Distributed On-Line System for Process Plant Monitoring

Q Ahsan, R. I. Grosvenor and P. W. Prickett

Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering 2006 220: 61
DOI: 10.1243/09544089JPME53

Distributed on-line system for process plant monitoring

Q Ahsan, R I Grosvenor, and **P W Prickett***
Cardiff University, School of Engineering, Cardiff, UK

The manuscript was received on 11 November 2004 and was accepted after revision for publication on 20 January 2006.

DOI: 10.1243/09544089JPME53

Abstract: This paper describes a methodology for monitoring industrial processes and plant that can be implemented cost-effectively within small-to-medium enterprises. The methodology is based on a network of 8-bit microcontrollers that communicate with each other on a controller area network bus. Ethernet connectivity is provided so that remote users can access the system on the internet. The software models developed for data acquisition nodes and the design of remote user interfaces and supervisory nodes are also explained. The system is aimed at providing specific maintenance guidance and fault identification, rather than gathering data for off-line analysis. Overly complicated processing is avoided to make real-time implementation possible, using 8-bit microcontrollers. The methodology emphasizes the use of process controller signals for fault detection and sensor signals for fault isolation. The suitability of the methodology is explored by acquiring signals from a laboratory-based process rig. Suitable monitoring techniques for the system in time and frequency domains are also discussed.

Keywords: process monitoring, microcontroller, remote user interface, controller signal

1 INTRODUCTION

This paper considers the development of a process monitoring system, including the design of signal acquisition hardware, associated networking issues, the required software architecture, and a real-time on-line user interface. Both low- and high-level languages are used to develop the prototype system, which is then evaluated using a laboratory-based process rig.

The computer control system of a modern automated plant should be capable of operating the process continuously with an optimal performance. Keeping the plant in the best possible condition is essential to support optimal production. This paper presents a low-cost but powerful tool that can monitor processes through the actions taken by the controller to indicate current process conditions and hence be built into optimal plant operation strategies.

One common approach used to maintain processes is time-based maintenance, where various components are repaired or replaced at a scheduled time according to their estimated life. In the absence of any real information, such time-based maintenance can represent a useful approach but may cause significant financial losses due to the needless replacement of healthy parts, the loss of production through shutdown time, equipment costs, and maintenance personnel time. For example, Hartley [**1**] states that 35 per cent of field visits for a large chemical company were routine checks where no problem was found. Although this in itself is a waste, more critically, this activity uses up time which could be concentrated on undertaking activities that are capable of producing truly significant business results [**2**]. These activities include the clear need to target maintenance efforts to actions that are really required, as highlighted by Trenchard *et al.* [**3**], who report that a major refinery saved $100K by identifying valves that did not require maintenance.

A sensible approach to satisfy this need is to deploy condition-based maintenance. The knowledge of a component's health is as vital as the knowledge of its failure in today's competitive industrial world [**1**]. It has been reported that improvements of 20–30 per cent in process efficiency have been made using intelligent condition-based monitoring [**4**]. In

**Corresponding author: Cardiff University, School of Engineering, Newport Road, Cardiff CF24 3AA, UK. email: prickett@cf.ac.uk*

such systems, components not performing at an expected level can be replaced or repaired immediately. If this policy is to work, however, it is essential that faulty components are located quickly and required action is taken at an early stage. To do this effectively entails checking the health and status of all the components in a plant, and hence, automated monitoring systems are required.

A potential solution is to distribute sensors in the plant and acquire and then analyse the signals using some form of computer-based system. Decisions can then be made and implemented. However, such a strategy is often not practically possible because of the associated high cost, networking problems, data storage issues, and the time needed for analysing the resulting huge amount of data. Researchers have worked on this problem from various aspects, and attempts have been made to reduce the number of sensors, the associated cabling, and the amount of data generated.

This paper examines approaches taken based on 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 and various monitoring techniques is given in section 2. The technological issues involved are addressed in section 3 with an assessment of the use of various hardware platforms and networking issues for monitoring. Overall, there is no general consensus on any particular method or technique to be the best, and a lot of work is still required in the area.

This paper presents one such approach based on a monitoring methodology that is technically achievable and economically feasible. The requirements from the system are given in section 4 and the proposed solution details are explained in section 5. The methodology is based on small-sized low-cost signal acquisition nodes collaborating with each other. Section 6 describes the practical implementation of the proposed methodology using 8-bit microcontrollers. The system is aimed at producing specific maintenance alarms rather than presenting raw data or trends to maintenance engineers for analysis, as it is almost impossible for control engineers to analyse raw data due to shortage of time and the huge amount of data [**5**]. To support the approach-embedded signal analysis, tools have been developed. Finally, section 7 considers the continuing development of the system and its potential uses.

The design and implementation of a system of this type from scratch for each new application is a large task encompassing signal acquisition, data analysis, communication protocols, remote user interface, and system configuration. The authors contend that the costs normally associated with systems capable of providing real-time monitoring have consequently been too high to allow their widespread adoption. The authors consider that the systems resulting from this research can provide a significant step towards more affordable and flexible monitoring systems. Once installed, the system is capable of providing results to the end user who may not be a technical person. Any updates or modifications needed may be downloaded over the internet; the developer/engineer does not have to travel to the installation site. The ease of use for the end user is another important aspect towards widespread deployment of the system.

2 MONITORING APPROACHES

If, within a computer-controlled process, the output quality deviates from a set or optimal value, it is normal for the controller to attempt to bring the process back to the required state. The changes in a controller signal can thus be seen as a response to an external disturbance or a fault within the system. The variations arising within the controller signal during this activity can, therefore, potentially be used as an indication of any abnormality. Most of the research related to process controller signals typically assumes certain conditions. These include assumptions such as the set-point changes are infrequent, and process, actuators, and sensors are noisy [**6**].

Examples of this approach include that taken by Gustafsson and Graebe [**7**] who implemented an algorithm on a digital signal processor (DSP) in real-time, providing control-loop performance monitoring (CLPM) with an emphasis on distinguishing between disturbances and controller relevant changes when a deviation from the nominal behaviour is observed. Schafer and Cinar [**5**] proposed a methodology to determine a benchmark and to monitor model-predictive control performance online. They proposed the use of ratio of historical and achieved performance for monitoring and the ratio of design and achieved performance for diagnosis. The diagnosis is limited to distinguish between root cause problems associated with the controller and those not caused by the controller.

Hu *et al.* [**8**] propose that the process controller's signal and switches are important sources of data for fault diagnosis and isolation (FDI). Their diagnostic approach was based on digital and analogue parameter acquisition and an expert system. Amplitude, variance, mean, gradient, and signal trend were

used with analogue signals. Abnormal conditions were detected using thresholds. The system was reportedly successful in fault classification and cause detection by searching functional trees. Fault trees for each module were coded and saved into a knowledge base. Fault tree analysis (FTA) was combined with case-based reasoning, as the inferring mechanism for the expert system, for on-line fault diagnosis for a railway train by Raaphorst *et al.* [**9**]. Input nodes of the indexed network were associated with fault symptoms. The existing symptoms posed questions to the network and the answers provided were used for fault code matching. Hu *et al.* [**10**] combined FTA with sequential and logical diagnostic models to achieve good results in fault source indication. They matched the actual controller signals against the expected ones, but considered digital sources only. Andrews and Dunnett [**11**] consider the traditional FTA methods as inaccurate and inefficient, especially for the non-trivial situations with branch point dependencies, and support the use of event tree analysis (ETA), a visual representation of all the possible events in a system. They propose the use of binary decision diagrams in ETA for computational efficiency.

Various studies have been made on using controller signals to assess control-loop performance. These are important here because they may indicate the root cause of the problems associated with the controller. Schafer and Cinar [**5**] point out that 60 per cent of all industrial controllers have some kind of performance problem. Ender [**12**] observed that one-third of the control loops are fit for purpose, one-third are open loop, and one-third have excessive variability. This is confirmed [**13**] for over 100,000 regulatory control loops and further discussed in regard to the various causes of poor control-loop performance and their respective contributions to the problem [**14**]. Xia and Howell [**15**] classify the loop status into seven categories and suggest using the identified category to narrow down the number of possible faults to be investigated. Horch [**16**] discusses control-loop performance assessment in detail and the detection of oscillation because of sticky valves in particular. He states that one of the most important problems in process industry is the widespread presence of oscillations. Hagglund [**17**] observes that oscillations in a control loop are generally far from sinusoidal, and the main reason for the oscillations is the valve friction. Rengaswamy *et al.* [**18**] mention that there are very few papers explicitly dealing with the problem of detecting faults in control loops and describe a semi-qualitative approach for detecting sinusoidal, square, and triangular oscillations in a control loop. They claim that their approach works quite fast and is suitable for real-time applications. Hagglund [**19**] proposes a method to identify the sluggish loop among hundreds or thousands of loops in a process plant. Thornhill *et al.* [**20**] use non-linearity to find the root cause among various loops in a refinery. They use the regularity of the zero crossings of filtered auto-covariance functions to find multiple oscillations existing simultaneously. Choudhury *et al.* [**21**] use higher order statistics to diagnose poor control-loop performance.

