International Journal of Sensors and Sensor Networks
2015; 3(1): 1-11
Published online March 21, 2015 (http://www.sciencepublishinggroup.com/j/ijssn)
doi: 10.11648/j.ijssn.20150301.11

An IEEE 1451.0-based Platform-Independent TEDS Creator Using Open Source Software Components

Paul Celicourt, Michael Piasecki

Civil Engineering Department, The City College of New York, New York, USA

Email address:
pcelico00@citymail.cuny.edu (P. Celicourt), mpiasecki@ccny.cuny.edu (M. Piasecki)

To cite this article:
Paul Celicourt, Michael Piasecki. An IEEE 1451.0-based Platform-Independent TEDS Creator Using Open Source Software Components. *International Journal of Sensors and Sensor Networks.* Vol. 3, No. 1, 2015, pp. 1-11. doi: 10.11648/j.ijssn.20150301.11

Abstract: This paper introduces a Graphical User Interface supported and platform-independent application to generate a Transducer Electronic Data Sheet (TEDS) based on the IEEE 1451.0 standard using Python programming language. Compared to other TEDS application development efforts, this application provides a help system that improves the usability as it requires little familiarity with the IEEE 1451 standard. It is built on the Hierarchical Model-View-Controller software design architecture to improve reusability and modularity, it is platform agnostic, light-weight and easy to install, it produces both binary and Text-based TEDS, supports a large array of physical units used in the hydrology field and also incorporates sensor data management provision. We have used the Consortium of Universities for the Advancement of Hydrologic Sciences, Inc.'s Observations Data Model (CUAHSI ODM) as a test case to demonstrate how backend demands on data management can be incorporated in front end applications such as the TEDS. We have tested the results of our application with examples provided in the IEEE 1451.0 documentation, and both results show agreement.

Keywords: IEEE 1451, Transducer Electronic Data Sheet, CUAHSI's Observations Data Model, Automatic Data Management, Python, Low-Cost Sensor Network

1. Introduction

Implementing a sensor network requires, in most cases, the use of proprietary protocols and programming languages to configure sensors and loggers. When using different sensors platforms the configuration task becomes more complex due to the need of having to learn different proprietary data formats and protocols which constitute a bottleneck for the expansion of sensor networks [1]. In response to this rising complexity the Institute of Electrical and Electronics Engineers (IEEE) have sponsored the development of the IEEE 1451 standard [2, 25, 46] to introduce a common standard. This standard is a family of eight (8) (sub)standards, as of the date of this writing, that provides the common interface and enabling technology for the connectivity of transducers to microprocessors, control and field networks, and data acquisition and instrumentation systems in a plug-and-play fashion [2]. The standard puts forth the concept of a Transducer Electronic Data Sheet (TEDS) which plays the role of an identification card attached to smart transducers enabling transducers self-identification, self-description, self-diagnosis, self-calibration, and plug-and-play functionality.

While offering substantial benefits, implementation of the IEEE 1451 standards, more importantly the TEDS for traditional sensors and sensor platforms, remains a tedious task especially for practitioners with no or little background in computer programming. Hence, the development of an application that incorporates an intuitive Graphical User Interface (GUI) to generate the TEDS represents a critical step to both reduce work load and also to avoid mistakes as manual compilation has shown to be tedious and error prone [1, 4, 11, 12, 45, 55-57]. Multiple mistakes in manually assembling the TEDS in addition to lack of knowledge of the IEEE 1451 standard have been reported in Rana et al. [11] where, for example, instead of providing the "Design operational upper and lower measurement limits" for the temperature sensor they use in their setup; they provide the operating temperature range for the Arduino board itself.

