI-LOOP PROCESS MONITORING

The Bytronic process rig was again used to check the monitoring system performance in a multi-loop environment. Two control loops were implemented on the process rig such that it operated in continuous and batch process modes simultaneously. The controller in the continuous loop was set to maintain a constant flow in the pipe between the pump and the tank. The batch process was set to allow storage of the incoming water and then to empty the tank once a pre-defined level was reached. The process repeated itself indefinitely. The two processes were interlinked with the controlled water batch flow from the continuous process being the input to batch process. Thus, a fault in the continuous process also disturbed the batch process. The distributed monitoring system was deployed to monitor both processes. The deployed monitoring system again consisted of three FENs and a SUIN, and the logic rules developed considered the loop interactions.

7.2.1 Continuous Process Monitoring

As stated, the continuous process objective was to maintain a pre-defined flow in the pipe. The pump power was the controlled variable and the flow signal was the feedback. A blockage type fault was simulated on this loop by closing a manual valve between the pump and the flow sensor. A relatively large deadband was set within the controller algorithm to cater for flow signal fluctuations caused by water turbulence. Two FENs

acquired power and flow signals for monitoring purposes. Figure 7.6 summarizes the control and monitoring arrangements for this loop.

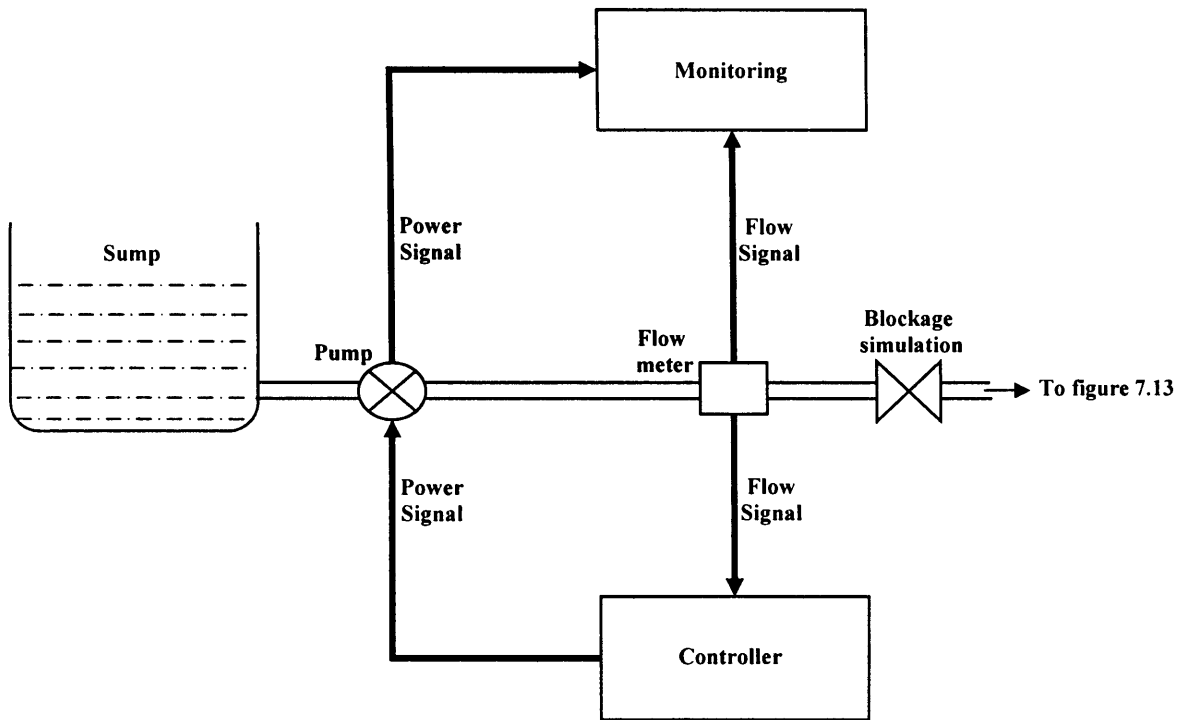


Figure 7.6: Continuous process

The nominal values were observed for the monitored signals when the process was in steady state. The pipe blockage faults led to increased pump power and this was used to provide fault indication. Mean power values were computed and it was found that the mean pump power remained virtually constant at a given constant flow. The threshold value was set on this basis and the method was found to be more suitable than the running sum or other statistical methods. The threshold values of 150 and 66 for power and flow signals respectively were used. These were values of 10 above and below the typically steady state power and flow values respectively.

The final deployed decision architecture set the two FENs monitoring this control loop to collaborate with each other to find the fault extent. Figure 7.7 describes the detection procedure. When the power monitoring FEN detected a fault it sent an event message to the flow monitoring FEN. On receiving this message, the flow monitoring FEN checked its own status and combined the two signals' information. The effect of a process delay

was allowed for since water flow did not change instantly with a change in pump power. A delay of 2.5 seconds (15 samples) was used in the flow FEN before it checked status and combined information with the received power information. If the fault was confirmed the flow FEN then sent the SUIN an ‘integrated’ error message. The SUIN performed the role of communicating the fault extent to remote users. It parsed the received error message for combined information of power and flow and updated its web page for mild or severe pipe blockage fault accordingly, as shown in figure 7.8 (when batch process monitoring reported normal conditions).

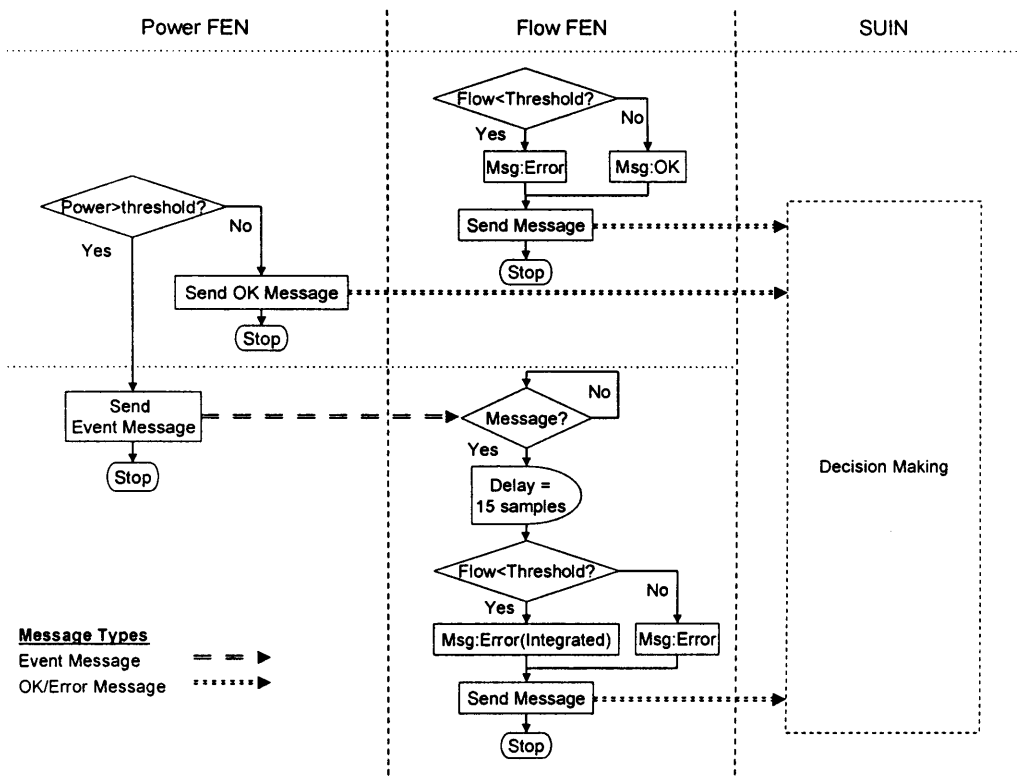


Figure 7.7: Fault detection procedure in continuous loop

Monitoring was started when the process was in steady state condition and figure 7.9a shows the acquired signals for normal operation. Mild blockage faults were reacted to by the controller and increased pump power with normal flow was observed, as shown in figure 7.9b. This combination was taken as a mild fault indication. Severe pipe blockages also caused drop in flow being beyond the extent to which the controller to compensate. Figure 7.9c shows the signals for the severe blockage condition. The decreased flow, (and maximum power) was taken as indication of severe blockage in the pipe.

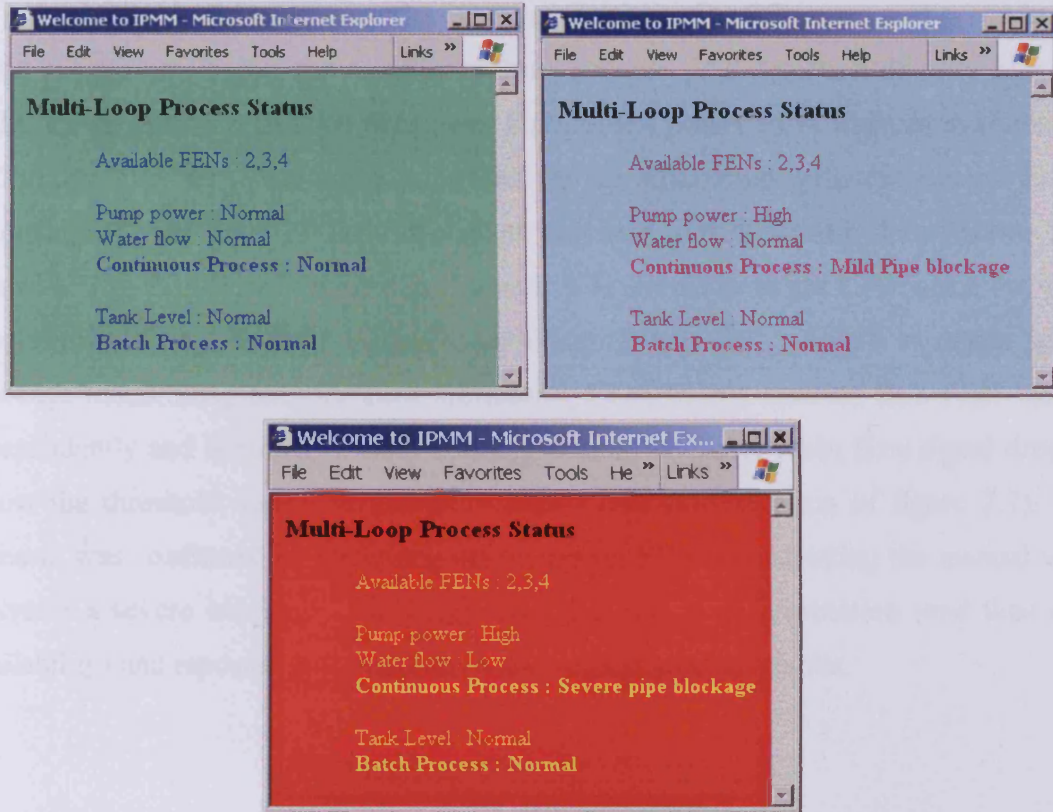


Figure 7.8: SUIN reporting process condition

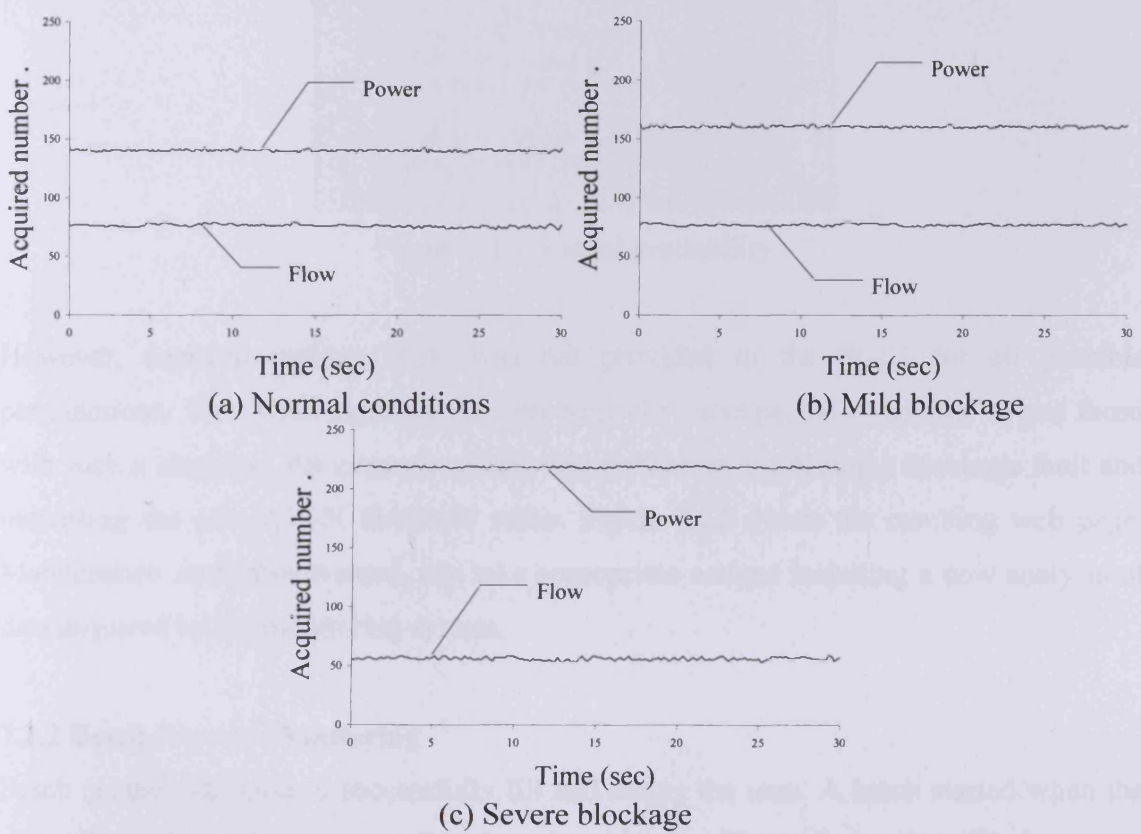


Figure 7.9: Continuous process monitoring signals

As mild blockages did not reduce flow the power monitoring FEN was used to initiate the event message to trigger the decision making process. The detection of mild blockage faults was therefore reliant on the power FEN. If the power FEN was not available (or faulty) then even with plug-and-play capabilities the monitoring system would not be able to distinguish the level of fault. However, the user was informed about power FEN unavailability (and hence reduced functionality) as shown in figure 7.10 (when the batch process monitoring reported normal conditions). It was still possible to detect severe blockage faults using only the flow monitoring FEN. In this case the flow FEN worked independently and initiated an error message directly to SUIN when flow signal dropped below the threshold value, as shown in figure 7.11 (modification of figure 7.7). This scenario was confirmed by switching off the power FEN and adjusting the manual valve to create a severe blockage. The SUIN had plug and play information (and thus node availability) and reported this situation as severe pipe blockage faults.

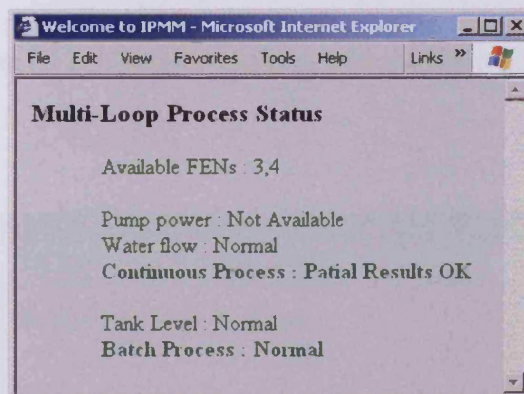


Figure 7.10: Partial availability

However, decision making logic was not provided to the SUIN for all possible permutations. The SUIN updated its web page for 'unexpected condition' when faced with such a situation. An example of this was created by producing a blockage fault and increasing the power FEN threshold value. Figure 7.12 shows the resulting web page. Maintenance staff, thus warned, can take appropriate actions including a new analysis of data acquired by the monitoring system.

7.2.2 Batch Process Monitoring

Batch control was used to successfully fill and empty the tank. A batch started when the controller issued a close command to the solenoid valve. The tank was then filled to a

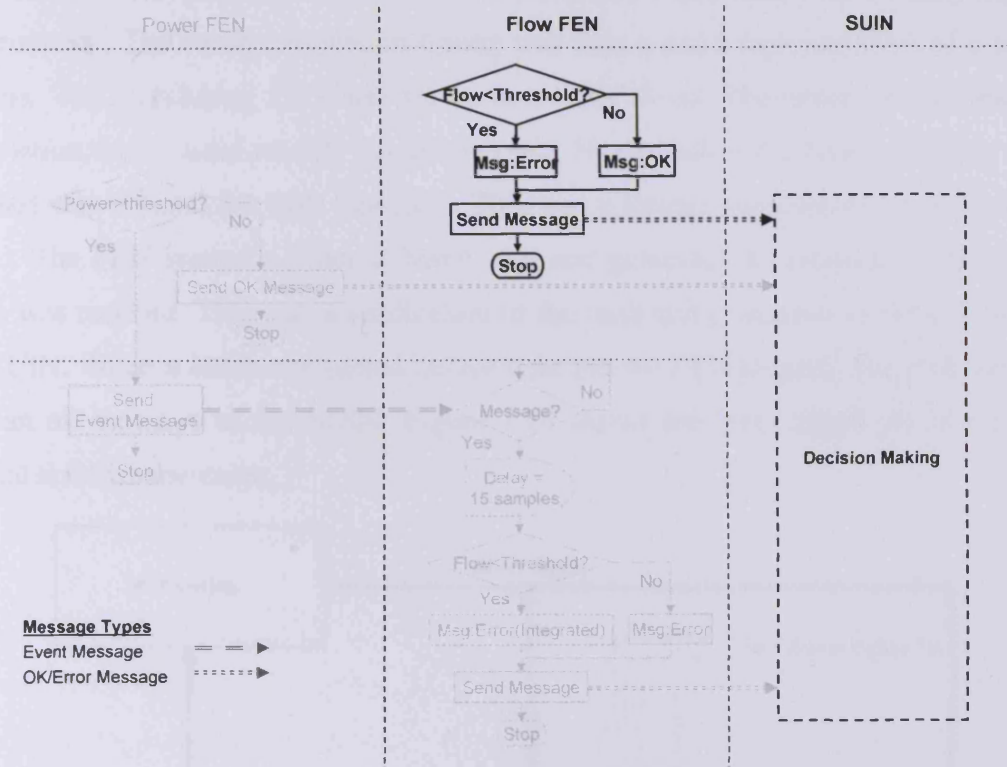


Figure 7.11: Decision making with partial availability

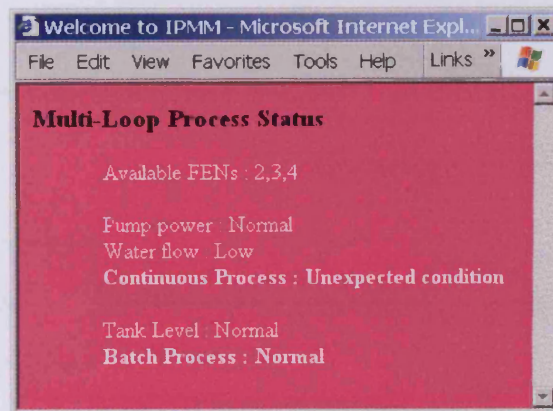


Figure 7.12: Detection of unexpected process conditions

pre-defined level. When sensed the solenoid valve was opened and the water was released into the outlet pipe. Figure 7.13 shows the control and monitoring arrangements. The FEN used for monitoring acquired the level signal and checked control commands to the solenoid valve. A leakage fault in the tank was simulated by opening a manual valve at the bottom of the tank. Digital control of the solenoid valve was used in conjunction with the level signal as the feedback.

The duration of all batches was found to be consistent when there was a steady incoming water stream. The batch completion timing was thus a good representation of a nominal process. The monitoring FEN was set-up in a timer mode. The observed nominal batch completion time (under normal conditions) was 38 seconds and a time-out value of 38.5 seconds was defined for fault detection. This was a simple implementation with the PIC MCU. The FEN started a timer at batch start and generated an interrupt if the time-out value was reached. This was an indication of the fault and generated an error message to the SUIN. When a batch completed before time-out the FEN stopped and reset timer and sent an ok message to the SUIN. Figure 7.14 shows the level signal profiles for both normal and leakage cases.

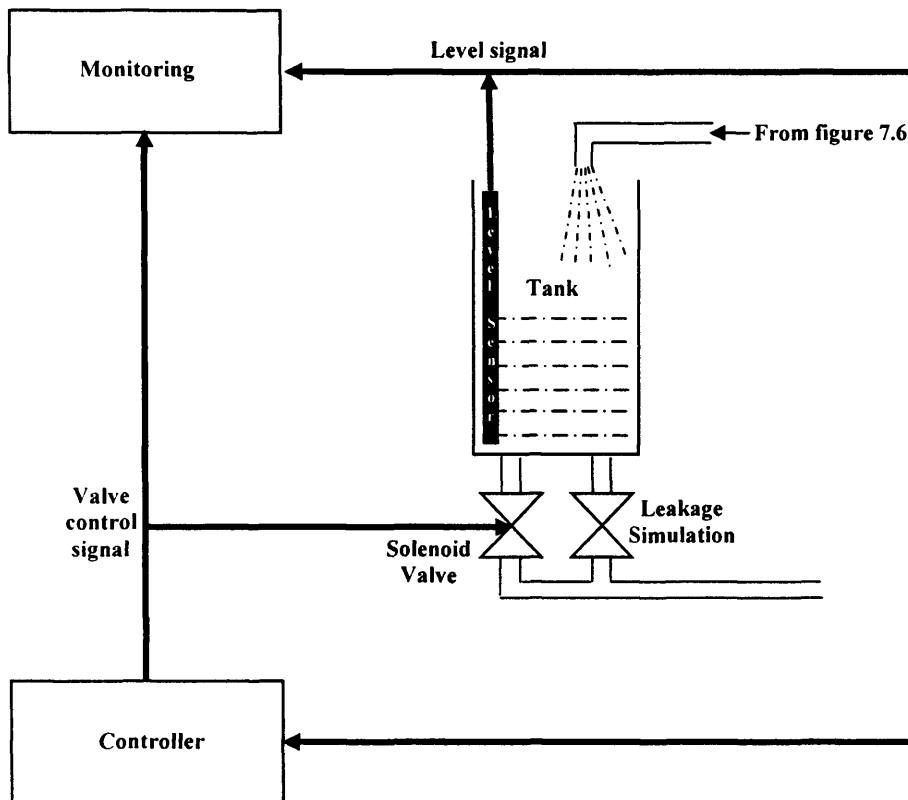


Figure 7.13: Batch process

In practice, the resolution of 0.5 seconds between conditions did not provide good results. Processes, such as this one, may show different timings in different batches. The monitoring system generated several false alarms with this threshold value. A higher threshold value was tried but the system was then unable to consistently detect the fault. Actual processes in industry use much larger tanks and the filling and emptying will take more time and potentially provide better opportunity to establish robust thresholds.

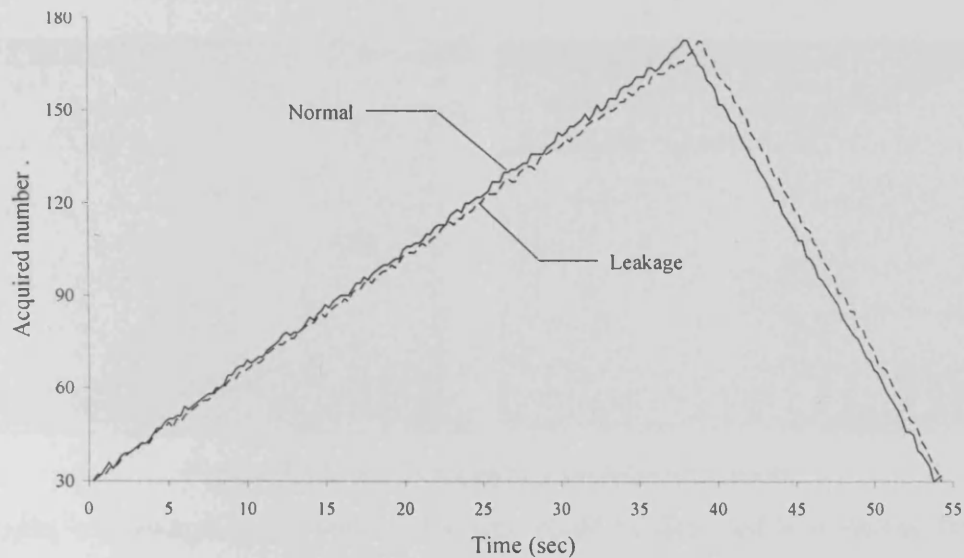


Figure 7.14: Level signal shows delay in batch completion for leakage

Another problem was that detection was only possible after batch cycle completion. A second approach was investigated with the level signal now used directly. The raw level signal variations did not show any useful difference between normal and leakage conditions, as is apparent from figure 7.14. The tank filled slower in the leakage case but running sum values did not display any useable patterns when thresholds were based on attaining certain levels. The difference in the number of samples over which running sum was calculated caused this behaviour and thus the running sums were re-calculated, now based on the batch timings. Computations were done on same number of samples this time and level running sum provided better results. Table 7.3 shows these results calculated at various normalised times within the batch cycle. The selected running sum was at 50% of the expected batch finish time and a threshold value 4960 was used to distinguish between normal and leakage conditions. This was configured in the decision making process as an early warning and generated an error message to the SUIN. The initial fault detection was subsequently confirmed by the previous timer based methods and results were posted on the web page as shown in figure 7.15 (when continuous process monitoring indicated normal conditions).

Batch time	Normal	Leakage	Difference
25% (9.5 sec)	1835	1795	40
50% (19 sec)	5009	4907	102
75% (28.5 sec)	9517	9323	194
100% (38 sec)	15363	15040	323

Table 7.3: Running sum values for level signal



Figure 7.15: SUIN reporting process condition

In principle, a blockage in the tank outlet pipe could be detected in a similar fashion. It was observed that a batch takes approximately 25 seconds to empty the tank. The continuation of a timer method and a 50% emptying time running sum would be feasible, but was not tested in the current work.

7.2.3 Combined Loop Monitoring

The two control loops on the process rig were interconnected with the water level in the tank being a function of the continuous process output. Water flow to the tank remained constant irrespective of the batch state and water continued to flow into the tank during the batch tank emptying phase. A fault in continuous process would therefore affect the batch process. A flow reducing problem (such as blockage) in the pipe also reduced the tank filling rate. The batch process monitoring FEN would then report it as a fault with the batch process, which was not the real case. This problem was resolved at second layer of computational hierarchy. The decision making was referred such that the SUIN only updated its web page for continuous system faults when both control loops simultaneously agreed on fault conditions.

More elaborate monitoring could be achieved by linking the flow rate (in the continuous process) to the water level fill rate (in the batch process) and hence the setting of thresholds. The various times required to fill the tank (at various flow rates) would have to be obtained through experimentation and the time-out values could then be determined. It would therefore become possible to reliably detect faults in the batch process even when faulty continuous process conditions are present. False alarm generation by the batch monitoring FEN would potentially cease and the system would detect multiple faults simultaneously. This approach however was not tested in the current work.

7.3 SUMMARY

The distributed monitoring system was again deployed on the process rig to test its capabilities in fault isolation. Blockage and leakage faults were simulated on a batch process and different fault symptoms were established for both faults. A fault isolation strategy was accordingly developed and implemented into the monitoring system. Running sum values of power, level, and flow signals were used and the FENs collaborated with each other (at the second computational hierarchy layer) to reduce load on the SUIN. The peer to peer communication between the FENs was thus found beneficial. Slightly different codes were used in the FENs to implement the devised strategy. The generalised nature of FEN code made this possible without disturbing the main code and functional arrangements.