Goulding *et al.* [**22**] discuss the use of multi-variate statistical process control (MSPC) for fault detection. They advocate the joint analysis of cause and effect for exposing faults, emphasizing that a strong relationship between MSPC and model- and signal-based fault detection schema is obvious. They compare partial least-squares (PLS) and principal component analysis (PCA) as two MSPC techniques and favour PLS for better indicating the changing process conditions. They state that PLS is suitable when plant variables can be partitioned into cause and effect. Desforges and Sandoz [**23**] report a toolbox for advanced MSPM based on PCA and PLS, in which, in one application, PCA reduced the number of variables from 36 to only 3. Desforges *et al.* [**24**] state that emphasis is now shifting from mere fault detection to ensuring continued operation under identified fault conditions. The research results are now commercially available, combining practical skills from industrial process control experts and academic capabilities in model-based controllers and statistical process monitoring [**25**]. In an example, Lennox *et al.* [**26**] explain the development of a condition monitoring system for a fed-batch fermentation process, where PCA is used to detect and isolate faults and PLS is used to estimate final product composition. They used multiway MSPC along with a radial-basis function neural network. Wong *et al.* [**27**] favour a hierarchical approach when dealing with multiple signals, arguing that the approach of classifying each variable first, and then classifying the overall process is more computationally efficient. They present two methods for extending the previous method to identify and classify individual process trends to multiple process trends. Ruiz *et al.* [**28**] describe the monitoring of a sequencing batch reactor (SBR) pilot plant, which is a non-linear and time-varying complex process used in wastewater treatment plants. MPCA with an automatic fuzzy classification algorithm is used for situation assessment by combining numeric and symbolic classification algorithms.

The development of fault models for continuous physical systems is very difficult because of the very large range of possible behaviour. Biswas *et al.* [**29**] urge the use of qualitative methods. They reduced the number of potential fault candidates by determining normal, above-normal, or below-normal

behaviours at the first stage and applied quantitative methods on the reduced set. Multiple fault candidates were searched for contradictions in the thermal bus system of space station Freedom, and candidates with no contradictions were reported to the user. Isermann [**30**] favours the use of heuristic approaches for fault diagnosis stating that the underlying physical laws for a process are either frequently not known in analytical form or are too complicated for calculations. He also observed that binary variable-based FTA has not proved successful because of the continuous nature of faults and symptoms. The methods of approximate reasoning were, therefore, considered more appropriate.

Another approach is to use the event timings for fault detection and diagnosis. This approach was taken by Davey *et al.* [**31**], who associated nominal times with each event, and time outs were used to indicate a fault within a Petri-net technique for machine tool failure diagnosis. The cause of the fault was diagnosed by analysing the event that caused the time-out. This method was claimed to be very effective providing rapid, accurate, and appropriate fault condition information. Another technique used time difference in successive acoustic emission signals to detect bearing wear, [**32**] and its application to a test rig achieved good results [**33**].

Studzinski [**34**] provides examples for computerized monitoring of environmental processes. These include a computer-aided system to support decision-making by the process operator at a wastewater treatment plant, an expert system to forecast weather using automatic monitoring of atmospheric parameters, and a three-module integrated computer system to support operational decision-making in a communal water network, where water pressure and flow were measured at nine points and communicated to the central computer using global system for mobile communication (GSM) telephony. In another application, a number of artificial neural networks (ANN) were arranged in parallel and hierarchical fashion to detect leakage from a treated water distribution system using flow and pressure sensors data [**35**]. The ANNs were trained using time series data collected from the sensors for several weeks or months so that they could learn the proper distribution patterns. Data from 14 pressure loggers were combined in the test trials. The pipe bursts were simulated by opening the outlet valves, and system performance was evaluated. The resultant three-dimensional pressure drop map was reported to be successful in the accurate indication of the fault location.

Another important aspect of fault diagnostics uses the frequency components available in the signals. 's domain' parameters were used in the controller performance assessment by Kendra and Cinar [**36**], who recommend the use of measures that coincide with classical and modern frequency domain design specifications. Higham [**37**] states that it is important to gather the whole signal from a turbine flowmeter, including the noise, to obtain its wide frequency response. He supported the use of distributed fieldbus nodes for turbine flowmeter signal acquisition and analysis, as the communication bottleneck in a centralized system restricted the frequency spectrum of the measurement signal to $\sim$1 Hz. Zhu *et al.* [**38**] removed the usual low-pass filters from the sensors to obtain the widest possible frequency responses. They also removed the signal mean value by using high-pass filtering. The resulting signals from flow and pressure sensors in various motor impeller conditions were analysed. Significant differences were observed in the frequency response for normal and faulty conditions. Similar observations were made when the flow path was partially blocked. Amadi-Echendu and Higham [**39**] analysed a turbine flowmeter signal to obtain information about the turbine condition. Inherent periodicity in the signal was observed to be different when a blade of the turbine flowmeter was damaged. Higham and Perovic [**40**] concentrated on pressure and flow measurements, which account for almost half of the measurements made in process industries. Their study on various kinds of pumps indicated that the differences in frequency spectra for these signals identified not only the pump faults but also their extent. It was recommended that the measurement systems should be designed to have the widest possible frequency response for such analysis. The sensors are usually designed to filter out frequency components $>$5 Hz to provide smoother signal to the controller, and hence, it is important that the monitoring system should acquire the signal before this filtering.

3 TECHNOLOGICAL ISSUES

Researchers have developed many techniques and methodologies for finding faults in a process. These techniques are implemented on various hardware platforms in a variety of ways. This section considers the approaches that are currently being investigated.

Personal computers (PCs) have been the most common choice because of their availability and flexibility. Typical applications include: monitoring vibration of turbines in a power plant [**41**], condition monitoring of feed rolls in plate mills [**42**], an on-line expert system for fault diagnosis in hydraulic systems [**43**], and an intelligent monitoring system for end milling [**44**].

Networking is often considered for use in monitoring systems to support tasks that a single computer cannot perform. Bonastre *et al.* [**45**], for example, present a distributed expert system for the

monitoring and control of chemical analysis process. They argue that the existing technology makes it feasible to overcome the network bottlenecks. In another case, a PC-based distributed system is deployed to monitor about 14 000 control loops worldwide [**46**], generating daily and monthly reports. The results have changed the mindset of the company, which had traditionally placed a low priority on loop hardware maintenance, as ∼30 per cent proportional integral derivative (PID) controllers were found in 'manual' mode at any given time. As a consequence, off-class production due to process control-related causes has been reduced by 53 per cent in 1 year. However, some caution is needed. Other research has also considered the problem of data transfer over a network. Manders *et al.* [**47**] report an on-line FDI system for a multi-tank fluid system using smart sensors. Signal-to-symbol transformation is used to reduce network load, but they observe that sampling rates of more than one sample per second are not feasible in their system because of network conditions. Ong *et al.* [**48**] developed a prototype system based on an agent-based peer-to-peer protocol with monitoring and diagnostics done by different agents and again indicated that the network speed was not satisfactory to meet system demands in providing on-line fault diagnosis.

Distributing processing power, including microprocessors, around the network seems to be providing some solutions to these restrictions. De Frutos and Giron-Sierra [**49**] employed an embedded PC in their distributed control system. Modems and telephone lines were used for data communication from these remote distributed nodes to a supervisory node, again based on a PC. Ehrlich *et al.* [**50**] proposed the use of fieldbus to connect various system components in their generic model for the deployment of smart sensors. Mittal *et al.* [**51**] observe that the use of multiple access bus networks is increasing for real-time applications. Baccigalupi *et al.* [**52**] describe an on-line monitoring and diagnostic system for electronic components using two DSPs. The use of DSPs was supported to facilitate mathematical manipulations. Another example using this technology is a real-time monitoring system for tool failure in a milling process [**53**].

A typical architecture evolving from this approach sees the use of microcontrollers and embedded processors to reduce cost and size, with a PC for diagnostics at a higher level [**54**]. A communication link is provided to send reports to the management layer. Nieva and Wegmann [**55**] suggested a layered protocol for generic data acquisition system (DAS), so that application layer programmers do not need to know the data acquisition details. Bolic *et al.* [**56**] use 8-bit microcontrollers for acquisition as well as a central unit that provides task arbitration and user interface in their simple task controlling system. Dassanayake *et al.* [**57**] propose a three-layer architecture for FDI using fieldbus, in which the low-layer nodes were implemented on microcontrollers. A DSP provided measurement and control capabilities, and an embedded PC was used to provide fault detection at the middle layer. As this PC uses controller efficiencies to detect faults rather than analytical models, less processing and memory storage are required. The consistency of a fault is checked, and a feature extraction routine is executed on data acquired for the consistent fault. A neural network procedure is used to generate a recognized fault code. Livani *et al.* [**58**] suggest exploiting the inexpensive availability of 8-bit microcontrollers for their communication model and consider the use of controller area network (CAN) bus as shared communication medium in real time.

4 REQUIREMENT SPECIFICATIONS

The earlier analysis of the strengths and weaknesses of current approaches to process monitoring produced a requirement for a complete monitoring system that could be formed from low-cost modules. These should be mounted very close to the origin of any acquired signal, preferably inside the machine or process. The overall system needed to be easy to use, with low maintenance costs and with little need for outside intervention following its initial installation.

Producing such a monitoring system required the evolution of a distributed network of nodes, each of which acquires its signals from specified sensors. The module needed to be general purpose and allows the acquisition of the signals commonly available from the sensors such as digital, analogue, timed, or frequency formats. The processing techniques used in the nodes should be selectable according to the computational power and memory available inside the microcontroller unit (MCU). In considering which techniques could be used, it was determined that several of the most common monitoring techniques lie outside the current capabilities of the microcontrollers. This analysis is presented in Table 1.

