Abstract—The IEEE 1451 family of standards was created to resolve the issues and problems associated with the proliferation and the heterogeneity of sensor networks. This paper will discuss the approach taken in the implementation of an IEEE 1451.2 microcontroller-based STIM in developing a suite of smart sensors for use in an autonomous vehicle. The autonomous vehicle sensor network environment was initially developed outside of the IEEE 1451 standard hence it would provide a very good benchmark system for a comparative analysis of the use of a standardized approach to sensor network development and conventional non-standardized approach. This paper also sheds its light on the importance and usefulness of the IEEE 1451 standards, which even though has been well applauded, has not enjoyed a very wide use.

Index Terms—Actuators, autonomous log-skidder, distributed processing, Network Capable Application Processor NCAP, sensor networks, smart sensors, Smart Transducer Interface Module STIM, Transducer Electronic Data Sheet TEDS, Transducer Independent Interface TII.

I. INTRODUCTION

SENSOR network developers have been faced with deciding which of the different available field bus technologies and device networks to implement. In doing this, they have to bear in mind that transducers are limited by the types of network they can support. There are over 60 sensor networking protocols from which a network developer can choose [1]. This proliferation of sensor networks increases the complexities of network design and any network developer is faced with the challenge of deciding what types of protocols to implement, this is more evident when designing multiple networks. On the other hand, transducer manufacturers are faced with the task of supporting as many network protocols and interface standards on their devices as much as would be cost-effective. Suffice it to say that most transducer manufacturers are SMEs (Small and Medium Enterprises) and are very inclined to cut production cost. The IEEE 1451 family of standards was created to address the problems due to these complexities encountered by the network developer and the transducer manufacturers.

This research work details an implementation of IEEE 1451 compliant transducers. This implementation is designed for use in a control network for an autonomous log skidder [2]. The log skidder is host to a suite of low-cost networked sensors that provide it with the capability for performing distributed measurement and control in real time.

Several implementations of a Smart Transducer Interface Module have been carried out using microprocessors [3], FPGAs and ASICs [4]. All these approaches have their various strong points and trade-offs. This paper describes an approach whereby the functional boundaries of the STIM and the NCAP [5] are implemented within a single microprocessor unit. The rest of this paper is organized as follows. Section 2 gives an overview of IEEE 1451.2. Section 3 describes the suite of sensors deployed on the control network for the autonomous vehicle. Section 4 describes how resources are allocated in a STIM/NCAP microprocessor and section 5 discusses the communication scheme used. Section 6 discusses the electronic data sheet format and Section 7 explains how software resources are put together to accomplish the STIM design. Section 8 looks at performance and productivity issues using an IEEE 1451.2-compliant sensor network versus a non-standardized conventional sensor network design. Sections 9 and 10 respectively provide a conclusion and a list of references.

II. OVERVIEW OF IEEE 1451.2

A. IEEE 1451 Family of standards

Due to the increasing pervasiveness of transducers (actuators and sensors) in manufacturing, industrial control, automotive, aerospace, building, and biomedicine to mention a few; the need arose to define a standard for addressing problems that surfaced from complexities associated with increasing numbers of sensor networking protocols. The objective of IEEE 1451 was to develop a smart transducer interface standard and there were initially a set of four sub-standards – 1451.1, 1451.2, 1451.3 and 1451.4; of which only 1451.1 and 1451.2 have been adopted. P1451.3 and P1451.4 are yet to be adopted and are still being reviewed, hence the prefix ‘P’ which indicates a proposed document. IEEE 1451.1 defines a Network Capable Application Processor NCAP information model which provides an object-oriented abstraction between a transducer and a network. This allows a transducer to be independent of the

Richard W. Wall is an Associate Professor in the Department of Electrical and Computer Engineering at the University of Idaho, Moscow, ID 83844-1023. (e-mail: rwall@uidaho.edu).

A. Ekpruke is a Computer Engineering graduate student at the University of Idaho. He has an undergraduate degree in Electrical Engineering from the University of Lagos, Nigeria. (e-mail: ekp2296@uidaho.edu).
IEEE 1451.2 defines a Smart Transducer Interface Module (STIM) which specifies an abstraction between transducer type and a microprocessor by specifying a digital communication interface and the capability for sensor self-identification.

IEEE P1451.3 proposes a Transducer Bus Interface Module (TBIM), which is a standard digital interface that can connect multiple physically separated transducers in a multi-drop configuration.