TEDS generators have been developed before; see [3, 5-10, 44, 61-62]. A key aspect to evaluate most of these TEDS generators is their level of convenience vis-à-vis the user's experience or expertise with this standard. In other words, these applications require the users to know the intricate details and technical terms of the standard. Acquiring detailed knowledge of the standard however is a time consuming and onerous task for one because it is an expansive standard, for another because it is filled with technical jargon. For example, the TEDS generator in [6] requires the users to understand how physical units are represented in the TEDS and they have to insert the proper and correct forms themselves, which has shown to be a likely source of error in TEDS creation; see table 1 for more details. Additionally, these applications suffer from a general lack of controlled vocabularies to allow sensors data to be organized, cataloged and retrieved painlessly. For example, in [3] temperature unit is “celsius” while in [6] it is “DEGREE CELSIUS”. A somehow similar issue has been raised by Hu et al. [37] prompting them to develop ontology to model terminologies defined in IEEE 1451.4 [63] and consequently, bridge the gap between IEEE 1451 and SensorML [65-67]. None of these applications have an interface or mechanism that would allow integration of controlled vocabularies such as the CUAHSI ODM [29] Physical Units controlled vocabularies, which we seek to use in the present application to improve on and ease the burden of inserting correct units. Another important aspect that has been overlooked and omitted in TEDS applications is the fact that the IEEE 1451 is flexible enough to allow practitioners to store ancillary information (metadata) about the sensors data values that in turn can be used to provide traceable heritage from raw measurements to usable information and be unambiguously interpreted in the form of Text-based TEDS. For this TEDS category, the IEEE 1451.0 documentation [12] only provides a suggestive model of the Text portion. Hence, this TEDS category is not fully documented to be seamlessly replicated.

Table 1. *Summary of the existing TEDS editors characteristics.*

References	Programming language	Comments
[10]	Labview	Poor documentation for TEDS implementation
[5]	Delphi	Poor documentation for TEDS implementation
[6]	C#	Manual decomposition of Physical units, Use IEEE 1451 keywords
[7]	ANSI C	Manual decomposition of Physical units, Use IEEE 1451 field type names abbreviations for GUI labels
[8]	C/C++	Poor documentation for TEDS implementation
[9]	C++	Poor documentation for TEDS implementation
[3]	Not reported	User-friendly GUI, Preliminary TEDS editor, No further improvement, Published prior to the acceptance and publication of IEEE 1451.0,
[44]	Web Technologies	Manual decomposition of Physical units, Heavily use abbreviations for GUI labels
[62]	Web Technologies	Well documented TEDS editor, but requires familiarity with IEEE 1451 and HEX manipulation

Finally, while a mature language, the use of Python programming language in sensor application is still at its infancy. A review of the current status of the IEEE 1451 standard based sensor applications [52] such as GUI for data management and visualization, communication drivers and data logging demonstrates that programming languages such as the C family, Java and LabVIEW (a proprietary language and programming environment) remain the de facto programming languages adopted. Out of 24 GUI applications reported in that review, only one [53] has used Python. A similar dominance of those programming languages has also been noted in TEDS creator applications (Table 1).

This paper presents a much improved and platform-independent TEDS generator application developed using Python programming language and incorporates provision for a unified standards-based automatic data storage system incorporating IEEE 1451.0 features [12, 16].

The paper is organized as follows: Section II provides an overview of the IEEE 1451 family of standards, Section III provides an understanding of the TEDS, Section IV presents the key design principles underlying the application, Section V shows an essential UML model of the application, Section VI presents some of GUIs and demonstrates some results and finally, Section V summarizes the paper and provides insight to future works.

2. Overview of the IEEE 1451 Standards

The anatomy of the IEEE 1451 standard can be succinctly explained using the Smart Transducer model as a reference (figure 1 (a) from [13]). It comprises analog or digital sensor or actuator elements, a processing unit, and a communication interface as a single block element [14]. The IEEE 1451 retrofits the Smart Transducer model and splits it into two major components: the Transducer Interface Module (TIM) and the Network Capable Application Processor (NCAP) linked by a Transducer Independent Interface (TII) (fig. 1 (b)).

