Proceedings of IMECE2005
2005 ASME International Mechanical Engineering Congress and Exposition
November 5-11, 2005, Orlando, Florida USA

IMECE2005-79403

INTELLIGENT SENSORS - STRATEGIES FOR AN INTEGRATED SYSTEMS APPROACH

Sanjeevi Chitikeshi[1], Ajay Mahajan[2], Pavan Bandhil[3]
Department of Mechanical Engineering and Energy Processes
Southern Illinois University at Carbondale
Carbondale, IL 62901

Lucas Utterbach[4]
Department of Computer Science
Southern Illinois University at Carbondale
Carbondale, IL 62901

Fernando Figueroa[5]
NASA
John C. Stennis Space Center
Technology Development and Transfer
Code HA30, Bldg. 8306
Stennis Space Center, Mississippi 39529-6000

1. Graduate Student 2. Professor 3. Graduate Student
4. Graduate Student 5. Scientist

ABSTRACT

This paper proposes the development of intelligent sensors as an integrated systems approach, i.e. one treats the sensors as a complete system with its own sensing hardware (the traditional sensor), A/D converters, processing and storage capabilities, software drivers, self-assessment algorithms, communication protocols and evolutionary methodologies that allow them to get better with time. Under a project being undertaken at the Stennis Space Center, an integrated framework is being developed for the intelligent monitoring of smart elements. These smart elements can be sensors, actuators or other devices. The immediate application is the monitoring of the rocket test stands, but the technology should be generally applicable to the Intelligent Systems Health Monitoring (ISHM) vision. This paper outlines progress made in the development of intelligent sensors by describing the work done till date on Physical Intelligent Sensors (PIS) and Virtual Intelligent Sensors (VIS).

INTRODUCTION

The need for intelligent sensors as a critical component for Integrated System Health Management (ISHM) is fairly well recognized by now. Even the definition of what constitutes an intelligent sensor (or smart sensor) is well documented and stems from an intuitive desire to get the best quality measurement data that forms the basis of any complex health monitoring and/or management system. If the sensors, i.e. the elements closest to the measurand, are unreliable then the whole system works with a tremendous handicap. Hence, there has always been a desire to distribute intelligence down to the sensor level, and give it the ability to assess its own health thereby improving the confidence in the quality of the data at all times.

Sensors are a critical component of complex and sophisticated systems of today's technology and their role is ever evolving in the smart systems of tomorrow. General theories to treat intelligent sensor systems have been reported in the literature since the mid 80's [1-3]. Parallel work was done in industry where sensors have been developed with built in expert systems and look-up tables [4,5]. These sensors, called smart sensors, were described as simple sensing devices with built-in intelligence. This intelligence included simple decision-making capabilities, data processing, conflict resolution, communications, or distribution of information. It was explained by Figueroa and Mahajan [6] that the autonomous sensor was defined as a sensor that had an expert system with extensive qualitative tools that allowed it to evolve with time into a better and more efficient system. It differed, at least in philosophy, from the previous models by having a dynamic knowledge base as well as embedded qualitative and analytical functions that gave it a higher degree of operational independence, self-sufficiency and robustness. The underlying philosophy behind the autonomous sensor was probably closest to Henderson's [7,8] logical sensor models that also endeavored to give more problem-solving capabilities to the sensor, but still stayed away from any type of dynamic models.

DeCoste [9] described a system, called DATMI that dynamically maintained a concise representation of the space of local and global interpretations across time that were consistent with the observations. Each of the observations was obtained from a sensor, and therefore the number of observations was equal to the number of sensors in the control system. The truth of the observations and the validity of the sensors was obtained by cross-referencing with possible and impossible states of the system. DATMI was designed for a complete control system comprising of multiple sensors and actuators, and was the inspiration for the formalized theory called DATA-SIMLAMT *(Dynamic Across Time Autonomous - Sensing, Interpretation, Model Learning and Maintenance Theory)* which was designed for and is applicable to each sensor in the control system [10].

