Applying Smart Sensor Technology to Existing Real-World (Legacy) Systems

Existing Assets and Open Standards Can Bring the Advantages of Smart Sensors to Legacy Systems

Robert N. Johnson
Telemonitor, Incorporated, Columbia, Maryland

1. INTRODUCTION

There is an increasing awareness of the advantages of using smart, communicating sensors (and actuators) in industrial control systems and data acquisition. However, these smart transducers are not being accepted and put into use as rapidly as some people had expected. One often-mentioned reason is the need to remove existing systems in order to install smart sensors.

This paper discusses how these existing (legacy) systems can migrate into a smart sensor-based architecture. It reviews the advantages of doing this and describes a pilot project at the University of North Carolina at Chapel Hill (UNC) to provide remote access to the university chilled water system. The project ultimately will allow the monitoring and limited control of chilled water systems in about one hundred buildings using existing campus assets.

2. WHAT IS A SMART SENSOR?

There are several different definitions of a smart sensor. One very good one, and the one we will use for this discussion, is found in the IEEE 1451.2-1997 standard. According to that document, a smart transducer is "A transducer that provides functions beyond those necessary for generating a correct representation of a sensed or controlled quantity. This functionality typically simplifies the integration of the transducer into applications in a networked environment."

That standard covers transducers, a general term encompassing both sensors and actuators. A smart sensor is therefore "A sensor version of a smart transducer."

The key concept is that a smart sensor must do more than give the correct answer or communicate in a digital format. A smart sensor adds value to the data in order to enable or support distributed processing and decision making.

There are several desirable functions that a smart sensor might provide. Not all of these functions need be in a single sensor in order for it to be considered smart, and there are many more functions that might be added to this list, but this list conveys the concepts that we are discussing.

Desirable features in a smart sensor include:

- Self identification
- Self diagnosis
- "Time aware" for time stamping and time correlation of multiple channels
- "Location aware" for position stamping and position correlation of multiple channels
- Higher order functions such as signal processing, data logging, event detection and reporting, and data fusion (deriving measurements from multiple channels)
- Conforms to standard data communications and control protocols

This last point is especially important because is helps to provide interoperability, either peer-to-

peer or between the smart transducers and an existing control or data acquisition system.

Figure 1 shows a general model of a smart sensor. This general model has been used to describe the IEEE 1451 family of standards for smart sensor interfaces, and the different members of that family each use a different partitioning of this model. Not all of the items shown in this figure necessarily are found in any given smart sensor.

The model shows the complete range of functions of a smart sensor from the real-world analog sensing element on the left through conditioning and conversion to the digital domain and finally to the communications network on the right. Note that there are sensor elements that are inherently digital and do not need the analog to digital converter or maybe even the signal conditioning. One of the simplest sensors is, for example, a limit switch.

The data storage can be used for data logging as well as for device parameters such as the Transducer Electronic Data Sheet (TEDS) defined in the IEEE 1451 standards.

The central box labeled “Application algorithms” provides the intelligence to make the sensor a smart device. This may include the correction engine that provides compensation and correction, and is where higher-level functions such as complex digital signal processing, logging, event detection, data fusion, etc. take place.

This figure shows only a single-channel device whereas real sensors may have multiple channels. For example, a temperature sensor may be included to support temperature compensation in the correction engine. Also not shown in this figure are the elements that provide time and location awareness. These may both be provided by something like a GPS receiver, or the location can be part of the user-entered data and the time can be synchronized over the communications network using traditional protocols such as Network Time Protocol (NTP) or newer ones such as IEEE P1588.

As noted in the bottom of the figure, smart transducers move intelligence closer to the point of action to make it cost effective to integrate and maintain distributed systems.

3. THE FUTURE OF SMART SENSORS

“As for the future, your job is not to foresee it, but to enable it.”

Antoine de Saint-Exupery

Of course we cannot actually foresee the future. We can, however, make intelligent extrapolations of current trends and estimate the future. More importantly, we can influence the future by making intelligent choices of which paths to pursue and which to avoid.

And smart sensors are very much an enabling technology that will influence the future applications of industrial controls and data acquisition systems.