The general specification of a node included the provision of some storage capacity to store the data before it is processed. Consideration was given to the sampling rates needed for analogue inputs; most physical processes require only a few samples per second for time-series analysis, but frequency analysis needs much higher sampling rates. The nodes needed communication facilities to share the gathered information with other nodes and some logic to send and receive messages in a collaborative way. It is also important for the network to accept as many nodes as needed up to some maximum limit

Table 1 Monitoring techniques and implementation

Technique	Implemented on distributed MCUs	Remarks[a]	References
MSPC	No	P	[**22, 23, 26**]
ANN	No	P	[**35, 59**]
Fuzzy methods	No	P	[**28**]
History based	No	M	[**20, 46**]
FTA	No	P	[**8, 9, 10**]
ETA	Yes		[**11**]
Petri net/time based	Yes		[**31, 32, 33**]
Semi-qualitative	Yes		[**15, 18, 29, 54**]
Hierarchical decomposition	Yes		[**27, 47, 48, 57**]
Simple SPC	Yes		[**7, 8**]
Sweeping filter	Yes for simple frequency analysis		[**60, 61**]
Quantitative models	Limited to simple small models	P and M	[**62**]
Expert systems/ KBS	Limited to very small systems	M	[**34, 43, 45**]
FFT	Limited FFT possible on newer MCUs	P and M	[**38, 39, 40, 63**]

[a]P indicates too much processing required and M indicates too much memory required.

(ten or larger). The system was required to be capable of detecting the available nodes automatically and to adapt to situations such as a node switching on or off. Another desired feature was that increasing number of nodes in the network should not increase the workload for any data acquisition node, although it is bound to increase the network traffic.

A network of MCUs with modest processing power is not expected to provide very detailed diagnostics all the time. For more detailed diagnosis, the system needed to act as DAS and provide the raw data to a higher layer computer for enhanced computations or for human analysis. The monitoring system may indicate such a situation on detecting an abnormality without being able to figure out the fault.

The system designed to meet the earlier requirements is outlined subsequently. This system is different in structure and operation to existing monitoring approaches. As such its performance, potential, and benefits cannot be directly compared with existing implementations. However, the following sections are intended to illustrate how effective the system will be.

5 MONITORING SYSTEM

The monitoring system outlined here was designed to fulfil the requirements specified in section 4. It consists of a distributed network of 8-bit microcontrollers communicating using CAN bus. Access to and from the outside world is via ethernet. The architecture developed is shown in Fig. 1. The system comprises two types of nodes: front end nodes (FENs) and synchronizing and user interface nodes (SUINs) both implemented on 8-bit microcontrollers. The FENs acquire the signals close to the sensor to achieve noise reduction and provide fault resolution to individual signals. CAN bus was chosen to connect the various nodes of the system because of its performance under noisy conditions and its robustness. The nodes are designed to work under a peer-to-peer paradigm, although one node, the SUIN, provides both the user interface and the task synchronization. With this peer-to-peer paradigm, the FENs check any set signal inter-relationships among themselves and provide the results to the SUIN for onward display to the remote user. Internet connectivity is provided by the SUIN, which thus acts as the interface to dynamic web pages. The system takes process-related decisions at its lowest level and 'normal' data are not presented for off-line processing, unless the system is specifically placed into a data acquisition mode by a remote user seeking to access and analyse any of the signals being acquired. This eliminates the need for data storage media, such as hard disks and so on, at the monitoring level.

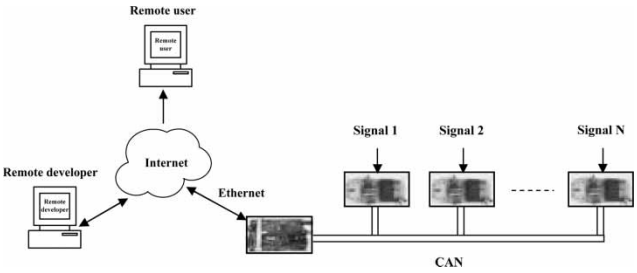


Fig. 1 Monitoring system overview

The monitoring problem is divided into clear and simple logical decisions to reduce the computations. This requires a clear understanding of the process in terms of any inter-relationships of its signals. To achieve this, 'normal' data are gathered from a healthy process under optimum performance conditions and presented to the developer for analysis. Various faults are then intentionally introduced into the process and the resulting data are captured. Fault-finding procedures are then formulated on the basis of this experimental process knowledge. This eliminates the requirement of a mathematical model of the process.

A configuration file is stored in each node according to the particular requirements of that node. This includes threshold and event definitions. It also includes the details of specific nodes that are to be

informed in the cases of specific fault detection. The configuration file also tells a node where else to look for specific types of information. This information is provided by the application developer according to the specific requirements of the application. A static data structure for the configuration file is favoured at the moment for this information storage. A larger dynamic configuration file is stored on the SUIN, which changes according to the current availability of various nodes and is used in system synchronization.

If a particularly computationally challenging task is to be performed in a particular application, a more powerful microprocessor may be used. Such a node can be added as a different type of FEN on the same CAN bus. This feature is currently being investigated [**63**] and will be included in this methodology in such a way as to preserve the low-cost basis of the monitoring approach. This will see the multiplexing of these more powerful modules, in order that they can be accessed as required through the CAN bus and the number of such modules can be kept to a minimum.

5.1 FEN design and embodiment

FENs are responsible for signal acquisition, after necessary conditioning, and for checks required on individual parameters. A FEN is implemented on a Microchip PIC 18F458 microcontroller and its printed circuit board (PCB) consists of a microcontroller, a crystal oscillator, a resistor, and a CAN bus transducer. The PCB size is $5 \times 9\,\mathrm{cm}^2$ with dual-in-line (DIL) chips and can be significantly reduced by using surface mount chips. The unit cost is less than £20; therefore, the price and size of FENs are no issues and multiple nodes may be used as required. General purpose FEN PCBs are used in this research with signal conditioning done separately. Assembly language is used for programming higher execution speed and shorter code. Figure 2 shows the software model for a FEN, where messages are written in capital letters and their width is given in number of bits.

Fault detection is based on threshold crossing for individual signals and on simple statistical measures such as mean values. When a fault condition is detected, the FENs coordinate with each other and check for fault consistency. Because sensor signals are generally noisier than controller signals, it is the controller signal that is used as the prime candidate for initial fault detection. The fault-detecting FEN initiates the coordination with other FENs to find the cause of the controller signal deviation. It is expected that the controller signal will behave differently from the normal in almost all possible fault

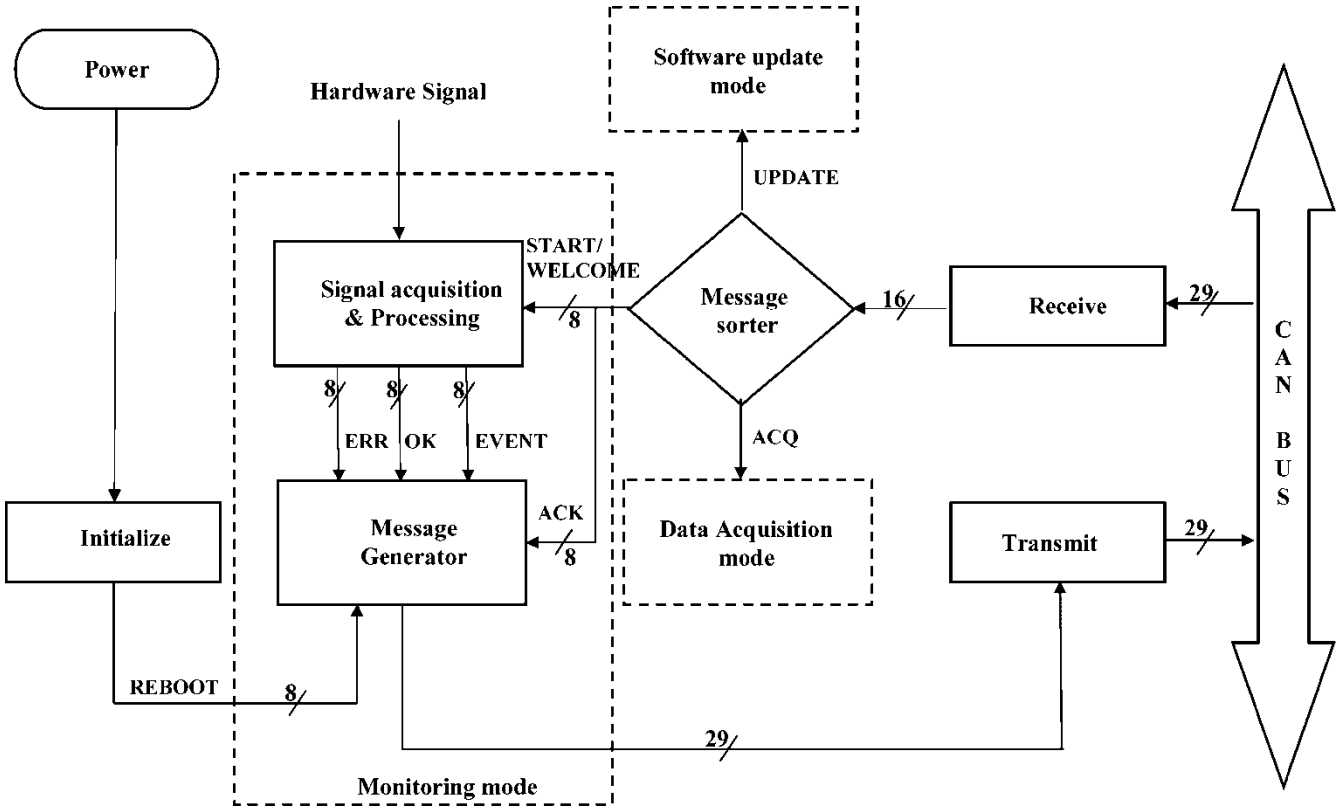


Fig. 2 FEN software model

conditions. Therefore, it is important to devise the monitoring strategy giving a key position to the controller signal.