The effectiveness of plug & play capabilities of the system, and its limitations, were investigated by deploying the monitoring system on a continuous flow process. The system continued working when the power FEN was switched off but with reduced functionality. Unavailability of an expected FEN was reported to the user. Unexpected conditions were also reported to the user via its web page when the system detected a combination of signals not anticipated in the system study phase.

Faults in interconnected control loops were also investigated using a continuous flow and a batch process together. The continuous process output (flow) was used as the batch process input. The monitoring system successfully detected a leakage fault in the batch process and mild blockages in the continuous process. It was however unable to locate a batch process fault when the continuous process reported severe blockage, as this fault affected batch process performance. A procedure to resolve this problem was conceived but not tested and implemented. The monitoring system's capabilities and limitations were thus determined for multi-loop processes.

The system's capabilities were thus evaluated with signal analysis in time domain. Its performance was next evaluated for frequency domain analysis by deployment on a machine tool. The details are provided in the next chapter.

TOOTH BREAKAGE DETECTION

This chapter reports work designed to evaluate the monitoring system's capabilities particularly in the frequency domain. The system was deployed on a machine tool to detect either tooth breakage. Knowledge of the machine tool and its monitoring was already available in the IPMM research group and had identified differences in frequency spectra for a new cutter and for a cutter with one broken tooth. The author applied sweeping filter technique (described in chapter 4.5.2) in combination with the monitoring system to implement a simple, low resolution frequency analysis. The performance of the system is detailed and discussed in this chapter. The need for multiple signal monitoring was again established and the benefits of the distributed monitoring system were emphasized. The work described was focused on monitoring system performance evaluation and was not intended to provide a complete solution to the complex problem of machine tool tooth breakage detection.

8.1 TOOTH BREAKAGE DETECTION THEORY

The Kondia B500 machine tool used has been the centre of previous research within the IPMM group which had identified machine signals via a machine audit (Jennings et al, 2001 B). Suitable interface arrangements were already in place and a large database of previously acquired signals was available for offline signal analysis and calibration purposes.

The previous research indicated the expected changes in the key signal frequency components when a tool breakage occurred. Johns (1998) identified two key frequencies in the axis drive motor current signal corresponding to 'tool rotation' and 'tooth passing' frequencies. The former depended on the spindle rotational speed used in cutting. The latter was the frequency at which cutting teeth entered the workpiece and was dependant on number of cutter teeth. For the tests for this work the case where the milling process was performed with a spindle rotating speed of 500 rpm and 4 tooth cutters was used. The tool rotation frequency was 8.33Hz under these conditions. Four teeth per revolution

caused a tooth passing frequency of $4 \times 8.33 = 33.33\text{Hz}$. Numerous other frequency components (harmonics etc) are typically present in machine signals for various reasons but were not considered in this research. When one of the 4 cutter teeth broke, the remaining 3 teeth produced a different frequency pattern. The tool rotation frequency remained as 8.33Hz but tooth passing frequency changed to $8.33 \times 3 = 25\text{Hz}$. Imbalance effects caused other frequency variations. For example, one of the teeth now has to cut extra metal to compensate for the broken tooth. This makes extra cutting effort for the tooth after the broken one occurs once per revolution, resulting in a 8.33Hz frequency component. The appearance of this component was ideal for detection purposes.

In other IPMM based work (Johns, 1998) a Wadkin v4-6 machine tool, whose axis drive system was controlled by dc motors, was used. Kondia B500 machine tool, however, used three phase ac motors for axes drives. The axes drive current signals were therefore tested to ensure consistency when using the stored signal and frequency analysis databases. Figure 8.1 shows the FFT derived frequency spectra for the X-axis drive motor current signals for new and broken tooth cutter conditions. The tooth passing frequency was visible in both cases and the change in the strength of the tool revolution frequency component (8.33Hz) was also observed. The emergence of the changed tooth passing frequency component (25Hz) on breakage showed presence of three teeth rather than four. The increase in strength of this component thus indicated tooth breakage. The initial FFT analysis thus confirmed the presence of both frequencies of interest with the ac motor signals in the Kondia machine. These set the specifications for the deployment of the distributed monitoring system and in particular for the set-up of the sweeping filters.

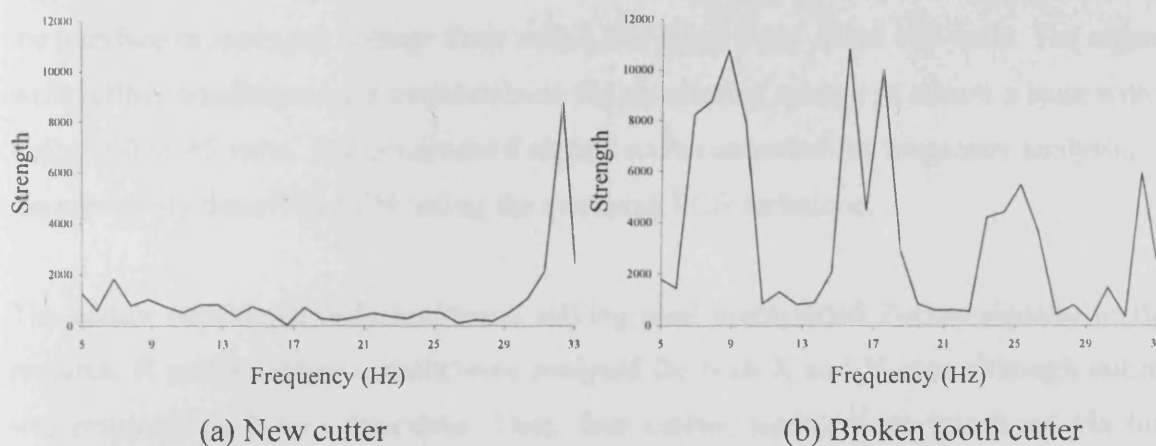


Figure 8.1: Frequency spectrum calculated with FFT

8.2 SWEEPING FILTER APPLICATION

Machine tool tooth breakage detection is a complex issue with a number of variables involved. A large body of research has been done in this field and its details are beyond the scope of this thesis. The author took a simplified approach and concentrated on only one out of several potential issues. The research reported in this chapter was aimed at detecting tooth breakage as a hard fault (a sudden event that need to be diagnosed in real time) to prove the monitoring system's capabilities. The spindle rotational speed was fixed at 500 rpm and the axis feed rate at 100 mm per minute. The same metal was used in all tests and only four teeth cutters were used. Cases of simultaneous breakage of multiple teeth were not considered. The monitoring system was deployed to detect changes in the frequency spectrum resulting from tooth breakage and successful detection proved its capabilities. However, it should not be considered as a complete solution for tooth breakage detection in real-life. Several other variables, such as rotational speed, depth of cut, material, etc, must usually be considered. The proposed monitoring system has potential to be useful but to complete a full solution would require more research than described in this chapter. Another aspect of this application was the higher amount of data generated and monitoring system's capability of tackling this data was also observed.

8.2.1 Monitoring Signals

The Kondia B500 machine tool can move in X, Y, and Z axes. Three 3-phase induction motors controlled the movements in these axes. Previous arrangements were available to acquire motor current signals for two phases for each of these 3-phase motors. The tapped signals were called 'R' and 'S' and were out of phase by 120°. These were available via the interface in analogue voltage form with a full range from -10 to +10 volts. The signals were further conditioned for connection to the monitoring system to ensure a span within range of 0 to +5 volts. The conditioned signals were connected for frequency analysing to the previously described FENs using the sweeping filter technique.

The author considered only horizontal milling (and disregarded Z-axes signals) in this research. R and S current signals were analysed for both X and Y axes although cutting was restricted to X-axis directions. Thus, four current signals were interfaced via four FENs with the SUIN integrating these outputs and provided combined monitoring results.

Cutting tests were performed under the selected conditions and the 8.33 and 25Hz frequencies were of interest. Each FEN was accordingly set up and swept a frequency range from 6Hz to 28Hz and calculated a measure of signal strength in each 1Hz frequency band within the range. Acquisition was run for 640ms to allow the signal to become relatively settled and to assure consistent filtering. The analogue filter was not fully settled in that time but its output provided enough amplitude difference to reliably indicate the strength measure for each band. The same filter settings were used for all bands thus making their results comparable (a fully settled analogue filter would provide higher filtered signal strengths for the same signal with the sweeping filter technique but would take a longer time). The acquisition time was a required compromise between detectable strengths and detection times.

The filter settings were stored in each FEN memory as a table and the built-in PWM generator (in the PIC 18F458) generated appropriate filter control frequencies according to the table. The filter output was acquired at each stage by the MCU and the maximum and minimum values were determined. The difference between them provided the strength measure for each frequency component. The processing requirements had again been decided upon to match the PIC MCU capabilities. The system study used the monitoring system in a mode where the total profile of the signal was obtained. Clear differences in acquired signal strengths were observed for new and broken tooth cutters. The R current signal for X-axes drive is shown in figure 8.2, as an example, for both cutter conditions. New and broken tooth cutters are also shown.

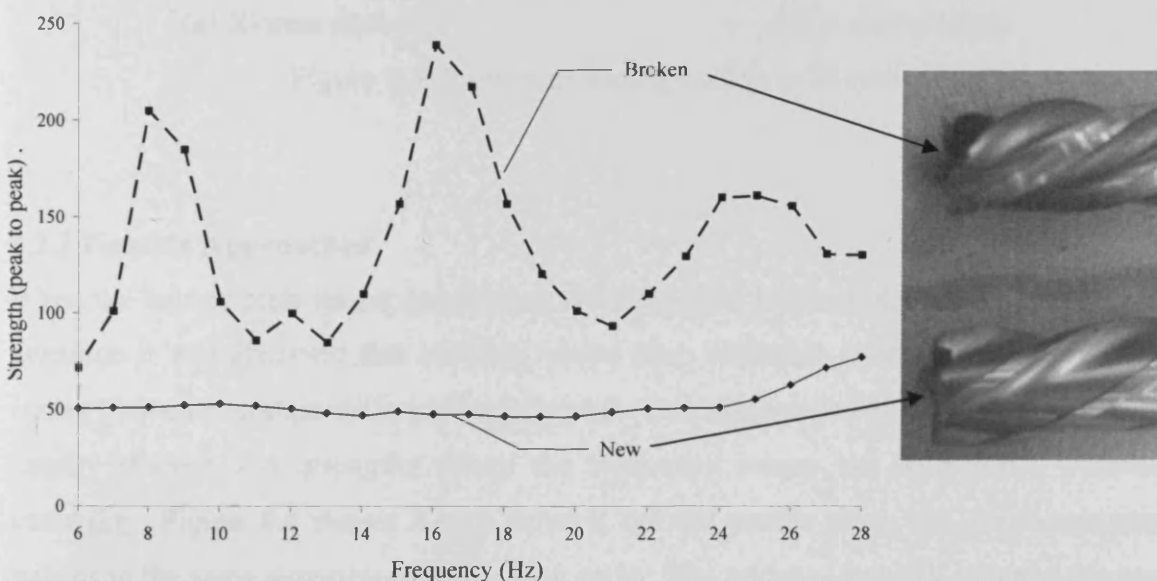


Figure 8.2: R current profile for X-axis drive signal with new and broken tooth cutters

Y-axis drive currents were also monitored in the same fashion and no clear pattern was observed in them since cutting was done only in X-axis. There was no movement in Y-axis and its motor was not loaded. The Y-axis FENs were included for completeness and would become important if other than single axis directional milling was to be monitored. The obtained Y axis spectra (not shown) confirmed that these nodes were, as expected, insensitive to X axis milling.

The lack of cutting information in axes drive currents other than the cutting axis was also confirmed from time series plots acquired during the system study. For example, figure 8.3 shows time series data for X and Y axes (R motors currents) with cutting in the X-axis only. The time series plot in figure 8.3a has a 1.5 seconds period component. This was consistent with the feed rate of 100mm/min combined with 10mm ball screw pitch and a 8-pole axis drive motor (Jennings, 2001 B). Figure 8.3b confirms no movement in Y-axis.

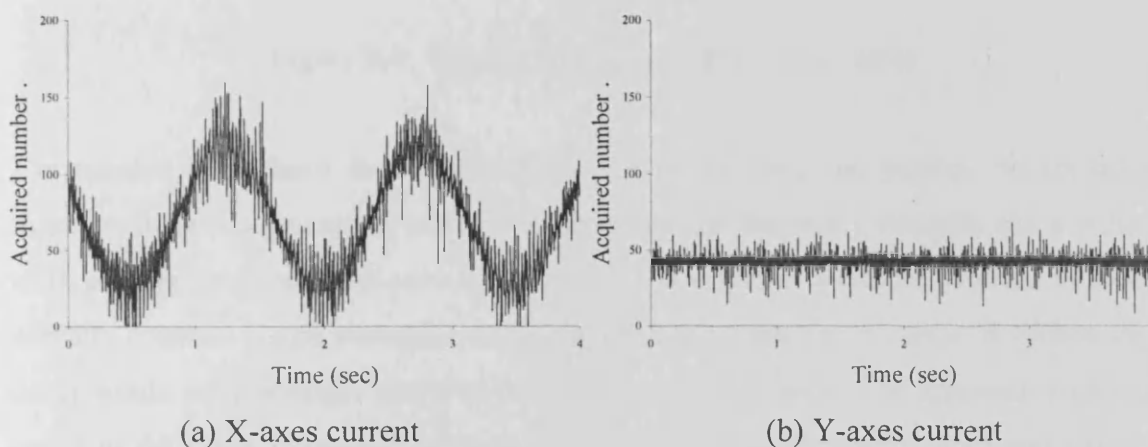


Figure 8.3: R currents during cutting in X-axes

8.2.2 Possible Approaches

When the cutting tests (at set conditions) were repeated to check the consistency of fault detection it was observed that variable results were obtained. A broken tooth did create strong frequency components, particularly at the two frequencies of interest. A new cutter usually showed low strengths across the frequency sweep but with some wideband variability. Figure 8.4 shows X-axis drive R current profile from two (time-separated) sweeps in the same experiment with a new cutter. The wideband profile showed the same

pattern in both cases but with a relatively large offset in the overall signal strength levels. This caused problems in threshold determination. The reason for this behaviour was not properly understood and was beyond the scope of this thesis.

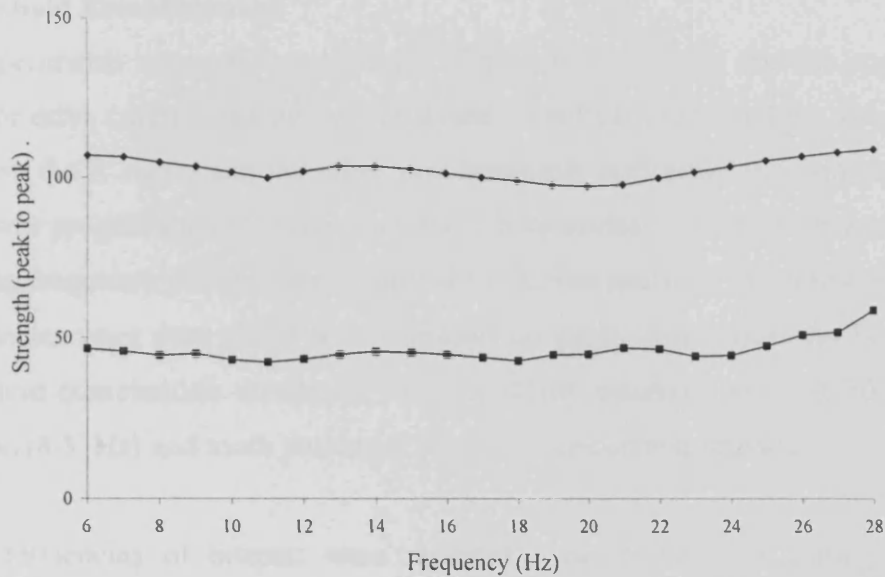


Figure 8.4: Varying signal strength for new cutter

The solution considered thus required analysis of the spectrum pattern. As previously stated tooth breakage caused increase in tool rotation frequency strength and a reduced tooth passing frequency. As seen in figure 8.2 a new cutter would be detected as having virtually constant signal strengths across the considered frequency range. A broken tooth cutter would have strength peaks at the frequencies of interest. The approach required a sweep of the entire frequency range or, if a reduction in computation times was required, across a more focused range of frequencies.

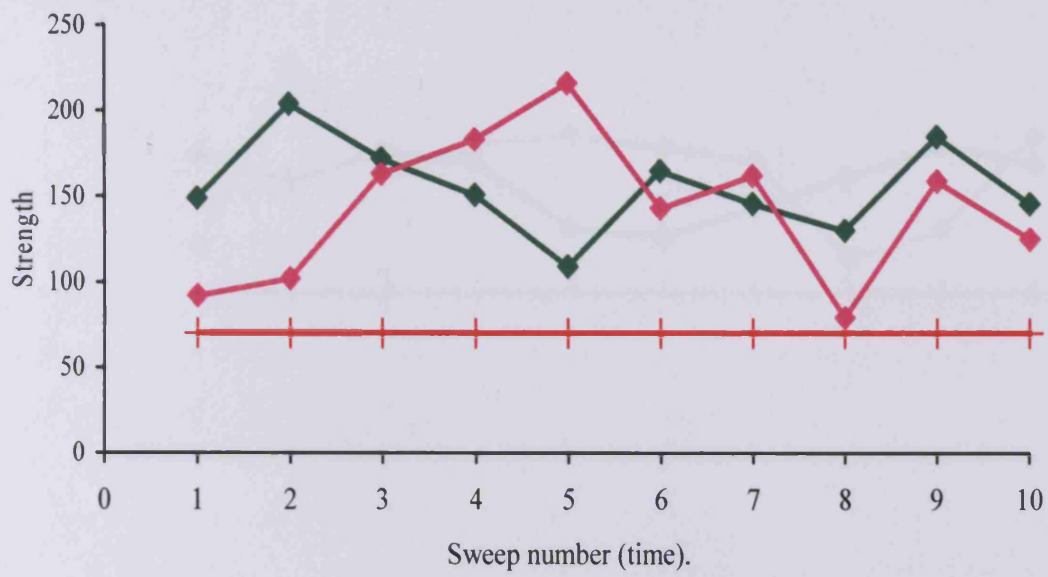
The hierarchical decision making process was again utilised via the use of multiple signals for tooth breakage detection. Two FENs performed independent analysis on the R and S current signals (for the appropriate cutting axis). The 2 FEN results were compared between themselves and provided a combined conclusion to the SUIN. Threshold values were established via a system calibration phase (details in section 8.2.3 following) for each signal for both frequencies of interest. Each FEN performed frequency analysis only for the two focused frequency sweeps, centred at 8.33 Hz and 25Hz in this case. A higher than threshold strength in a signal component indicated tooth breakage. Each FEN was

programmed to generate a broken tool alarm only when thresholds were exceeded at both frequencies.

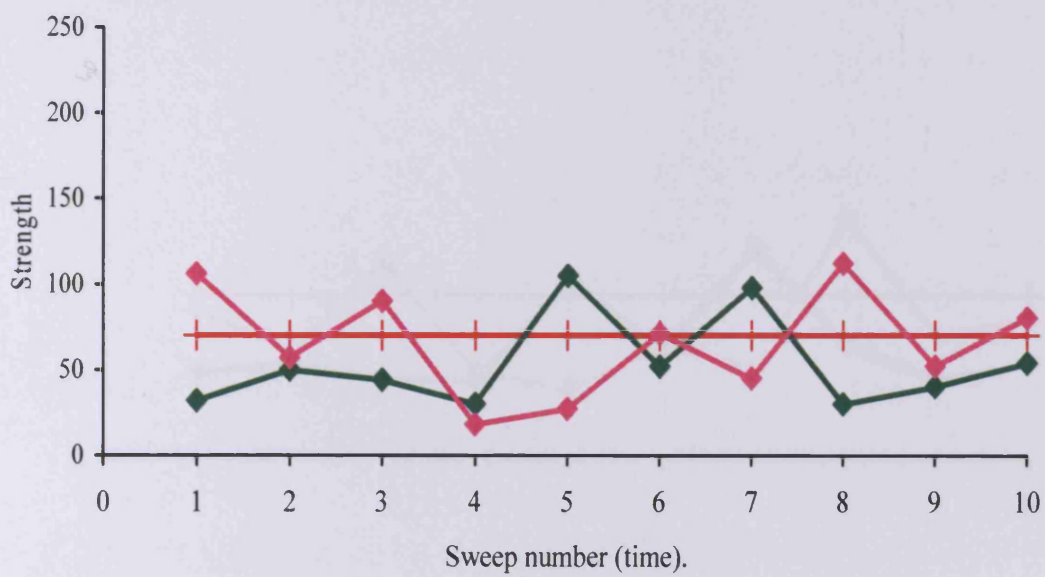
8.2.3 Threshold Establishment

Cutting experiments were performed in the X-axis direction only and the appropriate R and S motor drive current signals were analysed. One FEN employed the sweeping filter technique on the R signal and the other simultaneously performed the same on S signal. The filter was programmed to sweep only the 2 frequencies of interest for each FEN and the resulting frequency profiles were captured for further analysis. The threshold values at two frequencies were determined with emphasis on using same values for both R and S signals. These combination thresholds were set (fairly conservatively) at 70 and 95 for tool rotation (8.33Hz) and tooth passing (25Hz) frequencies respectively.

The two frequencies of interest were repeatedly swept during a cutting test. Each completed sweep was immediately followed by the next sweep throughout the cutting run. Figure 8.5(a) depicts the obtained strength values for tool rotation frequency from first ten sweeps during a test where a broken tooth cutter was used. It was observed that both R and S signals consistently exceeded the set threshold and provided indication of the fault. Figure 8.5(b) provides results from a similar test undertaken with a new cutter. Strength values for both R and S signals frequently (but not consistently) exceeded the set threshold confirming the previously observed uncertainty introduced by the 'global' frequency strength variations for new cutter. It was also observed that both signals crossed the threshold at different times and no simultaneous threshold violation was observed. The FENs performed similar analysis on tooth passing frequency during the same sweeps and the results obtained during the same cutting test are presented in figure 8.6. Signal strengths for both R and S currents consistently exceeded the set threshold for broken tooth cutter and uncertainty was observed with new cutter. Once again, the two signal never crossed the threshold simultaneously. This consistent behaviour from both signals for both frequencies of interest provided fault detection opportunity.



(a) Broken tooth cutter



(b) New cutter

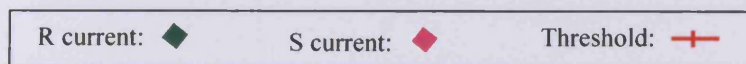
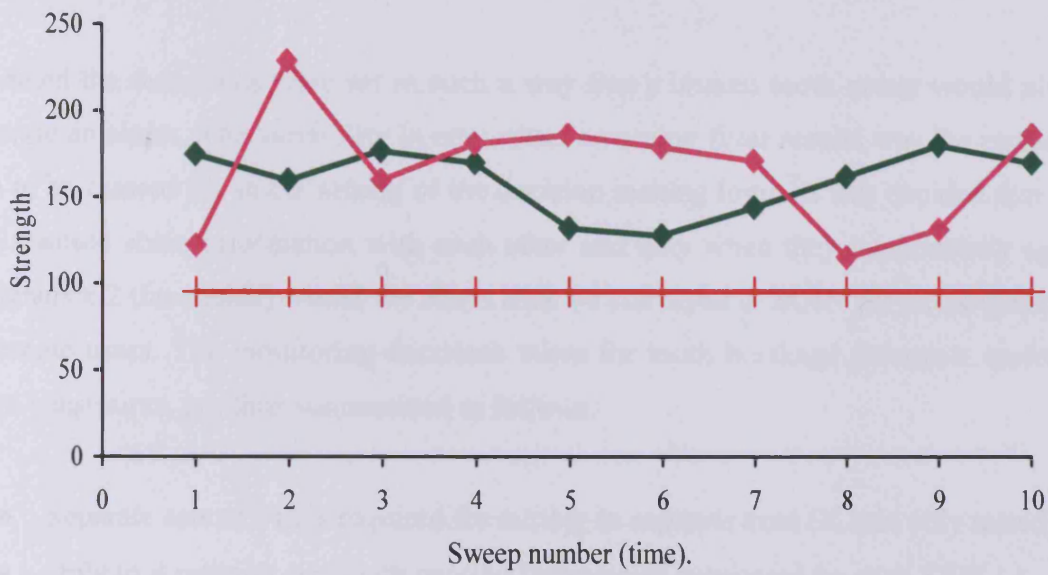
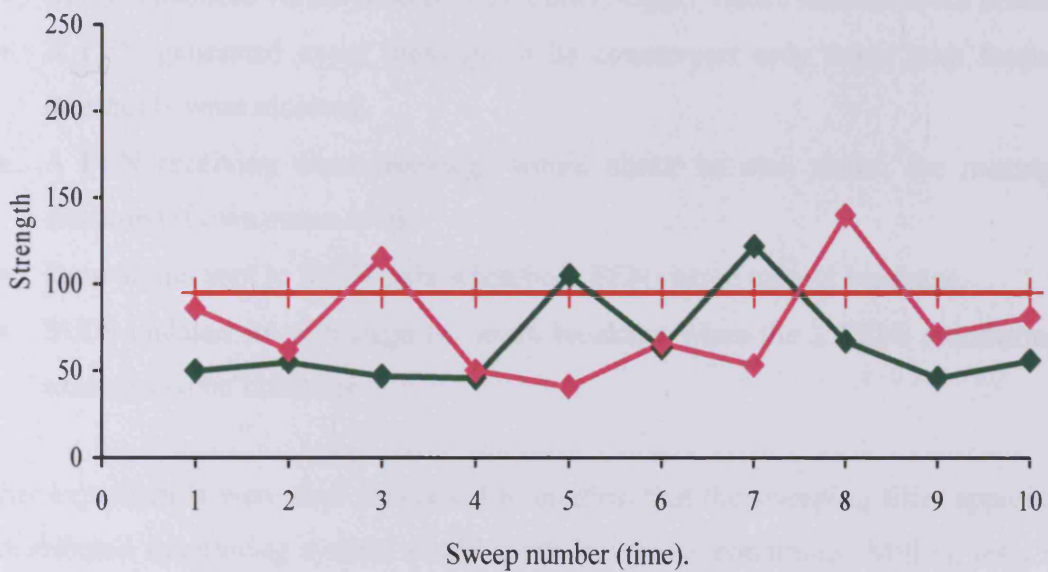


Figure 8.5: Tool rotation frequency (8.33Hz) for R and S signals at different times



(a) Broken tooth cutter



(b) New cutter

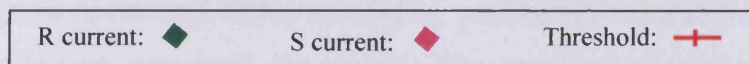


Figure 8.6: Tooth passing frequency (25Hz) for R and S signals at different times

8.3 MONITORING DECISIONS

As stated the thresholds were set in such a way that a broken tooth cutter would always generate an alarm. The variability in new cutter sweeping filter results was the remaining area to be catered for in the setting of the decision making logic. It was decided that both FENs would share information with each other and only when they unanimously agreed (2 signals x 2 thresholds) would the alarm state be conveyed to SUIN for onward delivery to remote users. The monitoring decisions taken for tooth breakage detection, under the given constraints, are thus summarised as follows:

- Separate sets of FENs required for cutting in separate axes (X axis only tested).
- Only tool rotation and tooth passing frequencies monitored by each FEN.
- Threshold for tool rotation frequency (8.33Hz) strength was set at 70.
- Threshold for tooth passing frequency (25Hz) strength was set at 95.
- Below threshold values indicate new cutter; higher values indicate tooth breakage.
- A FEN generated event message to its counterpart only when both frequency thresholds were violated.
- A FEN receiving alarm message would check its own status; the message is discarded if own status is ok.
- Error signal sent to SUIN only when both FENs agree on tool breakage.
- SUIN updated its web page for tooth breakage when the 2 FENs monitoring an axes agreed on breakage.