So what can we extrapolate into an estimate of the future? The following discussion is generally limited to evolutionary items to show a possible transition path to the revolutionary changes that smart sensors can enable if used to their full potential.

The first extrapolation of present trends is that someday virtually all non-trivial sensors will be smart to some degree. Certainly a much higher percentage of them will be self-identifying and communicating.

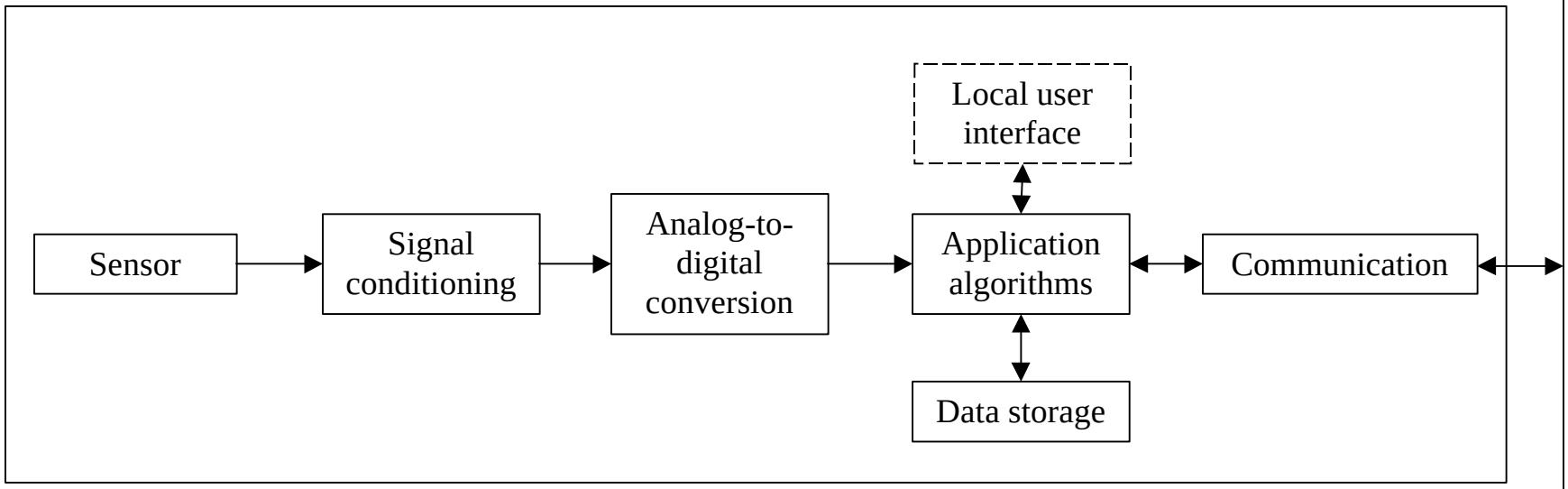
Open standards such as the IEEE 1451 family will evolve to define interoperability of smart transducers, but meanwhile we have to deal with a large installed base of “legacy” systems. We need a transition path to the future use of these new standards, and existing open standards and other assets may help define such a path

4. LEGACY IS NOT A FOUR-LETTER WORD

Some people seem to equate “legacy” with “obsolete,” whereas a lot of legacy devices have some very desirable characteristics:

Network

Network specific



Some points regarding "smart":

- Moving intelligence closer to the point of measurement/control.
- Confluence of transducers, computation and communication towards common goal.
- Goal: make it cost effective to integrate/maintain distributed systems.

Figure 1. A general model of a smart sensor.

- Industrial strength I/O
- Local high-level functions and operator interface
- Digital data interface (frequently RS-232 or RS-485)
- Already installed and paid for
- THEY WORK!

We as a society have a tremendous investment in these legacy systems that we need to exploit, not view as an impediment (although in some instances they do fit that description also).

The bottom line is that it is a much harder sell for the benefits of smart sensors if step one of the process is to break existing, proven applications.

5. CONVERTING LEGACY SENSORS TO SMART SENSORS

So how do we convert an installed device into a smart sensor? Basically by adding any missing elements from the previous list of desirable characteristics of a smart sensor.