To support the real-time handling of process events, multiple types of interrupts are used in all nodes. A timer interrupt is required to acquire signals on the correct sampling time. Another interrupt notifies the central processing unit about the completion of an A/D conversion. The receipt of a CAN message generates an interrupt so that it can be dealt with immediately, before it can be overwritten by another message. CAN messaging is structured to include the receiving node address and message-type filtering for broadcasting messages to minimize interrupt occurrences. A CAN interrupt is assigned the highest priority because the monitoring system cannot afford to lose any important messages. Very short codes are written for interrupt service routines (ISRs) so that no interrupts are missed. The computations are done in normal routines in the time between the interrupts. The ISRs for A/D conversion completion or pulse counting are used only to put the result in a buffer so that it can be used later outside the ISR. The details of the full interrupt structure for the proposed model are given in reference [**64**].

5.2 SUIN design and embodiment

The SUIN provides a remote user interface and task synchronization in the monitoring system. It has been developed using Dallas Semiconductor DS80C390 microcontroller-based Tiny InterNet Interface (TINI) hardware. The TINI board is a small PCB with a microcontroller, memory, and other required circuitry. It is available as a single inline memory module (SIMM) that can be inserted into a TINI socket providing connectors for various ports, including CAN and ethernet. The TINI socket is 10 $\times$ 16 cm^2 and the TINI board is mounted on it without requiring any extra space. The TINI board and socket were purchased for a total price of under £60. A Unix-like operating shell 'Slush' and development tools were downloaded from the internet as shareware codes. The developed software model is shown in Fig. 3, where messages are written in capital letters with their width given in number of bits. The application program was developed using Java application programme interface (API).

The SUIN is responsible for providing a remote user interface through the internet. The user does not need any propriety software because generally available software, such as Internet Explorer and Telnet, has been used. The SUIN provides a password-protected Telnet server through which a

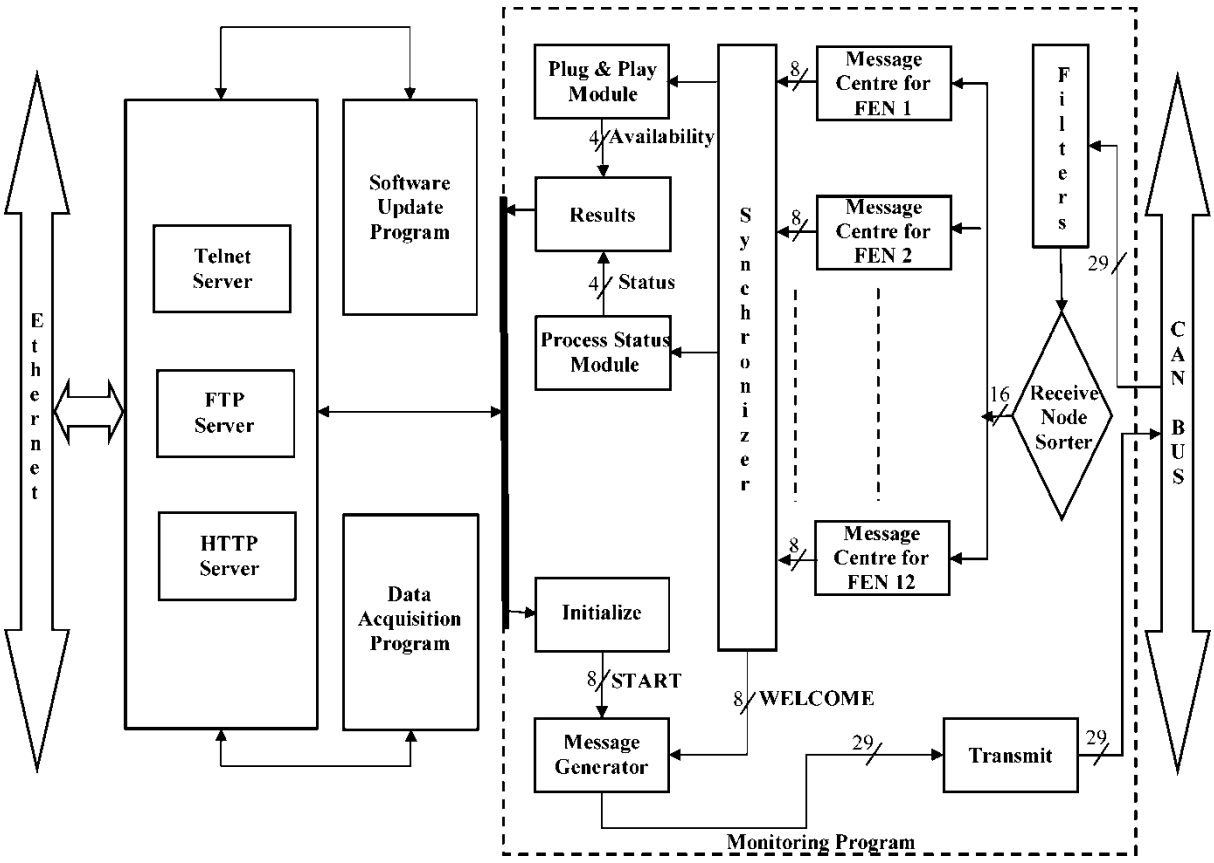


Fig. 3 SUIN software model

remote operator can login to start a monitoring program. The results are currently being stored in the SUIN file system as small-sized files that can be downloaded through file transfer protocol on Internet Explorer onto a remote PC. Use of TINI as a web server has also been reported [**64**] and the integration of both activities as HTML pages is ongoing. The SUIN files are updated with new monitoring results as soon as they become available from where they can be viewed by the user on time-activated web pages that automatically refresh themselves at regular intervals. The remote user interface and the monitoring process are developed as two different modules sharing information with each other. This application architecture provides the flexibility to run both activities on the same microcontroller and on separate microcontrollers. For complex monitoring systems, as well as in the cases of multiple remote users, it may be more suitable to use two separate microcontrollers running one module each.

An updated version of code can be uploaded to the SUIN by a remote developer. Old files can be replaced with new ones and completely new files can be added. New web pages may be added to the SUIN website giving the users more options. This increases the effectiveness of the same hardware in changing demands. Newer versions of FEN codes can be uploaded via the SUIN for onward delivery. CAN messages are used to upload the new code on FENs, as they are implemented with flash program memory. A new version of software can, therefore, be loaded in the whole system in a matter of minutes with no changes in hardware and no visits required to the installation site.

5.3 Fault detection and isolation

To support fault detection, each FEN acquires signals and checks them for any abnormality. The check may be applied to raw data, its running sum, running mean value, and so on. Thresholds are defined for each signal and signals are continuously matched against the upper and lower bounds. Any out-of-bounds data indicate either a fault or a disturbance; disturbances are generally expected to die out quickly and will be noted but ultimately ignored by the FEN. Any persistent abnormality is reported by the detecting FEN to other predefined nodes via a CAN message. Any FEN receiving such a message checks its own status and sends the combined information as an alarm message to the SUIN. The SUIN will therefore acquire knowledge of the situation and status of all the signals. It uses this combined status information to isolate the fault cause, according to the knowledge provided at the time of installation. The SUIN updates the process status in its results' file as soon as a change is detected. Interested users can check the monitoring results on the internet at any time. If a specific combination of signal conditions occurs which was not considered at the time of installation, the system will not be able to detect the fault cause. Such a condition will be reported to the user. The user may then set the monitoring system into a data acquisition mode to allow more detailed analysis. In this way, remote experts can undertake fault diagnosis with full access to all the available signals. Should a new fault be identified, knowledge of its cause and effect can be integrated in the existing code so that the system can automatically deal with similar situations in the future.

Most process faults do not occur abruptly, but develop gradually in the system causing a gradual deterioration in the performance. It is therefore recommended that some simple statistical measures should be taken periodically to see whether there are any trends in the data for each parameter. Trends such as increased mean value can give an indication of a developing fault. Thresholds for such statistical values should be predefined to reduce the computational load on FENs. Several threshold levels may be defined to grade the intensity of the problem. The system nodes will coordinate with each other when they detect such a trend and specific maintenance guidance will be provided. A maintenance activity can then be planned at an appropriate time to prevent any fault occurrences and financial losses. This functionality, when implemented in the future, will make the system even more useful.

5.4 Plug and play

A plug-and-play facility is provided in the code. The SUIN is able to broadcast enquiry messages on the CAN bus; all active FENs respond to this enquiry and the SUIN therefore knows the active nodes on the system. A suitable policy is then adopted for monitoring, according to the nodes present. Each node is allocated a 4-bit identification number for this purpose, as shown in the CAN identifier structure in Fig. 4. The SUIN gathers this information at start-up, and any FEN starting afterwards sends a message to it to update the information. It decides about the unavailability of an initially available FEN in case it does not receive any message from it within the time-out period defined by the developer at the time of installation. Typical messaging between the SUIN and the FENs is shown in Fig. 5. Figure 5(a) shows initial setup, Fig. 5(b) shows a case where an error is detected within the FEN and

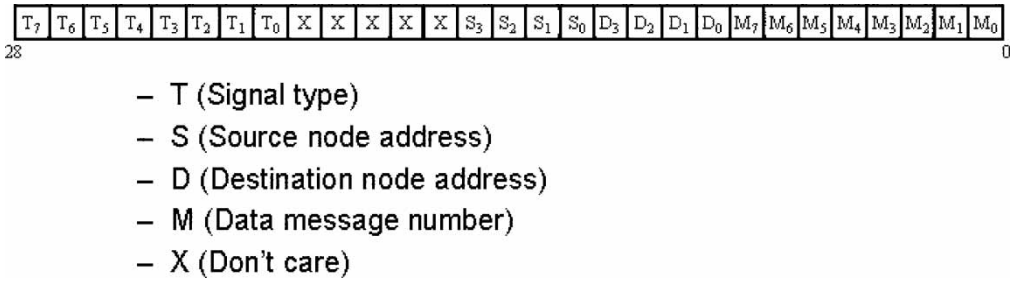
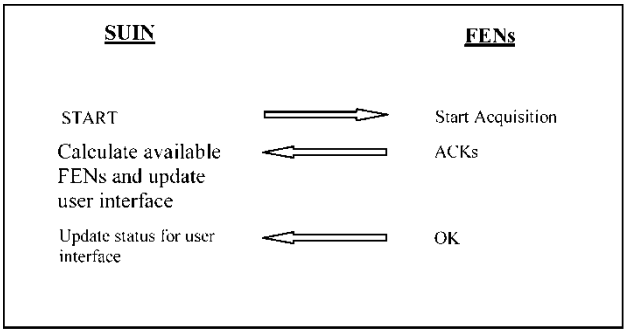


