XIX IMEKO World Congress
Fundamental and Applied Metrology
September 6−11, 2009, Lisbon, Portugal

SMART TRANSDUCER BLOCK ENABLES PLUG & PLAY TRANSDUCERS

Vítor Viegas [1,2] , *J. M. Dias Pereira* [1,2] , *P. Silva Girão* [2]

[1] ESTSetúbal-LabIM, Instituto Politécnico de Setúbal, Setúbal, Portugal
[2] Instituto de Telecomunicações, Lisboa, Portugal
vviegas@est.ips.pt , joseper@est.ips.pt , p.girao@lx.it.pt

Abstract − The paper presents a dedicated IEEE 1451.1 Transducer Block that enables plug & play transducers according to the recommendations of the IEEE 1451.4 standard. The Transducer Block is self-configurable based on the information stored in the Transducer Electronic Data Sheet (TEDS): basic TEDS and transducer type templates are used for transducer self-identification; and calibration templates are used for transducer self-calibration. The Transducer Block is also network-enabled by exposing its functionalities through a set of Web Services.

Keywords: Smart sensor, IEEE 1451, TEDS, Web Service

1. INTRODUCTION

The IEEE 1451 family of standards [1] simplifies transducer connectivity to existing networks by defining a set of standardized hardware and software interfaces where heterogeneous components can connect and work together.

The IEEE 1451.1 standard [2-5] defines an information model for connecting processors to networks and transducers. The processor, known as Network Capable Application Processor (NCAP), acts as a bridge between transducers and the network: on the field side, an abstraction layer provides high-level functions to interact with transducers; on the network side, an abstraction layer provides high-level services to handle network requests; in the middle, the NCAP application receives data from both sides (the field and the network), processes it and decides the next state of the system.

The IEEE 1451.4 standard [6] defines a hardware interface that allows analog legacy transducers to exchange digital data with the NCAP. It also defines the format of the Transducer Electronic Data Sheet (TEDS), a data structure embedded on the transducer that describes its identity, type, operation and attributes. This information is all the NCAP needs to create a self-configurable software interface – known as Transducer Block in the 1451.1 terminology – to *play* with the transducer.

2. TRANSDUCER BLOCK

We present a self-configurable Transducer Block that works with 1451.4 transducers connected to Data AcQuisition (DAQ) boards compliant with the DAQmx driver from National Instruments (NI). For this reason, the Transducer Block was named "DAQmxTBlock".

The DAQmxTBlock works with 1451.4 systems of type tier-2, meaning that it establishes a point to point connection to a single-node transducer and reads its TEDS through the Mixed Mode Interface (MMI). It does not support multimode features or extended TEDS capabilities.

The life-cycle of the DAQmxTBlock can be described as follows:

1) At design-time, using the Measurement and Automation Explorer (MAX), a software tool provided by NI, the developer creates *tasks* involving one or more *channels*. A *task* is a set of properties that completely describe the process of acquiring/updating data (such as number of samples, sampling frequency and trigger settings); a *channel* is a hardware input/output configured to acquire/generate a signal. Each channel can be automatically configured (without any human intervention) by reading the embedded TEDS of the attached transducer. If the transducer has no embedded TEDS, the developer can associate it a virtual TEDS in the form of a .ted file (this feature is useful for older transducers or DAQ boards that do not support the MMI). Detailed information about the configuration process can be found in [7-8].

2) At run-time, the DAQmxTBlock object is created, initialized and executed. During initialization, the DAQmxTBlock loads all pre-defined tasks and creates auxiliary objects to support them, more precisely one 1451.1-Parameter for each task and one DAQmxTChannel for each channel. The 1451.1-Parameter provides methods to read/write data from/to task transducers, as well as a metadata structure that describes the meaning of the data exchanged. The DAQmxTChannel provides methods to get information about the channel where the transducer is attached to, including methods to retrieve the underlying TEDS. All these objects register themselves on the network as Web Services [9] making possible to access their methods remotely.

3) During execution, the NCAP application interacts with the DAQmxTBlock and related objects. The DAQmxTBlock is destroyed when the NCAP application ends.

The DAQmxTBlock and related classes were developed using Visual Basic .NET and Visual Studio 2008. They were compiled as a Dynamic Link Library (DLL) in order to make them reusable for any Windows application. Their relations, represented by numbered circles in figure 1, can be described as follows:

1) The DAQmxTBlock owns n 1451.1-Parameters, each one corresponding to a pre-defined task. The 1451.1-Parameter acts as a networking-visible variable holding the data acquired/updated by the task. The properties of the task are used to fill the metadata structure of the 1451.1-Parameter.