For some sensors this will involve adding everything including analog signal conditioning and conversion to the digital domain, but for others the only requirement will be to convert an existing digital communications interface to a standard network protocol and add selected high level functions like data logging and event detection. All of these new functions can be added external to the existing sensor.

In addition to the legacy system itself, there generally are a number of other useful assets available to assist with the conversion. The primary ones are:

- Device serial ports
- Ethernet
- Control software

While some devices have strictly an analog output, others already have built-in serial ports. Frequently these serial ports either are totally unused or are used for configuration and then forgotten. UART-based (asynchronous) interfaces such as RS-232 and RS-485 are common, as is one of the Modbus protocols. Other devices have proprietary serial interfaces or protocols that may or may not be openly described to the public.

The second useful asset is Ethernet. Frequently Ethernet is already installed in the building, and may be available in the vicinity of the legacy system installation. There are legitimate concerns regarding the use of Ethernet in industrial applications, but it is a widely used, highly understood technology that almost certainly will evolve to be accepted as an industrial-grade network.

The third useful existing asset is enterprise software packages that support Web-based data interfaces. These interfaces include HTTP, XML, and the emerging class of protocols that are generally known as Web services.

Hypertext Transfer Protocol (HTTP) is the underlying protocol for Web pages and is generally available across any Internet Protocol (IP) network, including Ethernet and the Internet, regardless of the use of switches, routers, firewalls, or other network equipment.

The Extensible Markup Language (XML) is similar to the HTML used for Web pages except it includes provision for defining the meaning of the data in the page, not just the format for displaying it in a Web page. XML can be transported across networks using HTTP so it is rapidly becoming the standard for transferring machine-readable data across IP networks. Many of the recent control and enterprise management software packages support XML as a means of data exchange with remote devices.

Web services are a rapidly evolving technology for letting computers talk to each other over IP networks including the Internet. Basically they add the capability to remotely invoke processes to the data access that is provided by HTTP and XML.

There are competing camps for Web services (generally Microsoft versus non-Microsoft solutions) and a complete discussion of Web services is well beyond the scope of this paper, but it is likely that they will become increasingly important in the future.

These existing and emerging open standards will enable the future of smart sensing, and can be used to provide an evolutionary path to that vision of the future.

6. REAL-LIFE CASE STUDY

The University of North Carolina at Chapel Hill (UNC) is a large extended multi-campus facility spanning miles of area around Chapel Hill. They use chilled water extensively, both for cooling the buildings and for refrigeration of storage areas such those used by the medical department.

Not only are there multiple water chillers, but each building has a chilled water "bridge" that controls the amount of water from the main loop that is diverted through that building. Local process controllers are used to operate the chillers and bridges, but the only way to monitor the performance of the system is to physically visit each site.

The original motivation for this project was a mandate from the university to make all operational data available on the campus network in order to provide the information needed to operate all of the university utilities (chilled water and power) in a more efficient manner. As described, the chilled water system has installed instrumentation and control solutions that are working fine, but they did not have the network connectivity to enable relevant data to be widely distributed.

A second motivation was that the university decided to convert the chilled water department to a billed utility. This necessitated the collection and storage of flow rate and temperature drop data on a regular basis.

The UNC chilled water department needed a simple way to make these data available without redesigning the whole control system currently in place. In other words, UNC needed a convenient means to *encapsulate* the existing solutions that were present throughout the chilled water system and expose only what was necessary to meet the university mandate.

UNC had several valuable assets already in place to accomplish this. First, the installed process controllers had an RS-485 Modbus RTU interface that provided digital access to all the necessary data. Second, the upgraded enterprise-level plant control software that was to be used for the billing process already had a Web interface. And third, the UNC campus is wired with an extensive high-speed Ethernet network that included virtual private networks (VPNs) between the various remote sites.

What was needed was a way to encapsulate a local control solution in such a way as to provide a universal connection to the campus Ethernet LAN consisting of multiple clients supporting commonly available open-standard Web protocols. This basic configuration is shown in figure 2. Not shown in figure 2 are the firewalls and other equipment necessary to provide the VPNs, or the large technology-savvy student body. Perhaps when we illustrate islands of automation we need to remember that those may be shark-infested waters.