Further experiments were then conducted to confirm that the sweeping filter approach in the distributed monitoring system would work in various conditions. Milling tests were performed with varying depths of cut to investigate the monitoring system response. Cutting at 0.5, 1, 1.5, and 2 mm depths was performed (with the previously established threshold settings in operation). Other machining parameters remained constant and the frequencies of interest remained the same. The monitoring system was able to successfully distinguish between new and broken tooth cutters in almost all of these experiments. For discussion purposes, the system was run in monitoring mode to provide full result sets. The frequency profiles are collated and presented in figure 8.7.

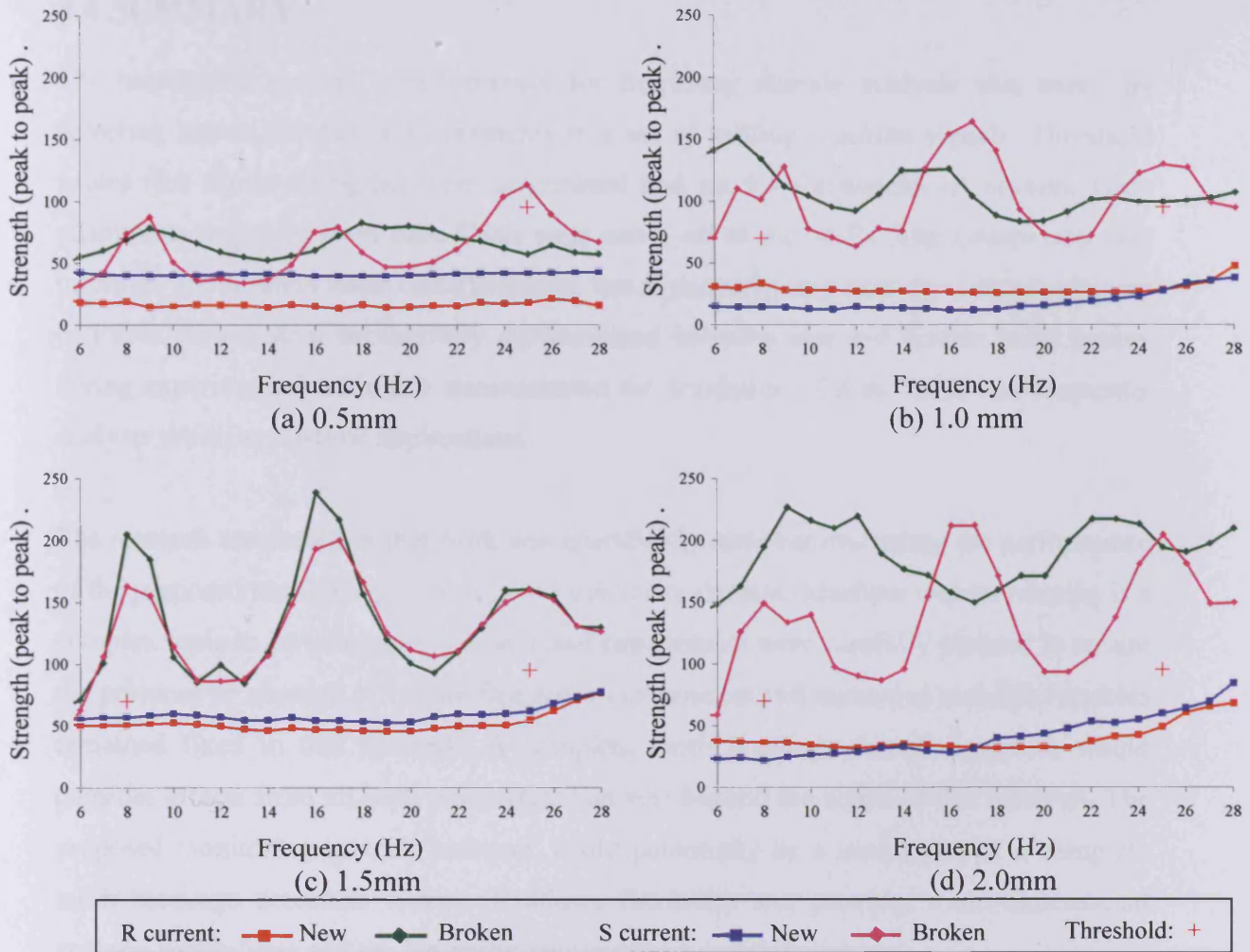


Figure 8.7: Frequency profiles for various depths of cut

As can be seen from the profile to thresholds comparisons, the monitoring system successfully detected tooth breakage for the 1.0, 1.5 and 2.0 mm depths of cut. It was unable to consistently detect tooth breakage for the 0.5mm depth of cut. In this case the tooth passing frequency strength was sometimes observed below threshold and the FENs (set to require all 4 threshold tests to be true for broken tooth) did not generate an event or error message. This limitation was mainly due to the use of fixed (conservatively set) thresholds. The thresholds had been set in order to minimize false alarms and were prone to error in marginal cases. A lower threshold value would potentially detect tooth breakage with a 0.5 mm depth of cut. Possibly an additional FEN could be used to provide a depth of cut measure and its input to the monitoring system might allow the threshold values to be determined dynamically. The monitoring system has the capability to include more FENs and would support such enhancements. The addition of a depth of cut measure FEN was not tested in the current research.

8.4 SUMMARY

The monitoring systems's performance for frequency domain analysis was tested by detecting known frequency components in a set of milling machine signals. Threshold values (for signal strengths) were determined and set for frequencies of interest. Only unanimous decisions from both FENs were conveyed to the SUIN. The system user was informed about faults when tooth breakage was reported by any axes drive monitoring set of FENs. The system successfully differentiated between new and broken tooth cutters during experimental testing. It demonstrated the feasibility of 8-bit MCUs as frequency analysis tools in real-time applications.

The research conducted in this work was specifically aimed at evaluating the performance of the proposed monitoring system in the frequency domain. Machine tool monitoring is a complex topic in its own right. The selected experiments were carefully planned to ensure the presence or absence of certain frequency components and numerous real-life variables remained fixed in this research. A complete tooth breakage detection system would consider effects from all such parameters and was beyond the scope of this research. The proposed monitoring system, however, could potentially be a useful part of a complete tooth breakage detection system. It allows flexibility and provides multi-dimensional analysis capabilities and the hierarchy allows robust decision making.

This application also highlighted network traffic reduction achieved by processing the signals at the first hierarchy layer (close to the source). A relatively higher sampling rate was required for frequency analysis and the signals were acquired at 100 samples per second. Six signals at this rate generated 51.84 MB of raw data in a day. This raw data (with network overheads) would have been transmitted on the network in a centralised processing system. The distributed hierarchical system processed this raw data at the source thus considerably reducing the network traffic.

The next test applied to the monitoring system was to check its performance for industrial systems. A process rig was used for this purpose that contained a commercial digital valve controller mounted on an industrial valve. The details of the monitoring system's deployment on this air flow process rig are provided in the next chapter.

AIR FLOW PROCESS MONITORING

This chapter describes the deployment and testing of the developed system on a commercial Digital Valve Controller (DVC) in order to ascertain its performance for industrial systems. The DVC has its own valve monitoring procedures that may be accessed through the PC based software supplied with it. However, the DVC, and hence the process valve to which it is fitted, must be taken out of active service for the diagnostics to be run. The DVC was used with a process valve to control the flow rate of air in a pipeline and the aim was to supplement the supplied diagnostic capabilities. The distributed monitoring system was used whilst the DVC maintained the normal set point control function. Two typical faults, namely deteriorating valve diaphragm condition and partial pipe blockage, were investigated. Different symptoms were identified for these faults and the monitoring system successfully determined either fault cause. A fault in supply air pressure (reduction) was also diagnosed. These, along with details of the user interfaces used, are described in the following sections.

9.1 PROCESS RIG

In this section of research the monitoring system was deployed on a process flowline containing an industrial control valve. Such valves are widely used for automatic flow control, especially in the petrochemical industry. A brief but useful introduction to pneumatic valve functionality can be found in Kempley (1980). The process rig is shown in figure 9.1. The DVC controlled the valve position under a Highway Addressable Remote Transducer (HART) protocol. HART provides digital communication capability on 4 to 20 mA analogue current loop lines and its details can be found in HCF LIT 34 (1999). A working schematic of the process rig is described in figure 9.2. The desired valve position was transmitted to the DVC which ensured correct valve positioning via its feedback mechanism. A 16-bit microprocessor embedded in the DVC generated the control commands. A pneumatic relay converted these into drive pressures on the valve actuator. The actuator moved the valve position, which was fed back to the microcontroller. Valve calibration and PID control parameters for this closed-loop control were already stored in the DVC memory.

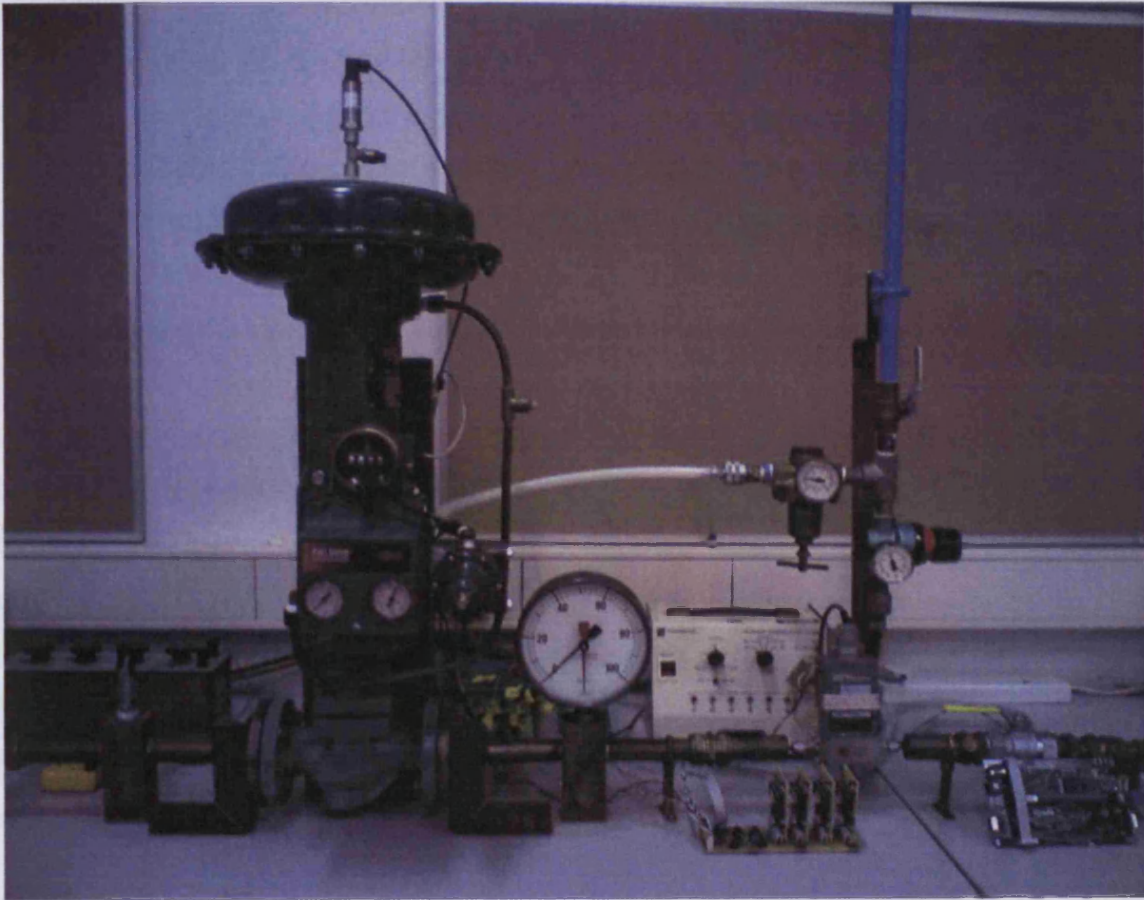


Figure 9.1: Process rig

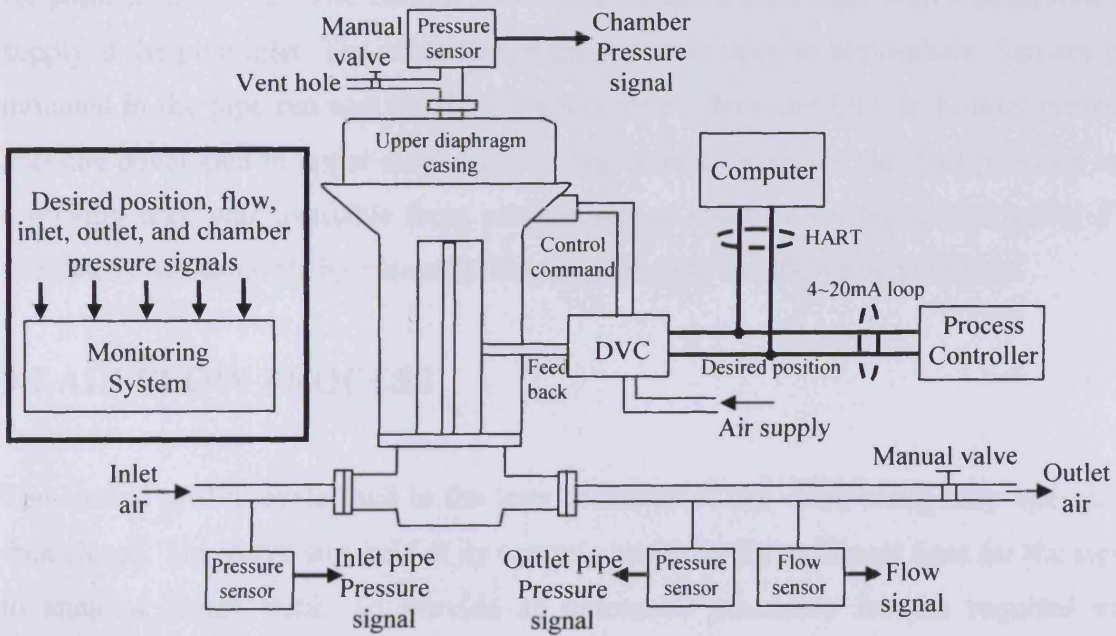


Figure 9.2: Process rig working schematic

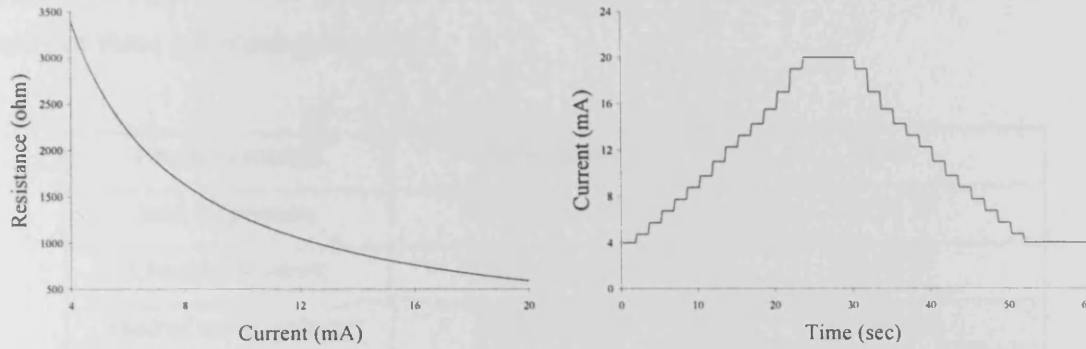
The DVC worked in 'Instrument' and 'Control' modes. Instrument mode specified whether the valve was 'Out-of-service' or 'In-service'. The DVC reacted to digital HART commands in the Out-of-service mode and off-line diagnostic routines and valve health checks, requiring specific valve movements, were performed. Changes to Control mode settings were also allowed in this mode. When the valve was in the in-service mode the desired position was communicated to the DVC through digital data in the 'digital' control mode and through the 4 to 20 mA analogue current in the 'analogue' control mode. The DVC mode selection was controlled (over HART) by devices such as a hand-held HART Communicator or a HART-enabled PC. The process rig used a PC with installed VLink2000 software to provide such facilities. The PC serial port was connected to the DVC via a serial-to-HART converter. Further details of the DVC, valve and the existing process rig can be found in Sharif (1999). Any air leakage through a faulty valve diaphragm reduced the effective actuator pressure and affected correct valve positioning. The VLink2000 software could be used to perform off-line diagnostics but required the process operation to be stopped. With the distributed monitoring system on-line evaluation of the diaphragm condition became possible.

The process rig was designed to control the valve in the in-service analogue mode. Current derived from a 24 volt power supply, via a resistance box, provided the desired set point to the DVC. The control valve was mounted on a pipe with a pressurised air supply at the pipe inlet. The other end of the pipe was open to atmosphere. Sensors were mounted in the pipe run and provided signals for air flow and inlet and outlet pressures. Pressure developed in upper diaphragm casing, referred to as the chamber pressure in the following text, was available from another sensor mounted on top of the valve. Fault simulation was possible by manually blocking the pipe and the valve vent hole.

9.2 AIR FLOW PROCESS

The control profile cycle used in the tests consisted of the valve being fully opened and then closed. The valve was held at its extreme positions for sufficient time for the signals to attain a steady state. To provide an automatic procedure for the required valve movement cycles a PIC 18F458 microcontroller was used as an open loop controller (of the control current to the DVC). The PIC MCU changed loop resistance via a

programmable digital potentiometer IC AD7376, using a Serial Peripheral Interface (SPI). Figure 9.3a depicts the determined non-linear performance. Adjusted MCU commands were used to compensate for this in order to attain linear characteristics. The AD7376 had limited resolution and this led to only 15 valve positions between fully open and fully closed. Table 9.1 shows the adjusted data commands sent by the MCU and the generated control signal is shown in figure 9.3b.



(a) DVC nonlinear behaviour

(b) Generated control signal

Figure 9.3: Linearization of control command

MCU command	Resistance (Ω)	Current (mA)
42	3404	4
35	2865	4.75
28	2326	5.75
23	1941	6.75
20	1710	7.75
17	1479	8.75
15	1325	9.75
13	1171	11
11	1017	12.25
10	940	13.25
9	863	14.25
8	786	15.5
7	709	17
6	632	19
5	555	20

Table 9.1: Control signal generation

Normal conditions were taken as 4 bar supply air pressure, 1.4 bar supply air pressure at DVC input, new diaphragm, and no blockage in pipe.

9.3 MONITORING SIGNALS

The process rig contained sensors for measuring inlet air pressure, air flow in the pipe, and chamber pressure above the diaphragm. These sensor signals were interfaced to three FENs and another FEN was interfaced with the desired analogue valve position command signal. The pipe outlet pressure was also monitored via a fifth FEN. Table 9.2 provides details of these monitoring signals.

Process variable	Description	Level
Inlet air pressure	Analogue voltage	0 to 15 V
Chamber pressure	Analogue voltage	0 to 15 V
Desired valve position	Control signal	4 to 20 mA
Air flow	Analogue current	4 to 20 mA
Outlet air pressure	Analogue current	4 to 20 mA

Table 9.2: Monitoring signals

Operational amplifiers (OP27) were used to provide signal isolation. Each 4 – 20 mA current signal was converted into a voltage using a RCV420 IC. During the system study the monitoring system was set to present the acquired signals for analysis. Figure 9.4 shows acquired profiles for each signal under normal operating conditions for one process cycle.

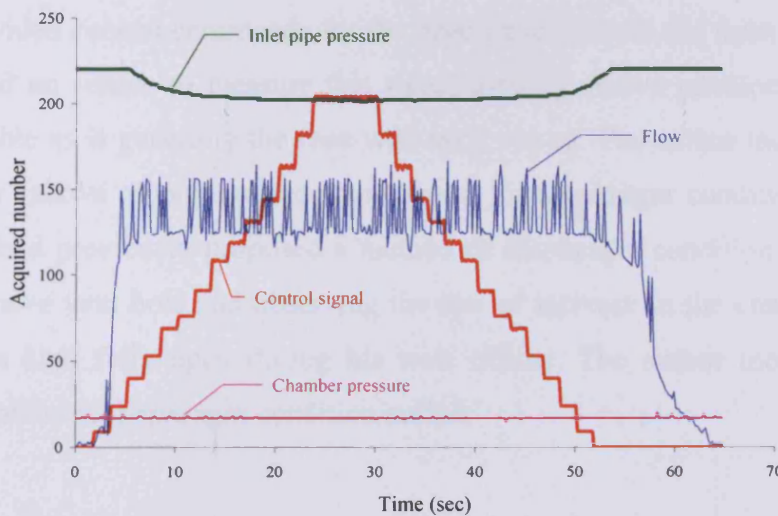


Figure 9.4: Normal condition signals

A small decrease in the inlet pipe pressure was observed whilst full air flow was being delivered. This was a practical constraint of the air supply system. It was also observed that maximum flow was established before the valve was fully opened. Although it was possible to establish the minimum valve opening required for full flow the control signal used in the reported research however moved the valve stem between the extreme positions of fully closed and fully open.

9.4 DIAPHRAGM CONDITION MONITORING

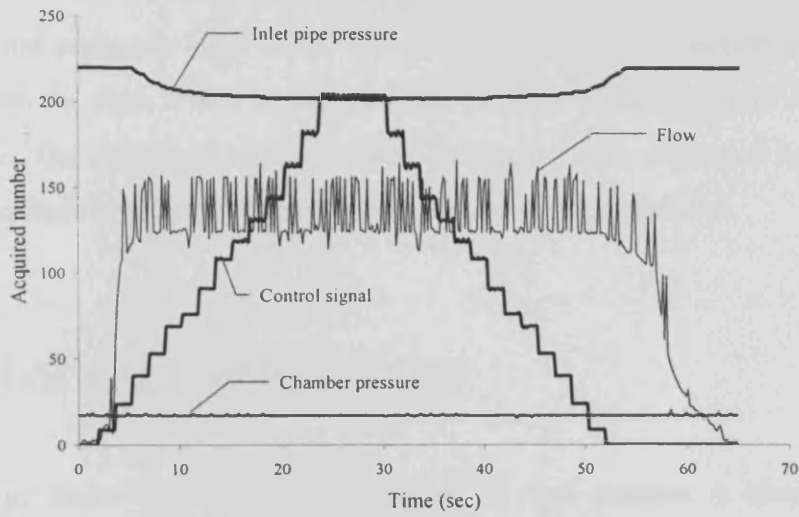
The diaphragm condition is an important factor for correct valve positioning. Normal wear and tear causes deterioration in the diaphragm condition over time. A leaking diaphragm will cause some air pressure to be lost. This causes pressure reduction on the actuator and a lesser valve opening is achieved for the same DVC command. The DVC would compensate on detecting the reduced opening through its feedback mechanism and the process would then remain in a normal state. Thus, often the incipient diaphragm fault is hidden by the feedback control system. Eventually severe diaphragm deterioration hampers valve opening and when detected the process has to be shutdown for diaphragm replacement to take place. Time based valve maintenance is typically performed to avoid this situation and the process is shutdown at predetermined times irrespective of the actual diaphragm condition. Online diaphragm monitoring can minimize such process shutdowns.

The DVC provided control commands for the pneumatic valve in the form of air pressure and the rig had no sensor to measure this signal directly. Valve position feedback was also not available as is generally the case with such valves. The author therefore used an upper chamber (above diaphragm) pressure signal for diaphragm condition monitoring. Sharif (1999) had previously proposed a method of diaphragm condition monitoring by blocking the valve vent hole and observing the rate of increase in the chamber pressure. The valve was kept fully open during his tests offline. The author modified Sharif's method and monitored diaphragm condition online.

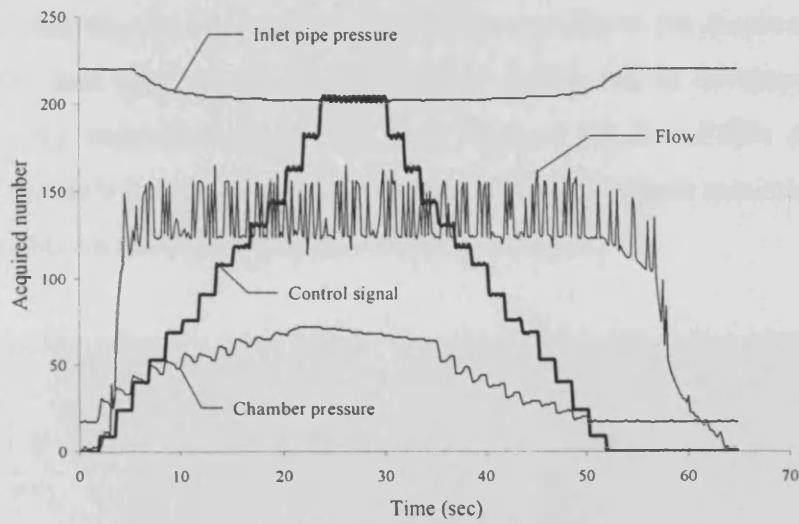
The chamber pressure signal, for new and deteriorated diaphragms was acquired. The deterioration of the diaphragm condition was simulated by making pin holes in it. After some initial testing with the chamber pressure transducer fitted, but with the vent hole still open to atmosphere, it was decided to monitor this signal with the vent hole blocked off. The monitoring system would control a solenoid valve for producing the temporary vent hole blockages during a test. This would also provide the monitoring system with the capability to perform such tests on a user demand. In the absence of this automated arrangement, the vent hole was manually blocked in this research over one process cycle. The chamber pressure was then observed to gradually increase with time for both new and faulty diaphragms. The relative sensitivity of tests with open and blocked vent hole for the signals collected in system calibration mode can be seen in Figure 9.5. When tested with a severe fault (large hole) in the diaphragm the chamber pressure rose very quickly and valve failed to open at all. This confirmed that a severe diaphragm fault would cause process disruption. The monitoring system was therefore required to generate an alarm before such severe diaphragm degradations developed. It was concluded that the vent hole can be blocked (for short duration testing) without hampering the normal process operation and this method was adopted.

Figure 9.6 shows chamber pressure for new, pin-holed, and large-holed diaphragms where the increase in this signal with fault severity is clearly visible. The same control signal was used for all cases and similar flow was obtained from new and pin-holed diaphragms, due to controller compensatory action. However, the increased chamber pressure for the large -holed diaphragm effectively disabled valve motion and no flow was observed.

Over a number of calibration tests it was observed that the acquired chamber pressure signal had a maximum value of 72 (ADC units) with a new diaphragm in place. The threshold value established for the FEN was set at 75 to detect diaphragm degradation. With a pin-holed diaphragm in place the pressure crossed this threshold between 15 to 19 seconds into the process cycle. The observed maximum value for pin-holed diaphragms over a set of repeated tests was 84. The monitoring system thus detected diaphragm deterioration before it started affecting the process. Use of multiple threshold values would quantize the level of diaphragm deterioration but was not tested in this research.