2) The DAQmxTBlock owns m DAQmxTChannel objects, each one corresponding to a hardware input/output configured to acquire/generate a signal.

3) A task can involve one or more channels making m≠n.

4) Each DAQmxTChannel connects to a transducer having access to its TEDS.

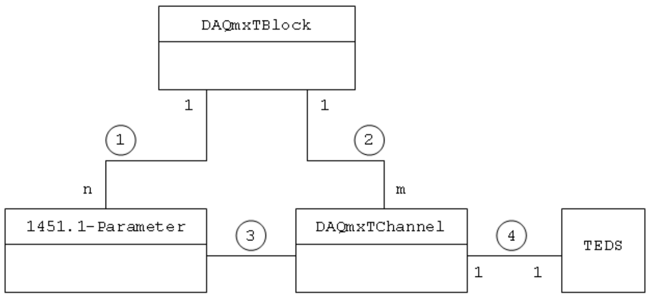


Fig 1. DAQmxTBlock context.

Table 1 briefly describes the methods implemented by the classes DAQmxTBlock and DAQmxTChannel.

3. APPLICATION EXAMPLE

To exemplify the use of the DAQmxTBlock, we built a small Web-enabled weather station that measures temperature, relative humidity, dew point and heat index. Two physical sensors are used: a thermoresistance to measure the temperature and a variable capacitor to measure the relative humidity. The dew point (DP) and heat index (HI) are computed from the sensed data using the following equations [10-11], where T and F represent the temperature in Celsius and Fahrenheit degrees respectively, and H represents the relative humidity (between 0 and 100):

$$DP = \frac{237.7 \times \left[\frac{17.271 \times T}{237.7 + T} + \ln\left(\frac{H}{100}\right)\right]}{17.271 - \frac{17.271 \times T}{237.7 + T} - \ln\left(\frac{H}{100}\right)} \qquad (1)$$

Table 1. DAQmxTBlock and DAQmxTChannel overview.

Class Name: `DAQmxTBlock`   Class ID: 1.1.1.4 (inherits from Block)
`GetDriverVersion()`: Returns the DAQmx driver major and minor versions.
`GetLastDAQWarning()`: Returns a string describing the last exception occurred in the DAQ infrastructure.
Class Name: `DAQmxTChannel`   Class ID: 1.1.2.5 (inherits from Component)
`GetChannelType()`: Returns the channel type (analog input/output, digital input/output or counter input/output).
`GetPhysicalName()`: Returns a string identifying the physical terminal where the transducer connects to (ex. “dev1/ai2”).
`GetRelatedParameter()`: Returns the name of the 1451.1-Parameter that consumes/provides data from/for this channel.
`GetTEDSBitStream()`: Returns the raw TEDS as a byte array.
`GetTEDSManufacturerID()`: Returns the manufacturer of the transducer from its TEDS.
`GetTEDSModelNumber()`: Returns the model number of the transducer from its TEDS.
`GetTEDSSerialNumber()`: Returns the serial number of the transducer from its TEDS.
`GetTEDSTemplateIDs()`: Returns the identifiers of the templates that compose the TEDS.
`GetTEDSVersionLetter()`: Returns the version letter of the transducer from its TEDS.
`GetTEDSVersionNumber()`: Returns the version number of the transducer from its TEDS.

$$\begin{aligned} HI = & -42.379 + (2.049 \times T) + (10.143 \times H) \\ & - (0.2248 \times T \times H) - (6.838 \times 10^{-3} \times T^2) \\ & - (5.482 \times 10^{-2} \times H^2) + (1.229 \times 10^{-3} \times T^2 \times H) \\ & + (8.528 \times 10^{-4} \times T \times H^2) - (1.99 \times 10^{-6} \times T^2 \times H^2) \end{aligned} \qquad (2)$$

In terms of hardware, the weather station includes the following equipments (figure 2):

1) A three-wire PT100 thermoresistance, part number 745691-01 from NI, compliant with the DIN 43760 standard.

2) A relative humidity sensor, model HIH4000-001 from Honeywell.