7. DEMONSTRATION PROJECT

A feasibility demonstration was conducted in 1999 and reported at Sensors Expo in the Fall of that year. The remote access interface consisted of a Web server with a Modbus interface that was connected to each process controller. We used the in-place campus (and inter-campus) Internet access for physical connectivity. We used a fire wall-friendly Web-based interface for the communications protocol.

The communications protocol for the demonstration was for the monitoring site to request a specific Universal Resource Locator (URL) that included parameters to specify the specifics of the data desired. This was sent to the Web server as an HTTP "Get" command. The Web server then returned the requested data in text form as an HTML Web page. These data could be displayed in a standard Web browser program, or programmatically collected by Access, Excel, Visual Basic, or any other program with a Web interface.

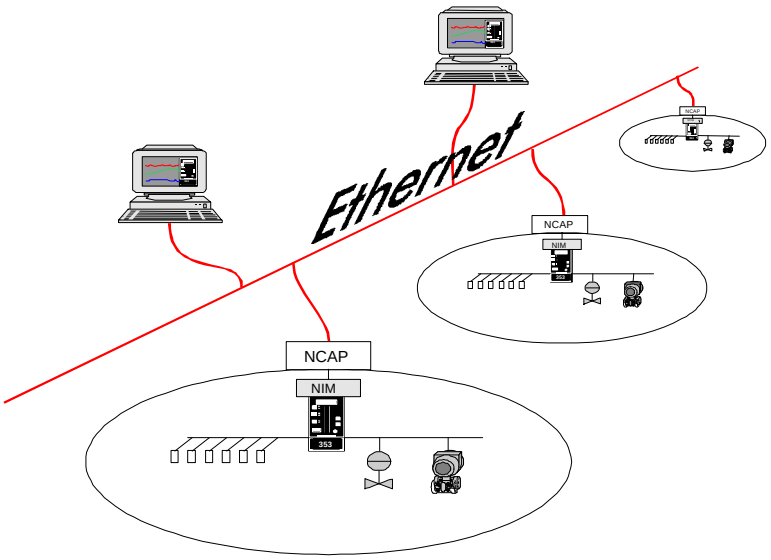


Figure 2. Bridging Islands of Automation with open standards.

The Web servers used for this demonstration program are shown in figure 3.

Figure 3. Web server.

Actual data from a chilled water Bridge is shown in figure 4. This unit is attached to the controller for a small water chiller plant located on an auxiliary site a few miles from the main UNC campus. This shot was captured from a computer in the main chiller water facility and illustrates the ability to monitor remote facilities using in-place infrastructure such as Ethernet and computers with web browsers. The trend plot is not very interesting, but for real-life data that frequently is the desired result!

At the time this screen shot was captured, the chilled water pump was not running (flow essentially zero), the inlet and outlet temperatures were both approximately 65º F, and the outside temperature was 34º F. The low outside temperature explains why there was not much demand for chilled water that particular day.

This project successfully demonstrated the feasibility and flexibility of using Web servers and Internet connectivity for remote monitoring of the UNC process data. The next phase consists of installing the system in approximately 100 buildings on multiple campuses.