(a) Open vent hole



(b) Blocked vent hole

Figure 9.5: Monitoring signals for normal process

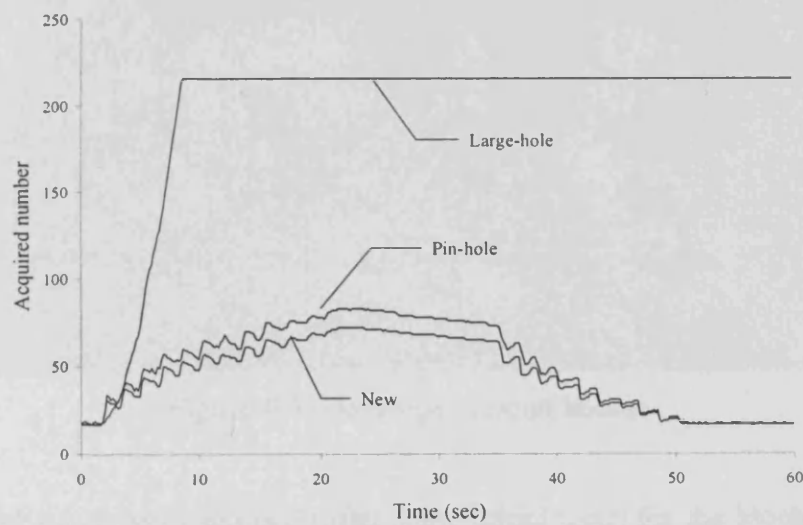


Figure 9.6: Chamber pressure signals for various diaphragm conditions

Diaphragms with relatively large scale leakage faults lead to a reduction in the controlled air flow through the pipe. It was recognized that pipeline blockage faults would also have similar effects. The detection of partial pipe blockage faults presented in the following section and the fault isolation studies are then presented in section 9.6.

9.5 PIPE BLOCKAGE MONITORING

Partial blockage faults were investigated for the air flow process. A blockage fault was simulated by partially closing a manual valve. Control and flow signals were acquired along with pressure signals at pipe inlet, pipe outlet and above the diaphragm (chamber). Five FENs were thus required for five monitoring signals but in developing the plug & play capability the manufactured circuits only allowed for four FENs at this stage of development. Figure 9.7 shows the developed circuit boards where mounting of only four FENs was possible on the signal conditioning (base) board.

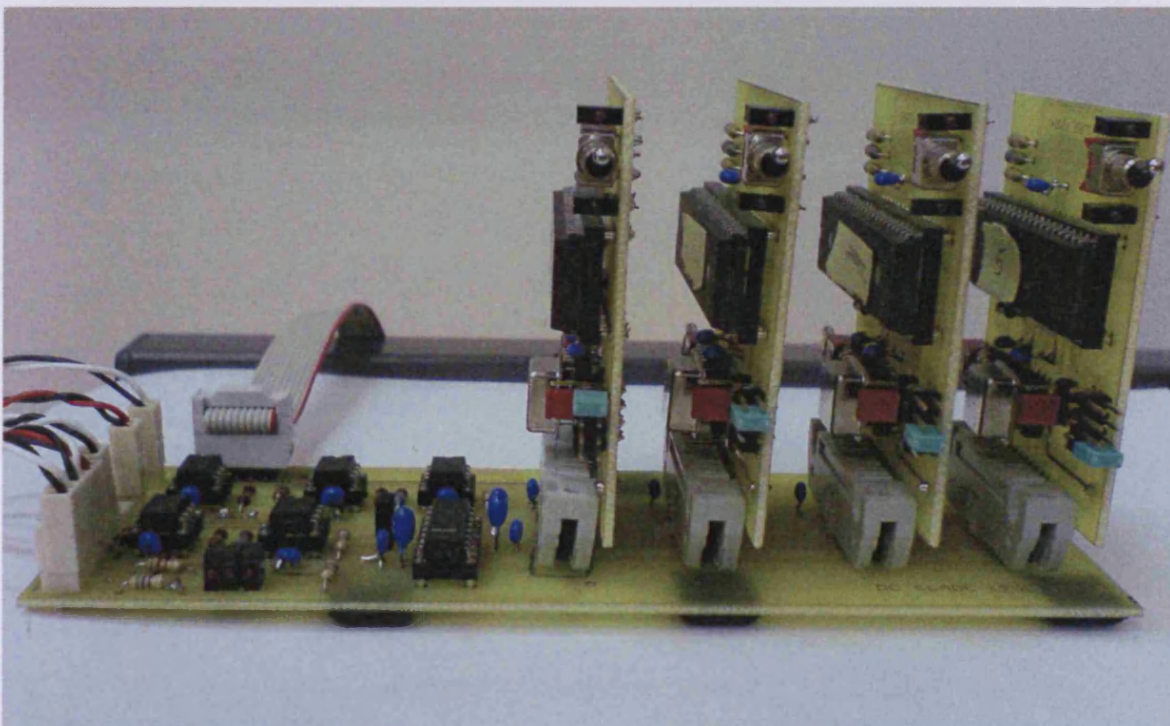
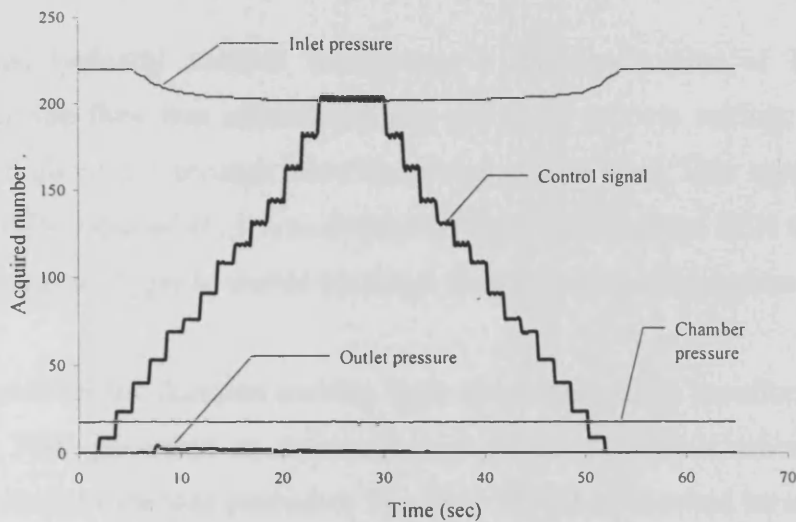


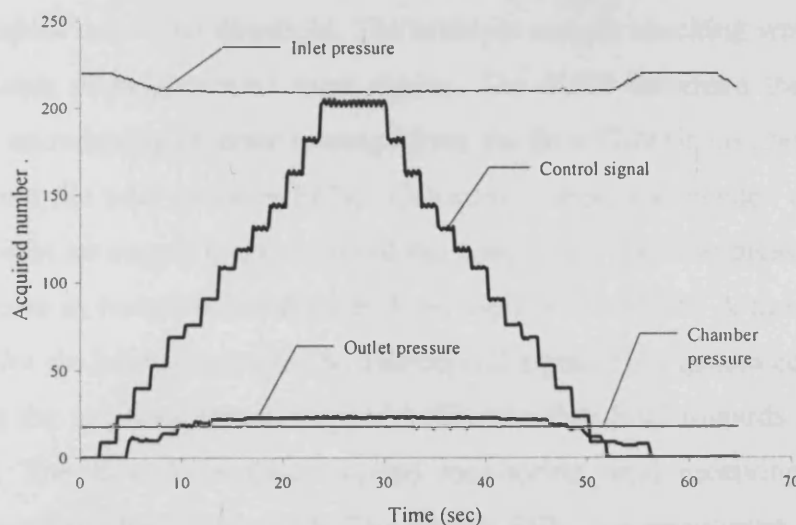
Figure 9.7: Developed circuit boards

The pipe outlet signal was excluded (after some initial tests) for the blockage fault tests, only due to the limits on the hardware available at the time of testing. Figure 9.8 shows

the results of the initial tests (with flow being the excluded signal) for normal and blockage cases. The pressure increase in the outlet pipe with a partial blockage present is clearly visible. Figure 9.9 then shows the signals acquired in the blockage fault case with flow forming the fourth FEN rather than outlet pipe pressure. The effect of the blockage is, as expected, reduced flow, when compared to figure 9.4 (with same control signal). From figure 9.4 and figure 9.9 the typical flow reduction during mid-cycle is of the order of 16%. Thus either flow or outlet pressure signals could provide an indication of a partial blockage fault. The pressure signal was dependant on blockage location and any blockage between the valve and the sensor could not be detected. The flow signal was unaffected by fault location and was adopted into the monitoring scheme.



(a) Normal case



(b) Partial blockage case

Figure 9.8: Signals with outlet pipe pressure sensor

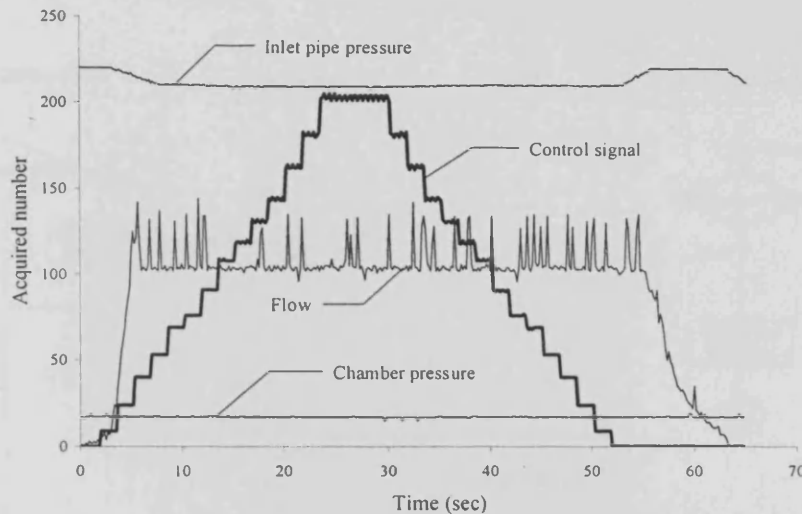


Figure 9.9: Signals with partial pipe blockage (flow sensor)

The flow signal typically attained values over a 'baseline' value of 120 in normal conditions once the flow was established. For the given process settings full flow was usually attained about 5.5 seconds after the process cycle start. This corresponded to a control signal FEN value of 40. It was decided to use 35 as a control FEN threshold value to effectively act as a trigger to enable blockage fault detection to commence.

Figure 9.10 describes the decision making logic developed as the monitoring scheme. A control signal FEN generated an event message to the flow FEN when its threshold indicating developed flow was exceeded. The flow FEN then checked its acquired values relative to a threshold of 120. It only generated an error message to SUIN on detecting 4 successive samples below the threshold. The multiple sample checking was performed in order to eliminate noise-generated false alarms. The SUIN informed the user about a blockage fault on receiving an error message from the flow FEN (in combination with the ok message from the inlet pressure FEN). This combination was needed to confirm that an absence of inlet air supply had not caused the symptoms. The inlet pressure FEN could report such a case in isolation and directly from itself to the SUIN. A threshold value of 190 was used for the inlet pressure FEN. The control signal FEN generated another event message when the acquired signal dropped below the threshold towards the end of the process cycle. The flow FEN then stopped monitoring until receiving another start monitoring event from the control FEN. The control FEN stop monitoring event typically occurred 48.5 seconds into the process cycle. A sub-message field in the CAN message header was used to differentiate between start and stop event messages.

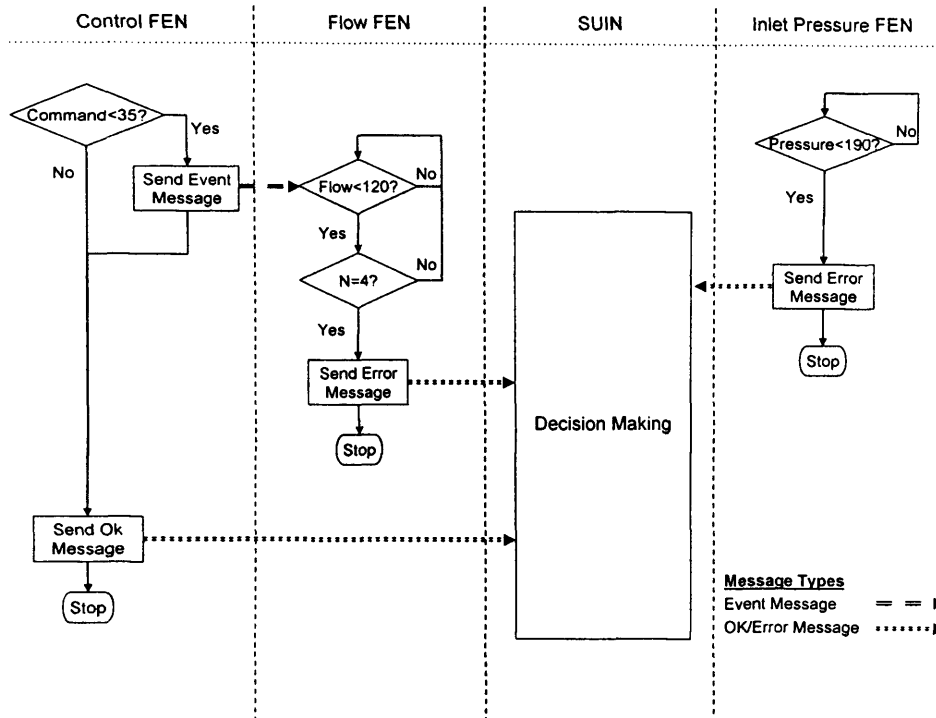
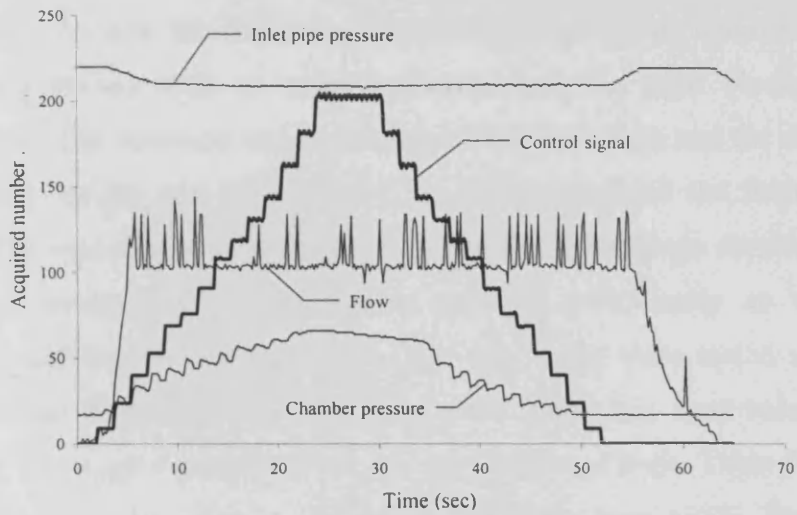


Figure 9.10: Developed decision making logic

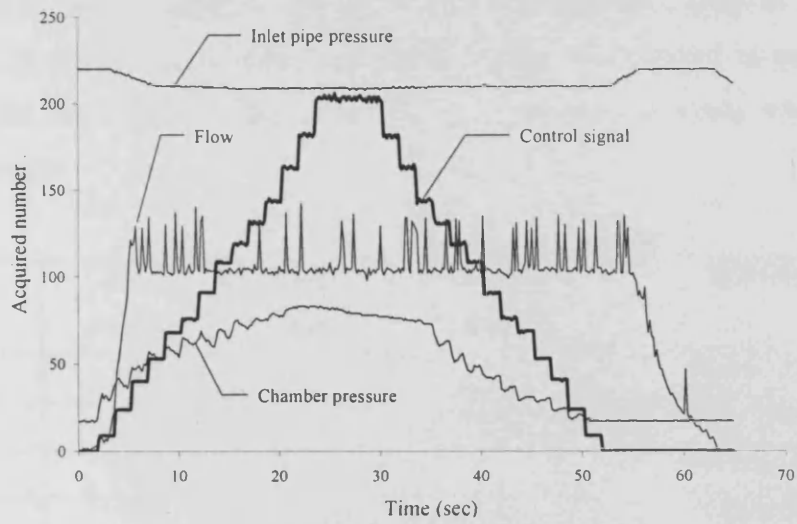
9.6 FAULT ISOLATION

The blockage faults reported in the previous section were from tests with the vent hole open to atmosphere. Prior to determining the fault isolation capabilities it was necessary to study the effects of a blocked vent hole on the signals used for pipe blockage detection. The vent hole was blocked for one valve cycle and the acquired monitoring signals for new, pin-hole and large-hole diaphragms are shown in figure 9.11.

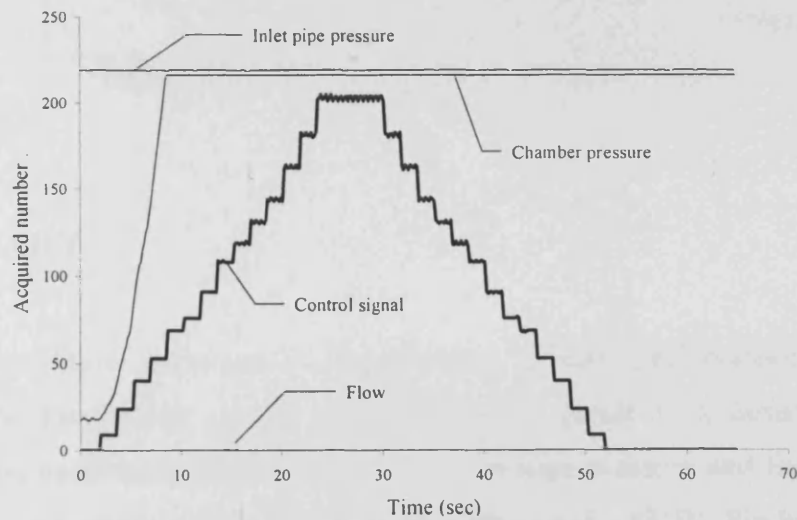
With the existing FEN/SUIN thresholds and decision logic still in place, the results obtained with a new diaphragm detected flow reduction combined with an ok message from the chamber pressure FEN. This was taken as indication of partial pipe blockage fault. No flow was observed for a diaphragm with a large hole (figure 9.11c) as no valve movement was possible. The FENs would report this extreme situation as simultaneous diaphragm and pipe blockage faults. The SUIN would not be able to distinguish effects as it receives the same messages from the FENs in both cases.



(a) New diaphragm



(b) Pin-hole diaphragm



(c) Large-hole diaphragm

Figure 9.11: Signals with partial pipe blockage fault

The author then devised the following monitoring strategy to counter this problem. Monitoring was started with an open vent hole and the pipe blockage fault was investigated first. The vent hole was then closed for a short time and the diaphragm fault was investigated. At the end of both tests the SUIN presented the fault status to any remote user. The vent hole was re-opened again and pipe blockage monitoring resumed. The diaphragm monitoring tests would be repeated periodically so that any slow deterioration could be detected over time. The two faults were tested under different conditions and this simplified the required isolation logic. Any flow reduction detected when the vent was open was interpreted as a pipe blockage fault. Table 9.3 summarizes SUIN monitoring decisions based on received messages from FENs. The control FEN always generated an ok signal to the SUIN and this was used only as an availability confirmation. Its part in the process monitoring regime was limited to generating event messages to the flow FEN so that pipe blockage detection is made when appropriate conditions prevail.

Chamber pressure	Inlet pressure	Flow signal	Control signal	SUIN decision
ok	ok	ok	ok	Normal conditions
ok	ok	error	ok	Pipe blockage fault
ok	error	X	ok	Low supply pressure
error	ok	X	ok	Diaphragm fault
error	error	X	ok	Diaphragm fault + Low supply pressure

Table 9.3: SUIN decision table (X = Don't care)

9.7 SUMMARY

Diaphragm condition deterioration significantly affects performance of industrial processes. The monitoring system was deployed to detect such deteriorations under normal process conditions. Chamber pressure, inlet pipe pressure and flow signals were monitored along with the control signal by four FENs and a SUIN. Pin holes were made in valve diaphragm to simulate condition deterioration. The monitoring system detected the deterioration before any significant process performance loss and thus provided fault prediction. A blockage in pipe was identified as another potential flow reducing problem

and fault symptoms were devised for it as well. The monitoring system was presented with both faults individually and it successfully isolated the two faults. Thus the distributed monitoring system based on 8-bit microcontrollers was proved useful in fault detection, isolation, and prediction for real time industrial problems.

Chapter 10

DISCUSSION

The research reported in this thesis made the following contributions towards knowledge:

- A novel sweeping filter technique was developed that enabled 8-bit microcontrollers to perform frequency analysis in real-time. Smart use of resources in this technique made it possible for 8-bit microcontrollers to process data, using techniques including frequency analysis, and to communicate the results.
- An effective methodology was developed for inter-node communication via a CAN bus where information was shared in real-time without overloading the network or the microcontrollers.
- A method was provided to implement a plug & play capability in the system without overloading the resources. The developed system automatically detected the available FENs. Although the partial unavailability of FENs impaired the system it was designed to continue to work although at a reduced functionality.
- Individual microcontrollers may be very limited in resources but much higher capabilities were achieved in the overall distributed system with their combined power. Effective process monitoring with fault detection, isolation, and prediction was achieved for a number of applications.
- A significant reduction in network load was achieved by processing data at the first hierarchy layer. Reductions will be of the order of giga bytes for applications involving analogue signals or frequency analysis.
- The results were provided on dynamic web pages via an embedded web server. Multiple remote users could access the results through computers, PDAs, and mobile phones simultaneously. Urgent alarm messages were communicated via SMS to the identified remote user's mobile phone.

- The system was implemented on easily installable compact circuit boards that were developed at a low cost. It was based on modular hardware and software units providing ease of deployment to new applications.

The strengths and weaknesses of the distributed hierarchical monitoring system were evaluated through its deployment for

- blockage and leakage fault detection and isolation in a batch process
- blockage fault detection in a continuous process
- fault isolation in a multi-loop (a batch and a continuous) processes
- tooth breakage detection in a machine tool
- diaphragm condition deterioration detection in an industrial valve
- fault isolation for flow reduction in an air flow process.

The system correctly detected and isolated the faults in all the test cases. The successful identification of fault level extents provided fault prediction and the opportunity to plan and schedule maintenance activities accordingly.

The investigations in this research were aimed at providing a cost effective generalised monitoring system that has the scope for wide deployment by SMEs. A review of various monitoring techniques and their applications was conducted and it was established that controller signals can provide an initial indication of a developing fault. Implementational issues were considered for various monitoring techniques in light of currently available technologies. The cost of PC based solutions was considered too high when all system constituent components (ADC cards, cabling) were included and a distributed network of 8-bit microcontrollers was deemed more suitable for the identified objectives. A hierarchical approach was taken towards individual situation assessment and each signal was evaluated by simple threshold checks for normal/abnormal behaviour. Any abnormality was reported to the second hierarchy layer where all first layer results were combined for total situation assessment. The combination of individual FEN results and their timings was used for fault detection and isolation. For conditions where no monitoring decision was possible at the first or second hierarchy layer it was proposed that the situation could be reported to the third layer for specialised processing and detailed analysis. The focus of this research was on the first and second layers which were expected to detect and isolate a large majority of faults. Computations at these layers

were restricted to simple but very effective methods because of the limited processing capabilities of 8-bit MCUs.

It was established by the literature review that no previously developed system provided monitoring results based solely on 8-bit microcontrollers. Such microcontrollers were previously mainly used as data acquisition devices and for command implementations with little intelligence. Any complete systems reported in the literature were limited to standalone applications where different MCUs did not collaborate with each other for total situation assessment. The author considers the research reported here as novel and significant because the 8-bit MCUs used combine their processed results intelligently to reach monitoring decisions. The decisions are also communicated to remote users over the Internet and mobile phones. It was a challenge to provide acquisition, processing, and communication tasks on a single microcontroller especially for analogue signals because of the magnitude of the data and processing required.

Compact circuit boards were developed by keeping the chip count low and not using any external memory. The limited memory resources of a microcontroller posed problems but efficient memory utilization and measures taken to reduce communication overheads provided a feasible solution. First level processing was performed at the acquisition nodes and only the results were communicated on a CAN bus. The use of 8-bit microcontrollers in this research reduced the cost significantly. The system developed in this research eliminated the need of data transmission on the CAN bus by indicating the normal/abnormal status of each parameter. Information was shared among the nodes and a decision was made on the basis of all available signal statuses.

Assembly language was used in the FENs to achieve high control over memory allocations and fast code executions. Minimal code size and fast execution were achieved. The complete control made available within the assembly language was preferred over the ease of using a high level language. Real-time operating systems are available for embedded devices but were not used for the FENs for similar reasons.

The situation in the SUIN was different as it involved complex tasks such as communicating via the Internet. The development of codes for the Internet protocols (TCP/IP, FTP, Telnet etc.) from scratch would be a very time consuming and tedious

task. Debugging the developed code would be another time consuming and intensive activity. It was therefore decided to utilise already developed and tested codes. Shareware codes which were available as Java APIs were used. An asynchronous message passing architecture was used for inter-node communication on the CAN bus. Code development becomes easier in an object oriented language under such an event-based environment. Java was thus a suitable choice for the SUIN programming. A real-time multi-tasking operating system was considered a necessity for the SUIN to synchronise its various activities and such a shareware operating system was used. The use of shareware codes would also help in reducing overall software cost when a system is made commercially available.

The system developed in this research provided useful results in a number of applications where time and frequency domain analysis was conducted on microcontrollers. The system detected deviations from nominal behaviour based on the established fault symptoms. The detection of fault severity provided the opportunity for failure prediction and the associated planning of a maintenance schedule. These factors would enhance process quality and productivity culminating in increased economic benefits and market advantages. In this effect, the developed system has a wide scope for future deployments in industry.

A major issue for any system is its ease of deployment to new applications. Easily deployable small-sized circuits were designed with modular software to make application porting easier. The developed system provided this ease of deployment in the sense that only a small fraction of code needed replacing for a specific application. A large part of the software for the FENs (assembly language) and SUIN (Java) remained unaffected with an application change. Small subroutines were developed covering various possible amendments and whole subroutines were replaced. The details of monitoring system's unaltered aspects and the application related alterations are as follows:

Unaltered aspects:

- The FEN (with new signal conditioning) and the SUIN hardware.
- Code architectures.
- Interrupt structure.

- Code for CAN bus access.
- Code for Internet access.

Alterations:

- Subroutine replacements in FENs were only required according to:
 - Signal format in hardware (analogue, pulse rate, etc.).
 - Pre-processing (raw value, running sum, trend etc.).
 - Threshold crossing criteria (low/high).
- Changes were required in SUIN decision making logic.
- A new SUIN web page design was required.

The requirement of a system study phase for every new application, no matter how similar it is to a previous application, is the tedious part of new deployments. A suitable processing method for each signal is determined by the engineer after careful analysis. This study is required for calibration, threshold establishment, and fault symptom location. It does however optimize the deployed system to that particular process implementation. Such optimization cannot be achieved through general process models and commissioning would be required for model-based approaches as well. The system study phase eliminates the requirement of mathematical models, which are, in themselves, difficult to obtain for real processes.

Thresholds were established through experimental data in this research as no mathematical model processing was deemed feasible with 8-bit microcontrollers. This provided the additional benefit that no prior process knowledge was required for system deployment. An engineer well versed with the monitoring system would be able to deploy the system on any application without detailed knowledge of the process. Any prior process knowledge can be incorporated for establishing fault symptoms but is not an essential requirement. Such benefits more than compensated for the requirement of a system study phase.

An advantage provided by the system is its ability to provide online data during the system study phase. Gathered data can be made available via the Internet to a remote developer who can analyse it from his office. No site visits are thus required by the

developer for signal analysis. Engineer's site visits are often costly and considerable money and time savings can be achieved by eliminating this requirement. The availability of data over the Internet also provides global access making signal analysis possible for plants operating in other countries as well. This international access can provide further economic benefits.

The monitoring system provided specific monitoring results from the first and second hierarchical layers thus reducing any data traffic on the Internet (after the system study phase). Without this feature much greater levels of data would need to be transported. The tooth breakage detection system, for example, produced data in excess of 50MB per day (100 samples per second per node) or 1.5GB per month. Many signals require higher sampling rates for their frequency analysis where data reduction would be even more significant. Further traffic reduction would be achieved for more elaborate processes with more signals involved. Local processing thus saved network costs and massively reduced any required PC based server-side processing.