This paper proposes the development of intelligent sensors as an integrated system approach. Over the years some work has been done in this area, but most of the work has been for customized applications. It is certainly now time to think of generic models for such types of sensors that can be quickly fitted in to any application. Under a project being undertaken at the Stennis Space Center, an integrated framework is being developed for the intelligent monitoring of smart elements. These smart elements can be sensors, actuators or other devices. The immediate application is the monitoring of the rocket test stands, but the technology should be generally applicable to the ISHM vision. This paper outlines progress made in the development of intelligent sensors by describing the following:

- A strategy for using a qualitative approach to process data and recognize problems in the data and/or in the health of the sensor itself.
- The introduction of a condition assessment sheet for each sensor that functions as a report card, and allows the system to make critical decisions during or after a run.
- The development of an integrated environment to run these intelligent sensors monitoring real processes.
- A Physical Intelligent Sensor (PIS) or a smart sensor that connects directly to an Ethernet bus and has processing and storage capabilities embedded in it. Its output is data as well as an indicator of the quality of the data.
- A Virtual Intelligent Sensor (VIS) that takes data from a traditional sensor and has the same output as that of a PIS.

FORMALIZED THEORY FOR INTELLIGENT SENSORS

DATA-SIMLAMT *(Dynamic Across Time Autonomous - Sensing, Interpretation, Model Learning and Maintenance Theory)* [10] is a philosophy that has been inspired by the need for autonomous sensors, and these in turn were inspired by the need for autonomous systems. Some of the terms that will be used in this paper are defined as follows:

Property - is a parameter that has different state values based on the sensor performance, e.g. an amplitude check that monitors the amplitude of the current data point compared to the past few readings. It could have state values of (N)ormal or (H)igh signifying normal state of affairs or a potential problem.

Concept - is a set of properties with same state values, e.g. amplitude is high for a certain duration of time.

Behavior - is a set of concepts, e.g. a normal operation followed by a duration of very high amplitude may signify a problem such as a spike.

Envisionment - is a known, hence pre-defined, concept/ behavior similar to a known *pattern* in the *pattern recognition* problem, and is stored in the sensors' knowledge bases.

Eight properties with their state values (at any given time) form a pattern which constitute a concept as shown in Table 1.

Table 1: Some Properties and theirs description

	Properties	States	Description
1	Deviation_check	**H**igh, **N**ormal, **Z**ero	Obtained by comparing the standard deviation of past few readings with a pre-defined limiting check value. This is essentially a **noise level** check. The limiting value can also be extracted from an FFT analysis or a curve fitting routine.
2	Amplitude_ check	**H**igh, **N**ormal, **L**ow	Obtained by comparing the difference of the present value with the moving average of the past few readings. This gives the notion of a **sudden increase in the amplitude**.
3	Limit_check	**H**igh, **N**ormal, **L**ow	Obtained by comparing the current reading to pre-defined high and low limits. This is a **specification limit** check.
4	Estimate_check	**G**ood, **B**ad	Obtained by comparing the current value with an estimated value obtained from an FFT analysis, curve fitting, Kalman filter, etc. This is essentially a check for the **validity of the assumed model**.
5	Zero_check	**N**o, **Y**es	Obtained by checking for a zero reading. This is essentially a check for a **power failure**.
6	Sign_check	**P**lus, **M**inus, **S**ame	Obtained by checking against the past few readings. This is a check for **trends in measurand behavior**.
7	STC_check	**V**alid, **I**nvalid	This is obtained by checking to see if the sensor could possibly have detected a fast change. It is useful in identifying an **impossible situation**.
8	MTC_check	**V**alid, **I**nvalid	This is obtained by checking to see if the measurand could have changed at the given rate. It is essentially used to **identify external disturbances** to the sensor due to influences other than the measurand.

A concept is defined as a period in time in which the properties have the same state values. A concept, as stated earlier, is defined by the eight properties and their unique state

values. The states are represented by the first letter of the words shown in the third column of the table. It is these first letters, or symbols, that make up the patterns. Two or more concepts, in a definite order, constitute a behavior.

Example of a behavior comprising of two concepts is:
Spike(Present) :- Previous_Noise(Low) + Amplitude(High)

Previous_Noise(Low) is defined by the pattern shown on the Fig. 1. Each column represents the state values of the properties defined above. Consecutive columns of the same state values (or symbols) can be bunched together. The pattern for this concept signifies that the standard deviation was normal (low noise level), the amplitude was normal, the value was between specified limits, the value agreed with an estimated value, the trend was positive, and the limiting values for the sensor and measurand time constants were not violated in any way.

N	N	N	N
N	N	N	N
N	N	N	N
G	G	G	G
N	N	N	N
P	P	P	P
V	V	V	V
V	V	V	V

A similar pattern can be generated for the concept, Amplitude (High) as shown in Fig. 1. Hence, the sensor knowledge base consists of numerous such patterns. It must be emphasized here that this example has been illustrated by using only 8 generic properties. The proposed work goes much further and attempts to identify many more generic and sensor specific properties.