The TIM consists of a Transducer Signal Conditioning and Data Conversion unit, a maximum of 255 sensors and actuators, and the TEDS files. This differentiates the IEEE 1451 smart transducer from the conventional smart transducer. The NCAP performs application processing and network communication functions. The IEEE 1451 smart transducer model (fig. 1(b)) demonstrates that both TIMs and NCAPs can be implemented using off-the-shelf components [15] like the Arduino (http://www.arduino.cc/) microcontroller as TIM and the Raspberry Pi (http://www.raspberrypi.org/) as a NCAP which is what we ultimate lively seek to accomplish.

We focus on the presentation of a straightforward creation of the various types of TEDS as defined in the IEEE 1451.0 standard [12]. The IEEE 1451.0 standard member provides a common basis to enable interoperability between the other members (named IEEE 1451.X). Within this interoperability philosophy, the IEEE 1451 standard defines a common set of commands for setting up, accessing sensors and actuators connected in various physical configurations such as point-to-point, distributed multi-drop and wireless configurations as well as reading and writing the data used by the system [13]. It defines the functions that are to be performed by a transducer interface module (TIM) and the common characteristics for all devices that implement the TIM. More importantly, it specifies the formats for the TEDSs and defines a set of commands to facilitate the setup and control of the TIM. It also defines Application Programming Interfaces (APIs) to facilitate communications with the TIM and with applications through a network link. As this paper is concerned with the IEEE 1451.0 member only, the bolts and nuts of the IEEE 1451 family of standards are not explained here. Further details can be found in [2] and [13].

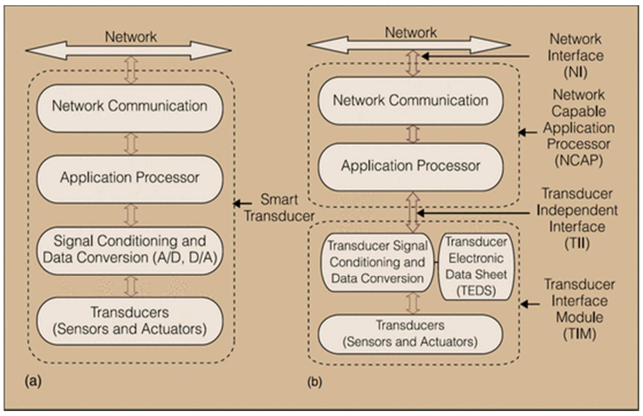

Network
Network Communication
Application Processor
Signal Conditioning and Data Conversion (A/D, D/A)
Transducers (Sensors and Actuators)
Smart Transducer
(a)
Network
Network Communication
Application Processor
Transducer Signal Conditioning and Data Conversion
Transducer Electronic Data Sheet (TEDS)
Transducers (Sensors and Actuators)
Network Interface (NI)
Network Capable Application Processor (NCAP)
Transducer Independent Interface (TII)
Transducer Interface Module (TIM)
(b)

Figure 1. *Smart Transducer model (a) vs IEEE 1451 Smart Transducer Model (b).*

3. TEDS Background

The IEEE 1451.0 defines 16 TEDS types [46] divided in two general categories based on the content of the TEDS information block and can accommodate a wide range of potential sensors and actuators. These are: the Binary TEDS which are divided into two sub-categories, Required and Optional TEDS; and the Text-based TEDS which are all Optional TEDS. However, the exceptions to the rule are the Optional Manufacturer-defined and End User's Application Specific TEDS which can be either a Text-based or Binary TEDS the choice of which is left up to the user's discretion. Fig. 2 depicts the division of the IEEE 1451.0 TEDS.

The IEEE 1451.0 format for binary TEDS consists of three major blocks: the Length block specifying the number of octets in the information block and the checksum, the Information block (a series of ordered entries) and the Checksum block to verify data integrity. In addition, each entry in the information block uses the Time-Length-Value (TLV) format where T is an octet representing the entry number, L represents the number of octets in the V, and V is the data value octets describing the property of the entry. There is an exception to this convention however; the User's Transducer Name TEDS, even when implemented as a binary TEDS, has the "TCName" user-dependent entry, a name to recognize the TIM that does to not follow the TLV rule. The data block of the Text-based TEDS also ignores this convention. It is also worthwhile noting that not all of the defined entries in any TEDS need to be included in the implemented TEDS information block. For example, the field indicating the time and date when a sensor was last calibrated can be omitted in the Calibration TEDS.