The reported system was equipped with a plug & play capability and showed its robustness with a degree of adaptability to the partial availability of a FEN. Reduced inputs hamper the system functionality but it continues working and provides any possible monitoring decisions. This feature is useful for future expansion as well since any new FEN would be detected by the SUIN and incorporated in the monitoring scheme. A new decision table could be uploaded to the SUIN via FTP for this. The plug & play capability in the developed system was achieved using smart but simple means and no heavy protocols were used. This was again in line with the use of resource limited microcontrollers.

It is also possible to upload new codes to the SUIN via the Internet. The SUIN can provide new FEN codes to appropriate PIC microcontrollers via the CAN bus. Both the SUIN and the FEN implementations are based on flash memories that can be re-written after system installation. A new software version can be loaded in the monitoring system within minutes in this way without taking the system offline. This ensures that software upgrades can be used without affecting the normal process and without involving hardware. A possible example of software update requirement is the case where the system reports a previously unanticipated situation. New analysis may be conducted in

such cases generating new knowledge. The new knowledge would then be incorporated in the system and improvement would be achieved without any site visit or process shutdown.

The SUIN communicates with FENs to manage plug & play activities and also performs FDI duties. Providing web access to remote users loads the microcontroller and thus very small web pages were designed. The use of graphics in a web page provides a better quality of service but it was not deemed appropriate in the current stage of development. More powerful microcontrollers will emerge in future and enhanced graphics may then be used. The system may then provide overall equipment effectiveness, downtime/uptime, productivity, etc. to users in the form of graphs and pie charts. Such features were considered too resource intensive at this stage. A newly available version of the TINI is based on the DS80C400 microcontroller. This has a built-in Ethernet controller and provides faster Internet access at 100 Mbps. Such technological enhancements provide system improvement opportunities and must be availed.

Technology is bound to develop; new and more powerful microcontrollers are already emerging in the market. The architecture and methodology proposed in this research will remain valid with new developments. It will become possible to apply more elaborate processing techniques on the acquired data with new microcontrollers. The accurate diagnosis of more faults may then be determined and at earlier times. The proposed architecture would benefit from the future technologies rather than diminishing with time. Microchip dsPIC series microcontrollers are an example of improved microcontrollers becoming available since the start of this research. These 16-bit microcontrollers have an on-chip DSP engine. These microcontrollers are also supported with a DSP routines library from the manufacturer. Any future implementations may take advantage of such new developments within the framework of the proposed architecture.

A new trend in machine connectivity is introduced by the M2M concept. Embedded systems can now communicate with remote users on their mobile phones. A user may get reports from a monitoring system or issue commands (new set points, for example) to control remote processes. It is expected that M2M will gain rapid popularity in future for various monitoring and control applications. A very basic set up was tested in this research sending SMS only. Full mobile phone features can be used for two-way

communication using voice, data, and text services. The list of possible features includes automated voice alerts, emails, user subscriptions, report on demand, etc. It will again be a challenge to include such elaborated services using resource limited microcontrollers.

Any system connected to the Internet remains under security threats. Providing secure connection for small devices is a challenge because the established security measures are too resource intensive for current microcontrollers. The security in the developed system was provided via a username and password based login which is effectively a base level protection. Any communication between the monitoring system and remote logged-in user can be tapped into on the Internet. Encryption techniques used to protect resources against such tapings are too computationally extensive to be implemented on 8-bit microcontrollers. Better security measures are therefore required before the commercial use of the proposed system. System security has become more important because codes in the microcontrollers are now programmed in flash memories which can be altered as well. Flash memory provides ease of later software updates but also creates threat from hackers. The development of appropriate security measures were deemed outside the scope of this research but would be required for wide spread deployment of the system.

CAN is a time tested protocol under noisy environments and was selected for its reliable communication over a pair of wires. It provides physical and data link layer protocols and requires higher layer software to be built on it. Several higher layer protocols are available for CAN but were not used in this research. Such protocols cover many aspects that are not needed for process monitoring and their implementation would consume significant resources. A simplified approach was therefore taken in this research and only the required features were implemented. This light version of the application layer communication enabled the microcontrollers to meet acquisition, processing, and communication requirements simultaneously.

General purpose microcontrollers were used as front end nodes in this research. It was observed that a FEN did not utilize all available features in a general purpose MCU. Only a few external pins were used and most of the parallel ports remained unused. The most powerful PIC microcontroller (available at the start of this research) was used to avail the best possible performance but an actual implementation can be attained on a smaller sized MCU having a lower pin count. Similarly, no communication interface was required other

than CAN and other interfaces remained redundant. It may be considered that a microprocessor may be designed specifically for sensor applications that contain only the required features. The Silicon area vacated by these unused peripherals may be utilised to build additional on-chip memory, processing power, and/or ADC resolution.

These small microcontrollers may be built inside the sensor assemblies and a sensor would thus provide its signal intelligently and only on the CAN bus. Such a sensor with built-in decision-making power would convert the physical parameter directly into MCU-compatible signal and no additional signal conditioning circuits would be required. This will further reduce the overall system cost. The sensor market consists of billions of items and huge saving can be achieved in this way. This creates a big business opportunity and business worth billions of pounds. The effectiveness of CAN-only intelligent sensors has been proved in this research for monitoring applications. Further research is needed to ensure that control applications could also benefit from such sensors because a process controller would also be using the same sensors as the monitoring system. Large scale deployment of such sensors is envisaged once their suitability for control applications is confirmed.

CONCLUSIONS & FUTURE WORK

11.1 CONCLUSIONS

Investigations were performed in search of a generalised, widely deployable and low-cost monitoring system. Various possible approaches were considered and a distributed hierarchical monitoring system based on 8-bit microcontrollers at its first and second hierarchy layers was implemented. The system was deployed on a number of processes and insight was gained from the research. The following contributions towards knowledge were made through this work.

- Development of a generic monitoring system based on 8-bit microcontrollers.
- Development of a novel sweeping filter technique for frequency analysis.
- Implementing plug & play capability using a light protocol.
- Development of detailed message structure for network traffic reduction.
- The developed system's performance was evaluated for batch and continuous processes, for a machine tool application, and for on-line monitoring of an industrial valve.

The following conclusions were drawn from this research.

- Effective process monitoring with fault detection, isolation, and prediction can be achieved with distributed microcontroller-based system where decisions are based on a hierarchical integration of individual signal statuses.
- The current generation of 8-bit microcontrollers have enough processing and communication power to realize an effective monitoring system. The hierarchical approach enables the system to utilize PC based resources when needed.

- 8-bit microcontrollers can perform frequency analysis in real-time using programmable analogue filters. The developed sweeping filter technique has increased their effectiveness in a monitoring system.
- Plug and play feature can be implemented on 8-bit microcontrollers. This enables them to be combined in a modular and adaptable distributed system.
- Significant reduction in network load is achieved by processing data at the first hierarchy layer. The savings may easily go to the order of giga bytes for applications involving analogue signals or frequency analysis.
- Appropriate measures are required for online system security, especially for flash memory based systems. Current security algorithms are devised mainly for PC applications. This mind set need to be changed and improvement in small systems security is required.
- Embedded system's connection with the Internet and mobile phone networks provides better remote user interfaces. A whole new range of online embedded services can be expected in near future.

11.2 FUTURE WORK

The developed monitoring system was deployed on various process applications and it was shown to be able to successfully detect and isolate various faults. Further improvements can be achieved by incorporating the following features in the system.

Process signals were analysed with one simulated fault at a time. Various faults were isolated based on the knowledge gained through these experiments. The cumulative effects of multiple faults were not investigated. Several kinds of process parameter deterioration might be present at any given time and further study is required to detect and isolate simultaneous faults. This should be set against the fact that the monitoring system will identify individual faults and the sequence in which they occur – hence making multiple fault diagnosis a simpler process.

A very basic M2M functionality was provided in the system. Further work is required for fully integrated M2M. New CAN message types are required for communication with this specialised node. The M2M node may be communicating with FENs directly and further research is required to assess the usefulness of such communications in the process monitoring context.

The availability of new microcontrollers should be exploited. Future FENs may utilize dsPIC rather than PIC microcontrollers. Now available dsPIC MCUs provide the same features on a single chip but also provide enhanced processing capabilities due to the built-in DSP engine. It would be possible to implement digital filters in these 16-bit microcontrollers while keeping the acquisition and communication features intact. It would thus become possible to detect more faults on the first hierarchy layer. Similarly, a new version of TINI can be used as a SUIN for faster Internet access. The DS80C400 microcontroller can work up to 75 MHz clock and would thus provide much higher execution speed. The architecture provided in this research would thus provide even better results with future technologies.

The acquired sensor signal may be checked for correctness at its acquisition. The self validating sensors (SEVA) approach may be used where the acquired signal is compared with an anticipated (modelled) signal. The comparison result is subsequently used to rate the confidence level on the acquired signal. Validating sensor signals is a field in itself and is being researched separately. New developments in this field may be checked for any possible implementation on microcontrollers. The use of validation results, if possible, will provide additional reliability to the monitoring system.

REFERENCES

- Ahsan, Q. Development of a Low Cost Analog Signal Acquisition System. MSc Thesis (2002). Cardiff School of Engineering, Cardiff University, UK.
- Ahsan, Q., Amer, W., Grosvenor, R. I. and Prickett, P. W. Sweeping Filter Technique for Frequency Analysis. In proceedings of Quality, Reliability, and Maintenance (QRM) 2004, 5th International Conference, Oxford University, UK. Professional Engineering Publishing Ltd, Bury St Edmunds and London UK. pp 185-188. ISBN 1 86058 440 3.
- Al-Habaibeh, A., Whitby, D. R., Parkin, R. M., Jackson, M. R., Mansi, M. and Coy, J. The Development of an Internet-based Mechatronic System for Remote Diagnostic of Machinery using Embedded Sensors. International Conference on Mechatronics, 2003. pp 297-302.
- Alheraish, A. Design and Implementation of Home Automation System. IEEE Transactions on Consumer Electronics. Vol. 50, No. 4, November 2004, 1087-1092.
- Amadi-Echendu, J. E., Zhu, H. and Atherton, D. P. Development of Intelligent Flowmeters through Signal Processing. IFAC SICICA 92, Malaga, Spain, (May 1992), 23-28.
- Amer, W. Design of a PIC Microcontroller Based Analogue Acquisition and Processing System. MSc Thesis (2002). Cardiff School of Engineering, Cardiff University, UK.
- Andrews, J. D. and Dunnett, S. J. Event-Tree Analysis Using Binary Decision Diagrams. IEEE Transactions on Reliability, Vol. 49, No. 2, June 2000, 230-238.
- Au, Y. H. J., Kaewkongka, T., Harris, A., Rakowski, R. T. and Jones, B. E. Bearing Condition Monitoring based on the Inter-Arrival Time Distribution of Accoustic Emission Events – Theory. Quality, Reliability and Maintenance Conference (2004), pp 67-70.
- Axelson, J. Network Security for Small Systems. Circuit Cellar, Issue 172, November 2004. pp 62-69.
- Bentham, J. 2000. TCP/IP Lean: Web Servers for Embedded Systems. Group West Publishers, Berkeley, CA. ISBN 1-929629-11-7.
- Bezergianni, S. and Georgakis, C. Controller Performance Assessment based on Minimum and Open-Loop Output Variance. Control Engineering Practice, 8, 2000, 791-797.
- Biswas, G., Kapadia, R. and Yu, X. W. Combined Qualitative-Quantitative Steady-State Diagnosis of Continuous-Valued Systems. IEEE Transactions on System, MAN, and Cybernetics – Part A: Systems and Humans, Vol. 27, No. 2, March 1997.
- Bolic, M., Drndarevic, V. and Samardzic, B. Distributed Measurement and Control System Based on Microcontrollers with Automatic Program Generation. Sensors and Actuators A, 2001, 90, 215-221.
- Bonastre, A., Ors, R. and Peris, M. Distributed Expert Systems as a New Tool in Analytical Chemistry. Trends in Analytical Chemistry, 2001, 20(5), 263-271.
- Bucci, G. and Landi, C. A Distributed Measurement Architecture for Industrial Applications. IEEE Transactions on Instrumentation and Measurement. Vol. 52, No. 1, February 2003. pp 165-174.
- Burkett, R. J. and Thornhill, N. F. The Assessment of Performance of Multivariable Controllers. Seminar on Control Loop Performance Assessment, London, November 2002, 11/1-11/3.
- Bytronic International Ltd. Documentation for the Bytronic Process Control Unit (IBM version). The Courtyard, reddicap Trading Estate, Sutton Coldfield, West Midlands, B75 7BU, England.

- Clarke, D. W. Sensor, Actuator and Loop Validation. *Advances in Control Technology*, IEE Colloquium on Advances in Control Technology. 25th May 1999, London. pp 1/1 – 1/10.
- Dassanayake, H.P.B., Roberts, C. and Goodman, C.J. An Architecture for System-wide Fault Detection and Isolation. *Proceedings of the Institute of Mechanical Engineers*, 2001, 215(I), 37-46.
- Davey, A., Grosvenor, R., Morgan, P. and Prickett, P. Petri-net Based Machine Tool Failure and Diagnosis. In *Proceedings: COMADEM '96*, 16-18 July, Sheffield – UK, 1996, 723-731.
- De Frutos, J. A. and Giron-Sierra, J. M. Design of a Distributed System Architecture Including an Automatic Code Generator. *Microprocessors and Microsystems*, 2002, 26, 207-213.
- Desborough, L. and Harris, T. Performance Assessment Measures for Univariate Feedback Control. *The Canadian Journal of Chemical Engineering*, 70, 1992, 1186-1197.
- Desborough, L. and Harris, T. Performance Assessment Measures for Univariate Feedforward/Feedback Control. *The Canadian Journal of Chemical Engineering*, 71, 1993, 605-916.
- Desforges, M. J., Marjanovic, O., Lennox, B. and Sandoz, D. J. Prototype for Hierarchical Process Condition Monitoring. *Computing and Control Engineering Journal*, (Oct 2002), 254-258.
- Developer Guidelines: AT Commands. 3rd edition. 2005. Publication number EN/LZT 108 7729, R3B. Sony Ericsson Mobile Communications AB, SE-221 88 Lund, Sweden.
- Divan, D., Luckjiff, G. A., Brumsickle, W. E., Freeborg, J. and Bhadkamkar, A. A Grid Information Resource for Nationwide Real-Time Power Monitoring. *IEEE Transactions on Industry Applications*. Vol. 40, No. 2, March/April 2004. pp 699-705.
- Eady, F. TCP/IP Stack Solution: A Detailed Look at the CMX-MicroNet. *Circuit Cellar*, Issue 172, November 2004. pp 54-61.
- Ehrlich, J., Zerrouki, A. and Demssieux, N. Distributed Architecture for Data Acquisition: a Generic Model. In *Proceedings: IEEE Instrumentation and Measurement Technology Conference*. Ottawa – Canada, 19 - 21 May 1997, 1180-1185.
- Eisenreich, D., and Demuth, B. *Designing Embedded Internet Devices: A practical guide to hardware and software design using the TINI microcontroller*. Elsevier (2003). ISBN: 1-878707-98-1
- Ettaleb, L. *Control Loop Performance Assessment and Oscillation Detection*. PhD Thesis, 1999. Dept. of Electrical and Computer Engineering, Faculty of Applied Science, The University of British Columbia. Vancouver, Canada.
- Feng, X., Velinsky, S. A. and Hong, D. Integrating Embedded PC and Internet Technologies for Real-Time Control and Imaging. *IEEE/ASME Transactions on Mechatronics*. Vol. 7, No. 1, March 2002. pp 52-60.
- Fletcher, M., Austin, J. And Jackson, T. Distributed Aero-Engine Condition Monitoring and Diagnosis on the GRID: DAME. *Comadem 2004*. pp 185-194.
- Frankowiak, M. R. *Intelligent Distributed Process Monitoring and Management System*. PhD Thesis (2004). Cardiff School of Engineering, Cardiff University, UK.
- Frankowiak, M., Grosvenor, R. and Prickett, P. A Review of the Evolution of Microcontroller-based Machine and Process Monitoring. *International Journal of Machine Tools and Manufacture*, 45, 2005, 573-582.
- Frankowiak, M. R., Grosvenor, R. I., Prickett, P. W., Jennings, A. D. and Turner, J. R. Design of a PIC based Data Acquisition System for Process and Condition Monitoring. *Comadem 2001*, Manchester, pp 481-488. ISBN 0080440363.

- Fu, Y. and Dumont, G. A. On-Line Evaluation of Control Loop Performance. IEEE Control Applications Conference, 1995. pp 655-656.
- Goulding, P. R., Lennox, B., Sandoz, D. J., Smith, K. J. and Marjanovic, O. Fault Detection in Continuous Processes using Multivariate Statistical Methods. International Journal of Systems Science, 31 (11), 2000, 1459-1471.
- Grosvenor, R. I. and Prickett, P. W. Intelligent Process Monitoring and Management – Time for a Return to Sensor-Based Methods? In Proceedings of Sensors and their Applications, XII, September 2003. pp 497-502. IOP Publishing Ltd.
- Gustafsson, F. and Graebe, S. F. Closed-Loop Performance Monitoring in the Presence of System Changes and Disturbances. Automatica, Vol 34 (1998), No. 11, 1311-1326.
- Hagglund, T. Automatic Detection of Sluggish Control Loops. Control Engineering Practice, 7 (1999), 1505-1511.
- Hagglund, T. A. Control Loop Performance Monitor. Control Eng. Practice, Vol 3 (1995), No. 11, 1543-1551.
- Harris, T. J. Assessment of Control Loop Performance. Canadian Journal of Chemical Engineering, 67, 1989, 856-861.
- Harris, T. J. Observations and Thoughts on Controller Assessment and Performance Monitoring (2004). [Assessed on 11th Feb 2005] at <http://www.appsci.queensu.ca/tjh/publications/ControlSystems2004.pdf>.
- Harris, T. J., Boudreau, F. and Macgregor, J. F. Performance Assessment of Multivariable Feedback Controllers. Automatica, Vol. 32, No. 11, 1996, 1505-1518.
- Hartley, J. Field Based Systems, Asset Management and Advanced Diagnostics, Seminar on Control Loop Performance Assessment, London, November 2002, 10/1-10/6.
- Hawkins, F. Condition Monitoring in Defence – The Challenges Ahead. Comadem 2004, pp 25-35.
- HCF LIT 34. (1999). Application Guide. HART Communication Foundation, 9390 Research Boulevard, Suite I-350, Austin, Texas, 78759 USA.
- Higham, E. H. and Perovic, S. Predictive Maintenance of Pumps based on Signal Analysis of Pressure and Differential Pressure (Flow) Measurements. Transactions of the Institute of Measurement and Control 23, 4 (2001). pp 226-248.
- Hopkins, A. Virtual Instrumentation of a Process Rig, MSc Thesis, 2001, School of Engineering, Cardiff University, UK.
- Horch, A. Condition Monitoring of Control Loops. PhD Thesis (2000). School of Electrical Engineering, Royal Institute of Technology, Stockholm, Sweden. ISBN 91-7170-638-0.
- Hu, W., Starr, A.G. and Leung, A.Y.T. Operational Fault Diagnosis of Manufacturing Systems. Journal of Material Processing Technology, 133, 2003, 108-117.
- Huang, B., Shah, S. L. and Kwok, E. K. Good, Bad or Optimal? Performance Assessment of Multivariable processes. Automatica, Vol. 33, No. 6, 1997, 1175-1183.
- Insam, E. Interface Ethernet and Embedded Systems. Circuit Cellar, Issue 172, November 2004. pp 44-53.
- Isermann, R. Supervision, Fault-Detection and Fault-Diagnosis Methods – An Introduction. Control Engineering Practice, Vol. 5, No. 5, 1997, 639-652.
- Isermann, R. and Balle, P. Trends in the Application of Model-Based Fault Detection and Diagnosis of Technical Processes. Control Engineering Practice, Vol 5 (1997), No. 5, 709-719.

- Jennings, A. D., Kennedy, V. R., Prickett, P. W., Turner, J. R. and Grosvenor, R. I. (2001, A). A Distributed Data Processing System for Process and Condition Monitoring. 14th COMADEM, (2001), 375-381, ISBN 0080440363.
- Jennings, A. D., Prickett, P. W. and Grosvenor, R. I. (2001, B). Machine Tool Audit: Kondia B500 Milling Machine. 2001, School of Engineering, Cardiff University, UK.
- Johns, C. Machine Tool Axis Signals for Tool Breakage Monitoring. PhD Thesis (1998). Cardiff School of Engineering, Cardiff University, UK.
- Kaewkongka, T., Au, Y. H. J., Harris, A., Rakowski, R. T. and Jones, B. E. Bearing Condition Monitoring based on the Inter-Arrival Time Distribution of Acoustic Emission Events – Experiment. Quality, Reliability and Maintenance Conference (2004), pp 71-74.
- Kandasamy, N., Hayes, J. P. and Murray, B. T. Time Constrained Failure Diagnosis in Distributed Embedded Systems: Application to Actuator Diagnosis. IEEE Transactions on Parallel and Distributed Systems, Vol. 16, No. 3, March 2005, 258-270.
- Kempley, J. 1980. Valve Users Manual: A technical reference book on industrial valves for the control of fluids. British Valve Manufacturers Association. Mechanical Engineering Publications Ltd, London. ISBN: 0 85298 4286.
- Kendra, S. J. and Cinar, A. Controller Performance Assessment by Frequency Domain Techniques. J. Proc. Cont. Vol. 7, No. 3, 1997. pp 181-194.
- Kerkeni, I. T., Arantes, L. And Moalla, M. An Agent-Oriented Architecture for F.M.S. Control/Monitoring. IEEE Conference on Control Applications, CCA 2003, Istanbul, Turkey, June 2003. IEEE Society Press. Pp 1024-1028.
- Keyif, I., Kapucu, A. R., Durakbasa, T., Eldem, V., Gokmen, B., Wetherilt, J. and Stratton, D. MCM: A new Technology for Motor Condition Monitoring. Comadem 2004. pp 70-79.
- Kimmich, F., Schwarte, A. And Isermann, R. Fault Detection for Modern Diesel Engines using Signal- and Process model-based Methods. Control Engineering Practice, 13, 2005, 189-203.
- Lacoste, R. PIC Spectrum – Audio Spectrum Analyzer. Circuit Cellar, (1998), Issue 98.
- Lee, T. and Hsiung, P. Embedded Software Synthesis and Prototyping. IEEE Transactions on Consumer Electronics. Vol. 50, No. 1, February 2004, 386-392.
- Lennox, B., Montague, G. A., Hiden, H. G., Kornfeld, G. and Goulding, P. R. Process Monitoring of an Industrial Fed-Batch Fermentation. Biotechnology and Bioengineering, 74 (2), (2001), 125-135.
- Livani, M. A., Kaiser, J. and Jia, W. Scheduling Hard and Soft Real-Time Communication in a Controller Area Network. Control Engineering Practice, 7 (1999), 1515-1523.
- Lynch, C. B. and Dumont, G. A. Control Loop Performance Monitoring. IEEE Transaction on Control Systems Technology, Vol. 4, No. 2, March 1996. pp 185-192.
- Manders, J., Barford, L.A. and Biswas, G. An Approach for Fault Detection and Isolation in Dynamic Systems from Distributed Measurements. IEEE Transaction on Instrumentation and Measurement, 2002, 51(2), 235-240.
- Mittal, A., Manimaran, G. and Murthy, C. S. R. Dynamic Real-Time Channel Establishment in Multiple Access Bus Networks. Computer Communications, 26 (2003), Issue 2, 113-127.
- Mosca, E. & Agnoloni, T. Closed-Loop Monitoring for Early Detection of Performance Losses in Feedback Control Systems. Automatica, Vol 39, Issue 12 (2003), 2071-2084.

- Mounce, S. R., Khan, A., Wood, A. S., Day, A. J., Widdop, P. D. and Machell, J. Sensor-Fusion of Hydraulic Data for Burst Detection and Location in a Treated Water Distribution System. *Information Fusion*, 4, 2003, 217-229.
- Musciano, C. and Kennedy, B. *HTML & XHTML: The Definitive Guide*. 4th Edition. O'Reilly & Associates Inc. USA. pp 450. ISBN: 0-596-00026-X.
- Niemeyer, P. and Knudsen, J. *Learning Java*. First edition, 2000, O'Reilly & Associates Inc, Sebastopol, CA 95472. ISBN 1-56592-718-4
- Nieva, T. and Wegmann, A. A Conceptual Model for Remote Data Acquisition Systems. *Computers in Industry*, 2002, 47, 215-237.
- Patton, R. A Benchmark Study Approach to Fault Diagnosis of Industrial Process Control Systems. IEE Seminar on Control Loop Assessment and Diagnosis. 16th Jun 2005. pp 61-79.
- Paulonis, M. A. and Cox, J. W. A Practical Approach for Large-Scale Controller Performance Assessment, Diagnosis, and Improvement. *Journal of Process Control*, 13 (2003), 155-168.
- PIC 18FXX8 Data Sheet (2001). Microchip Technology Incorporated., USA.
- Plesnyaev, E. A. and Pazderin, A. V. Data Acquisition System Faults Detection. IEEE Conference on Control Applications, Istanbul, Turkey, June 2003. pp 1390-1394.
- Prickett, P. A Petri-net Based Machine Tool Maintenance Management System. *Industrial Management and Data Systems*, 97(4), 1997, 143-149.
- Proakis, J. G. and Manolakis, D. G. (1996). *Digital Signal Processing, principles, algorithms and applications*, 3rd ed, Prentice Hall Inc. ISBN 0-13-394289-9.
- Rappaport, T. S. 2002. *Wireless Communications: Principles and Practice*. Second edition. Prentice Hall Inc. USA. ISBN 0-13-042232-0.
- Raaphorst, A.G.T, Netten, B.D. and Vingerhoeds, R.A. Automated Fault-Tree Generation for Operational Fault Diagnosis. In proceeding: *Electric Railways in a United Europe*, IEE, 27-30 March, 1995, 173-177.
- Rengaswamy, R. Hagglund, T and Venkatasubramanian, V. A. Qualitative Shape Analysis Formalism for Monitoring Control Loop Performance. *Engineering Application of Artificial Intelligence*, 14 (2001), 23-33.
- Ruiz, M., Colomer, J., Colprim, J. and Melendez, J. Multivariate Statistical Process Control for Situation Assessment of a Sequencing Batch Reactor. *Control Conference 2004*, University of Bath, UK. ID-115.
- Schafer, J and Cinar, A. Multivariable MPC System Performance Assessment, Monitoring and Diagnosis. *Journal of Process Control*, 14 (2004), 113-129.
- Sharif, M. A. M. *Application of Intelligent Instrumentation in Process Plant Condition Monitoring and Fault Diagnosis*. PhD Thesis (1999). Cardiff School of Engineering, Cardiff University, UK.
- SIMATIC (1998). SIMATIC S5/PC/505 Automation Systems. Siemens Catalog ST50.
- Soderholm, P. and Parida, A. Health Management of Complex Technical Systems. *Comadem 2004*, 214-221.
- Stallings, W. 2004. *Computer Networking with Internet Protocols and Technology*. Pearson Education Inc. NJ. ISBN 0-13-191155-4
- Stipanicev, D. and Marasovic, J. Networked Embedded Greenhouse Monitoring and Control. IEEE Conference on Control Applications, Istanbul, Turkey, June 2003. pp 1350-1355.