H
H
H
B
N
P
I
I

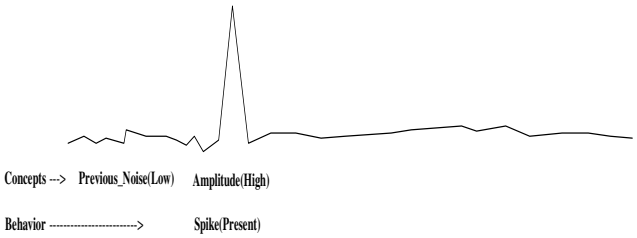

Figure 1: Spike behavior and Amplitude behavior

Fig 2 shows a typical example in how a behavior pattern, or envisionment, is recognized in real time data. As shown in Fig. 3 the numeric sensor data is converted to the symbolic data set in real time and a pattern recognition is done. The lightly shaded segment has been identified as Previous_Noise (Low) concept and the heavily shaded portion has been identified as Amplitude (High) concept. Together, in that order, the behavior of Spike (Present) has been identified this would cause the sensor to take appropriate action, which in this case could be to send a predicted value to the main controller rather than the actual data which is probably faulty.

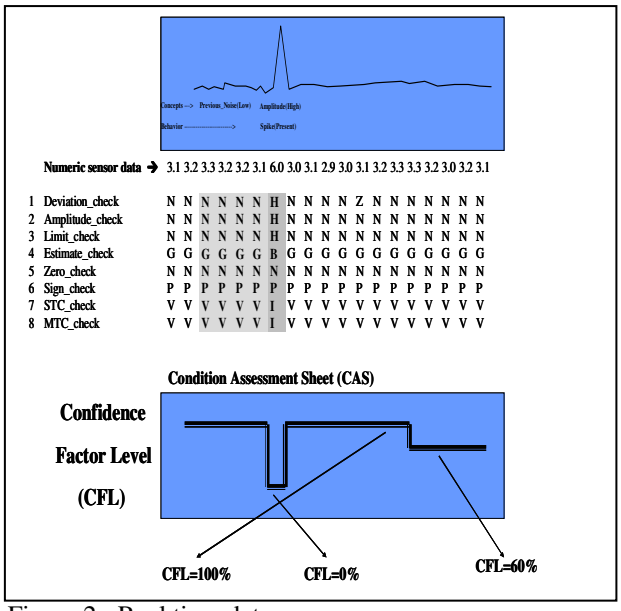

Figure 2: Real time data

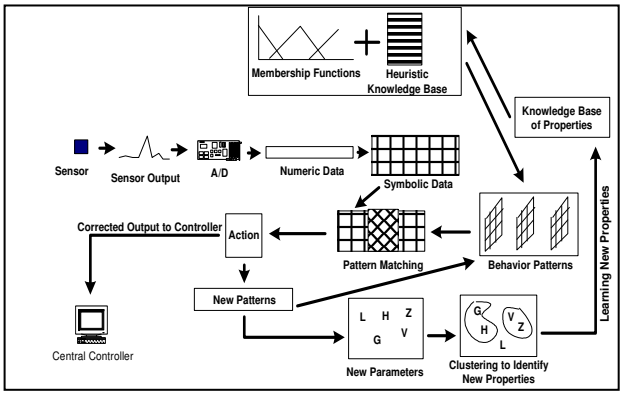

Figure 3: Pattern Recognition in real time data

CURRENT IHMS NETWORK MODEL

Fig. 4 shows the layout of the sensors and the governing Intelligent Systems Health Monitoring (ISHM) vision.

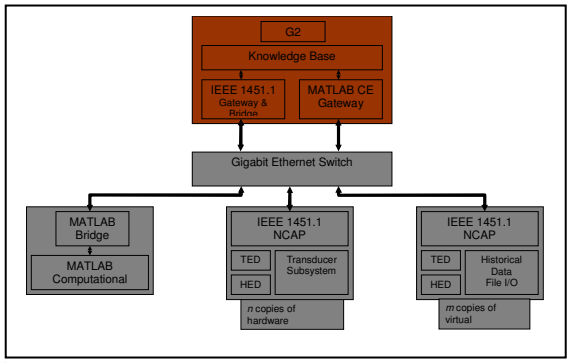

Figure 4: ISHM Layout

The whole system of intelligent and traditional sensors is tied together in the governing system designed in the G2 software environment. The overall system is being designed by

Stennis Space Center, but the scope of this work resides solely on the single sensor level. In this layout the brown is the software control of the IHMS system. The grey is external systems including from left to right

- an external Matlab program to do computation that G2 cannot e.g. FFT's, etc.
- hardware smart sensors from 1…..n
- virtual sensors from 1…..m

All of these external systems connect to G2 through a bridging program written in C\C++. The central system collects the data from the sensors and external programs and then applies it to the model of the system contained in its knowledgebase, as shown in Fig. 5. The model that this work is currently being applied to is that of a liquid oxygen tank at Stennis Space Center. Since this paper deals primarily in the single sensor realm a detailed discussion of the overall system will not be given.