The TEDS for smart sensors are typically stored and shipped on a nonvolatile memory embedded in the sensor or the TIM. However, the standard allows the TEDS to be stored in other places within the user's system; in this case they are known as Virtual TEDS. These permit traditional analog sensors to also enjoy the benefits of TEDS without needing to be retrofitted with an embedded nonvolatile memory. This is achieved by generating and storing the TEDS on the sensor platform or downloading it from an online repository managed by the sensor manufacturer. Once implemented and loaded to the TIM or somewhere within the system a copy of each TEDS is transferred to the NCAP; this action utilizes the TEDS information to help identify which TIMs are attached to it and further identify which transducers are attached to a TIM [16].

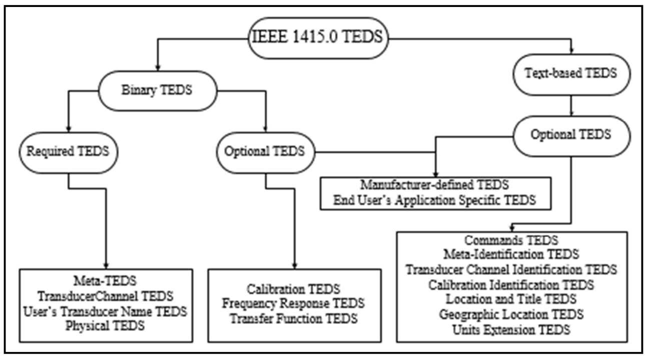


Figure 2. *IEEE 1451.0 TEDS types and their categorization.*

4. Application Design Principles

The TEDS generator is built upon the following key design principles:

4.1. Minimal Dependency on Python's Non-Native Package

Our application has been developed in Python, an interpreted, interactive, and object-oriented programming language. It is built using primarily python's built-in packages to make installation and utilization as easy and convenient as possible. This design principle allows the application to be as light-weight as possible in terms of memory usage and also permits it to run on resource constrained devices like the Raspberry Pi. Moreover, because the application is developed in Python it is platform agnostic which means that it is able to run common platforms such as Windows, Mac OS, Linux, Android, etc., without alteration.

4.2. Open-Source GUI Development Toolkit (WxPython)

Our application provides a GUI to enable users to create the implemented TEDS. The stand-alone WxPython [49] open-Source GUI development toolkit was selected for its simplicity in both utilization and installation compared to the use of other toolkits such as PyQt4 [50]. The package is free for anyone to use in commercial and non-commercial applications and is also platform agnostic.

4.3. Model-View-Controller Architecture

A key strategy in the development of our application was the introduction of the Model-View-Controller, MVC, approach developed by Tryge Reenskaug [17-24, 51]. It permits the separation of the View, Controller, and Model components such that alterations and further developments on one component will not affect the usability of the other two [17]. It thus greatly aids in meeting our objective for software reusability and ease-of-use, parallel implementation of components, and modularity. The design rules and conventions for reusable application/classes suggested by [26] were followed wherever they are applicable to our application.

4.4. Help System

Besides the translation of the IEEE 1451 technical terms into more comprehensible terms, our application has been augmented with a help system in the form of a tooltip that gives instructions to the user with regard on how to select entries in the Graphical User Interface. In addition, an asterix (*) is added to the essential fields required to create the TEDS.