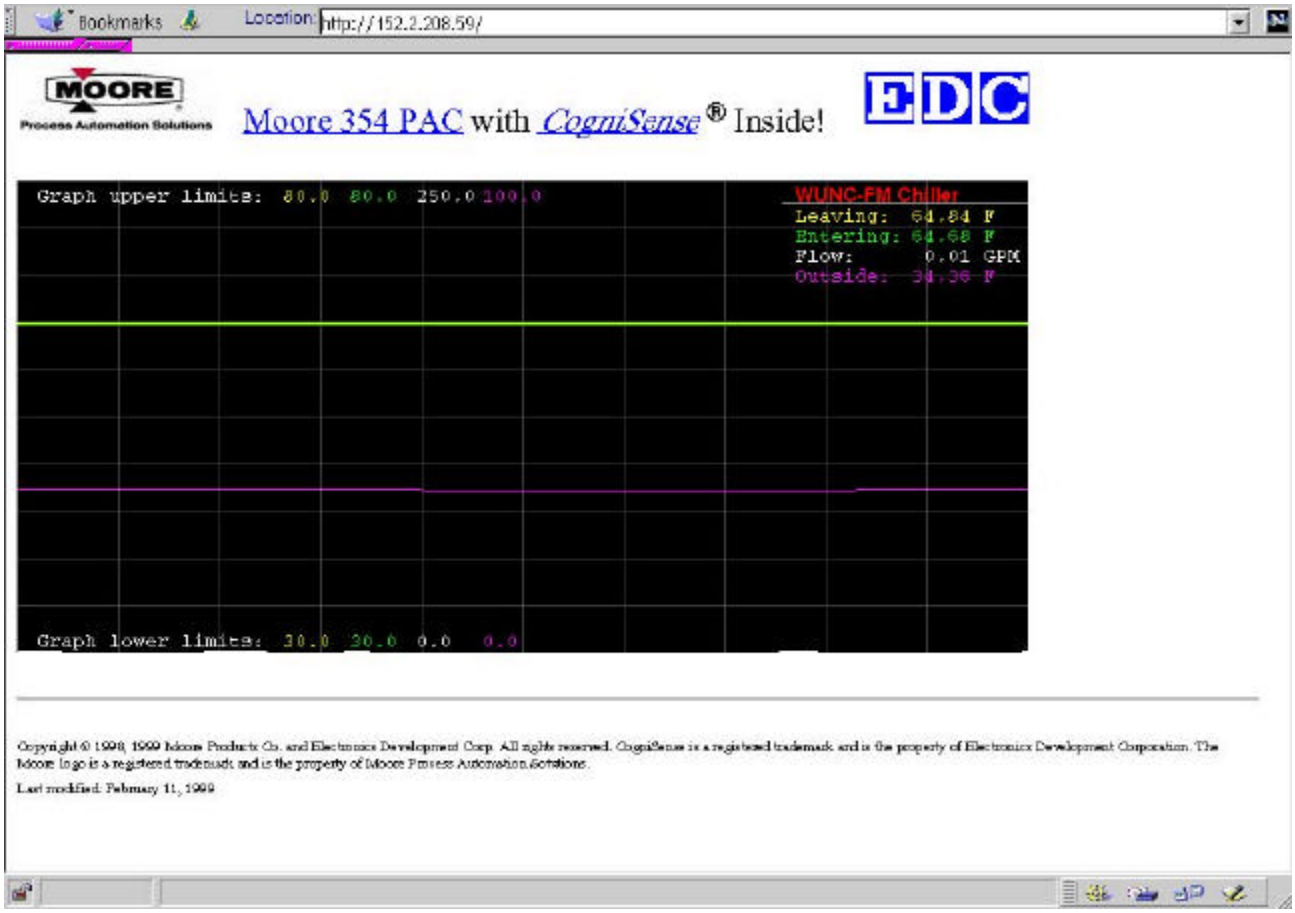


Figure 4. Monitoring Chilled Water Plant at UNC.

For the next phase we added an option to return the data in XML format instead of straight Web pages to simplify the parsing of the data and storing it in a database. Also, the Web server has been embedded in the process controller instead of being in a separate box, but still uses the RS-485 Modbus interface.

8. WEB SERVER FEATURES

A graphical Web page like the one shown in figure 4 is valuable for setup, configuration, and troubleshooting, but the real day-to-day work is done by a database server somewhere that is collecting and storing data and using that data for something like the UNC billing application or for trend analysis. Even for a relatively small enterprise-level system like the 100 buildings at UNC, it is not really practical to have an operator visit each Web site regularly and note the status and collect data.

The Web server needs to be more sophisticated than just serving static (unchanging) pages or acting as a simple protocol bridge. Some examples of important features include:

- Simple network configuration
- Standalone data logger
- Dynamic data display
- Standard file system (with FTP)
- Real-time clock
- Multi-level user control
- Firewall-friendly