IEEE P1451.4 proposes a standard interface that will allow analog transducers to operate in a mixed-signal mode. By this, the transducer starts up in a digital signal communication mode for self-identification and control purposes; then it switches to analog signal mode for operational purposes. Finally, a committee has also been set up to propose a sensors and actuators standard for wireless communication protocol, this is known as IEEE P1451.5.

B. Components of IEEE 1451.2

The IEEE 1451.2 standard does not define a new field bus or networking protocol, however, it defines a common interface to any field bus or network type. The scope [5] of IEEE 1451.2 is:

(i.) to define a digital interface for connecting transducers to microprocessors.

(ii.) it describes a transducer electronic data sheet (TEDS) and its data format

(iii.) it defines an electrical interface, read and write logic functions to access the TEDS and a wide variety of transducers.

Understanding this standard involves getting acquainted with a new set of terminologies which are briefly summarized in the following paragraphs of this section.

The Smart Transducer Interface Module (STIM) provides an object-oriented approach to transducer design. In this context, a transducer refers to a sensor and an actuator. A STIM specifies a standard digital communication interface between a transducer and a microprocessor, and it also defines a TEDS that enables plug and play of the STIM into a network. A STIM can support a maximum of 255 channels, either separately or in groups.

A Transducer Electronic Data Sheet (TEDS) provides a permanent repository of information related to the transducer. This makes self-identification and STIM “plug and play” possible. The TEDS is stored in a non-volatile memory and the TEDS data is obtained on demand by the Network Capable Application Processor (NCAP).

The Network Capable Application Processor (NCAP) is on a higher level of abstraction than the STIM and it integrates the STIM into a digital network; thus making it possible for any STIM to operate in any digital network. The NCAP drives the STIM and provides local node intelligence in a network.

All communications between the STIM and the NCAP occurs through a ten wire digital interface that is known as the Transducer Independent Interface (TII). The TII basically incorporates a synchronous serial interface along with other lines for handshaking and support. Fig. 1 shows the relationship between the various components of the IEEE 1451.2 architecture.

III. DESCRIPTION OF LOG SKIDDER SENSORS

In order to fulfill its mission, the autonomous log skidder was initially fitted with an array of conventional sensors for real-time gathering and processing of data [2]. Fig. 2 shows a picture of the log-skidder and Fig. 3 shows a simple direct network of these sensors to a CPU sub-system.

These sensors consist of an array of six Polaroid Ultrasonic Ranging units [6], two SEI optical shaft encoders [7], an Applied Geomechanics tilt-meter [8], a camera visioning system and a BEI Gyrochip gyroscope [9]. These sensors are connected to the CPU subsystem either directly using the CPU I/O buses or via RS232 serial communication ports. This CPU subsystem is a Versalogic VSB-6 single board computer with a pentium/K6 processor. This subsystem processes sensor data and provides a centralized and real-time control of the vehicle.

This approach to real-time networking suffers from several inherent shortcomings and problems. These include a lack of intelligence at each sensor node which leads to the need for a centralized control and data processing CPU. Consequently, this system is prone to a single point of failure at the CPU sub-system and there is also an attraction to design one large monolithic program for this control system.

Fig. 2 Log skidder – Tracked ASV-30 Vehicle with A-Frame
This research work details a different approach to designing this network. It has been shown in earlier works [10] that a distributed network design allows each node to provide some autonomous intelligent control and this results in a more efficient and fault-tolerant network. Kang Lee and Richard Schneeman [10] also showed in their paper that a distributed design is more cost-effective due to the much reduced cost of microprocessors, availability of higher speed networks and higher-level languages and development environments.

The major contribution of this work to the existing log skidder control network is the provision of local intelligence at each sensor node. This consequently results in a distribution amongst the nodes of the entire processing in this control network. As a demonstration of the strengths of this approach, the Polaroid Ultrasonic Ranging unit was developed into a STIM.

Fig. 4 illustrates a distributed design approach for the autonomous vehicle sensor suite. There is local intelligence at each node, and the ability of a node to identify itself within the network provides for better feedback mechanisms for system configuration and device failure detection. The design provides the flexibility of including additional sensors into the network with little or no additional configuration. Another advantage of this design is that the processing requirements are distributed over the entire system and network traffic is reduced [10]. It can be seen that a distributed sensor network built with smart sensors as defined by the IEEE 1451.2 would on the long run lead to a more efficient and fault tolerant network.

IV. MICROPROCESSOR RESOURCE ALLOCATION

In developing a STIM, we took a slightly different approach from most other microprocessor STIM implementations. The advantages of using a microprocessor is that it can easily be reprogrammed to accommodate new STIM design enhancements and most microprocessors come with on-chip resources such as EEPROMs, flash memory, A/D and D/A [4]. In this work, the functional boundaries of the STIM and NCAP (refer to Fig. 1) are defined within the same microprocessor unit. The implication is that the microcontroller would alternatively act as a STIM and an NCAP.