4.5. TEDS-Based Controlled Vocabularies and Ready-Made Metadata

The CUAHSI Observations Data Model (ODM) [29] is a relational database designed to store time series observations with sufficient ancillary information (metadata) about the data values to provide traceable heritage from raw measurements to usable information [27-28]. [28] provides a detailed description of the tables and associated attributes in this data model, a listing of the fields contained in each table, a description of the data contained in each field and its data type, specific constraints imposed on each field, and discussion on how each field should be populated. Because the authors deal with a substantial amount of time series generated data, the motivation arose to integrate information system requirements up front into the TEDS. A key component for the successful functioning of a distributed time series database system is the provision and subsequent adherence to controlled vocabularies. We demonstrate using CUAHSI ODM Controlled Vocabularies how these can be integrated into our system, in this case the inclusion of the physical units CV, to aid discovery, search and publication of sensor data at the NCAP level. We have also integrated a “crawler” that automatically interrogates the CVs we are using for integration of the latest version.

Currently available Python packages intended to manipulate units do not natively support conversion of units with the naming convention adopted by CUAHSI like *'kilograms per 1000 square meter'* or by the National Institute of Technology and Standard [35]. In addition, there are a number of peculiarities and subtleties in the unit names and abbreviation that these packages cannot manage out-of-the-box such as “*gallon”* represented by lower (g) and uppercase (G) abbreviations. On top of those issues, these packages could not be used because they perform what we may call “*meshed dimensional conversion*” by analogy to a meshed sensor network. A special Python package having the capabilities to perform *Dimensional Analysis*, *Unit Reduction and Conversion* has been developed according to the unit definition and conversion algorithm proposed by [36] after pre-processing the units. In addition, a *Compact Representation of the units* according to [34], a structure that encodes only the exponents of a physical unit decomposed into the product of the seven Système International (SI) base units and two SI supplementary units (radian and steradian) as a vector in a well-defined order (e.g., joule (m^2*kg/s^2) is [0,0,2,1,-2,0,0,0,0]), is also a feature of our newly developed Python package to meet the unit representation specification in IEEE 1451. A full description of our package however, is beyond the scope of this paper and we just introduce its existence for the time being.

4.6. TEDS Structure Model

Each entry in the TEDS data block requires the field types (Type), generally a sequence of numbers, to be known. In addition, each entry has a specified data type (Int, float, etc.) and is represented with a certain length or number of octets. Some data blocks also contain nested entries such as parent entries with children entries where both parents and children have to follow the TLV format. This means that the V in the parent entry is a sequence of TLVs. Generating these TLVs is an involved and onerous task that complicates the creation of TEDS thus prompting the desire to develop a *one-size-fits-all* [30] method to generate the desired TEDS. Therefore, we need a model/pattern of the TEDS structure defined in IEEE 1451.0 for each TEDS type to be implemented. In our application, the TEDS structure is modeled using the following general pattern (fig. 4):

TEDS Type = Field Type: {Field Name: {Key1: Field Type, Key2: Number of bits, Key3: Value, Key4: Data type, Children: Children_Fields/None}}.

The Python dictionary type allows the capture of more structured information than a simple list of elements and its nesting feature allows building up and access complex information structures directly and easily. The strength of this data structure is that it can accommodate and store an unlimited number of data structures and types.

Adoption of this strategy also permits i the model to offer an all-encompassing to encapsulate the user inputs and the *one-size-fits-all* method processing the inputs to create the TEDS. In addition, if the IEEE standard is subject to some future revisions, the same method can be used without modification to accommodate new entries by simply updating the TEDS structure model with only, if at all, minimal changes needed in the View and Controller classes.