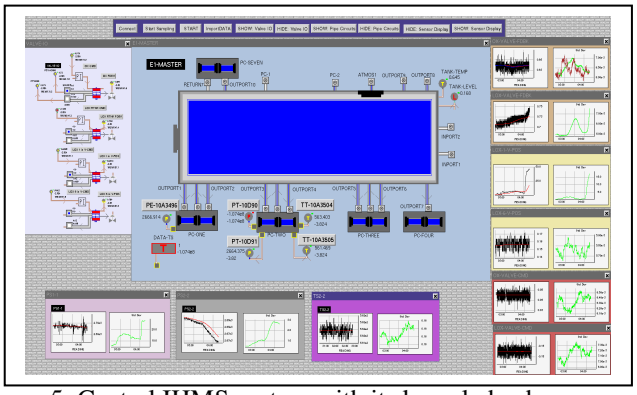

Figure 5: Central IHMS system with its knowledge base

Different types of sensors comprising a dynamic system will be instantiated as smart sensors that will fit within an object oriented integrated framework that uses embedded knowledge to monitor all the elements within the system. The concept of smart sensors may be extended to other types of elements as well as processes. Each element will have a specification sheet (SS) that will be fitted in to TEDS, and a new entity called the condition assessment sheet (CAS) that will be fitted into HEDS (see Fig. 2 for an example implementation). The CAS shows the condition (or the health) of the element and its confidence in its own working for the duration of the operation. This "health" information is provided over and beyond the numeric output of the sensor. These smart elements will have decision making capability derived from embedded knowledge bases and their own intrinsic specification sheets.

Networks of elements with autonomous character will cooperate to perform as a system composed of a collection of processes, each managing a collection of sensors, actuators, and other components. The emphasis is on knowledge bases that support each element of the hierarchy and the relationships between them. A key feature of the proposed framework is the evaluation of condition for all elements performed both autonomously and using feedback from other higher-order elements. The proposed effort will develop and validate a hierarchical intelligent architecture composed of a system, processes, and sensors. Each element, as mentioned before, will be a *smart or intelligent* entity, where they possess the capacity to perform actions, assess those actions, and modify actions based on self assessment and external assessment of results by others. Implementing this level of intelligence involves embedding agents within each element that communicate, integrate, and adapt based on access to knowledge bases and autonomous learning algorithms.

The smart elements will be developed in the MATLAB environment which is very conducive for research purposes, while the system integration will be done using the G2 software, which is designed to handle complex intelligent systems. G2 software has been chosen since it offers the opportunity to develop layered behaviors analogous to the hierarchical autonomous architecture we seek to develop. SSC provides an ideal test bed for this development effort due to ready access to a broad range of rocket engine test stands, associated data acquisition systems, and archival data. In addition, there is a large experienced user base, which can be drawn from to develop, refine, and validate the architecture elements.

PHYSICAL INTELLIGENT SENSOR (PIS)

The PIS or smart sensor is a combination of a sensing element, a data acquisition chip, a microprocessor and an Ethernet connection that allows one to directly connect the sensor to an Ethernet Bus. A PC is connected to the microprocessor to down load the software to the RAM and also debug the program. Once the software is running, this PC is removed and all the data is collected by a remote PC thru the Ethernet Bus. Several such smart sensors can be connected to the same Ethernet Bus and controlled by the single remote PC.