3) DAQ system from NI composed by a chassis (NI cDAQ-9172), a general-purpose analog input module (NI-9205), and a universal analog input module with signal conditioning for resistive sensors (NI-9219).

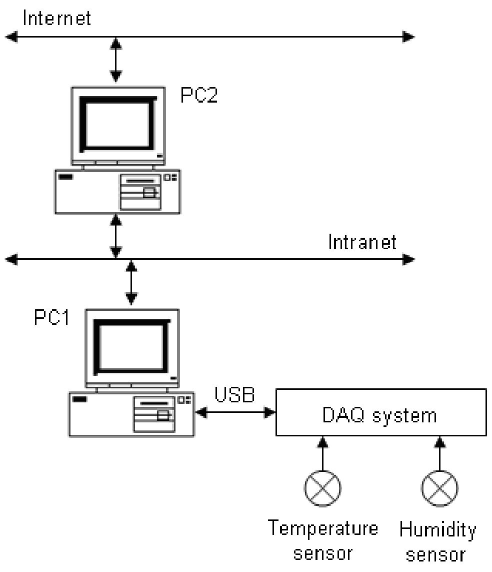


Fig 2. Weather station apparatus.

4) Personal Computer number one (PC1) with Windows XP operating system. This computer executes the NCAP application that acquires data from sensors, calculates meteorological variables and handles Intranet requests. The PC1 connects to the DAQ system by means of a high-speed Universal Serial Bus (USB).

5) Personal computer number 2 (PC2) with Windows XP operating system. This computer executes the Human Machine Interface (HMI) application that requests meteorological variables from the NCAP and presents them to the user. The HMI application has a Web interface that is accessible by a common Internet browser.

Both sensing elements are legacy sensors that do not support the embedded TEDS feature or the MMI interface. For that reason, we had to create two virtual TEDS files and manually associate them to the sensors.

The NCAP application was developed using the library IEEE1451Dot1.dll, which was developed by the authors and presented in [12]. The NCAP application includes three processing Blocks:

1) The top-level NCAPBlock, which represents the NCAP process as a whole and keeps track of all underlying network-visible entities.

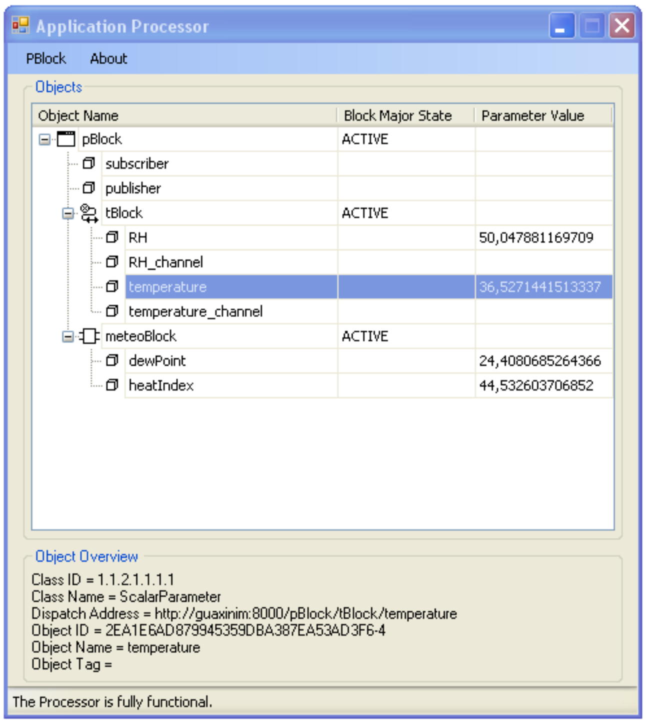
2) The DAQmxTBlock, which is in charge of acquiring data from both sensors and presenting it as 1451.1-Parameters. It also provides access to channel properties by means of DAQmxTChannel objects.

3) The MeteoBlock, which is a dedicated 1451.1-FunctionBlock that computes the dew point and heat index from the temperature and relative humidity. The two computed variables are presented as 1451.1-Parameters as well.

The NCAP application starts by creating all processing Blocks and their owned objects. Every remotable object registers itself on the network as a Web Service. After initialisation, the NCAP application enters on a timed loop where it updates all meteorological variables once per second.

As shown in figure 3, the front panel of the NCAP application is composed by a tree that lists hierarchically all network-visible objects. Two columns are provided to show the state of all Blocks and the value of all 1451.1-Parameters. An overview of the selected object is given in the text box at the bottom.