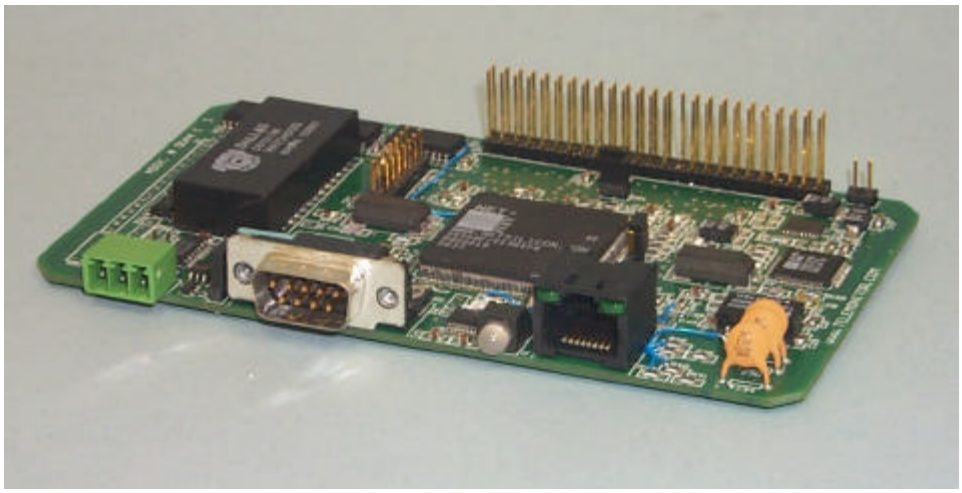


Figure 5. Embedded Web server.

Figure 5 illustrates one example of an embedded Web server that exhibits some of these characteristics. This board-level product is approximately 3 x 5 inches.

Two of the important Web server features mentioned earlier are discussed individually.

Easy Network Configuration

The network configuration support is frequently not considered in the selection of a server, but it should be. Unlike a PC that has a keyboard and monitor, a true embedded server may have no operator interface at all. In that case, how do you set the network configuration in order to install it on your local network?

Dynamic Host Configuration Protocol (DHCP) is appropriate for some applications, but certainly not for all. The remote sensors may be on a local sub-net that does not have a DHCP server. Or maybe the application cannot stand to have the sensor's IP address change every time power is cycled. And even if DHCP is appropriate, how do you find out to what IP address the server is set?

A network configuration utility like the one shown in figure 6 can find all the devices on the local sub-net, view and change the network settings and other useful information such as a self-identification tag for that sensor. Note than some of the devices are using DHCP and the configuration program reports their network configuration settings.

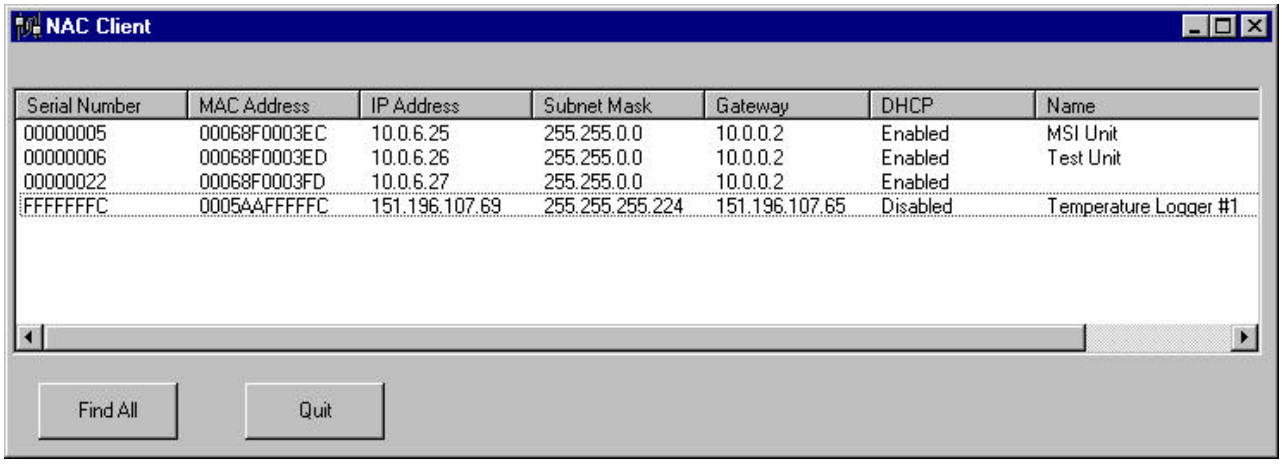


Figure 6. Network configuration utility.