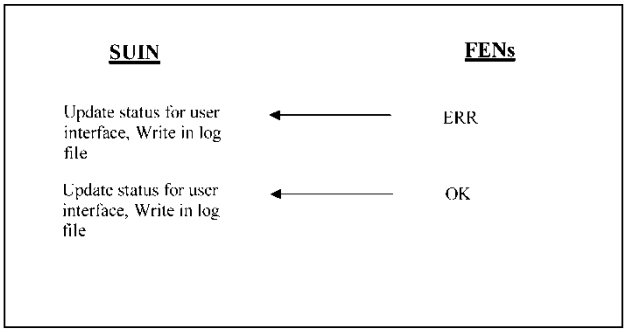
Fig. 4 CAN message identifier

subsequently corrected, and Fig. 5(c) shows the messaging arising when a new FEN is added. The block arrows show messages involving multiple FENs, whereas normal arrows depict messages to or from a single FEN. The SUIN CAN circuitry, built into the microcontroller, consists of 14 identical message centres (MCs). One MC is used to transmit messages and another MC is dedicated to receive broadcast messages. The remaining 12 MCs are available to receive messages from 12 nodes, then providing automatic sorting of messages from various senders. This reduces the load on the processor and also eliminates the possibility of one node's message being overwritten by another's.

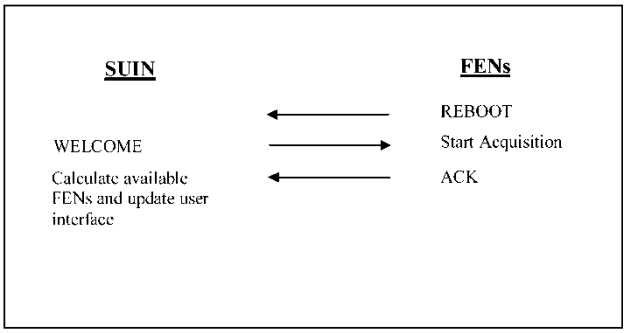
The plug-and-play facility also helps when a node turns faulty, as the SUIN can then adopt a reduced functionality model comprising the remaining nodes only. This is also useful in processes where some optional plant components are switched off when not required. Such process changes do not require modifications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



(a) Initial setup



(b) Error detected and corrected



(c) A FEN boot-up

Fig. 5 Typical messages between SUIN and FENs

5.5 Communication issues

A shared communication bus, CAN, was used for messaging between all the nodes in the monitoring system. Therefore, it was important that traffic load on this bus remained within limits. Thus, the monitoring strategy is such that a minimum number of messages should be generated and sending of acknowledgements is not required. To achieve this, checks on an individual signal are performed on the acquiring FEN and the communication of raw data between the nodes is discouraged. Further, a FEN will only initiate a message on detecting either an out-of-tolerance signal or a predefined event. It also generates a good-health message, 'OK', if it has not generated a message for a predefined length of time. This feature is required so that other nodes do not take it as being a switched-off node in this plug-and-play scenario. These messages are sent only to the interested nodes and are not broadcast. This is to avoid message overwriting in the receive buffers of a FEN. Simple statistical measures, such as maximum, minimum, running sum, or mean value, are calculated by a FEN as required and

provided to other nodes on request. The acquired data are stored in a circular buffer so that the most recent data remain available for such calculations.

5.6 Frequency analysis

The proposed methodology is based on dealing with individual signals at the first level. In the time domain, these signals are checked against the threshold values for amplitude or mean value. There may be certain scenarios where a frequency-domain analysis of a signal may be more fruitful than the time-domain analysis; the work of Amadi-Echendu *et al.* [**65**] in flow processes provides an example of this. A FEN may be used to detect the presence of certain frequency components in the signal. The power of these frequency components may be checked against the predefined thresholds to generate fault symptoms, just as in the time domain. Fast Fourier Transform (FFT) is the most commonly employed technique for frequency analysis, but is generally considered too computationally taxing for 8-bit microcontrollers. An FFT method for the 8-bit PIC microcontrollers is available [**66**] and example applications have been implemented [**67**], but the resolution is very low. The provided 64 Hz resolution may be good for audio applications, but not for monitoring industrial processes. The windowing required before applying FFT is also considered a very time-consuming and computationally expensive task. Microchip's new 16-bit microcontrollers (dsPICs) have a software library for windowing and FFT routines but are limited to 256 bits, which may still not provide satisfactory results.

To overcome these limitations, a technique based on 'sweeping filters' has been developed. In this technique, the microcontroller sweeps a range of frequencies of interest to find the peak-to-peak amplitudes of various frequency components present in a signal. A programmable analogue filter is configured as a narrow bandpass filter to detect any frequency in that band. The microcontroller then changes the programmable filter settings to select another centre frequency for the bandpass filter. The whole range of interest is scanned in this way, and the relative amplitudes of all the bands are calculated. A frequency resolution of 1 Hz is achieved for the normal frequency range of interest in industrial processes. Figure 6 shows the detection of a 9 Hz square wave, as an example, in a test signal in the laboratory. The microcontroller successfully detected the fundamental 9 Hz component and the harmonic at 27 Hz as expected. Amer *et al.* [**61**] provide an example from industry where a machine tool tooth breakage was detected with this technique.

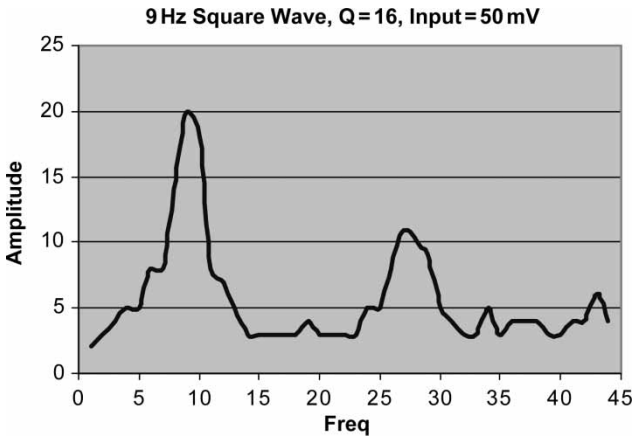


Fig. 6 Detection of 9 Hz square wave signal by sweeping filter technique

Further details of the sweeping filter technique can be found elsewhere [**60**].

6 SYSTEM OPERATION AND PERFORMANCE

A laboratory-based process rig has been used to deploy and test the monitoring system. On this rig, a centrifugal pump transfers fluid from a sump to a tank. Various types of faults, such as a blockage in the pipe, a leakage in the tank, and so on, can be introduced into the test rig. The rig is interfaced to a PC, which acts as the process controller, using Profibus fieldbus. To support this research, controller software has been developed and implemented for the test rig. Two controller modes were produced: one based on level control in a batch process and another based on flow control as a continuous process. Figure 7 shows the process rig with block diagrams explaining the batch and the continuous processes.

6.1 Batch process monitoring

The batch process is shown in Fig. 7(b), where the fluid is transferred from the sump to the tank until it is filled and a batch is completed. The controller is implemented on the PC to fill the tank according to a preset pattern after an initial idle time to let the flow establish in the pipe. The fluid is released from the tank on completion of the batch and the next batch may be started. The fluid level in the tank provides the feedback signal for the controller, which calculates the power required by the centrifugal pump as the control signal. A monitoring system consisting of three FENs and a SUIN was deployed. One FEN acquired the controller signal for pump power as an analogue voltage signal. The second FEN acquired the feedback signal from the level