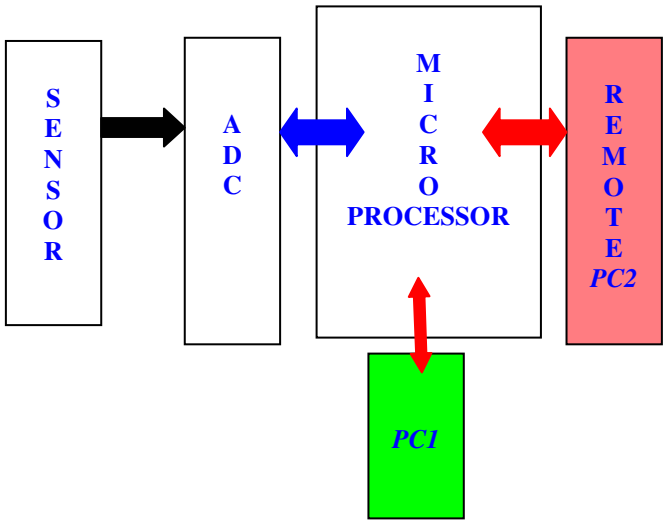

Figure 6: Block diagram of the PIS system

Fig. 6 is the block diagram of the PIS system. As shown in the block diagram the first block is the sensor connected to the Analog to Digital Converter (ADC), where the analog data from the sensor is converted into digital data. The ADC is then connected to the microprocessor) which is then connected to a computer with a programming cable (RS-232). The program is downloaded onto the microprocessor. In this current work ADC

0801 and ADC7794 are being used for Analog to Digital conversion. In these former one is parallel out converter and the later one is serial out converter. The microprocessor being used is the RCM3300. This microprocessor is compatible with Dynamic 'C' (9.01) software along with assembly language. Hence, one can do the software coding directly in Dynamic 'C' (9.01), thus reducing the complexity of the problem, and can see the results on the computer (stdout window). One can also send the programmed data from the microprocessor to a remote system using TCP\IP Protocols thru an Ethernet bus. The analog data from the sensor is given to the ADC. According to the specifications of the ADC, if the analog data from the sensor exceeds 5V then the analog voltage is scaled as 5V and is sent to the ADC. The same case is applied when the actual voltage is dropped below 0v i.e. dropped to negative voltage, then the voltage is scaled to 0V and is sent to the ADC. The analog data is converted into digital data and is supplied to the microprocessor (RCM3300), where this data is controlled by the software program installed on the microprocessor.

Fig. 7 shows the functional block diagram of the AD7794. The analog output from the sensor is given to the multiplexer and then the multiplexer output is given to the buffer. Then in order to amplify the signal from the sensor we have to give this signal to an operational amplifier due to the possibility of small strength signals. These signals are sent to the ADC where these analog signals are sampled and are converted into digital signals which are then fed to the microprocessor. This data is processed using the smart software program. The digital data can be available at the (digital out) DOUT pin of the AD7794 Evaluation board. When the data is ready at the DOUT pin then the SCLK pin goes high indicating to the microprocessor that the data is ready. The microprocessor then accepts the data and makes the (digital in) DIN pin high indicating the data transfer is complete. This process continues until there is data available at the DOUT pin.

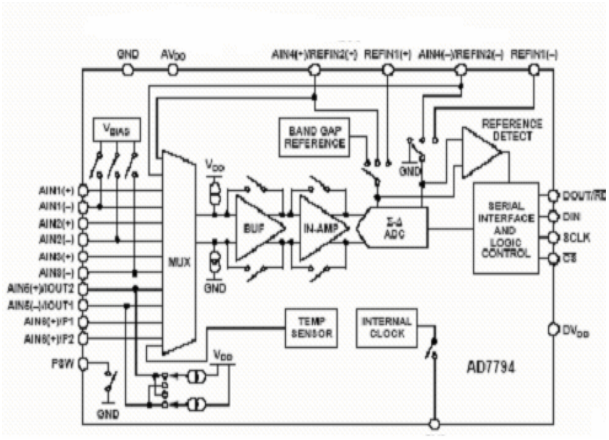

Figure 7: Functional Block diagram of the AD7794

The SCLK has a Schmitt-triggered input, making the interface suitable for opto-isolated applications. The serial clock can be continuous with all data transmitted in a continuous train of pulses. Alternatively, it can be a non continuous clock with the information being transmitted to or from the ADC in smaller batches of data. Clock In/Clock Out. The internal clock can be made available at this pin. Alternatively, the internal clock can be disabled and the ADC can be driven by an external clock. This allows several ADCs to be driven from a common clock, allowing simultaneous conversions to be performed. Chip Select Input. This is an active low logic input used to select the ADC. CS can be used to select the ADC in systems with more than one device on the serial bus or as a frame synchronization signal in communicating with the device. CS can be hardwired low, allowing the ADC to operate in a 3-wire mode with SCLK, DIN, and DOUT used to interface with the device.