The HMI application was developed using LabVIEW 8.6 as shown in figure 4. Four numeric indicators are used to present the meteorological variables, which can be refreshed on demand. The buttons named “Get TEDS” are used to retrieve the TEDS of the corresponding sensor, which is presented in the form of a human-readable table.

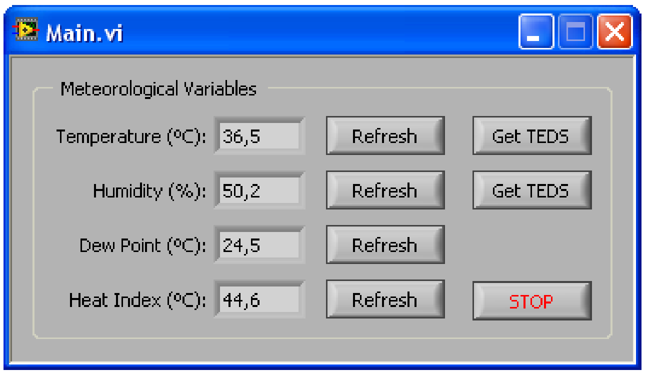


Object Name	Block Major State	Parameter Value
pBlock	ACTIVE	
subscriber		
publisher		
tBlock	ACTIVE	
RH		50,047881169709
RH_channel		
temperature		36,5271441513337
temperature_channel		
meteoBlock	ACTIVE	
dewPoint		24,4080685264366
heatIndex		44,532603706852

```
Class ID = 1.1.2.1.1.1.1
Class Name = ScalarParameter
Dispatch Address = http://guaxinim:8000/pBlock/tBlock/temperature
Object ID = 2EA1E6AD879945359DBA387EA53AD3F6-4
Object Name = temperature
Object Tag =
```

The Processor is fully functional.

Fig 3. NCAP application running on the PC1.



Temperature (°C):	36,5
Humidity (%):	50,2
Dew Point (°C):	24,5
Heat Index (°C):	44,6

Fig 4. HMI application running on the PC2.

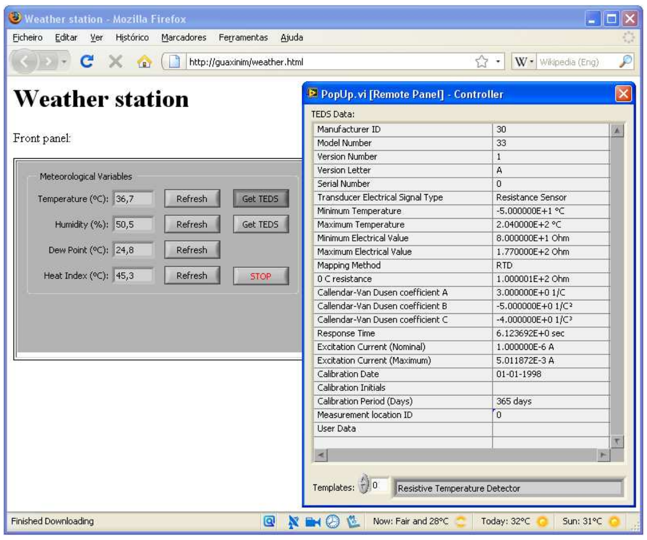
Both applications interact using exclusively the client/server communication model (the HMI application is the client and the NCAP application is the server). Whenever the user clicks one of the buttons in figure 4, the HMI application creates a proxy at run-time and passes it the dispatch address of the remote service. If both parts bind successfully, the proxy executes the remote call, collects the results and presents them to the user.

In addition, the HMI application is Web-enabled by using the Web Publishing Tool provided by LabVIEW. This tool publishes the HMI application on the World Wide Web (WWW) making it accessible on any computer with an Internet browser.

4. RESULTS

The proposed solution was tested in real conditions. The NCAP application worked as expected: sensor readings performed well, the dew point and heat index were computed correctly and all Web Services were visible on the Intranet. Unfortunately, the front panel seemed to be frozen, only responding to user actions once per second. This problem has to be solved in the future.

The HMI application also worked as expected: all remote calls performed well and all returned values were consistent (see figure 4 again). Both TEDS structures were retrieved successfully and the Web interface worked without problems (see figure 5).