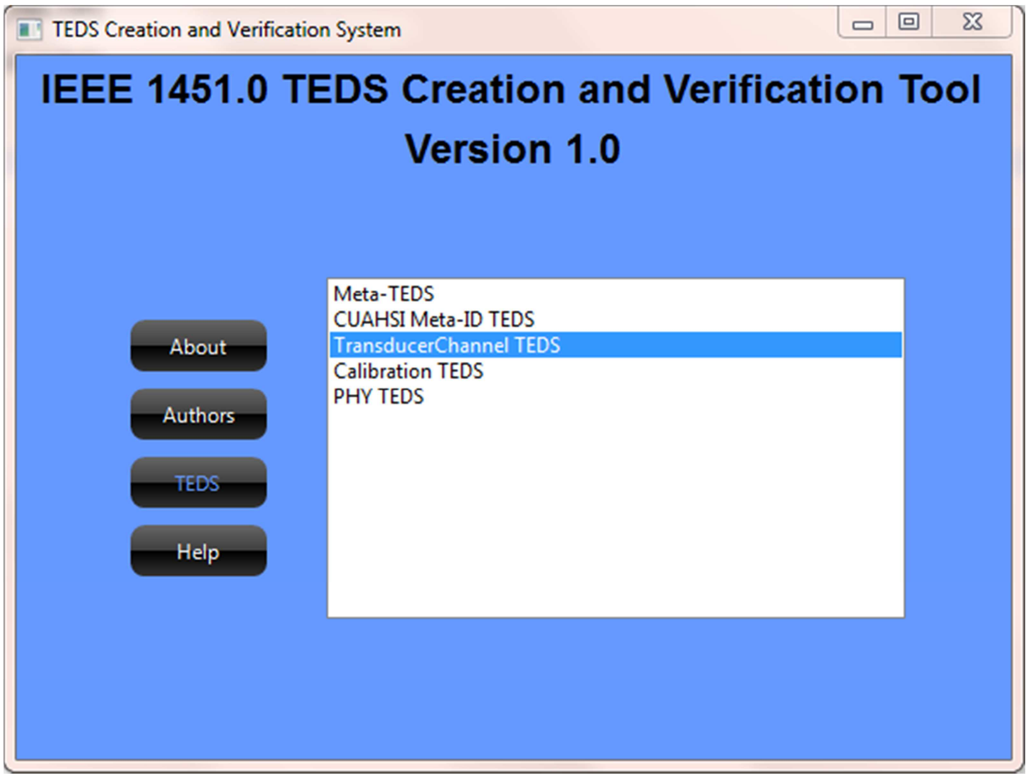
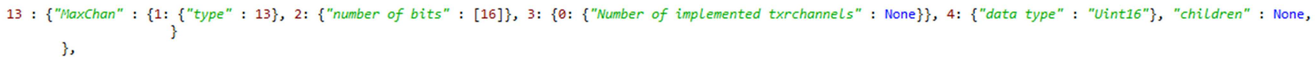

TEDS Creation and Verification System
IEEE 1451.0 TEDS Creation and Verification Tool Version 1.0
About
Authors
TEDS
Help
Meta-TEDS
CUAHSI Meta-ID TEDS
TransducerChannel TEDS
Calibration TEDS
PHY TEDS

Figure 3. *The Main View of the application.*



```
13 : {"MaxChan" : {1: {"type" : 13}, 2: {"number of bits" : [16]}, 3: {0: {"Number of implemented txrchannels" : None}}, 4: {"data type" : "Uint16"}, "children" : None,
                    }
        },
```

Figure 4. *Example of the TEDS model used as a helper to create the TEDS.*

5. A Simplified UML Model of the Application

The MVC architecture while conceptually simple nevertheless requires a substantial effort when implementing. The goal of the MVC triad is to separate application components into three independent action blocks. Each one of them however, still requires considerable coding work in addition to needing to stipulate data and information flow interfaces that require careful consideration. The three components are tasked as follows:

a. The Model handles data storage, provision and processing tasks and operates independently of the other two. It is also responsible for broadcasting messages about changes occurring on the data to its dependent view-controller pairs.
b. The View requests and presents the data and the state of the application held by the Model to the user through a GUI. In order to obtain information about the Model contents, the View component must register itself as a dependent or an observer of the Model using an Observer Pattern [41].
c. The Controller is the interface or the glue between the View and the Model and handles the user interaction with the application. It receives and registers user actions and determines the action to be taken, for example, calling a method. Like the View, the Controller registers itself as an observer of the Model.