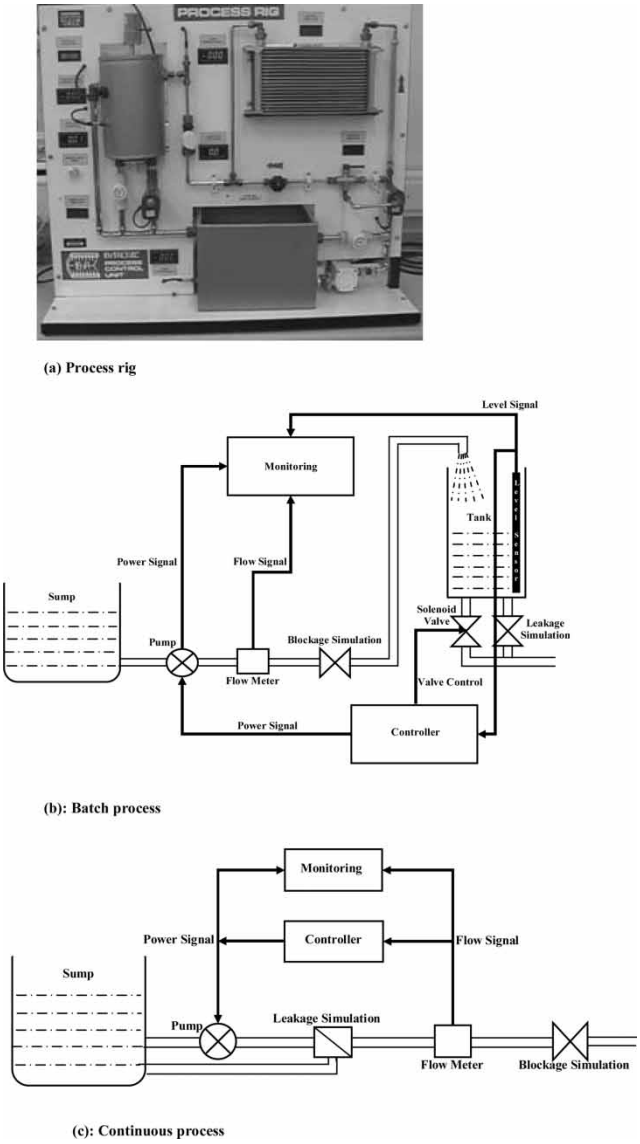


Fig. 7 Process rig with batch and continuous processes

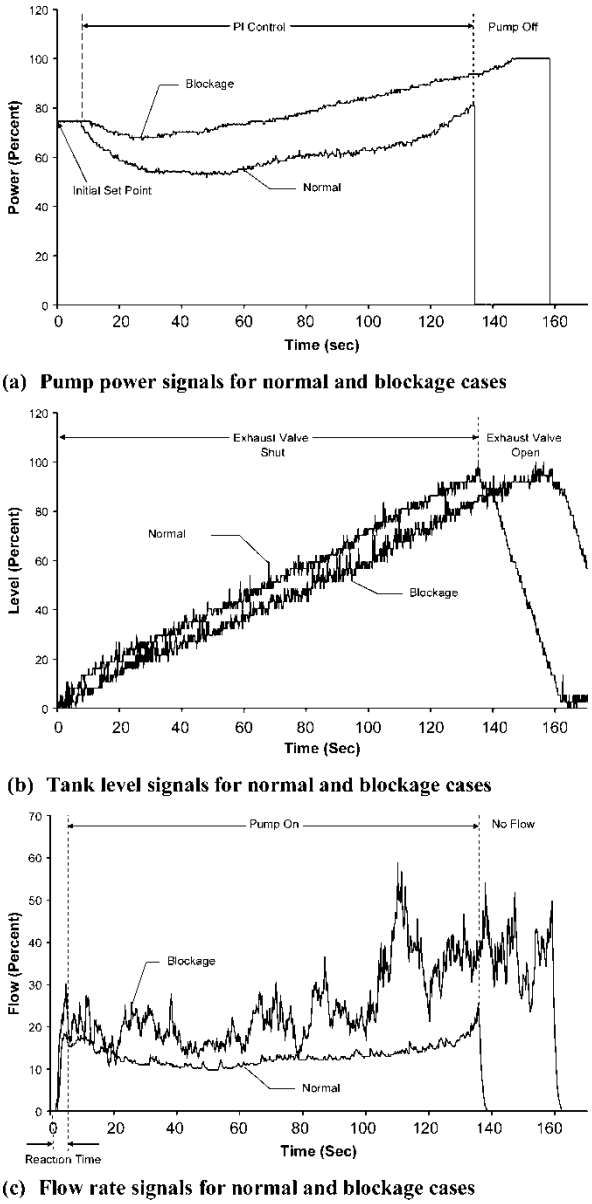


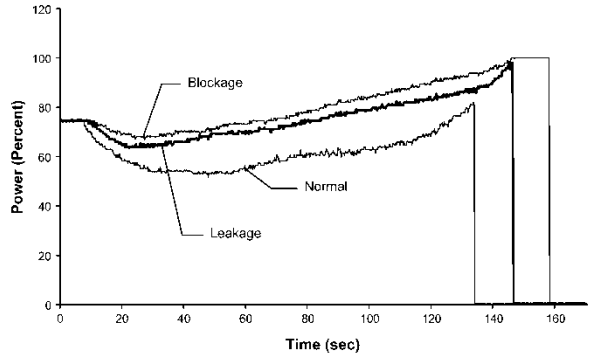
Fig. 8 Batch processing: sensor signals for normal and blockage cases

sensor in the tank as a 4–20 mA current signal. The third FEN acquired the flowrate signal from the flow sensor in the pipe in the form of pulses per second.

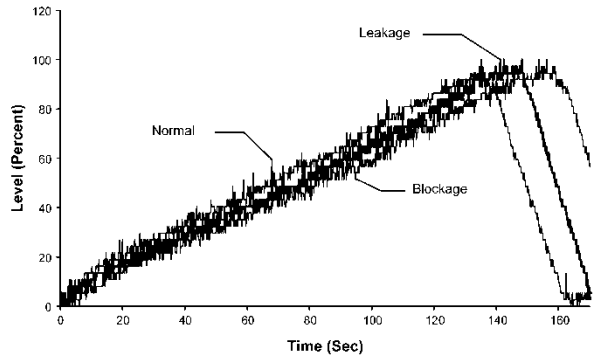
To establish the fault detection capabilities of the deployed system, various fault conditions were created. For example, in Fig. 8, the signals acquired by FENs under normal conditions are compared with those arising with a blockage. It can be observed in Fig. 8(a) that the controller signal is significantly increased when a blockage fault occurs. Signals from the tank level and the flowrate sensors, acquired concurrently by the designated FENs, are shown in Figs 8(b) and (c), respectively. The *y*-axis values have been changed as percentages for the sake of clarity, but the FENs work on the actual numbers they acquire. The variation in each signal can be seen for normal and abnormal conditions. The calculation of a running sum for the tank level signal, Fig. 8(b), shows the accumulating error in the process. The flowrate signal, Fig. 8(c), again provides an indication of a fault, with evidence of the greatest difference between normal and abnormal conditions being seen. It was thus established that the deployed system can identify the presence of a fault and that the fault diagnosis, in this example, a pipe blockage, may be confirmed by the fusion of information from the FENs. The next stage of this research was to investigate the ability of the system to distinguish between fault causes.

To test the ability of the system to distinguish between faults, a range of fault conditions were

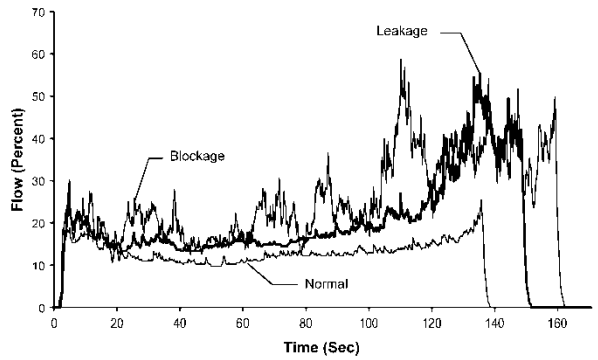
considered. Scenarios were created by introducing blockage and leakage faults in the test rig, acquiring signals from the appropriate FENs, and then overlaying these signals to determine whether they could be individually identified with their associated faults. As an example of this, the signals that were acquired when there was leakage from the tank are shown in Fig. 9 together with those acquired under normal conditions and with a blockage in the pipe. It can be observed in Fig. 9(a) that the controller signal is significantly increased when a fault occurs. However, the actual fault causing the controller signal deviation is not clear from the signal alone, as both faults cause approximately the same data trends. Additional information is required for proper fault diagnosis and identification. In this case, signals from the level and the flowrate sensors were available and were being acquired and monitored by FENs. Figure 9(b) shows the behaviour of the feedback signal from the level sensor and Fig. 9(c) shows the flowrate signals that were acquired at the same time. The variation in each signal can be seen for both normal and abnormal conditions. The tank level signal, Fig. 9(b), may indicate a possible fault, which is unlikely to always correctly identify the cause in isolation. The flowrate signal, Fig. 9(c), again provides an indication of a fault, with evidence of the greatest difference between normal and abnormal conditions and it does appear to identify a different pattern of behaviour between the leakage and the blockage. This indication, when viewed in conjunction with the other two signals, could be used by an expert to distinguish between fault conditions and to establish fault isolating rules. These rules are then provided to the monitoring system, and the FENs communicate with each other accordingly. The SUIN gets the alarm messages accordingly and conveys the identified fault cause to the end user.



(a) Pump power signals for normal, leakage and blockage cases



(b) Tank level signals for normal, leakage and blockage cases



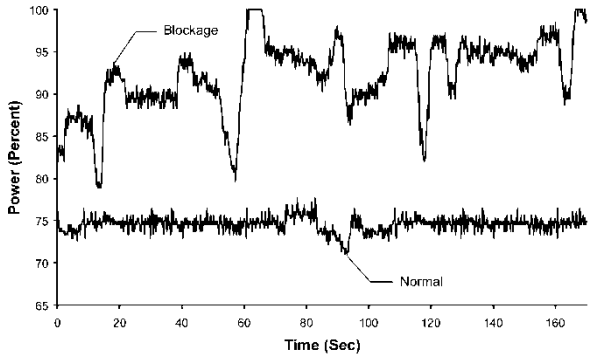
(c) Flow rate signals for normal, leakage and blockage cases

Fig. 9 Batch processing: sensor signals for normal, leakage, and blockage cases

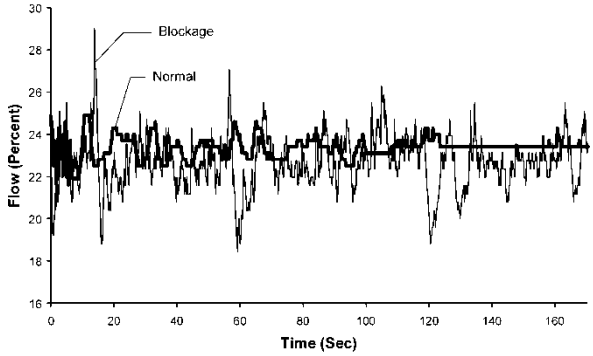
6.2 Continuous process monitoring

Figure 7(c) shows the continuous process implemented on the same test rig by changing the controller on the PC. In this mode, the flowrate signal was used as feedback to maintain a continuous flow in the pipe. The power signal to the pump was used as the controller signal in this application. The tank level sensor, although available, was not applicable because of the dynamic nature of the process. Faults were introduced to again initially simulate a pipe blockage. Figure 10(a) shows the pump power signal under normal and blockage conditions. The signal provides a clear indication of a fault. Figure 10(b) shows the flowrate signal under normal and abnormal conditions, with the increased range variation giving a clear indication of the presence of a fault.