The decision of which microcontroller that could best be used for this implementation had to be made. There are a variety of microcontrollers available: PIC, Basic Stamp, Rabbit and Analog Device ADuC812. The ADuC812 was specifically designed with IEEE 1451.2 functionalities in mind. Microcontroller selection is usually dependent on three main factors and how the interaction of these factors would affect your overall system design.

These factors are the amount of memory available for program and data, the CPU time the microcontroller is capable of providing and the sufficiency of I/O resources on the microcontroller. Other features and resources such as serial ports, timers, counters, pulse width modulation units, A/D units [11] and even TCP/IP ports simplify and enhance system design and reliability.

Based on the criteria for this project, the Rabbit 3000 microprocessor [12] was selected for the STIM module. Some of the reasons for this include the relatively large size of its flash memory (256 - 512K) which would make developing and storing very large TEDS easily feasible. There is also 128K of SRAM, up to 56 programmable I/O ports, an SPI port and a TCP/IP port. The development environment also provides a lot of features that ease code development, this include libraries for SPI, TCP/IP, RS-232 communication. The Rabbit 3000 system can also support very large code; up to 50000 lines of C-language program can be supported by this unit.

V. TII COMMUNICATION MODULE

The IEEE 1451.2 standard defines a 10-wire digital interface used for communication between the NCAP and STIM. These physical lines can be grouped into four main categories [5] as shown by table 1.

<table>
<thead>
<tr>
<th>Group</th>
<th>Lines</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data</td>
<td>DATA OUT</td>
<td>DOUT</td>
</tr>
<tr>
<td></td>
<td>DATA IN</td>
<td>DIN</td>
</tr>
<tr>
<td></td>
<td>DATA CLOCK</td>
<td>DCLK</td>
</tr>
<tr>
<td></td>
<td>N IO ENABLE</td>
<td>NIOE</td>
</tr>
<tr>
<td>Triggering</td>
<td>N TRIGGER</td>
<td>NTRIG</td>
</tr>
<tr>
<td>Support</td>
<td>POWER</td>
<td>POWER</td>
</tr>
<tr>
<td></td>
<td>COMMON</td>
<td>COMMON</td>
</tr>
<tr>
<td></td>
<td>N ACKNOWLEDGE</td>
<td>NACK</td>
</tr>
<tr>
<td></td>
<td>N STIM DETECT</td>
<td>NSDET</td>
</tr>
<tr>
<td>Interrupt</td>
<td>N IO INTERRUPT</td>
<td>NINT</td>
</tr>
</tbody>
</table>
In this work, the TII module is implemented entirely in software and its communication lines are software registers. This is because the STIM and the NCAP functional boundaries are defined within the same microcontroller unit. This potentially eliminates the need for physical TII lines. In most other microcontroller implementations of STIMs, the SPI interface is used in addressing the requirements of the TII [13]. However, in this case an SPI interface would not be required in designing the TII since all communications between the STIM and NCAP are to be carried out internally using registers. Except for the POWER and COMMON lines; the DOUT, DIN, NIOE, NTRIG, NACK, NSDET and NINT lines are defined as global registers. This approach also obviates the need for a DCLK register; this is because the timing requirements are bounded by the clock speed of the Rabbit 3000 microcontroller. At a maximum operating speed of 29.1MHz by this processor, it can be shown that the timing requirements of the standard are adhered to.

Another preclusion resulting from the above argument is that it becomes unnecessary for data transfer on the DOUT and DIN registers to be done bit-wise. Data is transferred over these data buses in byte-sizes and most significant byte is transferred first as specified in the IEEE 1451.2 standard. All the necessary communication between the STIM and the NCAP functional boundaries are implemented in software by the TII module. It is also important to note that the TII provides a master-slave interface between the NCAP and the STIM [13].

VI. DESIGNING THE TEDS

In order to develop a STIM from a Polaroid Ultrasonic Ranging unit and a Rabbit3000 processor, at least a Meta TEDS and one channel TEDS is required. The final log skidder implementation requires six channels of Polaroid Ultrasonic Ranging units, however the design procedure described by this paper allows for easy expansion simply by including five more channel TEDS. It is important to note that there are six optional TEDS, these are:

- Calibration TEDS,
- Meta-identification TEDS,
- Channel- Identification TEDS,
- Calibration-Identification TEDS,
- End-User's Application Specific TEDS, and
- Industry Extensions TEDS

Due to the large flash memory available on the Rabbit3000 device, incorporating all these optional TEDS would not require any additional hardware, so does increasing the number of channel TEDS in the current STIM design.