The main interface part in the smart sensor is the interfacing between the ADC and Microprocessor. The output data from the ADC is a 24bit serial data or 8 bit parallel data, so there are three pin connections between the ADC and the microprocessor. The following are descriptions of the pins:

1. Interrupt pin from the ADC indicating that the data is ready at the ADC for transfer (SCLK pin from the ADC).
2. Data transfer pin (DOUT/READY pin from ADC).
3. Data acknowledgement pin from the microprocessor (DIN to the ADC).

When there is data ready at the ADC then the interrupt pin is high indicating that the data is ready for the microprocessor. When the microprocessor receives the interrupt signal from the ADC then the serial/parallel data is transferred to the microprocessor through the data transfer pin of ADC. When the data is transferred then the microprocessor sends a data acknowledgement signal to the ADC indicating the completion of the data transfer. As the data transfer includes serial/parallel data transfer, the clock of the microprocessor and ADC are made the same, so that there is a minimal probability of data loss. The clock adjustment and the port pins (I/O) of microprocessor are controlled by the software (Dynamic 'C') in the microprocessor. The actual physical set-up of this interface connections is shown in Fig. 8.

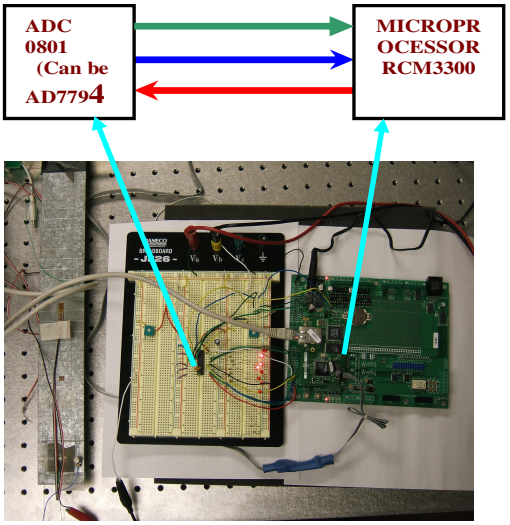

Figure 8: Interface between ADC and the Microprocessor

The microprocessor and the PC1 are connected by an RS32 programming cable. The code that has to be stored on the microprocessor has to be first written on PC1 and is then downloaded to the microprocessor. It is compiled and run by using the SRAM (or) FLASH (each of 512KB) of the microprocessor. There is a memory back up of 8MB in the form of serial flash on the RCM3300 core module to create files and to store data. This then has to be transferred to the remote PC2 for further analysis using TCP\IP protocols and Ethernet cable connected to the microprocessor and the remote PC2. The physical set-up is shown in Fig. 9.

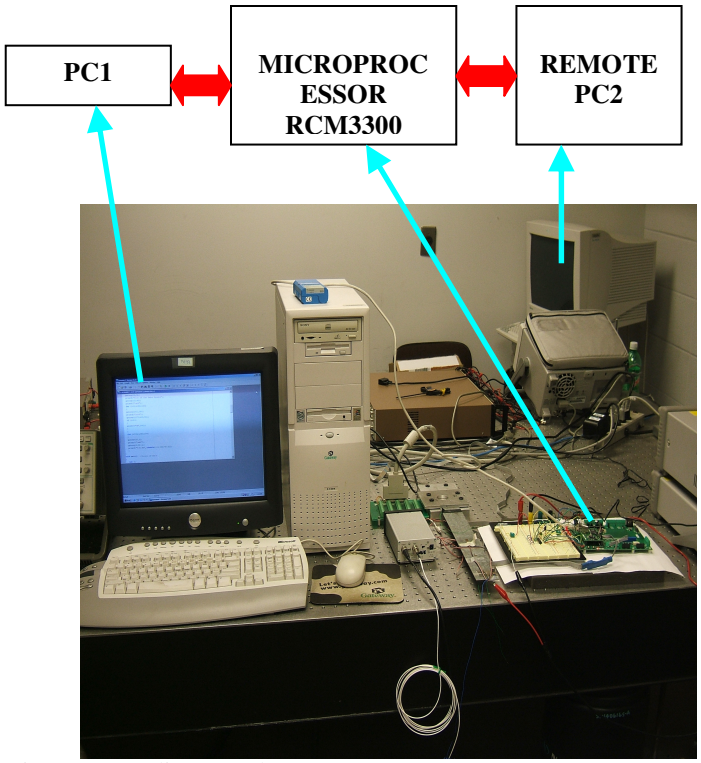

Figure 9: Sending Results to Remote computer

VIRTUAL INTELLIGENT SENSOR (VIS)

The realization of IHMS Intelligent sensors brings with it the need for virtual implementations of these sensors. The applications for intelligent sensors create a need for a way to test whether or not these sensors are applicable to various systems. The sensors themselves are somewhat limited in the amount of data storage available to archive and store programs, so a method for more powerful versions of these sensors is also useful. Having a software implementation of these sensors also lends itself to being an ideal test bed for future sensor development.