Following the same approach as before, different faults were then considered simultaneously. This produced the results shown in Fig. 11, which indicated that the real cause of fault is difficult to determine with these two signals. This means that, currently, the system would clearly indicate that a fault had occurred, but that the exact cause of this fault would need further consideration. This is possible using the architecture deployed here, because information from some additional sensors may also be acquired to support accurate fault identification. Additionally, more advanced signal analysis tools could be deployed at the higher levels in the architecture.

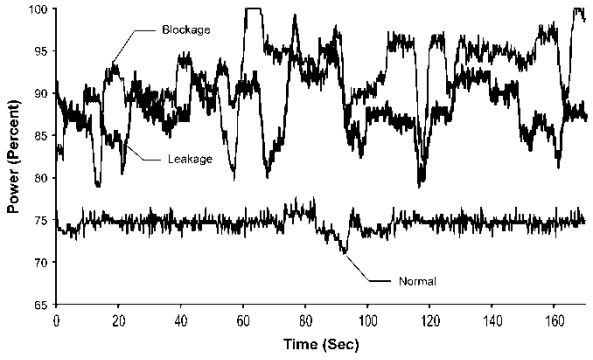


(a) Pump power signals for normal and blockage cases

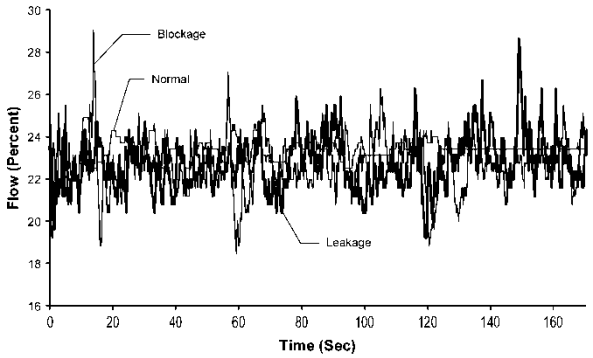


(b) Flow rate signals for normal and blockage cases

Fig. 10 Continuous processing: sensor signals for normal and blockage cases



(a) Pump power signals for normal, leakage and blockage cases



(b) Flow rate signals for normal, leakage and blockage cases

Fig. 11 Continuous processing: sensor signals for normal, leakage, and blockage cases

7 DISCUSSION AND CONCLUSIONS

The developed system fulfils the requirement specifications set before its development. The circuit designed for FEN is general purpose, and various types of signals can be acquired by setting the hardware jumper and appropriate flag in the configuration file at time of installation. This is demonstrated by acquiring signals in analogue voltage, current, and pulsed formats. The resulting PCB is small, making it easier to place the board close to sensor. The circuit was implemented using DIL integrated chips for testing, and the board size can be significantly reduced by using surface mount technology. The cost of one FEN board is approximately £15 and the SUIN cost is limited to a minimum by using shareware operating system and Java APIs. The overall system cost is, therefore, affordable for most small-to-medium enterprises.

The maintenance costs are minimal; the system can be checked and upgraded remotely requiring no site visit by the engineer. It is easy to use for the end user, as no expertise is required to interpret the results and web browsing is widely understood. The MCU used for FEN can acquire analogue signals up to 30 000 samples per second, which is sufficient for time-series analysis for the most common process signals. A limited frequency analysis capability, sufficient for most applications, is also achieved. The selected MCU also has 1536 bytes data memory, in addition to the system and peripheral special function registers, for data and variable storage. The developed program fits well within the program memory inside the MCU, and no external memory is therefore implemented. Several signal-processing subroutines are developed and the one specific to a signal's particular requirements can be selected for a particular FEN. The required automatic detection of nodes in the system has been achieved and up to 12 FENs can be used in a particular application. Each FEN is responsible for its own signal acquisition and processing. Additional FENs do not put any extra workload on already existing nodes. The SUIN uses separate MCs for every FEN eliminating the risk of message overwriting. The developed system has also achieved the capability of providing raw data for detailed signal analysis if required.

The proposed methodology is in-line with the technological advances, and the selected MCU may be replaced by the next generation MCUs if deemed necessary while keeping the same operational principles with enhanced processing capabilities. It is important to note that a FEN can be implemented on any microcontroller and it is not required to have the same MCU on all FENs. Similarly, all FENs may have different clock frequencies.

The proposed methodology has great potential for supporting processes of all types. It can provide an integrated approach to process monitoring and management and fault detection and identification. This work is an attempt to bridge the gap between existing monitoring techniques and the real-time implementation issues for wide scale industrial use. It is observed that monitoring can result in great economic benefits, but the cost and difficulty of installing a monitoring system can impede its deployment. The use of microcontrollers has been reported in several applications and a methodology is described in this paper to use them as a complete monitoring system. Only simple mathematics and logic decisions are used to make computations possible on 8-bit microcontrollers. Software models for FENs as well as for a remote user interface and synchronizing node are discussed, and communication issues are resolved. The process controller signals are used for fault detection and the sensor signals for fault isolation. An attempt is made to detect the developing fault before it grows out of control and causes quality deterioration and economic losses.

The resulting system has great potential. It is low cost, easy to install, and acts independently at each level to reduce the time and effort required for the fault detection and identification. It has even greater potential than has thus far been demonstrated, with the advent of ever more powerful microcontrollers, making it possible to expand and enhance the system to truly support the embedded intelligent process monitoring and management.

ACKNOWLEDGEMENT

Q Ahsan is a PhD student sponsored by the Ministry of Science and Technology, Pakistan.

REFERENCES

1 **Hartley, J.** Field based systems, asset management and advanced diagnostics. Seminar on control loop performance assessment, 2002, 10/1–10/6 (IEE, London).
2 **Drucker, P.** Managing for business effectiveness. *Harv. Bus. Rev.* 1963, **41**(3), 53–61.
3 **Trenchard, A., Desborough, L.,** and **Miller, R.** Managing control systems as asset. Seminar on control loop performance assessment, 2002, 1/1–1/3 (IEE, London).
4 Emerson process management, available from http://plantweb.emersonprocess.com/customer/chemical_index.asp, last updated 12 May 2003, accessed 28th July 2004.
5 **Schafer, J.** and **Cinar, A.** Multivariable MPC system performance assessment, monitoring and diagnosis. *J. Process Control*, 2004, **14**, 113–129.
6 **Mosca, E.** and **Agnoloni, T.** Closed-loop monitoring for early detection of performance losses in feedback control systems. *Automatica*, 2003, **39**(12), 2071–2084.
7 **Gustafsson, F.** and **Graebe, S. F.** Closed-loop performance monitoring in the presence of system changes and disturbances. *Automatica*, 1998, **34**(11), 1311–1326.
8 **Hu, W., Starr, A. G., Zhou, Z.,** and **Leung, A. Y. T.** A systematic approach to integrated diagnosis of flexible manufacturing systems. *Int. J. Mach. Tools Manuf.*, 2000, **40**, 1587–1602.
9 **Raaphorst, A. G. T., Netten, B. D.,** and **Vingerhoeds, R. A.** Automated fault-tree generation for operational fault diagnosis. In Proceeding of the Electric Railways in a United Europe, IEE, 27–30 March 1995, pp. 173–177.
10 **Hu, W., Starr, A. G.,** and **Leung, A. Y. T.** Operational fault diagnosis of manufacturing systems. *J. Mater. Process. Technol.*, 2003, **133**, 108–117.
11 **Andrews, J. D.** and **Dunnett, S. J.** Event-tree analysis using binary decision diagrams. *IEEE Trans. Reliab.*, 2000, **49**(2), 230–238.
12 **Ender, D. B.** Process control performance: not as good as you think. *Control Eng.*, 1993, **40**(10), 180–190.
13 **Miller, R. M.** and **Desborough, L. D.** Web enabled control loop assessment. *Chem. Eng.*, 2000, 76–80.
14 **Patwardhan, R. S.** and **Shah, S. L.** Issues in performance diagnostics of model-based controllers. *J. Process Control*, 2002, **12**, 413–427.
15 **Xia, C.** and **Howell, J.** Loop status monitoring and fault localisation. *J. Process Control*, 2003, **13**, 679–691.
16 **Horch, A.** *Condition monitoring of control loops*. PhD Thesis, School of Electrical Engineering, Royal Institue of Technology, Stockholm, Sweden, 2000, ISBN 91–7170–638–0.
17 **Hagglund, T. A.** Control loop performance monitor. *Control Eng. Pract.*, 1995, **3**(11), 1543–1551.
18 **Rengaswamy, R., Hagglund, T.,** and **Venkatasubramanian, V. A.** Qualitative shape analysis formalism for monitoring control loop performance. *Eng. Appl. Artif. Intell.*, 2001, **14**, 23–33.
19 **Hagglund, T.** Automatic detection of sluggish control loops. *Control Eng. Prac.*, 1999, **7**, 1505–1511.
20 **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, pp. 1–6 (Elsevier).
21 **Choudhury, M. A. A. S., Shah, S. L.,** and **Thornhill, N. F.** Diagnosis of poor control-loop performance using higher-order statistics. *Automatica*, 2004, **40**, 1719–1728.
22 **Goulding, P. R., Lennox, B., Sandoz, D. J., Smith, K. J.,** and **Marjanovic, O.** Fault detection in continuous processes using multivariate statistical methods. *Int. J. Syst. Sci.*, 2000, **31**(11), 1459–1471.
23 **Desforges, M. J.** and **Sandoz, D. J.** A prototype for hierarchical process condition monitoring. *IEE Seminar on Sensors and Actuators*, 2001, 7/1–7/6.
24 **Desforges, M. J., Marjanovic, O., Lennox, B.,** and **Sandoz, D. J.** Prototype for hierarchical process condition monitoring. *Comput. Control Eng. J.*, 2002, **13**(5), 254–258.
25 Perceptive Engineering Ltd., available from http://www.perceptive-engineering.co.uk/html/about_us1.ht ml, accessed on 6 August 2004.