The TEDS is defined as a structure data type and it is copied into the flash memory on system initialization, both TEDS implemented in this work are read-only and cannot be updated by the NCAP in runtime.

VII. PUTTING THE STIM AND NCAP TOGETHER

There are five main categories of software functional modules used in this project. It is important to note here that the major part of a STIM design consists of software development. Also, depending on the state of the microcontroller, it may act as a STIM or an NCAP. The five program modules are:

- **TEDS module** – this uses a structure data type for storing and initializing the TEDS (meta-TEDS and Channel TEDS).
- **STIM module** – this provides the functions required to access the STIM. The STIM module functions respond to NCAP command functions. This module also contains functions that interface to the sensor and read sensor inputs. When functions within this module are accessed, the microcontroller can be considered to be acting as a STIM.
- **NCAP module** – these modules cause the microcontroller to function as an NCAP. These functions drive the STIM functional module. Here, the necessary commands are issued to the STIM module.
- **TII module** – this drives the communication between the STIM and NCAP modules. The TII module provides two categories of functions; these are the handshaking functions and the data transfer functions.
- **Kernel** – At the heart of any real-time embedded system is a real time operating system RTOS. The kernel of any RTOS contains its core functions. In this implementation, the kernel module contains the code statements and functions that coordinates all the other modules and consequently causes the microcontroller to toggle between a STIM and an NCAP. Consequently, the microcontroller can be depicted as a state machine [14] driven by events external to it (Fig. 5). In general, there are two paradigms for designing RTOSes; these are event triggering and time triggering. In time triggered RTOSes, system activities are initiated at predefined instances with respect to a global or common time reference. In event triggered RTOSes (such as the one described by this paper) system activity is usually in response to the occurrence of a particular event, triggered by the system environment; in this case retrieval of sensor data in response to the NCAP trigger (Fig. 5). As shown in Fig. 5, the text in bold represents the processes which occur on system start-up and initialization (i.e. sensor self-identification) and the text in italics represents the state changes that occur during run-time of the microcontroller RTOS. The microcontroller can be in one of two states (NCAP or STIM) as dictated by the current event.

![Fig.5. State machine representation of STIM/NCAP microcontroller implementation](image-url)
VIII. RESULTS

A. Non-standardized network versus IEEE 1451.2 compliant network

Consider the two different networks shown in Fig. 3 and 4 (centralized network - Case I and distributed network – Case 2). Let us assume that each of these networks can be realized with n number of sensor nodes. In the first case, (Fig. 3), each sensor is connected to the CPU subsystem using a different network protocol (say, I²C, SPI, RS232, proprietary protocols etc.) depending on the type of interface provided by each sensor node. Hence, the extreme scenario would require that there will be n different protocols (i.e. one for each sensor or transducer). In the second case, (Fig. 4), a distributed approach is taken and there is intelligence at each node since each node represents a STIM. Only a single networking protocol is required for the bus and the sensors are intelligent since they are based on IEEE1451.2. This implies that for n sensors, a network developer only needs to deploy IEEE 1451.2 compliant sensors and a single network protocol for whatever network bus of choice.

In Case I, the network developer needs to learn how to deploy at most n numbers of protocols in order to integrate each sensor onto the network. Also, he is limited by the number of protocols that the CPU subsystem hardware can support as well as the choice of transducers that can feature those protocols. Also, the inclusion of a new node to this network requires additional development time and effort.

Let us assume that the time it takes for the network developer to learn a new protocol is time t and that the effort it takes to integrate a node into the network requires f time units. Thus in the extreme case of n protocols and n sensors; the time taken to develop a network would be approximately:

\[ n(t + f) \text{ time units} \]  

(1)

Also, each additional sensor integrated into the network subsequently would require at least f time units.

In Case II, the network developer needs to learn how to develop IEEE 1451.2 compliant STIMs and also needs to decide on only one networking bus protocol to implement. This is a one time process and for every node that is integrated into the sensor network he does not need to learn how to implement a different network protocol since each node has plug and play capabilities. This translates into a significant savings in time for the network developer working with STIM modules.

Using the assumptions derived from Case I, if it takes t time units to learn to develop IEEE 1451.2 and f time units to implement the network bus. By implication, the time that would be required to deploy any number of sensors would be:

\[ (t + f) \text{ time units} \]  

(2)

Also, integrating any additional sensor would be rather effortless due to the plug and play features of each node.