To better illustrate the complexity of the interactions of all components in our application, we have developed a UML representation that depicts the dependencies and inheritance, as shown in Fig. 5. The UML model shows that our application is actually built on a variant of the MVC pattern named Hierarchical MVC [31, 33] which is a hierarchy of parent-child MCV layers. We introduced this as a convenient way of structuring the Main GUI (Fig. 3) from which users can choose the TEDS to be implemented (e.g., Fig. 6) which themselves are based on the MVC pattern. Hence, the application is built as layered/hierarchical MVC in which a link is made between the main layer and the children layers. To better illustrate the HMVC setup, we provide the following example: Fig. 3 is modeled in the UML Model by the classes/objects: Main View, Main Controller and Model that together constitute the parent MVC layer; The Main Controller follows up on the user actions (click a button to implement a TEDS) on this GUI and calls the MVC layer associated to the clicked button. We use a common model (IEEE1451Dot0Model) instead of using a separate model for each controller-view pair.

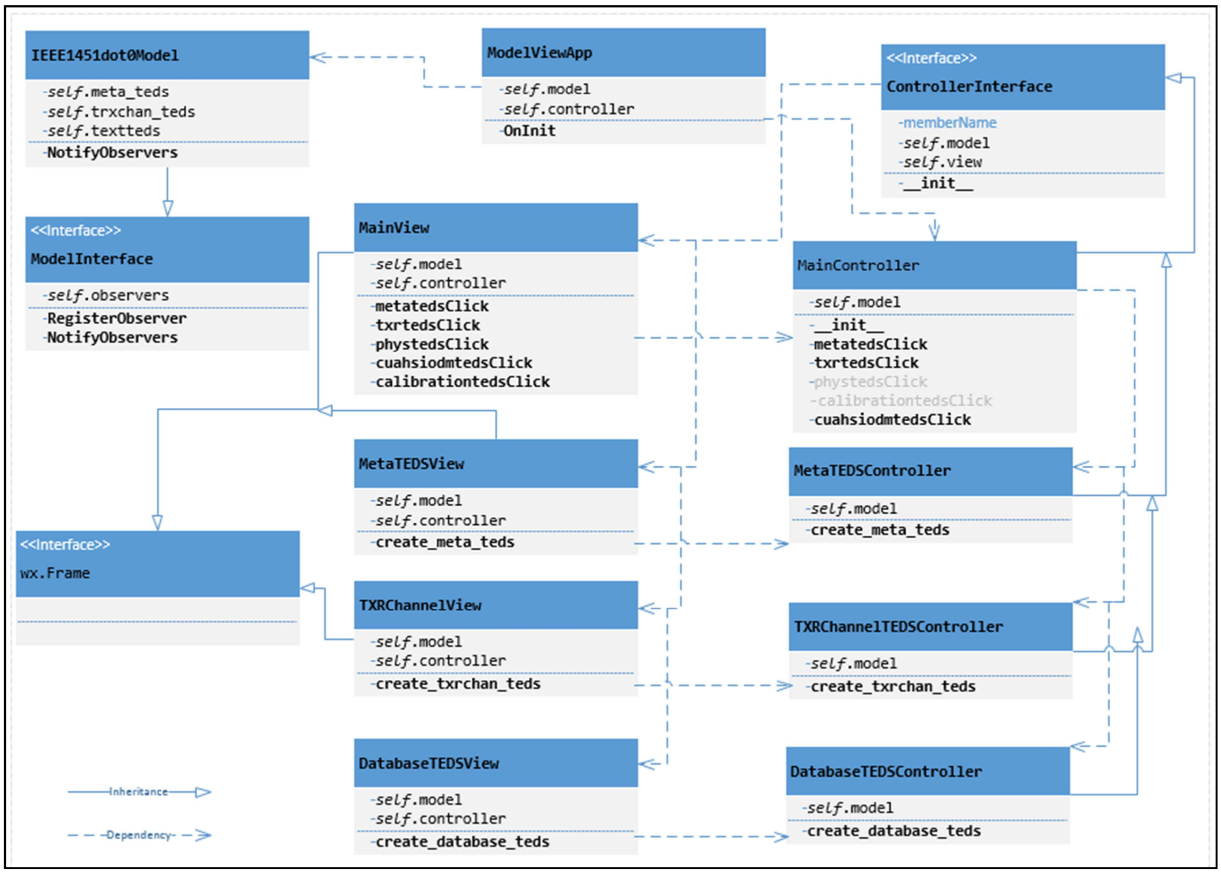
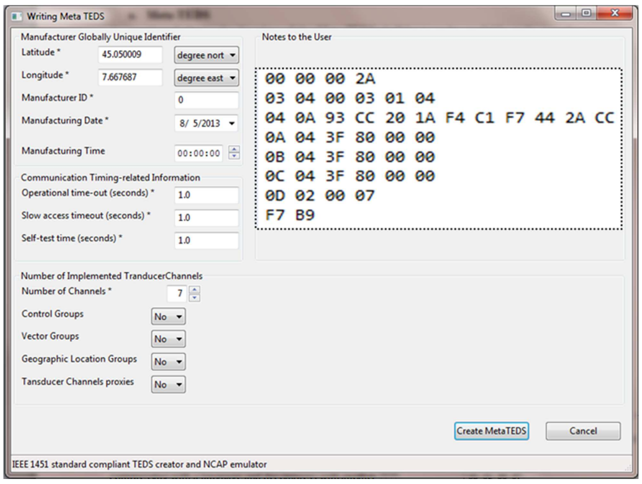