26 **Lennox, B., Montague, G. A., Hiden, H. G., Kornfeld, G.,** and **Goulding, P. R.** Process monitoring of an industrial fed-batch fermentation. *Biotechnol. Bioeng.*, 2001, **74**(2), 125–135.

27 **Wong, C. F., McDonald, K. A.,** and **Palazoghu, A.** Classification of abnormal plant operation using multiple process variable trends. *J. Process Control*, 2001, **11**, 409–418.

28 **Ruiz, M., Colomer, J., Colprim, J.,** and **Melendez, J.** Multivariate statistical process control for situation assessment of a sequencing batch reactor. Control Conference, University of Bath, UK, 2004, ID-115 (Hadleys, Essex).

29 **Biswas, G., Kapadia, R.,** and **Yu, X. W.** Combined qualitative–quantitative steady-state diagnosis of continuous-valued systems. *IEEE Trans. Syst. Man Cybern. – Part A: Syst. Humans*, 1997, **27**(2), 167–185.

30 **Isermann, R.** Supervision, fault-detection and fault-diagnosis methods – an introduction. *Control Eng. Pract.*, 1997, **5**(5), 639–652.

31 **Davey, A., Grosvenor, R., Morgan, P.,** and **Prickett, P.** Petri-net based machine tool failure and diagnosis. In Proceedings of the COMADEM' 96, Sheffield, UK, 16–18 July 1996, pp. 723–731 (Academic Press, Sheffield).

32 **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. 5th International Conference on *Quality, reliability, and maintenance*, 2004, pp. 67–70 (Professional Engineering Publishing, London and Bury St Edmunds).

33 **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. 5th International Conference on *Quality, reliability, and maintenance*, 2004, pp. 71–74 (Professional Engineering Publishing, London and Bury St Edmunds).

34 **Studzinski, J.** Application of monitoring technologies in environmental engineering. 5th International Conference on *Quality, reliability, and maintenance*, 2004, pp. 43–46 (Professional Engineering Publishing, London and Bury St Edmunds).

35 **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. *Inf. Fusion*, 2003, **4**, 217–229.

36 **Kendra, S. J.** and **Cinar, A.** Controller performance assessment by frequency domain techniques. *J. Process Control*, 1997, **7**(3), 181–194.

37 **Higham, E. H.** Casting a crystal ball on the future of process instrumentation and process measurements. IEEE 9th Instrumentation and Measurement Technology Conference IMTC/92, pp. 687–691 (IEE, New York).

38 **Zhu, H., Higham, E. H.,** and **Amadi-Echendu, J. E.** Assessing plant condition from analysis of pressure and differential measurement signals. IEEE Instrumentation and Measurement Technology Conference IMTC/94, 10th Anniversary, Advanced Technologies in I and M, 1994, pp. 825–828 (IEE, Hamamatsu, Japan).

39 **Amadi-Echendu, J. E.** and **Higham, E. H.** Additional information from flowmeters via signal analysis. *IEEE Trans. Instrum. Meas.*, 1990, **39**(6), 998–1003.

40 **Higham, E. H.** and **Perovic, S.** Predictive maintenance of pumps based on signal analysis of pressure and differential pressure (flow) measurements. *Trans. Inst. Meas. Control*, 2001, **23**(4), 226–248.

41 **Ramakrishna, K.** Development of a computerized on-line vibration monitoring, analysis and assessment system for power plant machinery. *Int. J. COMADEM*, 2001, **4**(2), 5–12.

42 **Jeng, J. J.** and **Wei, C. Y.** An on-line condition monitoring and diagnosis system for feed rolls in the plate mill. *J. Manuf. Sci. Eng.*, 2002, **154**, 52–57.

43 **Angeli, A.** An online expert system for fault diagnosis in hydraulic systems. *Expert Syst.*, 1999, **16**(2), 115–120.

44 **Tseng, P. C.** and **Chou, A.** The intelligent on-line monitoring of end milling. *Int. J. Mach. Tools Manuf.*, 2002, **42**, 89–97.

45 **Bonastre, A., Ors, R.,** and **Peris, M.** Distributed expert systems as a new tool in analytical chemistry. *Trends Anal. Chem.*, 2001, **20**(5), 263–271.

46 **Paulonis, M. A.** and **Cox, J. W**. A practical approach for large-scale controller performance assessment, diagnosis, and improvement. *J. Process Control*, 2003, **13**, 155–168.

47 **Manders, J., Barford, L. A.,** and **Biswas, G.** An approach for fault detection and isolation in dynamic systems from distributed measurements. *IEEE Trans. Instrum. Meas.*, 2002, **51**(2), 235–240.

48 **Ong, S. K., An, N.,** and **Nee, A. Y. C.** Web-based fault diagnosis and learning system. *Int. J. Adv. Manuf. Technol.*, 2001, **18**, 502–511.

49 **De Frutos, J. A.** and **Giron-Sierra, J. M.** Design of a distributed system architecture including an automatic code generator. *Microprocess. Microsyst.*, 2002, **26**, 207–213.

50 **Ehrlich, J., Zerrouki, A.,** and **Demssieux, N.** Distributed architecture for data acquisition: a generic model. In Proceedings of the IEEE Instrumentation and Measurement Technology Conference, Ottawa, Canada, 19–21 May 1997, pp. 1180–1185 (IEEE, Ottowa, Canada).

51 **Mittal, A., Manimaran, G.,** and **Murthy, C. S. R.** Dynamic real-time channel establishment in multiple access bus networks. *Comp. Commun.*, 2003, **26**(2), 113–127.

52 **Baccigalupi, A., Bernieri, A.,** and **Pietrosanto, A.** A digital-signal processor-based measurement system for on-line fault detection. *IEEE Trans. Instrum. Meas.*, 1997, **46**(3), 731–736.

53 **Baek, D. K., Ko, T. J.,** and **Kim, H. S.** Real time monitoring of tool breakage in a milling operation using digital signal processor. *J. Mater. Process. Techn.*, 2000, **100**, 266–272.

54 **Roberts, C., Dassanayake, N., Lehrasab, N.,** and **Googman, C. J.** Distributed quantitative and qualitative fault diagnosis: railway junction case study. *Control Eng. Pract.*, 2002, **10**, 419–429.

55 **Nieva, T.** and **Wegmann, A.** A conceptual model for remote data acquisition systems. *Comp. Ind.*, 2002, **47**, 215–237.

56 Bolic, M., Drndarevic, V., and **Samardzic, B.** Distributed measurement and control system based on microcontrollers with automatic program generation. *Sensors Actuators A*, 2001, **90**, 215–221.

57 Dassanayake, H. P. B., Roberts, C., and **Goodman, C. J.** An architecture for system-wide fault detection and isolation. *Proc. Instn Mech. Engrs*, 2001, **215**(I), 37–46.

58 Livani, M. A., Kaiser, J., and **Jia, W.** Scheduling hard and soft real-time communication in a controller area network. *Control Eng. Pract.*, 1999, **7**, 1515–1523.

59 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 1: tool usage – cutting force relationship. *Int. J. Mach. Tools Manuf.*, 2000, **40**, 599–608.

60 Ahsan, Q., Amer, W., Grosvenor, R. I., and **Prickett, P. W.** Sweeping filter technique for frequency analysis. 5th International Conference on *Quality, reliability, and maintenance*, 2004, pp. 185–188 (Professional Engineering Publishing, London and Bury St Edmunds).

61 Amer, W., Ahsan, Q., Grosvenor, R. I., and **Prickett, P. W.** Machine tool signal analysis using sweeping filter technique. 5th International Conference on *Quality, reliability, and maintenance*, 2004, pp. 189–192 (Professional Engineering Publishing, London and Bury St Edmunds).

62 Kimmich, F., Schwarte, A., and **Isermann, R.** Fault detection for modern diesel engines using signal- and process model-based methods. *Control Eng. Pract.*, 2005, **13**, 189–203.

63 Siddiqui, R. A., Ahsan, Q., Grosvenor, R. I., and **Prickett, P. W.** The role of emerging technologies in eMonitoring. COMADEM Proceedings, 2005, vol. 18, pp. 263–271 (Cranfield University Press, UK).

64 Ahsan, Q., Grosvenor, R. I., and **Prickett, P. W.** A reduced traffic software model for distributed monitoring systems. COMADEM Proceedings, 2004, vol. 17, pp. 204–213 (University of Birmingham, UK).

65 Amadi-Echendu, J. E., Zhu, H., and **Atherton, D. P.** Development of intelligent flowmeters through signal processing. IFAC SICICA 92, Malaga, Spain, May 1992, pp. 23–28 (Pergamon Press, Elsevier, UK).

66 Palacherla, A. Implementation of fast fourier transforms. Microchip Application Notes AN542, 1997.

67 Lacoste, R. PIC spectrum – audio spectrum analyzer. *Circuit Cellar*, 1998, **98**, 24–30.