Fig. 10 shows an early realization of the governing system. The following objects in this G2 KB are the result of an earlier attempt to encapsulate the behavior of a Virtual Intelligent Sensor in G2.

- SMALL-SENSOR-SYSTEM window contains an instance of our sensor
- TEMP-SENSOR-READOUT window contains the GUI to monitor the values of the sensor.
  - the first graph is the actual reading and the average
  - the second graph (disabled in this image) is standard deviation over 5 seconds and 5 minutes
  - the table contains the values of the 20 most recent history properties for zero, limit and deviation check
- RULES-OP window contains the rules that are inferred upon every time the sensor receives a reading and then inserts appropriate values into the history lists. This populated history is then used to realize the DATA-SIMLAMT philosophy.

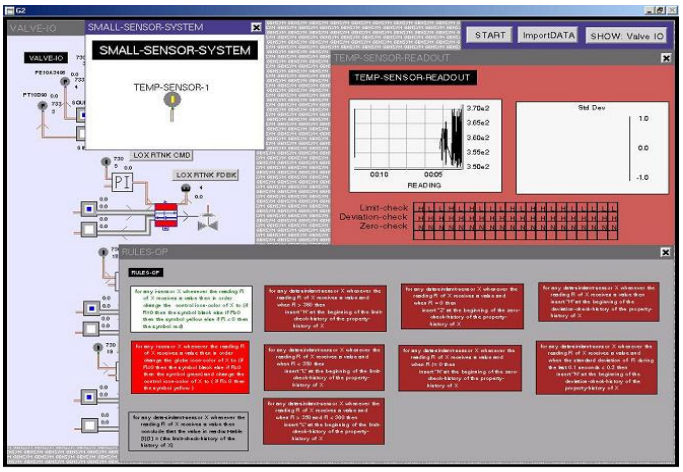

Figure 10: Early realization of Governing system

It was later decided that the VIS should consist of the actual code going into the sensor and not G2 rules. These rules were created as a proof of concept, as the history lists will later be populated with data from a Virtual Intelligent Sensor using the C\C++ routines actually used on the sensor as well. Since the governing system is created in G2 it grants a very powerful palette of tools that suit themselves very well to some of the ideas stated DATA-SIMLAMT. The G2 environment will later be used to do further processing of the data, including data fusion, where we can infer even more about the sensors reliability by cross checking it values with those of other sensors.

Another Advantage to the integrating system being developed in G2 is that it grants a common ground for the development of the virtual sensor. The development of the Virtual Intelligent Sensors is best started in c\c++. As mentioned earlier, in order for G2 to interface with external systems or programs, a "bridge" must be erected. This "gateway bridge", as Gensym has coined it, is an external c\c++ program written using GSI API functions to allow for G2 to connect to it through TCP. Now the task is to take this bridge and embed a software version of the sensor into it. Since the sensor itself uses routines to process the data, the routines themselves are being coded into a separate program and then this program is run by our bridge to simulate the sensor. The common ground between the PIS and the G2 KB is that many of the routines used for event detection have already been developed in C\C++.

Most of the routines needed for the sensors have already been written by NASA's Glen Research Center, including noise detection, spike detection, drift detection, etc. These routines as well as others developed for the PIS are taken and modularized. Each routine being a separate program, they are then compiled. The bridge, being the connection between G2 and the external systems, facilitates all of the information transfer as well as external processing. The goal is to run the various routines as an intelligent sensor and feed that information to the G2 system as if it were an actual sensor.

Upon execution, the bridge waits for a G2 knowledgebase to connect to it. Upon receiving a connection the bridge simply loops its polling function after initializing the needed variables. The following image shows one of the G2 modules being used for testing the routines, in this case the Noise Detection routines. Upon pressing the button "Start VIS, Noise" button, G2 calls a function remotely in the bridge, this function then proceeds to fork off a process and run the routine on our input data then subsequently pipe the data back into the bridge. That data is fed as raw data back into G2 into the VIS-READING variable and the processed noise data is fed back into the VIS-NOISE-READING variable in G2.