Considering the autonomous log skidder, the protocols involved in deploying all the sensors which are described in Section III are I2C, RS232 and a number of proprietary protocols. The disadvantages of deploying a network using this approach in terms of time-costs have been explained in the previous paragraphs. By decentralizing the network and using STIMs the effort was considerably reduced. The bus selected for the network in this design was a Controller Area Network CAN bus.

B. Microprocessor Resource Utilization – Memory and I/O

The available memory on the Rabbit 3000 module for program and data code is 256K of flash memory and 128K of battery backed SRAM. The entire software written for the 1-channel ultrasonic transducer STIM was programmed into the flash memory. Fig. 6 shows the memory utilization of the flash memory and Table 2 shows the memory utilization for a 1-channel ultrasonic transducer STIM and a 6-channel ultrasonic transducer STIM. The data for six channels is provided in this investigation since that is the maximum number of ultrasonic transducers that is required on the autonomous vehicle. Table 2 also shows that all 128K of battery-backed SRAM can be available for other use, the memory utilizations for the Channel TEDS and Meta TEDS data is also depicted.

Consequently, from Table 2, it is clear that memory is not a bottleneck for STIM implementation using the Rabbit 3000 microcontroller; the likely bottleneck is the available I/O resource as shown by Table 3. Thus, a microcontroller STIM is likely to be limited by I/O lines and A/D channels [4]. As shown by Table 3, the Rabbit 3000 has 56 programmable I/O lines; the ultrasonic transducer STIM designed in this work requires two I/O lines for each channel (i.e. one I/O line for input and the other for output). This implies that without the use of additional hardware logic, a total of 28 channels can be implemented using the available I/O on this microcontroller. This is a far cry from the maximum 255 channels that the IEEE 1451.2 standard stipulates for a STIM. One way of increasing the number of I/O lines would be to use memory mapped I/O [15]. Wall and King have showed in [15] that memory and CPU speed could be used to address I/O imitations.

![Fig. 6. Flash Memory Utilization on Rabbit 3000 Microcontroller](image-url)
TABLE 2: MEMORY UTILIZATION FOR 2 DIFFERENT STIM CONFIGURATIONS

<table>
<thead>
<tr>
<th>STIM Configuration</th>
<th>Memory Used (bytes)</th>
<th>Available Flash (bytes)</th>
<th>Available SRAM (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Channel STIM</td>
<td>36542</td>
<td>225602</td>
<td>131072</td>
</tr>
<tr>
<td>• 1 Channel TEDS</td>
<td>38</td>
<td>324</td>
<td></td>
</tr>
<tr>
<td>• Meta TEDS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-Channel STIM</td>
<td>36732</td>
<td>225412</td>
<td>131072</td>
</tr>
<tr>
<td>• 6 Channel TEDS</td>
<td>228</td>
<td>324</td>
<td></td>
</tr>
<tr>
<td>• Meta TEDS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IX. CONCLUSION

This research work has demonstrated an implementation of smart sensors using the IEEE 1451.2 standard in a vehicle control network. It was demonstrated in this paper that a single microprocessor could be used to develop a STIM and its corresponding NCAP by time-sharing the microprocessor's resources, and toggling it between STIM and NCAP functionalities. This would translate to significant cost savings for very large sensor networks and result in I/O savings on the microprocessor unit. The microprocessor selected in this paper was the Rabbit 3000 microcontroller. Its on-board I/O resources were determined to be the limitation to using a microcontroller in STIM developments. It was determined that memory was not a limitation and by using memory mapping, the I/O resources could be increased thus increasing the number of channels for a single STIM.

TABLE 3: I/O UTILIZATION

<table>
<thead>
<tr>
<th>STIM Configuration</th>
<th>I/O Lines Used</th>
<th>Available I/O Lines</th>
<th>Available Flash Memory (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Channel STIM</td>
<td>2</td>
<td>54</td>
<td>225602</td>
</tr>
<tr>
<td>6-Channel STIM</td>
<td>12</td>
<td>44</td>
<td>225412</td>
</tr>
<tr>
<td>28-Channel STIM</td>
<td>56</td>
<td>0</td>
<td>224576</td>
</tr>
</tbody>
</table>

Also, this paper has shown that a sensor network built with IEEE 1451.2 compliant sensors allows for easier expansion and integration of new nodes. One of the objectives of this paper is to encourage the increased use of IEEE 1451.2 standard by demonstrating the process involved in developing a STIM and highlighting a control network where these STIMs are utilized.

Finally, by using standardized components and interfaces, the task of network development and sensor integration is expected to become less cumbersome.

REFERENCES