Data Logger with Time-stamped Records

A data logger is a very valuable but also somewhat complex feature to configure. The logger configuration controls must allow you to specify several key parameters such as exactly what data to log, and at what sampling interval.

Note that the data logger configuration Web page shown in figure 7 is a good example of a Web-based configuration utility.

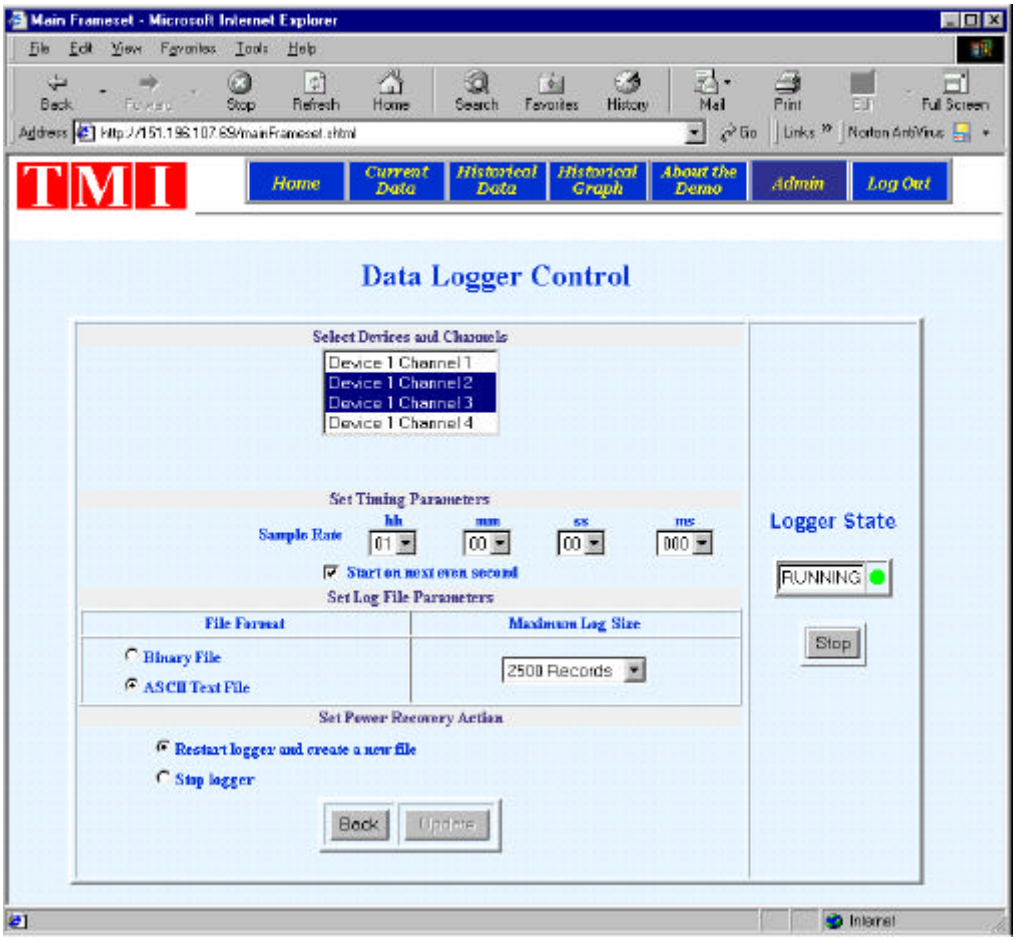


Figure 7. Data logger configuration page.

The logger should continue to run even when a browser is not logged into the unit. This seems obvious, but it eliminates the use of client-side Web page devices such as Java Applets and ActiveX controls. The logger program must be a standalone application in the server.

The server should provide access to the logger data through a firewall-friendly Web interface such as HTTP and XML.

And finally, the logger should log system events (alarms, power loss, etc.) as well as the data.

Figure 8 shows a sample graphical Web page display from such a logger. These data are also available as an ASCII or binary file via FTP, as well as in XML format through an HTTP page request.