- Studzinski, J. Application of Monitoring Technologies in Environmental Engineering. Quality, Reliability and Maintenance Conference 2004. pp 43-46.
- Suzudo, T., Nabeshima, K. and Takizawa, H. Software Integration for monitoring Systems with High Flexibility. Progress in Nuclear Energy, Vol 43 (2003), No. 1-4, 405-411.
- Tanenbaum, A. S. 1996. Computer Networks. 3rd edition. Prentice Hall Inc. NJ. ISBN: 0-13-394248-1.
- Tanenbaum, A. S. and Steen, M. V. 2002. Distributed Systems: Principles and Paradigms. Prentice Hall Inc. NJ. ISBN 0-13-088893-1
- Tansel, I. N., Arkan, T. T., Bao, W. Y., Mahendrakar, N., Shisler, B., Smith, D. And McCool, M. Tool Wear Estimation in Micro-Machining. Part I: Tool Usage-Cutting Force Relationship. International Journal of Machine Tools and Manufacture, 40,2000,599-608.
- Thornhill, N. F., Cox, J. W. and Paulonis, M. A. (2003, A). Diagnosis of Plant-wide Oscillation through Data-driven Analysis and Process Understanding. Control Engineering Practice, 11, 2003, 1481-1490.
- Thornhill, N. F., Huang, B. and Zhang, H. (2003, B). Detection of Multiple Oscillations in Control Loops. Journal of Process Control, 13, 2003, 91-100.
- Thornhill, N. F., Shah, S. L. and Huang, B. Detection of Distributed Oscillations and Root-Cause Diagnosis. CHEMFAS4, Cheju Island, Korea. 7-8 July 2001.
- Tokatli, F., Cinar, A. and Schlessor, J. E. HACCP with Multivariate Process Monitoring and Fault Diagnosis Techniques: Application to a Food Pasteurization Process. Food Control, 16, 2005, 411-422.
- Trenchard, A., Desborough, L. and Miller, R. Managing Control Systems as Assets. IEE seminar on Control Loop Performance Assessment, 26th Nov 2002, London, 1/1-1/3.
- Tullis, J. P. Hydraulics of Pipelines: Pumps, Valves, Cavitation, Transients. Wiley-Interscience Publication, USA. ISBN: 0-471-83285-5.
- Turner, J. R., Jennings, A. D., Prickett, P. W. and Grosvenor, R. I. The design and Implementation of a Data Acquisition and Control System using Fieldbus Technologies. Comadem 2001, Manchester. Pp 391-398. ISBN 0080440363.
- Tyler, M. and Morari, M. Performance Assessment for unstable and nonminimum-phase systems. In IFAC Workshop on On-line Fault Detection and Supervision in the Chemical Process Industries. Newcastle upon Tyne, England. 1995.
- Tyler, M. and Morari, M. Performance Monitoring of Control Systems using Likelyhood Methods. Automatica, 32 (8), 1996, 1145-1162.
- Valdastri, P., Menciassi, A., Arena, A., Caccamo, C. and Dario, P. An Implantable Telemetry Platform System for In Vivo Monitoring of Physiological Parameters. IEEE Transactions on Information Technology in Biomedicine. Vol. 8, No. 3, September 2004. pp 271-278.
- Venkatasubramanian, V., Rengaswamy, R., Yin, K. and Kavuri, S. N. A Review of Process Fault Detection and Diagnosis. Part I: Quantitative Model-based Methods. Computers and Chemical Engineering, 27, 2003, 293-311.
- Venkatasubramanian, V., Rengaswamy, R., and Kavuri, S. N. A Review of Process Fault Detection and Diagnosis. Part II: Qualitative Models and Search Strategies. Computers and Chemical Engineering, 27, 2003, 313-326.
- Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. N. and Yin, K. A Review of Process Fault Detection and Diagnosis. Part III: Process History based Methods. Computers and Chemical Engineering, 27, 2003, 327-346.

Ward-Smith, A. J. Internal Fluid Flow: The fluid dynamics of flow in pipes and ducts. 1980. Clarendon press, Oxford. ISBN 0 19 856325 6.

Werneck, M. M. and Abrantes, A. C. S. Fiber-Optic-based Current and Voltage Measuring System for High-Voltage Distribution Lines. IEEE Transactions on Power Delivery. Vol. 19, No. 3, July 2004. pp 947-951.

Xia, C. and Howell, J. Loop Status Monitoring and Fault Localisation. Journal of Process Control, 13 (2003). 679-691.

Yang, M., Manabe, K., Hayashi, K., Miyazaki, M. and Aikawa, N. (2003,A). Data Fusion of Distributed AE Sensors for the Detection of Friction Sources during Press Forming. Journal of Materials Processing Technology, 139, 2003, 368-372.

Yang, S. H., Chen, X. and Alty, J. L. (2003, B). Design Issues and Implementation of Internet-Based Process Control Systems. Control Engineering Practice, 11, 2003, 709-720.

Yang, H. and Eagleson, R. (2003). Design and Implementation of an Internet-Based Embedded Control System. IEEE Conference on Control Applications, Vol 1, 23-25 Jun 2003.

WEB REFERENCES

Apple Computers, Using mod_ssl on Mac OS X [WWW]
<URL:<http://developer.apple.com/internet/serverside/modssl.html>> [Accessed 10 Nov 2005]

Aspentech. Aspen Watch [WWW]
<URL:<http://www.aspentech.com/includes/product.cfm?ProductID=87>> [Accessed on 1st May 2005]

Automation World. News [WWW]
<URL:<http://www.automationworld.com/articles/Departments/211.htm>> [Accessed 14 March 2003].

CAN in Automation: CAN Dictionary, [WWW]
<URL:http://www.can-cia.org/can/can_dictionary2.pdf> [Accessed 27 Nov 2002]

CAN in Automation: Controller Area Network (CAN) – Protocol, [WWW]
<URL:<http://www.can-cia.org/can/protocol>> [Accessed 27 Nov 2002]

CAN in Automation: Home, [WWW]
<URL:<http://www.can-cia.org>> [Accessed 27 Nov 2002]

Cisco Systems, Documentation [WWW]
<URL:http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ethernet.htm#wp1020549> [Accessed 3 Apr 2005]

Cunningham and Cunningham Inc. Microsoft Internet Explorer [WWW]
<URL:<http://c2.com/cgi/wiki?MicrosoftInternetExplorer>> [Accessed on 9th May 2004].

Dallas Semiconductor, 80C390 Dual CAN High-Speed Microprocessor data sheet, DS80C390.pdf [Accessed 8 May 2003]

Electronics Talk. PIC micro tops 8bit market. [WWW]
<URL:www.electronicstalk.com/news/ari/ari145.html> [Accessed on 19th November 2003].

Embedded Star. Microchip Rolls Out Four Billionth PIC Microcontroller. [WWW]
<URL:<http://www.embeddedstar.com/press/content/2005/9/embedded18912.html>> [Accessed on 26th December 2005].

Emerson Process Management. Customer Proven – Chemical [WWW]
<URL:http://plantweb.emersonprocess.com/Customer/Chemical_index.asp> [Accessed 7 June 2003].

Emerson Process Management, Plant Web [WWW]
<URL:http://plantweb.emersonprocess.com/Whatis_PlantWeb/index.asp> [Accessed on 1st May 2005]

Emerson Process Management, Results [WWW]
<URL:<http://plantweb.emersonprocess.com/home/index.asp>> [Accessed on 1st May 2005]

Honeywell AlarmScout [WWW]
<URL:<http://www.loopscout.com/alarmscout>> [Accessed on 2nd May 2005]

Honeywell LoopScout [WWW]
<URL:<http://www.loopscout.com>> [Accessed on 2nd May 2005]

Honeywell LoopScout Overview [WWW]
<URL:http://www.loopscout.com/Info/LSOverview_01Jan03_42.pdf> [Accessed on 2nd May 2005]

Honeywell Process Solutions [WWW]
<URL:hpsweb.honeywell.com/Cultures/en-US/Products/AssetApplications/AssetManagement/LoopScout/default.htm> [Accessed on 2nd May 2005]

IPMM group, Cardiff University [WWW]
<URL:www.processwatch.cf.ac.uk> [Accessed on 11th April 2005].

KVASER Advanced CAN Solutions, The CAN Protocol [WWW]
<URL:<http://www.kvaser.se/can/protocol/index.htm>> [Accessed 5 Dec 2002]

Leroy Davis, CAN Bus [WWW]
<URL:http://www.interfacebus.com/Design_Connector_CAN.html> [Accessed 8 Dec 2002]

Matrikon [WWW]
<URL:www.matrikon.com> [Accessed on 3rd May 2005]

Matrikon ProcessDoctor [WWW]
<URL:http://www.matrikon.com/products/processdoc/success_stories.asp> [Accessed 15 Mar 2005]

Mattec Corporation [WWW]
<URL:www.mattec.com> [Accessed on 30th April 2005]

Mattec Corporation ProHelp [WWW]
<www.mattec.com/service/updates/phepm620.htm> [Accessed on 30th April 2005]

Mattec Corporation THE-MAN-A-ger [WWW]
<URL:www.mattec.com/products/manager/index.htm> [Accessed on 30th April 2005]

MAX263-MAX268 Data sheet. Maxim Integrated Products. [WWW]
<URL:http://www.maxim-ic.com/quick_view2.cfm/qv_pk/1186> [Accessed on 12th June 2003].

Microchip Technology Inc. [WWW]
<URL:<http://www.microchip.com>> [Accessed on 5th October 2002].

Microchip Technology Inc. Fischer, R. L. and Burch, J. AN214. [WWW]
<URL:<http://ww1.microchip.com/downloads/en/AppNotes/00214a.pdf>> [Accessed on 19th November 2005]

NOAA, Glossary of Section 508 Terms [WWW]
<URL:www.cio.noaa.gov/itmanagement/508_Glossary.html> [Accessed 15 Dec 2005]

Oxford English Dictionary. [WWW]
<URL:http://dictionary.oed.com/cgi/entry/00314100?query_type=word&queryword=monitor&first=1&max_to_show=10&sort_type=alpha&search_id=mwqt-cfQU2Y-3504&result_place=3> [Accessed on 8th April 2005].

Palacherla, A. Implementation of Fast Fourier Transforms. Microchip Application Notes AN542 (1997). [WWW]
<URL:http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2048> [Accessed on 13th May 2003].

PHYTEC America, CAN TECHNOLOGY [WWW]
<URL:http://www.sfe-dcs.com/CAN/CAN_Nets.html> [Accessed 11 Dec 2002]

Ramu, B. K. A. Implementing FIR and IIR Digital Filters using PIC18 Microcontrollers. Microchip Application Notes AN852 (2002). [WWW].
<URL:http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2048&fragment6_NextRow=201> [Accessed on 20th May 2003].

Relx Software. What is Event Tree Analysis [WWW]
<URL:<http://www.relexsoftware.com/resources/eventtree.asp>> [Accessed on 10th April 2005].

Relx Software. What is Fault Tree Analysis [WWW]
<URL:<http://www.relexsoftware.com/resources/faulttree.asp>> [accessed on 10th April 2005].

Robert Bosch GmbH, Controller Area Network [WWW]
<URL:<http://www.semiconductors.bosch.de/de/20/can/index.asp>> [Accessed 28 Nov 2002]

SoftSwitching Technologies [WWW]
<URL:www.softswitch.com/igrid-general.htm> [Accessed 29th March 2005]

Sony Ericsson. GR47/GR48 [WWW]
<URL:http://www.sonyericsson.com/spg.jsp?cc=gb&lc=en&ver=4002&template=pp1_1&zone=pp&lm=pp1&pid=10086> [Accessed on 22nd Nov 2005].

Sony Ericsson. Where to Buy [WWW]
<URL:http://www.sonyericsson.com/spg.jsp?cc=gb&lc=en&ver=4002&template=ph1_2&zone=ph&pid=10086> [Accessed on 29th Nov 2005].

USB Developer web site. [WWW]
<URL:<http://www.usbdeveloper.com/GSMPage/gsmpage.htm>> [Accessed on 10th Dec 2005].

W3schools: HTML, Introduction to HTML. [WWW]
<URL:http://www.w3schools.com/html/html_intro.asp> [Accessed 18 May 2004]

W3schools: TCP/IP, Introduction to TCP/IP. [WWW]
<URL:http://www.w3schools.com/tcpip/tcpip_intro.asp> [Accessed 7 May 2004]

W3schools: XML, XML Tutorial. [WWW]
<URL:<http://www.w3schools.com/xml/default.asp>> [Accessed 21 Jul 2004]

Walchem Corporation [WWW]
<URL:www.walchem.com> [Accessed on 5th May 2005]

Walchem Corporation WebAlert [WWW]
<URL:www.walchem.com/walchem/literature/Brochures/webalert.pdf> [Accessed on 5th May 2005]

APPENDIX A

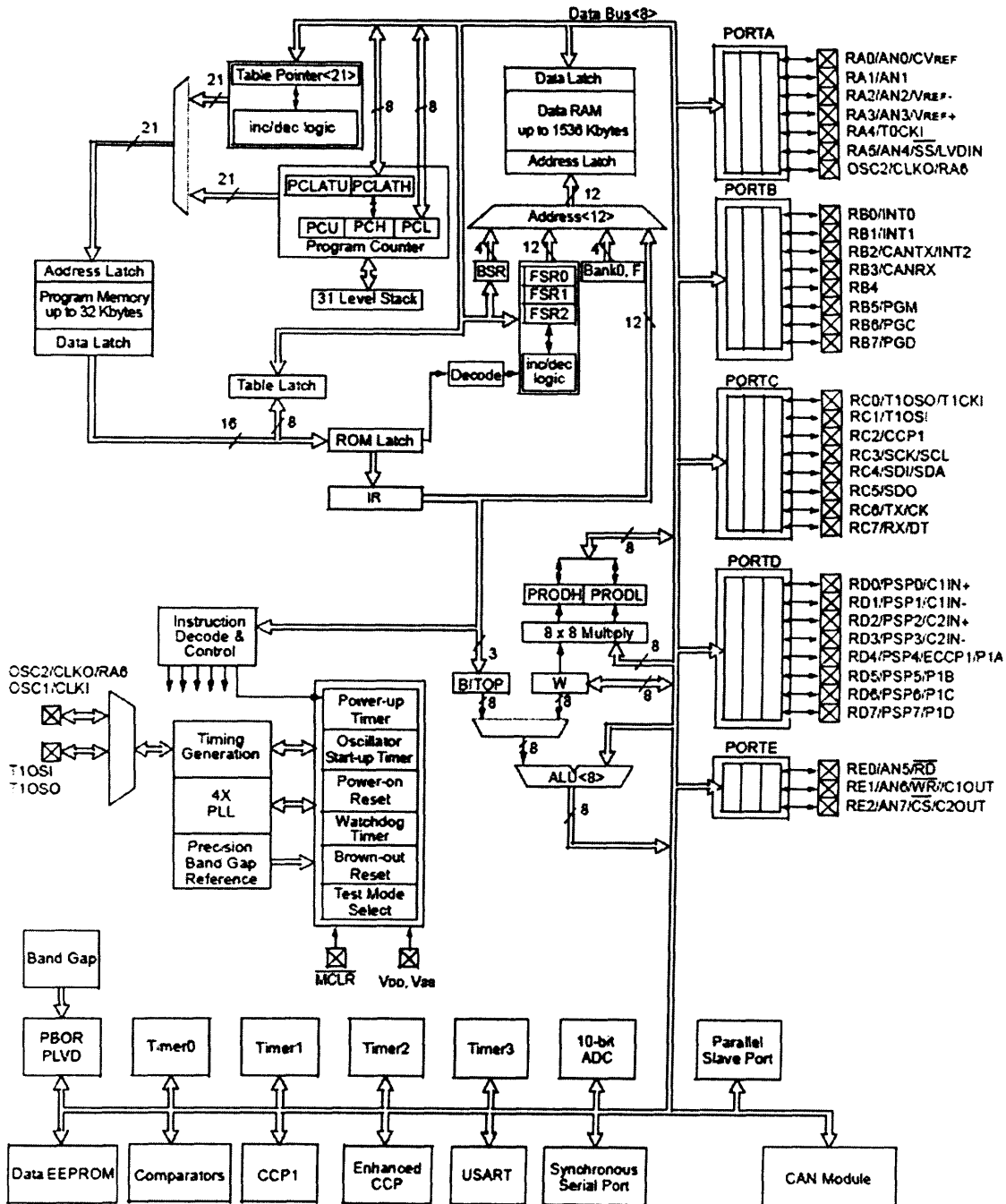
Transducers and sensing techniques (Frankowiak, 2004)

Physical phenomena	Measurement method	Sensing technique
Pressure	<ul style="list-style-type: none"> • Strain gauges • Variable capacitance • Linear var. diff. transformer • Piezo-electric effect 	<ul style="list-style-type: none"> • Pressure applied to the strain gauge, resulting in a resistance change; • Capacitance changes due to movement of a dielectric caused by pressure; • Transformer core moves due to pressure; • Output voltage resulting from applied pressure.
Temperature	<ul style="list-style-type: none"> • Thermocouple • Thermistor • Pyrometer 	<ul style="list-style-type: none"> • Electromotive force due to dissimilar metallic junctions; • Resistance variation due to temperature changes; • Heat wavelength radiation.
Flow	<ul style="list-style-type: none"> • Orifice plate • Venturi tube • Pilot tube • Turbine • Magnetic • Ultrasonic 	<ul style="list-style-type: none"> • Differential pressure due to restriction in the flow area; • Diff. pressure due to smooth and gradual reduction in tube diameter; • Differential pressure between static pressure and fluid flow; • Turbine rotor generates a electrical signal proportional to the flow rate; • Changes in the inductive voltage in a coil due to flow rate variations; • Measurement of acoustic wavelength changes due to flow rate variations.
Level	<ul style="list-style-type: none"> • ON/OFF switches • Continuous level 	<ul style="list-style-type: none"> • Beam breaking, capacitance, conductivity and float type level switches; • Capacitance (dielectric variation), differential pressure (level column), ultrasonic (wavelength reflection) and radioactive (absorbed radiation).
Displacement	<ul style="list-style-type: none"> • Angular and linear 	<ul style="list-style-type: none"> • Potentiometers, capacitance (parallel metal plates), inductive coil (permeable core), pulse counting, encoders and ON/OFF switches.
Velocity	<ul style="list-style-type: none"> • Linear • Angular 	<ul style="list-style-type: none"> • Time measurement based on pulse sensing; • Pulse sensing, electro-mechanical and digital tacho-generators.
Vibration	<ul style="list-style-type: none"> • Magnetic 	<ul style="list-style-type: none"> • Permanent magnet within a coil field, generating electrical signal.
Acceleration	<ul style="list-style-type: none"> • Strain gauges • Piezo-electric crystal 	<ul style="list-style-type: none"> • Changes in resistance due to applied forces; • Voltage variations due to strain in the crystal.
Force	<ul style="list-style-type: none"> • Weight • Force/torque 	<ul style="list-style-type: none"> • Load cells based on strain gauges principle; • Strain gauge and magnetic permeability changes due to tension variation.

APPENDIX B

PIC 18F458 Microcontroller

Schematic Block Diagram



Microcontroller Features

High-Performance RISC CPU:

- Linear program memory addressing up to 2 Mbytes
- Linear data memory addressing to 4 Kbytes
- Up to 10 MIPS operation
- DC – 40 MHz clock input
- 4 MHz-10 MHz oscillator/clock input with PLL active
- 16-bit wide instructions, 8-bit wide data path
- Priority levels for interrupts
- 8 x 8 Single-Cycle Hardware Multiplier

Peripheral Features:

- High current sink/source 25 mA/25 mA
- Three external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter with 8-bit programmable prescaler
- Timer1 module: 16-bit timer/counter
- Timer2 module: 8-bit timer/counter with 8-bit period register (time base for PWM)
- Timer3 module: 16-bit timer/counter
- Secondary oscillator clock option – Timer1/Timer3
- Capture/Compare/PWM (CCP) modules; CCP pins can be configured as:
 - Capture input: 16-bit, max resolution 6.25 ns
 - Compare: 16-bit, max resolution 100 ns (Tcy)
 - PWM output: PWM resolution is 1 to 10-bit
Max. PWM freq. @:8-bit resolution = 156 kHz
10-bit resolution = 39 kHz
- Enhanced CCP module which has all the features of the standard CCP module, but also has the following features for advanced motor control:
 - 1, 2 or 4 PWM outputs
 - Selectable PWM polarity
 - Programmable PWM dead time
- Master Synchronous Serial Port (MSSP) with two modes of operation:
 - 3-wire SPI™ (Supports all 4 SPI modes)
 - I²C™ Master and Slave mode
- Addressable USART module:
 - Supports interrupt-on-address bit

Advanced Analog Features:

- 10-bit, up to 8-channel Analog-to-Digital Converter module (A/D) with:
 - Conversion available during Sleep
 - Up to 8 channels available
- Analog Comparator module:
 - Programmable input and output multiplexing
- Comparator Voltage Reference module
- Programmable Low-Voltage Detection (LVD) module:
 - Supports interrupt-on-Low-Voltage Detection
- Programmable Brown-out Reset (BOR)

CAN bus Module Features:

- Complies with ISO CAN Conformance Test
- Message bit rates up to 1 Mbps
- Conforms to CAN 2.0B Active Spec with:
 - 29-bit Identifier Fields
 - 8-byte message length
 - 3 Transmit Message Buffers with prioritization
 - 2 Receive Message Buffers
 - 6 full, 29-bit Acceptance Filters
 - Prioritization of Acceptance Filters
 - Multiple Receive Buffers for High Priority Messages to prevent loss due to overflow
 - Advanced Error Management Features

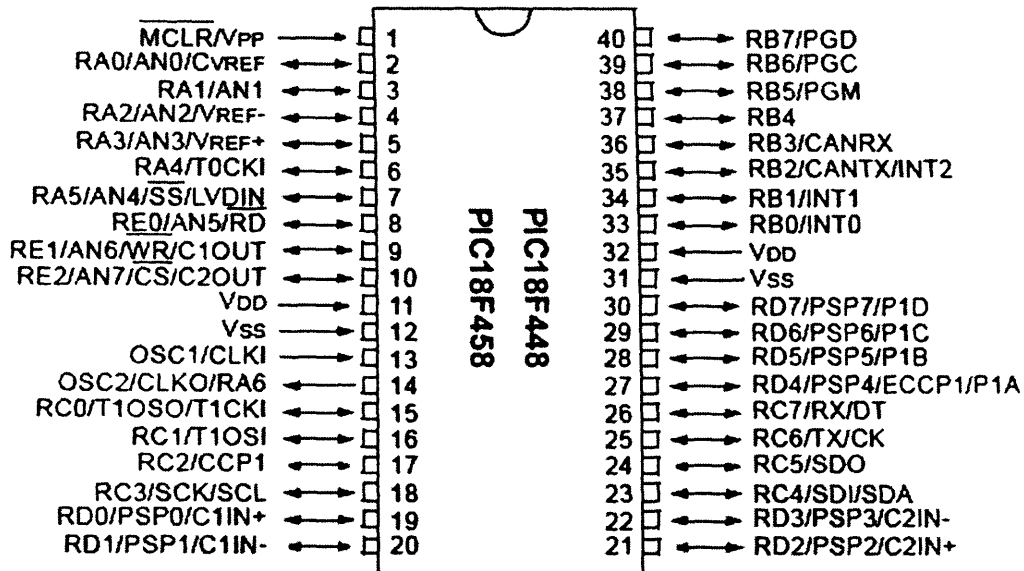
Special Microcontroller Features:

- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator
- Programmable code protection
- Power-saving Sleep mode
- Selectable oscillator options, including:
 - 4x Phase Lock Loop (PLL) of primary oscillator
 - Secondary Oscillator (32 kHz) clock input
- In-Circuit Serial Programming™ (ICSP™) via two pins

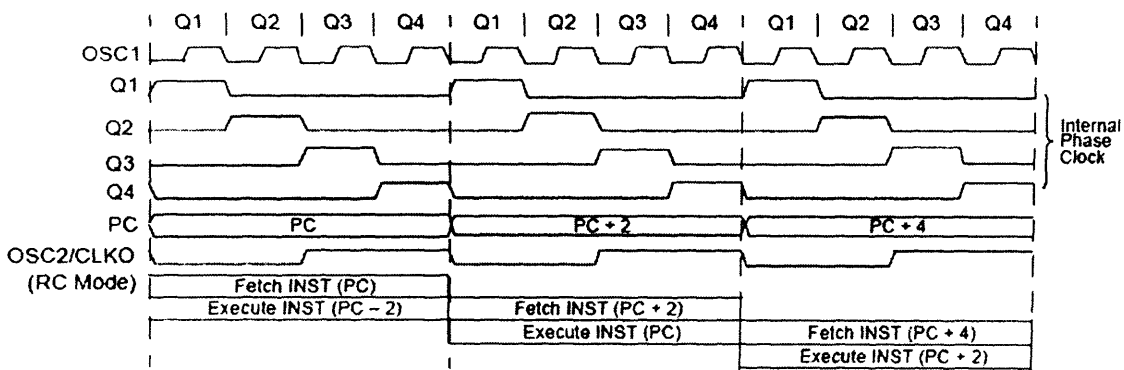
Flash Technology:

- Low-power, high-speed Enhanced Flash technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Industrial and Extended temperature ranges