Fig. 11 shows the values of these variables in one second intervals, the top being a pressure reading and the bottom being the noise reading, 1 for Excessive noise and 0 for Acceptable noise. The variable VSS-BRIDGE is the G2 side of the interface. With the modular fashion of the VIS additional algorithms are simple to test allowing for a useful framework for further development. The final product VIS will be composed of a suite of all of the routines used on the PIS.

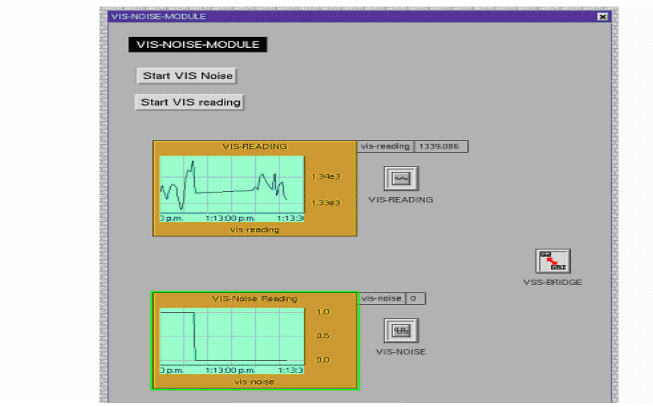

Figure 11: Some VIS routines in G2

The VIS should prove to be very useful in both confirming proper PIS behavior and testing the usefulness of the PIS in many different applications, as well as allowing the use of existing traditional sensors within this new IHMS paradigm. The integrated sensor software will then be able to take in both serial traditional sensor data routed thru the VIS and the newly developed PIS. The output of both the PIS and VIS will be identical, i.e. data as well as an indicator of the quality of the data.

CONCLUSIONS

This paper presents the progress made in the development of intelligent sensors as an integrated systems approach, i.e. one treats the sensors as a complete system with its own sensing hardware (the traditional sensor), A/D converters, processing and storage capabilities, software drivers, self-assessment algorithms, communication protocols and evolutionary methodologies that allow them to get better with time.

Under a project being undertaken at the Stennis Space Center, an integrated framework is being developed for the intelligent monitoring of smart elements. These smart elements can be sensors, actuators or other devices. This paper focuses only on the sensors. The immediate application is the monitoring of the rocket test stands, but the technology should be generally applicable to the Intelligent Systems Health Monitoring (ISHM) vision. This paper outlines specific progress made in the development of intelligent sensors by describing the work done till date on Physical Intelligent Sensors (PIS) and Virtual Intelligent Sensors (VIS).

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of NASA for funding this work under Grant NNS04AB79G.

REFERENCES

1. Ghani, N., "Sensor integration in ESPRIT," *IFAC Proceedings,* Karlsruhe, FDR 1988, pp. 323-328.
2. Pinkava, J., "Towards a theory of sensory robotics," *Robotica*, Vol. 8, 1989, pp. 245-256.
3. Lozano-Perez, T., Mason, M. T., and Taylor, R., "Automatic synthesis of fine motion strategies for robots," *International Journal of Robotics Research*, Vol. 3, No. 1, 1984, pp. 2-24.
4. AbdelRahman, M. and Smith M. L., "The Impact of AI On Sensing Technology," *SENSORS*, September 1991, pp. 16-22.
5. Studt, T., "Smart Sensors Widen Views on Measuring Data," *R&D Magazine*, March 1994, pp. 18-20.
6. Figueroa, F. and Mahajan, A., "Generic Model of an Autonomous Sensor," *Mechatronics*, Vol. 4, No. 3, pp. 295-315, 1994.
7. Henderson, T and Shilcrat, E., "Logical Sensor Systems," *Journal of Robotic Systems*, 1(2), 1984, pp. 169-193.
8. Henderson, T., Hansen, C. and Bhanu, B., "The Specification of Distributed Sensing and Control," *Journal of Robotic Systems*, 2(4), 1985, pp. 387-396.
9. DeCoste, D., "Dynamic Across-Time Measurement and Interpretation," *Artificial Intelligence*, Vol. 51, 1991, pp. 273-341.
10. Mahajan, A. and Figueroa, F., "Dynamic Across Time Autonomous - Sensing, Interpretation, Model learning and Maintenance theory (DATA-SIMLAMT)," *Mechatronics*, Vol. 5, No. 6, 1995, pp. 665-693.