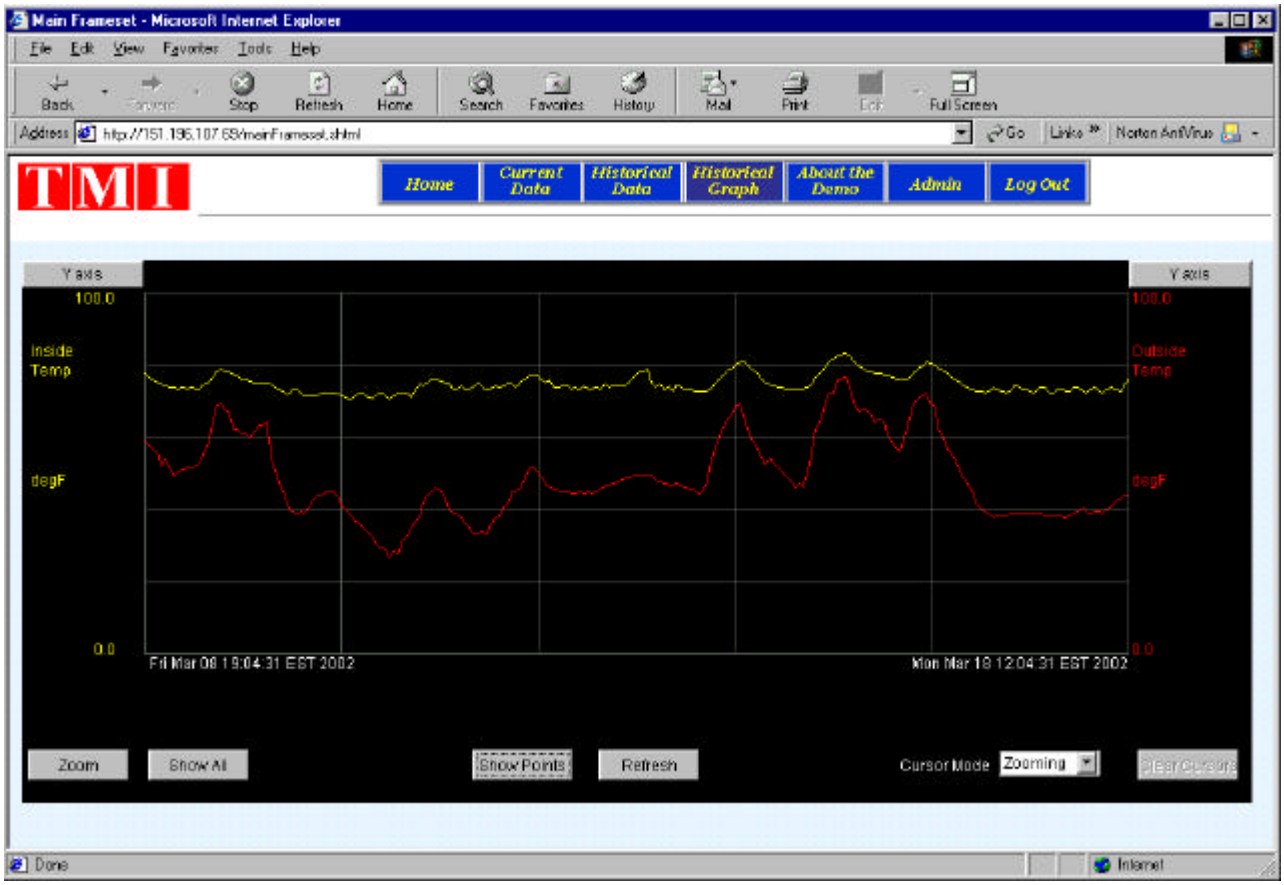


Figure 8. Data logger graphical display.

9. SUMMARY

We expect to continue to move toward smart sensors in the future, but we still need to preserve and use our investment in legacy devices and working systems.

Embedded Web servers and open standards can help with the transition by taking advantage of existing assets:

- Many existing devices already have serial ports
- Many installations already have Ethernet
- Many enterprise software packages already support Web interfaces.

Ethernet and the Internet protocol suite already have had a major impact on office operations and business data collection and management. They potentially can have the same impact on industrial control and data acquisition applications, and the techniques described in the paper should be considered for an evolutionary path toward that goal.

Robert N. Johnson is president of Telemonitor, Inc. He can be reached by voice at 410-312-6621, or by email at robertj@telemonitor.com.