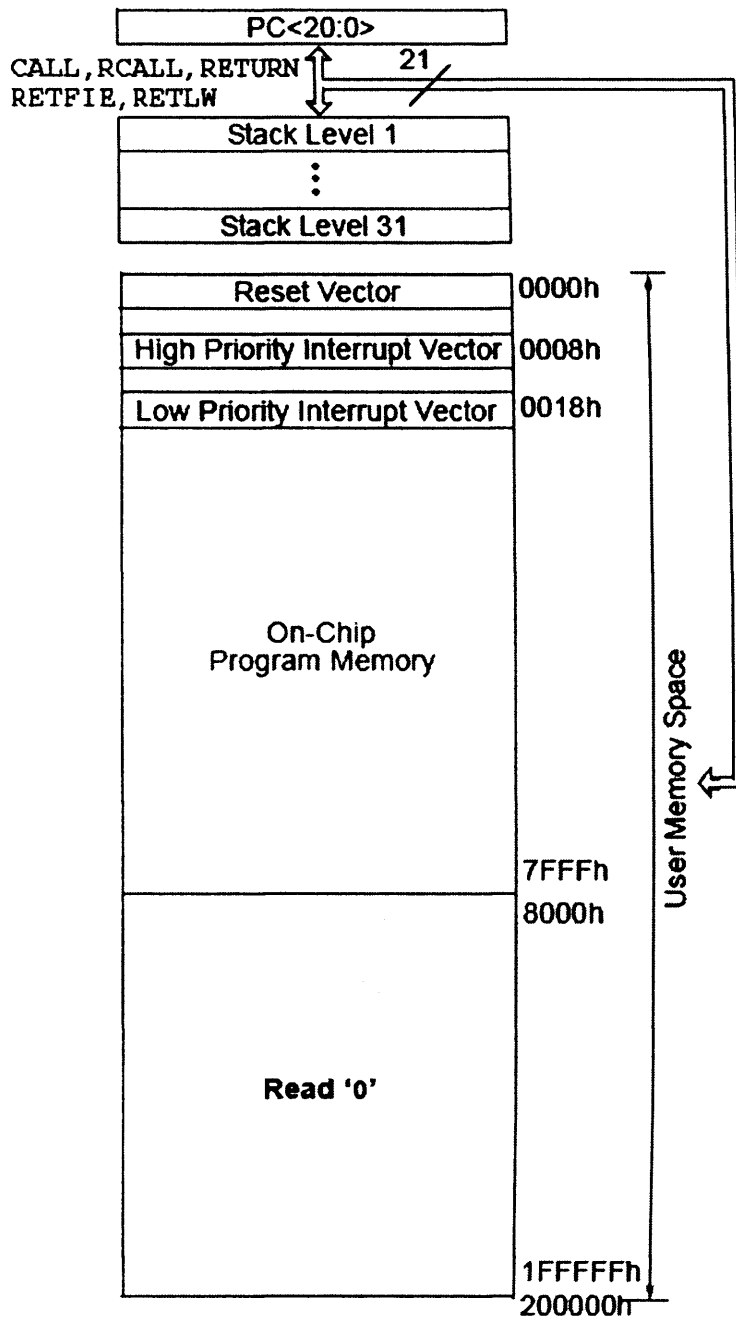
Pin Diagram



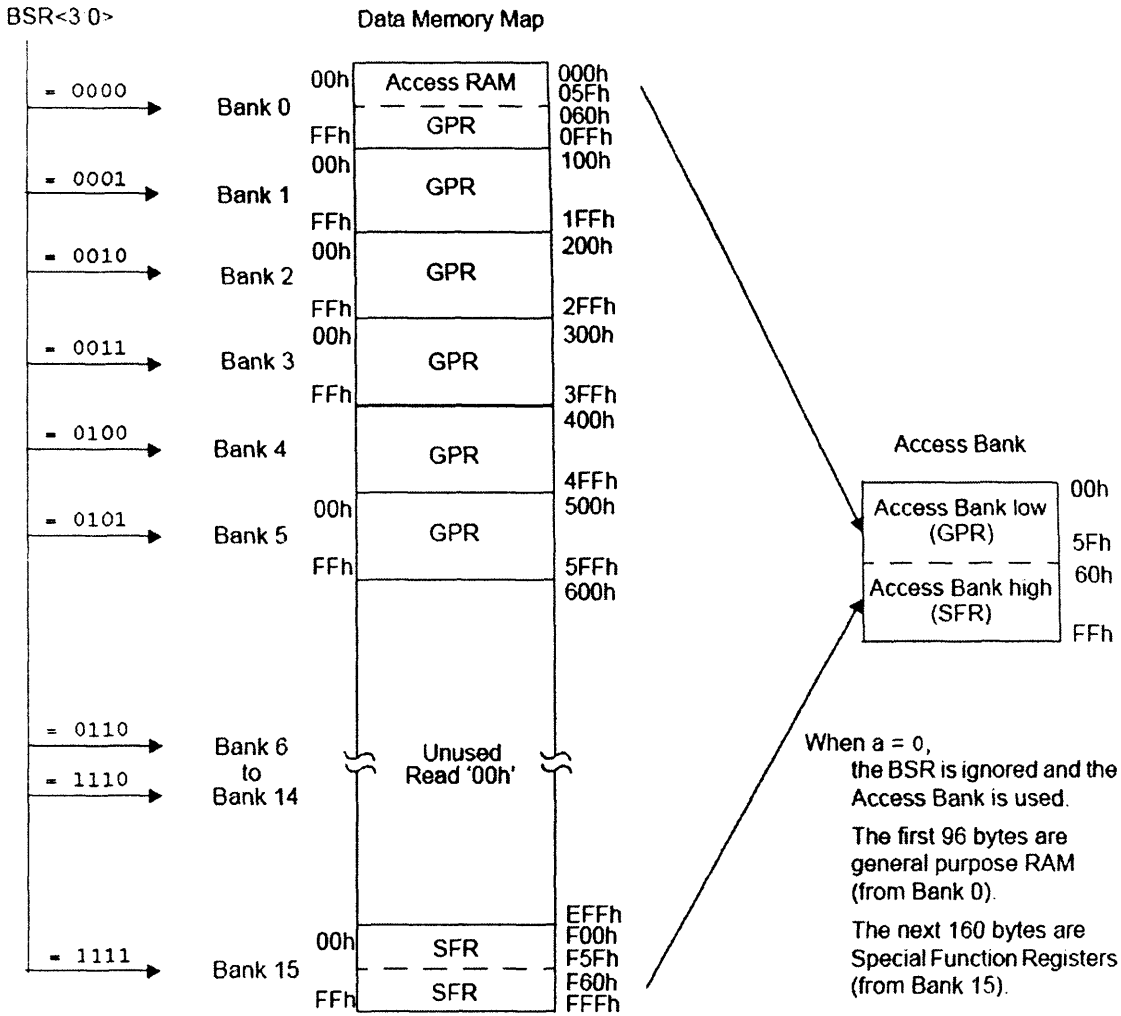
Pipelined Execution of Code



Program Memory Map



Data Memory Map

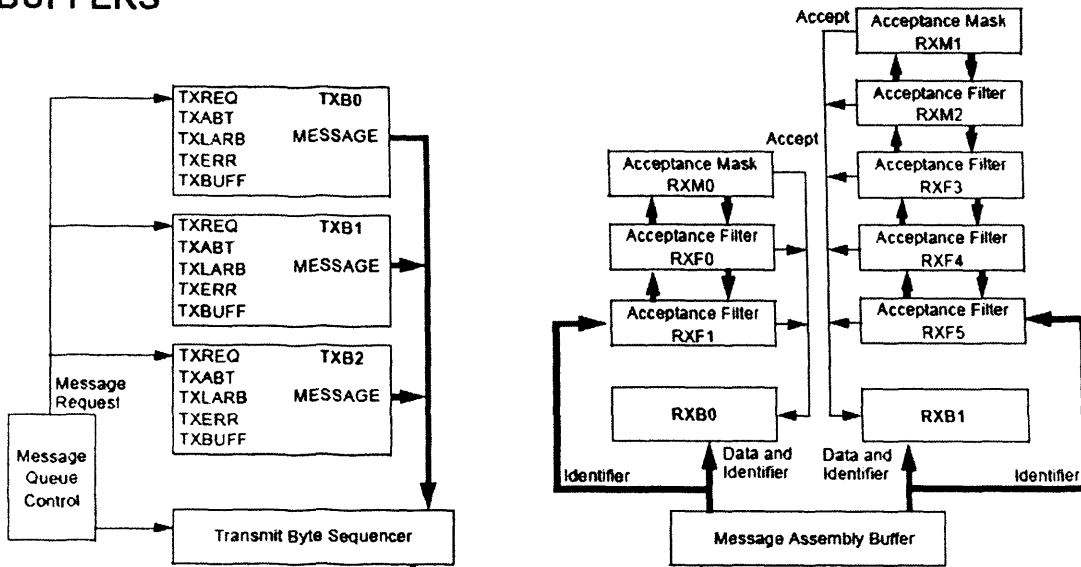


When a = 0, the BSR is ignored and the Access Bank is used. The first 96 bytes are general purpose RAM (from Bank 0). The next 160 bytes are Special Function Registers (from Bank 15).

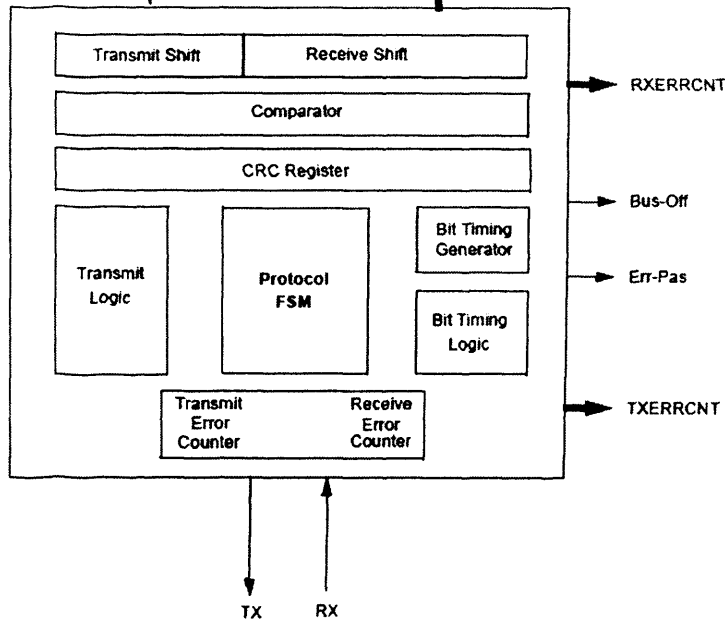
When a = 1, the BSR is used to specify the RAM location that the instruction uses.

CAN Buffers and Potocol Engine

BUFFERS



PROTOCOL ENGINE

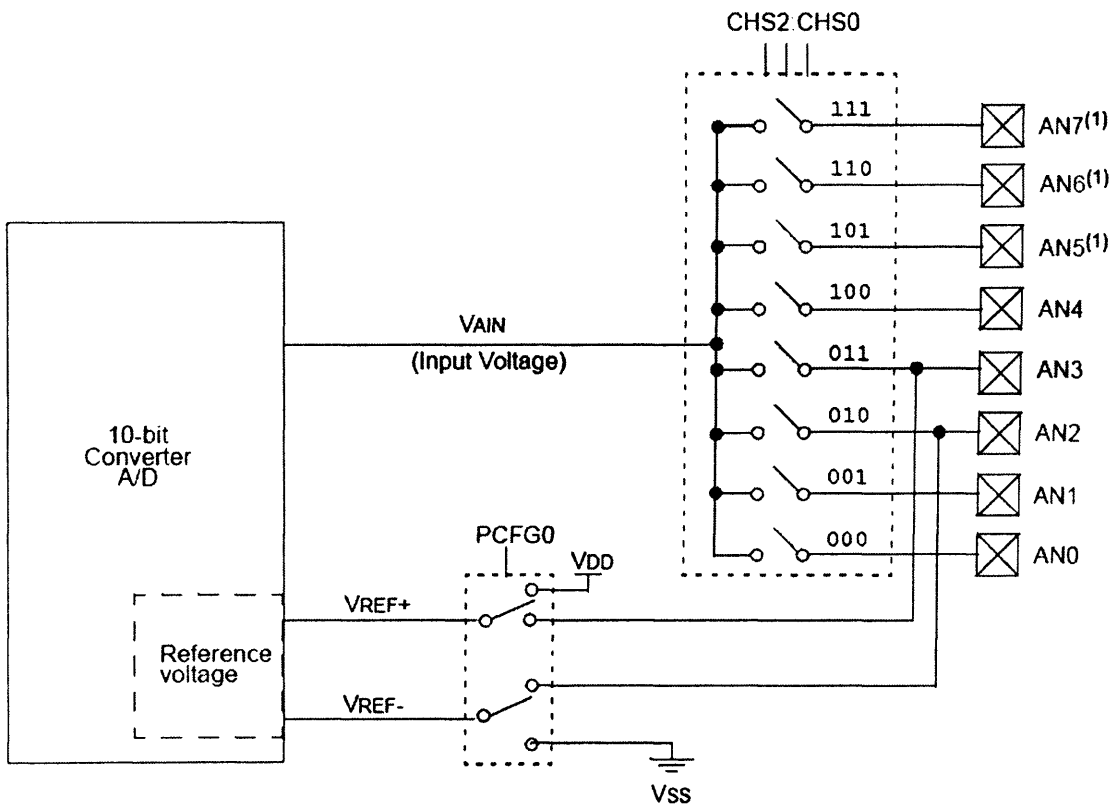


CAN Filter/Mask Truth Table

Mask bit n	Filter bit n	Message Identifier bit n001	Accept or Reject bit n
0	x	x	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

Legend: x = don't care

A/D Block Diagram

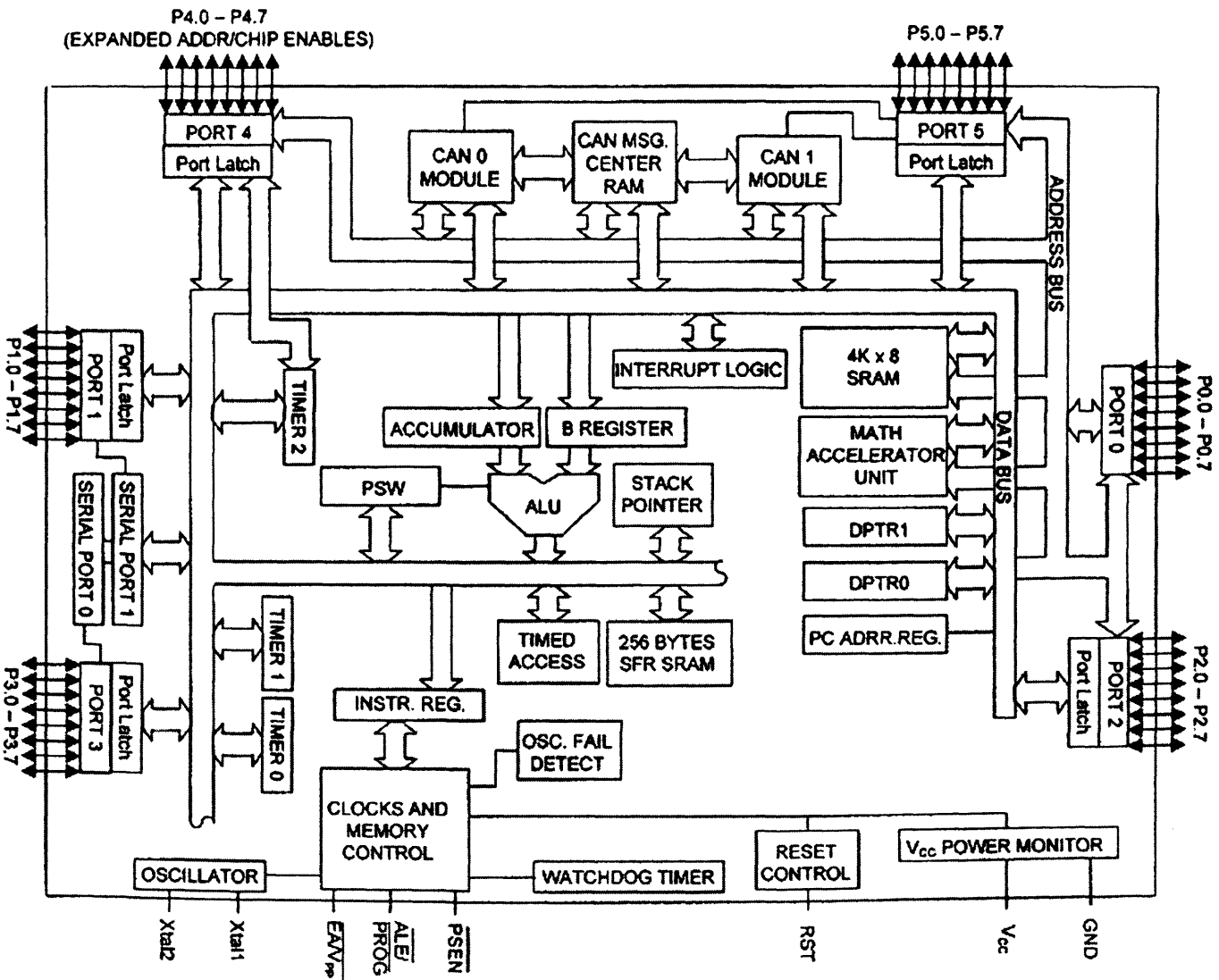


Note 1: Channels AN5 through AN7 are not available on PIC18F2X8 devices.

Note 2: All I/O pins have diode protection to VDD and VSS.

DS80C390 Microcontroller

Schematic block diagram



Microcontroller Features

FEATURES

- 80C52 compatible
 - 8051 instruction-set compatible
 - Four 8-bit I/O ports
 - Three 16-bit timer/counters
 - 256 bytes scratchpad RAM
- High-Speed Architecture
 - 4 clocks/machine cycle (8051=12)
 - Runs DC to 40 MHz clock rates
 - Frequency multiplier reduces EMI
 - Single-cycle instruction in 100 ns
 - 16/32-bit math coprocessor
- 4 kB internal SRAM usable as program/data/stack memory
- Enhanced memory architecture
 - Addresses up to 4 MB external
 - Defaults to true 8051 memory compatibility
 - User-enabled 22-bit program/data counter
 - 16-Bit/22-bit paged/22-bit contiguous modes
 - User-selectable multiplexed / non-multiplexed memory interface
 - Optional 10 bit stack pointer
- Two full-function CAN 2.0B controllers
 - 15 message centers per controller
 - Standard 11-bit or extended 29-bit identification modes
 - Supports DeviceNet, SDS, and higher layer CAN protocols
 - Disables transmitter during autobaud
 - SIESTA low power mode
- Two full-duplex hardware serial ports
- Programmable IrDA clock
- High integration controller includes
 - Power-fail reset
 - Early-warning power-fail interrupt
 - Programmable watchdog timer
 - Oscillator-fail detection
- 16 total interrupt sources with 6 external
- Available in 64-pin QFP, 68-pin PLCC

TINI Stick Memory Map

Flash ROM 1 (512K)	000000h
Flash ROM 2	080000h
Image of SRAM 0	100000h
SRAM 0 (512K)	180000h
SRAM 1 (512K)	
Image of SRAM 1	280000h
SMC Ethernet Controller	300000h
Available peripheral code & data space	308000h
Real Time Clock	
Available peripheral code & data space	320000h
Unused	
	800000h
	BFFFFFFh

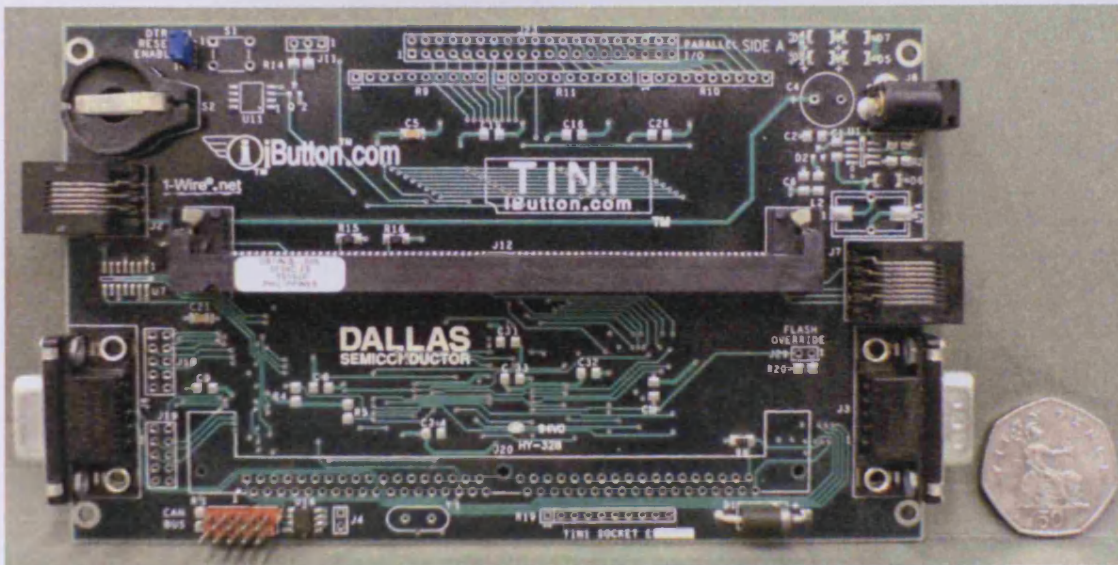
TINI Flash ROM Map

Boot Loader	0
Firmware (tini.hex)	64K
API (tiniapi.hex)	192K
Primary Java Application	448K
	512K

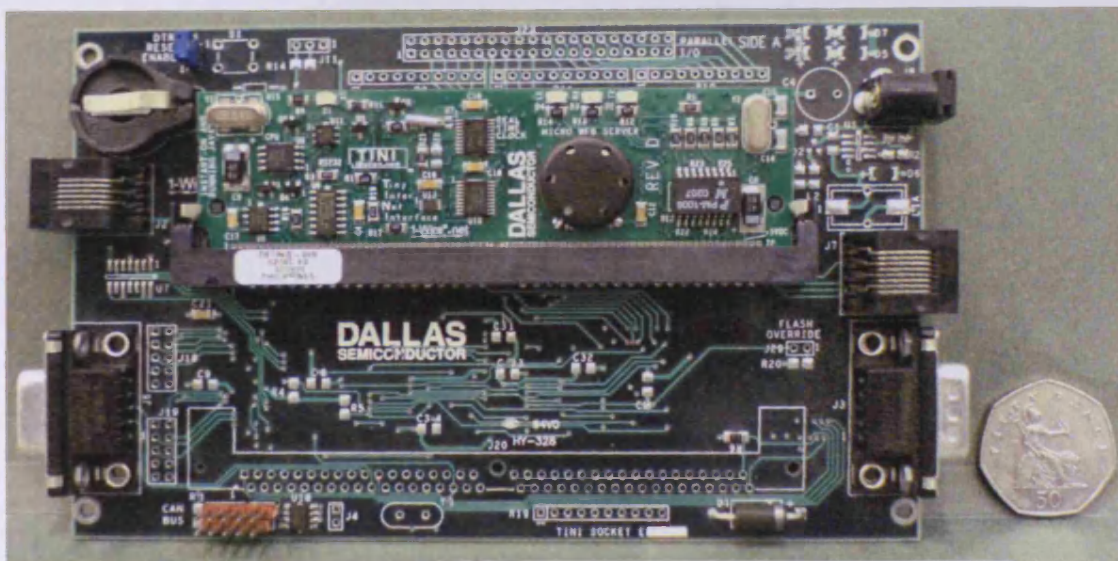
TINI Stick



TINI Socket



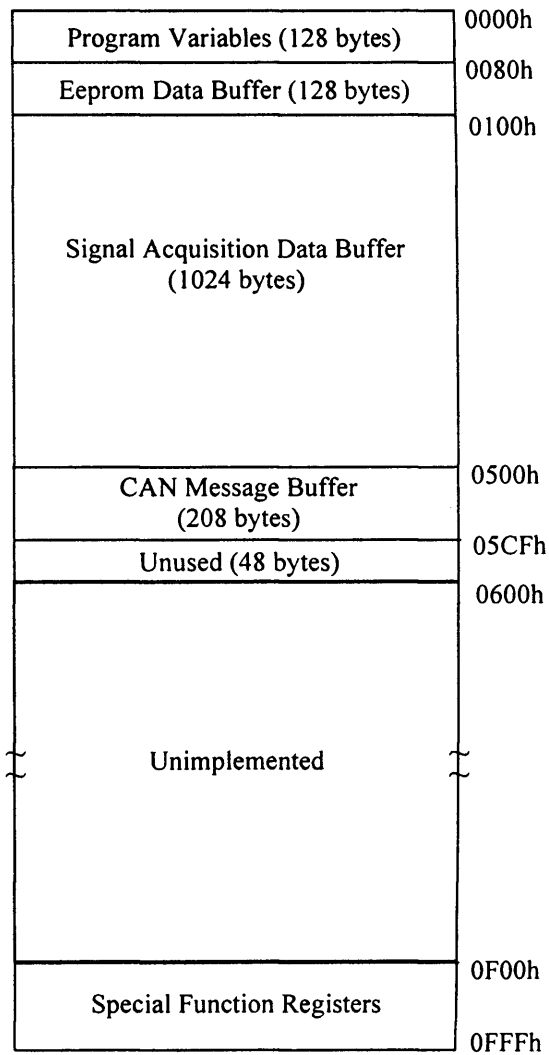
TINI Stick Mounted on Socket



APPENDIX C

System Related Details

FEN RAM Memory Map

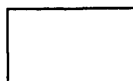


FEN Program Memory Map

Boot up code	0000h
Interrupt (High priority)	0008h
Interrupt (Low priority)	0018h
Initialisation code	001Eh
Main Loop	0800h
Write Eeprom subroutine	0A00h
Read Eeprom subroutine	0C00h
Delay subroutine	0E00h
High priority interrupt service routine (for CAN)	1000h
Low priority interrupt service routine (for Sampling)	1800h
Process_CAN subroutine	2000h
	3000h
	5000h
Process_Sample subroutine	5000h
Software Update mode subroutine	5200h
	5800h
Data Acquisition mode subroutine	5800h
Monitoring mode subroutine (Changed with applications)	6000h
	7FFFh



Memory currently used.



Unused memory left for application specific modifications and future extensions.

FEN Pseudo-codes for interrupts and associated subroutines

CAN Interrupt

Store message in a circular buffer.

Increment stored-messages counter.

CAN Routine

While stored-messages counter > 0

Process next CAN message.

Decrement stored-message counter.

TMR0 Interrupt

Timer 0 is used to generate sampling time interrupts. Its ISR is selected according to the nature of the signal to be acquired.

For Analogue voltage signals:

Start A/D conversion.

For Frequency signals:

Transfer pulse-count to accumulator.

Subtract previous count from the current count.

Adjust result in case of overflow.