Weather station - Mozilla Firefox
Ficheiro Editar Ver Histórico Marcadores Ferramentas Ajuda
http://guaxinim/weather.html
Wikipedia (Eng)
Weather station
Front panel:
Meteorological Variables
Refresh
Get TEDS
STOP
PopUp.vi [Remote Panel] - Controller
Finished Downloading
Now: Fair and 28°C
Today: 32°C
Sun: 31°C

Temperature (ºC): 36,7
Humidity (%): 50,5
Dew Point (ºC): 24,8
Heat Index (ºC): 45,3

TEDS Data:

Manufacturer ID	30
Model Number	33
Version Number	1
Version Letter	A
Serial Number	0
Transducer Electrical Signal Type	Resistance Sensor
Minimum Temperature	-5.000000E+1 °C
Maximum Temperature	2.040000E+2 °C
Minimum Electrical Value	8.000000E+1 Ohm
Maximum Electrical Value	1.770000E+2 Ohm
Mapping Method	RTD
0 C resistance	1.000001E+2 Ohm
Callendar-Van Dusen coefficient A	3.000000E+0 1/C
Callendar-Van Dusen coefficient B	-5.000000E+0 $1/C^2$
Callendar-Van Dusen coefficient C	-4.000000E+0 $1/C^3$
Response Time	6.123692E+0 sec
Excitation Current (Nominal)	1.000000E-6 A
Excitation Current (Maximum)	5.011872E-3 A
Calibration Date	01-01-1998
Calibration Initials	
Calibration Period (Days)	365 days
Measurement location ID	0
User Data	

Templates: 0 Resistive Temperature Detector

Fig 5. Accessing the HMI application on the Internet.

5. CONCLUSIONS

This paper described a Transducer Block that works with pre-defined DAQmx tasks and 1451.4-enabled transducers. The Transducer Block was successfully tested on a NCAP application intended to do meteorological measurements. For each task, the Transducer Block automatically created a Parameter object to hold transducer data and a DAQmxTChannel object to get channel information (including TEDS structures).

The objects of the NCAP application were exposed as Web Services and were consumed by a client application built in LabVIEW. The client worked as expected, not only locally but also across the Internet by means of its Web interface.

REFERENCES

[1] Eugene Y. Song, Kang Lee, "Understanding IEEE 1451 – Networked Smart Transducer Interface Standard", IEEE Instrumentation and Measurement Magazine, Vol. 11, No. 2, pp. 11-17, April 2008.

[2] IEEE Std 1451.1, "IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Network Capable Application Processor (NCAP) Information Model", USA, 2000.

[3] Vítor Viegas, J. M. Dias Pereira, P. Silva Girão, "A Brief Tutorial on the IEEE 1451.1 Standard", IEEE Instrumentation & Measurement Magazine, Vol. 11, No. 2, pp. 38-46, April 2008.

[4] Kang Lee, Eugene Song, "A Wireless Environmental Monitoring System Based on the IEEE 1451 Standards", Instrumentation and Measurement Technology Conference (IMTC), Sorrento, Italy, April 2006.

[5] A. Stepanenko, K. Lee, R. Kochan, V. Kochan, A. Sachenko, "Development of a Minimal IEEE 1451.1 Model for Microcontroller Implementation", IEEE Sensors Applications Symposium (SAS), Houston, Texas, USA, February 2006.

[6] IEEE Std. 1451.4, "IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Mixed-Mode Communication Protocols and Transducer Electronic Datasheet (TEDS) Formats", USA, 2004.

[7] "Upgrading Your System for Smart TEDS", NI Tutorial, http://zone.ni.com/devzone/cda/tut/p/id/2925.

[8] "Upgrading Your System for Virtual TEDS", NI Tutorial, http://zone.ni.com/devzone/cda/tut/p/id/4470.

[9] Gustavo Alonso, Fabio Casati, Harumi Kuno, Vijay Machiraju, "Web Services – Concepts, Architectures and Applications", Springer, Germany, 2004, ISBN 3-540-44008-9.

[10] http://en.wikipedia.org/wiki/Dew_point

[11] http://en.wikipedia.org/wiki/Heat_Index

[12] Vítor Viegas, J. M. Dias Pereira, P. Silva Girão, "Next Generation Application Processor Based on the IEEE 1451.1 Standard and Web Services", International Instrumentation and Measurement Technology Conference (I2MTC), Victoria, British Columbia, Canada, May 2008.