*Store the result in a *circular data buffer.*

ADC Interrupt

*Transfer the conversion result to a *circular data buffer.*

**The circular data buffer mentioned in TMR0 and ADC interrupts is the same because only one of the two interrupts will be active on one node.*

FEN Main Loop

Main

```
;Check if a new CAN message is available
MOVWF    CAN_RCount,W
CPFSEQ   CAN_WCount
CALL     CAN_PROC

;Check if a new sample is available
;FSR0 and FSR1 (writer and reader)
MOVWF    FSR0H,W
CPFSEQ   FSR1H
GOTO     New_Sample
MOVWF    FSR0L,W
CPFSEQ   FSR1L
GOTO     New_Sample
GOTO     No_Sample      ;No new sample available for processing
```

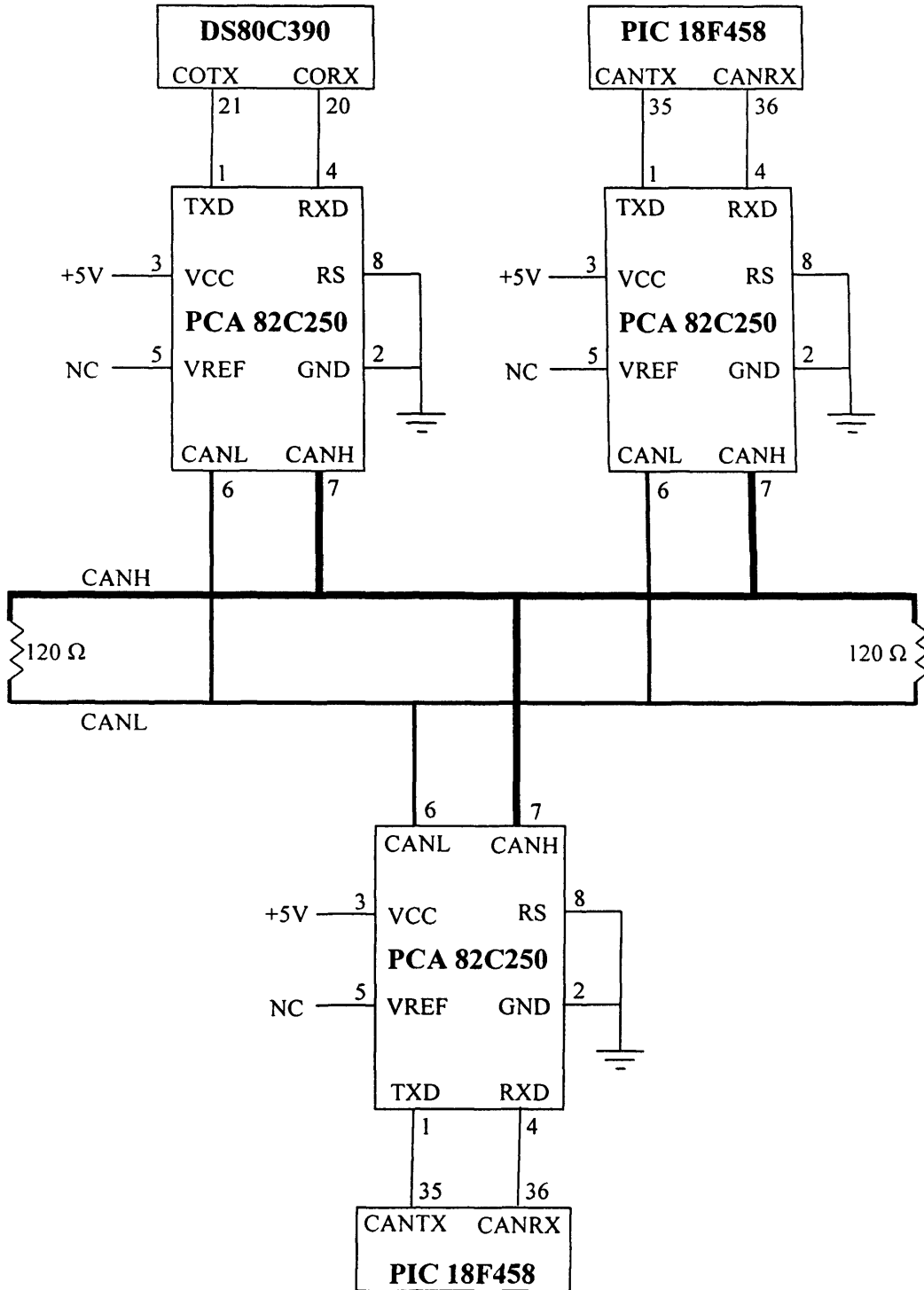
New_Sample

```
;Process the new sample
CALL    Process_Sample
```

No_Sample

```
GOTO    Main
```

CAN Bus Connections



FEN CAN Initialisations

```

BCF          TRISB,2      ;CANTX
BSF          TRISB,3      ;CANRX

;Setting for CAN control registers
MOVLW       H'80'
MOVWF       CANCON        ;Configuration mode
CLRF        CANSTAT
CLRF        COMSTAT

;Setting for CAN I/O control register
MOVLW       H'20'        ;Disable CAN capture (Don't use RC2 pin)
MOVWF       CIOCON        ;Tx pin High when inactive

;Setting for CAN baud rate Registers for 125000 bps for 40 MHz oscillator
MOVLW       H'49'        ;Tq = (2*1)/Fosc, Prescaler is 10 for 40 MHz crystal
MOVWF       BRGCON1      ;Sync jump width time = 2*Tq
MOVLW       H'AB'        ;Propagation time = 4*Tq, Sample once
MOVWF       BRGCON2      ;Phase seg1 time = 6*Tq
MOVLW       H'04'        ;Phase seg2 time = 5*Tq
MOVWF       BRGCON3      ;CAN not used for wake-up

;Setting for CAN transmit registers
;Transmit Buffer 0
MOVLW       H'03'
MOVWF       TXB0CON,BANKED ;Priority level 3 (highest priority)
MOVLW       H'06'        ;EID28-EID21
MOVWF       TXB0SIDH,BANKED
MOVLW       H'08'
MOVWF       TXB0SIDL,BANKED ;Extended identifier and EID20-EID16
MOVLW       NodeNum      ;NodeNum defined for each node
MULLW       D'16'        ;Multiply node number with 16
                                ;(lower nibble --> higher nibble in PRODL register)
MOVF        PRODL,W      ;Move result in W
ADDLW       D'01'        ;Make SUIN the destination
MOVWF       TXB0EIDH,BANKED ;EID15-EID8
MOVLW       H'00'
MOVWF       TXB0EIDL,BANKED ;EID7-EID0
MOVLW       H'00'
MOVWF       TXB0DLC,BANKED ;TXRTR bit clear, 0 data bytes

;Setting for CAN receive registers
;Receive Buffer 0
MOVLW       H'40'
MOVWF       RXB0CON        ;Receive valid messages with extended identifier

;Set Receive Mask 0 to check only the destination node number (same for all FENs)
MOVLW       H'00'
MOVWF       RXM0SIDH,BANKED ;EID28-EID21
MOVLW       H'00'
MOVWF       RXM0SIDL,BANKED ;EID20-EID16
MOVLW       H'0F'
MOVWF       RXM0EIDH,BANKED ;EID15-EID8
MOVLW       H'00'
MOVWF       RXM0EIDL,BANKED ;EID7-EID0

;Set Receive Filter 0 to accept messages for current node only

```



```

MOVLW      H'00'
MOVWF      RXF0SIDH,BANKED    ;EID28-EID21
MOVLW      H'08'
MOVWF      RXF0SIDL,BANKED    ;Extended identifier, EID20-EID16
MOVLW      NodeNum            ;Current node
MOVWF      RXF0EIDH,BANKED    ;EID15-EID8
MOVLW      H'00'
MOVWF      RXF0EIDL,BANKED    ;EID7-EID0

```

;Set Receive Filter 1 to accept broadcast messages (same for all FENs)

```

MOVLW      H'00'
MOVWF      RXF1SIDH,BANKED    ;EID28-EID21
MOVLW      H'08'
MOVWF      RXF1SIDL,BANKED    ;Extended identifier, EID20-EID16
MOVLW      H'00'              ;For broadcast message
MOVWF      RXF1EIDH,BANKED    ;EID15-EID8
MOVLW      H'00'
MOVWF      RXF1EIDL,BANKED    ;EID7-EID0

```

```

CLRF       CAN_RCount
CLRF       CAN_WCount
LFSR       FSR2,400H          ;Start address of CAN buffer for , 12 bit operation

```

;CAN interrupt configuration

```

BCF        PIR3,RXB0IF      ;Clear to initialize
MOVLW      01H               ;Set high priority for RXB0 interrupt
MOVWF      IPR3
MOVLW      01H               ;Enable RXB0 interrupt
MOVWF      PIE3

```

```

BSF        RCON,IPEN        ;Enable interrupt priorities
MOVLW      0C0H              ;Enable all high & low priority interrupts globally
MOVWF      INTCON

```

```

MOVLW      H'00'
MOVWF      CANCON            ;Normal mode for CAN

```

SUIN CAN Initialisations

```
static CanBus myCanBus;
static CanFrame myFrameT = new CanFrame(); //Broadcast transmission
static CanFrame myFrameR = new CanFrame(); //Reception

System.out.println( "Configuring CANBUS0" );

try {
    // Create a new CanBus object for CAN bus 0 of TINI
    myCanBus = new CanBus( CanBus.CANBUS0 );

    // Set up the CANBUS speed (125 Kbps)
    myCanBus.setBaudRatePrescaler( 7 );
    myCanBus.setTSEG1( 13 );
    myCanBus.setTSEG2( 7 );
    myCanBus.setSynchronizationJumpWidth( 1 );

    myCanBus.setTransmitQueueLimit(2);

    // Define 29 bit mask
    myCanBus.set29BitGlobalIDMask(0x0000FE00); //Global mask for MC 2 ~ 14
                                                //(MC 15 has its own mask)

    myCanBus.enableController();

    System.out.println( "Enabling Message Center 1 for transmission & 2 ~ 14 for reception." );
    myCanBus.setMessageCenterTXMode( 1 ); //Transmit only
    for(int i=2; i<15;i++) {
        myCanBus.setMessageCenterRXMode( i ); //Receive message from Node i
        myCanBus.setMessageCenterMessageIDMaskEnable( i, true ); //true means enabling the mask
        myCanBus.set29BitMessageCenterArbitrationID( i, 0x00000000 | (i << 12) ); //ID
        myCanBus.enableMessageCenter( i );
    }
}
catch( Exception e ) {
    System.out.println(e + ": Unable to set up CAN bus");
}
// ----- CAN BUS set up completed here -----
```

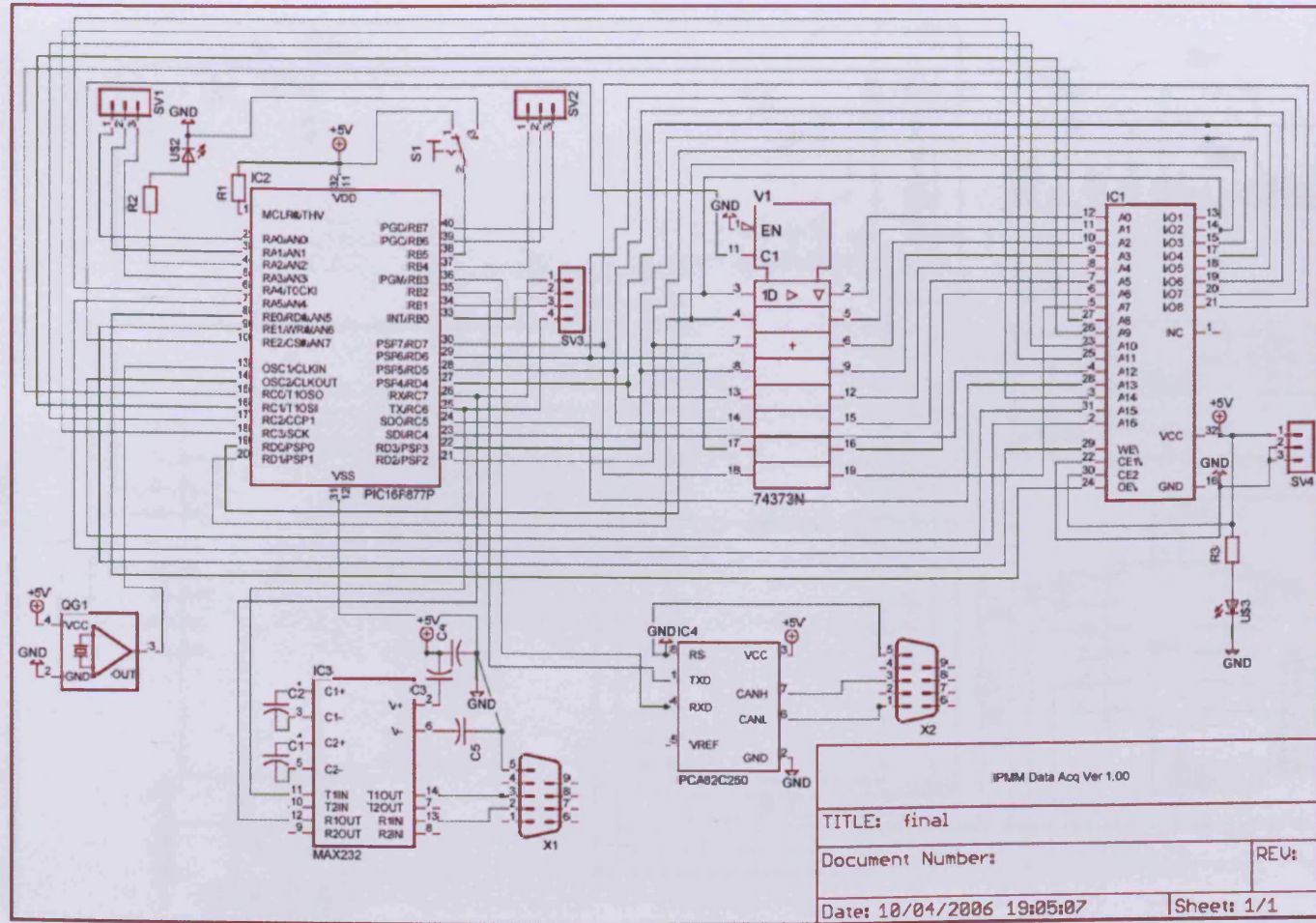
Some Useful Slush Commands

Command	Description
date	Set the system date and time
del	Remove the named file
ftp	Connect to a remote FTP server
help	Display usage information for Slush commands
ipconfig	Configures and displays the network settings
java	Executes a Java program
kill	Kill the identified process
ls	List the contents of the current directory
md	Make the named directory
netstat	Displays all TCP connections
passwd	Set the password for the specified user
pwd	Present working directory
rd	Remove the named directory
sendmail	Send email to designated recipients
startserver	Start up the specified server
stopserver	Shut down the specified server
useradd	Add a new user account
userdel	Delete the specified user account
who	List all currently logged in users

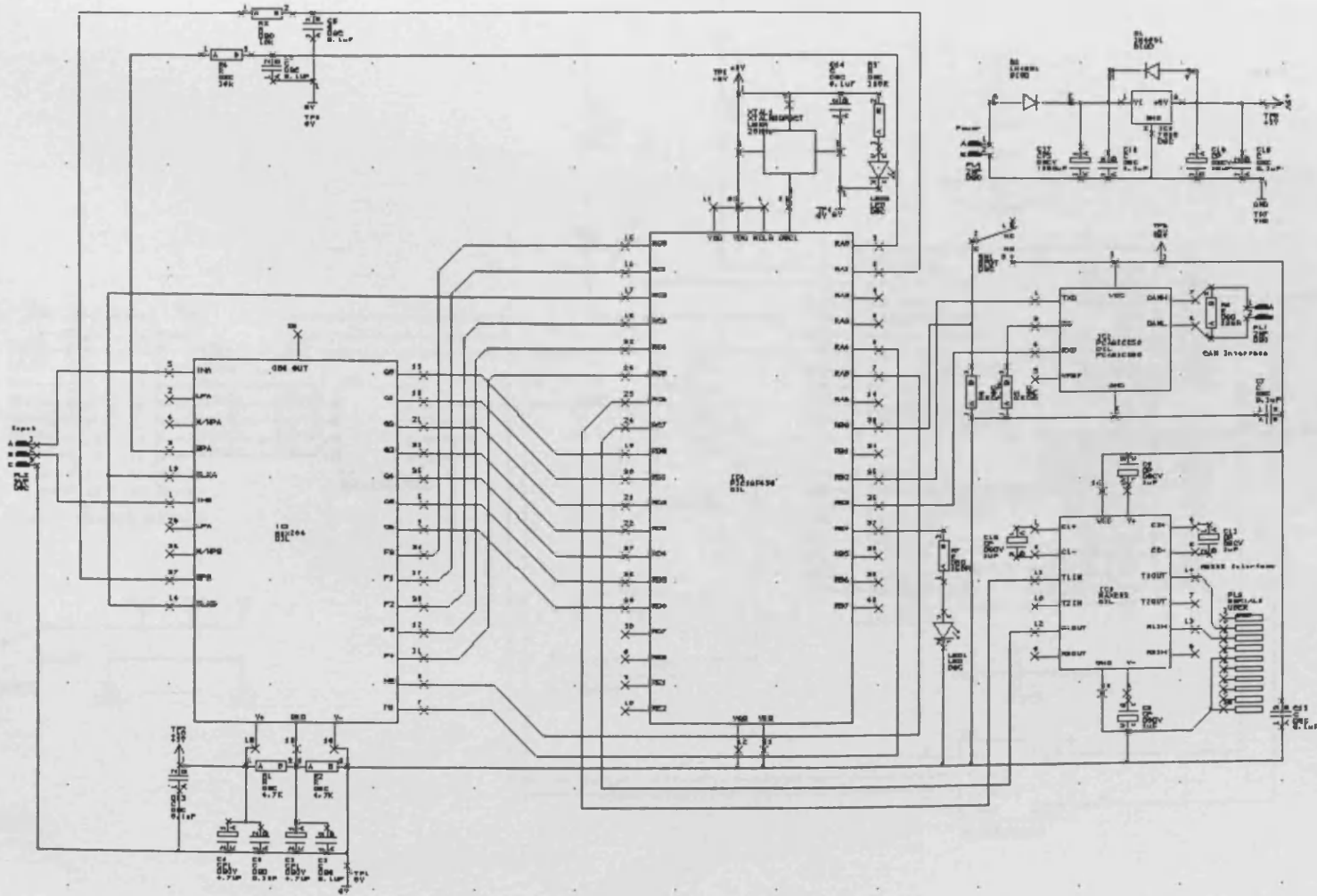
Some Useful AT Commands

Command	Description
AT	Attention command
AT&F	Set to factory default
AT+CHUP	Hang up call
AT+CLCK	Lock facility (including all incoming barring services 'AC')
AT+CLIP	Enable/disable calling line identification (CLI)
AT+CMAR	Master reset
AT+CMGF	Select message format
AT+CMGS	Send message
AT+CMSS	Send message from storage
AT+CSCS	Select character set
AT+CSIL	Silent mode
AT+CSQ	Signal strength

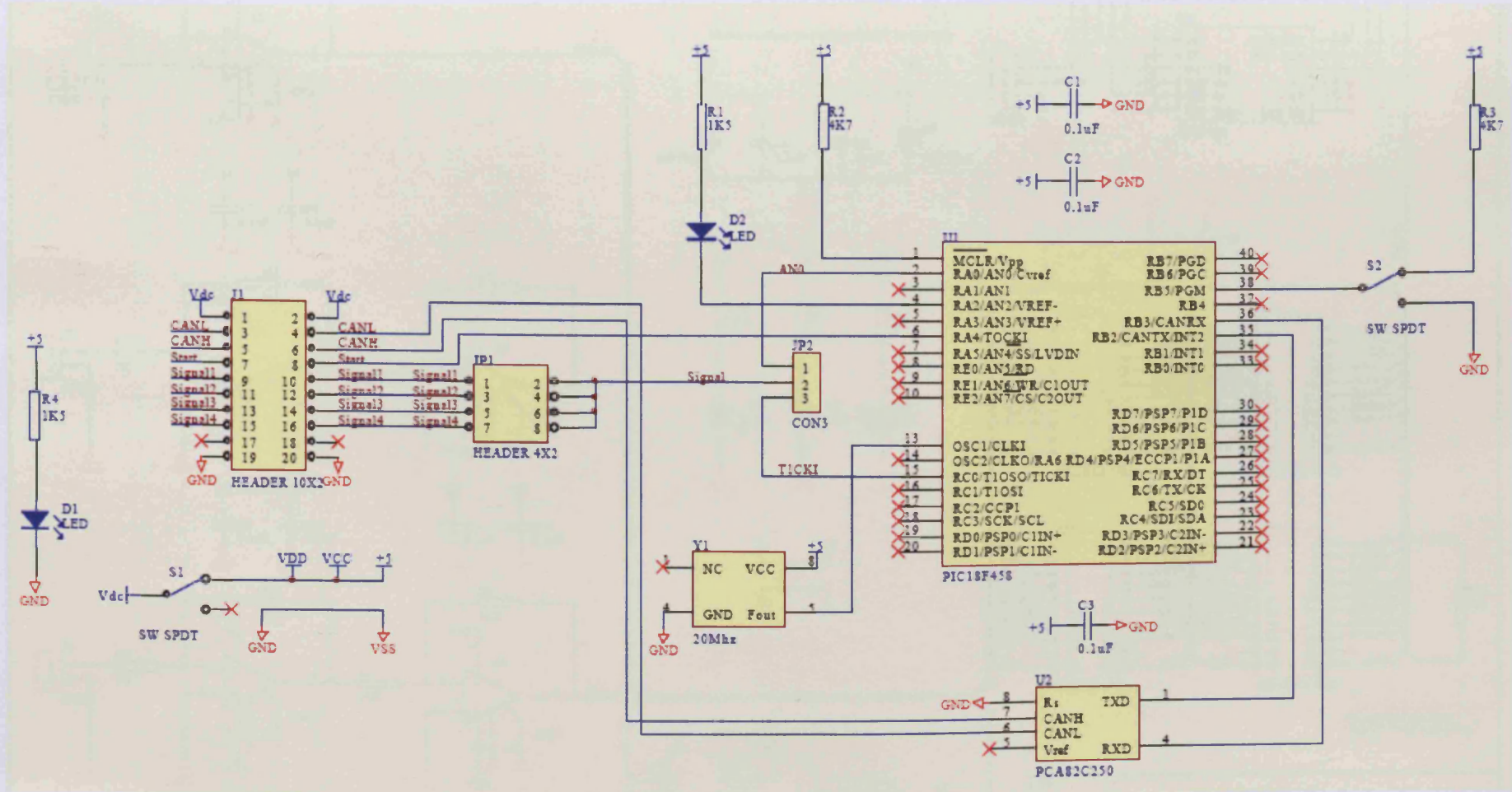
Time Analysis Circuit Diagrams



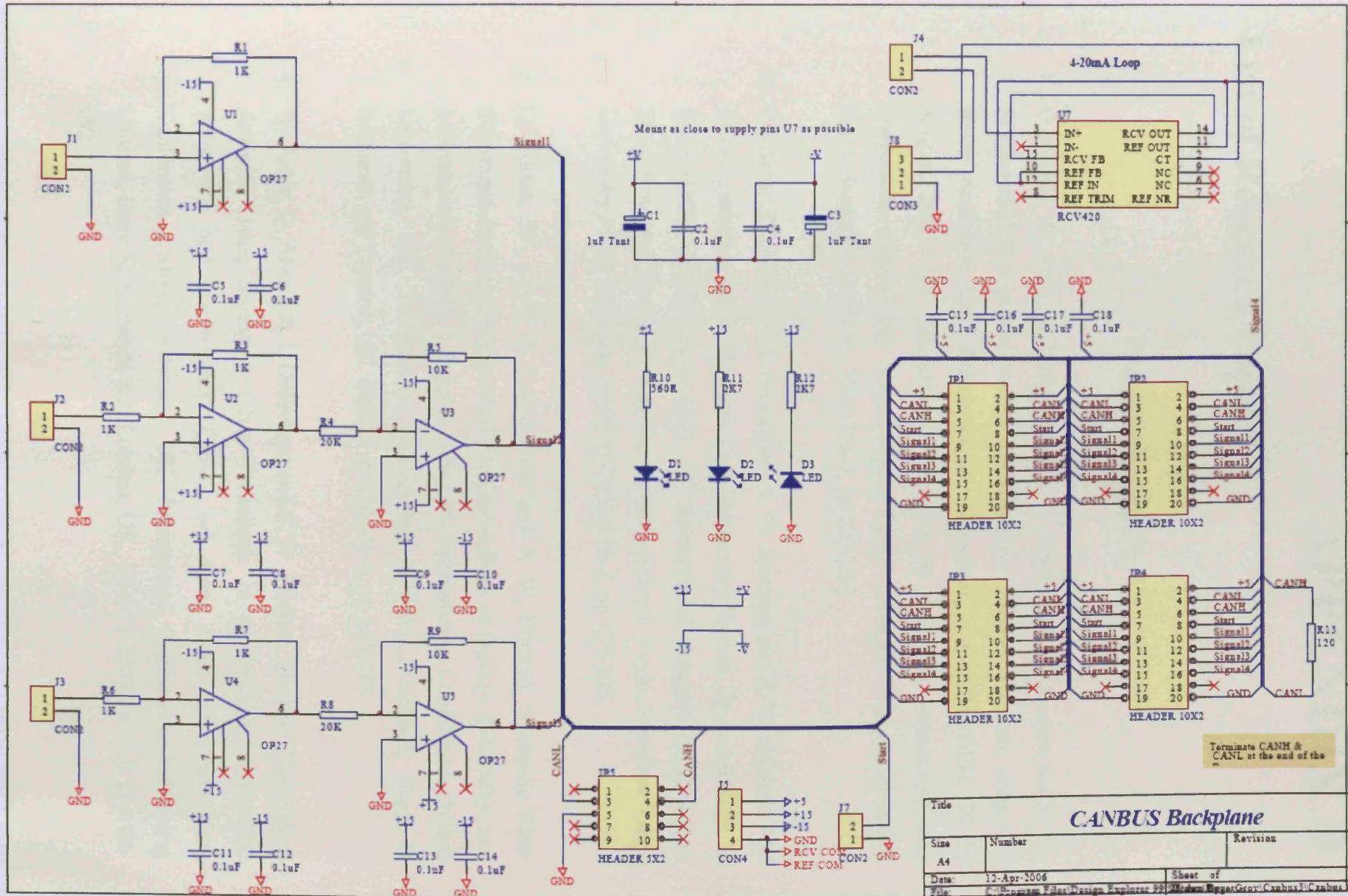
Frequency Analysis Circuit Diagram



FEN Circuit Diagram



Backplane Circuit Diagram



APPENDIX D

List of Research Papers

Published papers

1. Q. Ahsan, W. Amer, R. I. Grosvenor, A. D. Jennings, P. W. Prickett and M. R. Frankowiak. Design of a Process and Condition Monitoring System using PIC based Analogue Data Acquisition. In proceedings of COMADEM, 27-29 August 2003, 16th International Congress on Condition Monitoring and Diagnostic Engineering Management. Vaxjo, Sweeden. Vaxjo University Press, Sweeden. ISBN 91-7636-376-7, pp 227-235.
2. W. Amer, Q. Ahsan, R. I. Grosvenor, A. D. Jennings and P. W. Prickett. PIC Micro-controller based Machine Tool Monitoring System. In proceedings of COMADEM, 27-29 August 2003, 16th International Congress on Condition Monitoring and Diagnostic Engineering Management. Vaxjo, Sweeden. Vaxjo University Press, Sweeden. ISBN 91-7636-376-7, pp 219-225.
3. Q. Ahsan, W. Amer, R. I. Grosvenor and P. W. Prickett. Sweeping Filter Technique for Frequency Analysis. In proceedings of Quality, Reliability, and Maintenance (QRM), 1-2 April 2004, 5th International Conference. Oxford University, UK. Professional Engineering Publishing Limited, Bury St Edmunds and London, UK. ISBN 1 86058 440 3, pp 185-188.
4. W. Amer, Q. Ahsan, R. I. Grosvenor and P. W. Prickett. Machine Tool Signal Analysis Using Sweeping Filter Technique. In proceedings of Quality, Reliability, and Maintenance (QRM), 1-2 April 2004, 5th International Conference. Oxford University, UK. Professional Engineering Publishing Limited, Bury St Edmunds and London, UK. ISBN 1 86058 440 3, pp 189-192.

5. Q. Ahsan, R. I. Grosvenor and P. W. Prickett. A Reduced Traffic Software Model for Distributed Monitoring Systems. In proceedings of COMADEM 2004, 17th International Congress on Condition Monitoring and Diagnostic Engineering Management. Cambridge, UK. Central Printing Services, University of Birmingham, UK. ISBN 0-954 1307-1-5, pp 204-213.
6. Q. Ahsan, R. I. Grosvenor and P. W. Prickett. Distributed Control Loop Performance Monitoring Architecture. In proceedings of Control Conference, 6-9 September 2004. University of Bath, UK. ISBN 0 86197 130 2.
7. Q. Ahsan, R. A. Siddiqui, R. I. Grosvenor and P. W. Prickett. Adaptable eMonitoring System for Multi-Loop Processes. In proceedings of COMADEM 31st August – 2nd September 2005, 18th International Congress on Condition Monitoring and Diagnostic Engineering Management. Cranfield, UK. Cranfield University Press, UK. ISBN 1 871315 91 3, pp 211-220.
8. R. A. Siddiqui, Q. Ahsan, R. I. Grosvenor and P. W. Prickett. The Role of Emerging Technologies in E-Monitoring. In proceedings of COMADEM 31st August – 2nd September 2005, 18th International Congress on Condition Monitoring and Diagnostic Engineering Management. Cranfield, UK. Cranfield University Press, UK. ISBN 1 871315 91 3, pp 263-271.
9. Q. Ahsan, W. Amer, R. I. Grosvenor and P. W. Prickett. A Compact Monitoring System for Process Valves. In proceedings of 10th IEEE International Conference on Emerging Technologies and Factory Automation, 19-22 September 2005. Catania, Italy. ISBN 0-7803-9402-X, pp 1043-1046.
10. W. Amer, Q. Ahsan, R. I. Grosvenor and P. W. Prickett. Machine Tool Condition Monitoring System using Tooth Rotation Energy Estimation (TREE) Technique. In proceedings of 10th IEEE International Conference on Emerging Technologies and Factory Automation, 19-22 September 2005. Catania, Italy. ISBN 0-7803-9402-X, pp 529-536.

11. Q. Ahsan, W. Amer, R. Siddiqui, M. Al-Yami, R. I. Grosvenor and P. W. Prickett. Distributed Process Monitoring and Management. In proceedings of IEEE International Conference on Engineering and Intelligent Systems, 22-23 April 2006. Islamabad, Pakistan. pp 336-341.

Accepted papers for publication

1. Q. Ahsan, R. I. Grosvenor and P. W. Prickett. Distributed On-line System for Process Plant Monitoring. Accepted for Publication in Journal of Process Mechanical Engineering, Part E, Proc IMechE Vol 220, 2006.

Submitted papers for publication

1. R. A. Siddiqui, W. Amer, Q. Ahsan, R. I. Grosvenor and P. W. Prickett. Multi-band Infinite Impulse Response Filtering using Microcontrollers for e-Monitoring Applications. Submitted to Journal of Microprocessors and Microsystems.