Figure 5. *An Essential UML model of the TEDS generator application.*



Manufacturer Globally Unique Identifier
Latitude * 45.050009 degree nort
Longitude * 7.667687 degree east
Manufacturer ID * 0
Manufacturing Date * 8/ 5/2013
Manufacturing Time 00:00:00

Communication Timing-related Information
Operational time-out (seconds) * 1.0
Slow access timeout (seconds) * 1.0
Self-test time (seconds) * 1.0

Notes to the User

```
00 00 00 2A
03 04 00 03 01 04
04 0A 93 CC 20 1A F4 C1 F7 44 2A CC
0A 04 3F 80 00 00
0B 04 3F 80 00 00
0C 04 3F 80 00 00
0D 02 00 07
F7 B9
```

Number of Implemented TranducerChannels
Number of Channels * 7
Control Groups No
Vector Groups No
Geographic Location Groups No
Tansducer Channels proxies No

IEEE 1451 standard compliant TEDS creator and NCAP emulator

Figure 6. *Meta-TEDS Created for the Arduino Uno.*

6. TEDS Implementation

Our work has been motivated in the context of developing affordable and smart hydro-climate sensor stations that are easily deployed in the field. Our goal is to provide a complete sensor-to-end user package in which data management aspects in addition to data submission and retrieval are handled automatically thus providing a solution that is as hands-off as it can be. As mentioned before we choose the Observations Data Model of the CUAHSI HIS [43] a system to store point based time series data using international standards such WaterML2.0 [58-59] and Water One Flow web services to provide computer-to-computer communications. One consequence of this specific adoption is the need to provide compliant metadata to adequately annotate the time series data.

The IEEE 1451.0 standard is flexible enough that it can be used to embed textual information into a TEDS where each block is an XML "instance document", the only constraint imposed. Researchers have taken steps to either suggest some additional entries [1] or develop new TEDS [38] that have the potential to fulfill some application specific needs such as reporting on sensor health. While possible, our application does not demand this extra work of supplying metadata and stays within the confinement of the TEDS list enumerated above. This is because the required metadata can easily be described using the CUAHSI Water Markup Language (WaterML) [39]-[40] making the information block of the Text-based TEDS comply with the XML "instance document" constraint imposed by IEEE 1451.0. The metadata is encoded in WaterML at the TIM level for transmission to the NCAP and an application at the NCAP level will consume the WaterML description to dynamically process and store the data properly into the ODM.

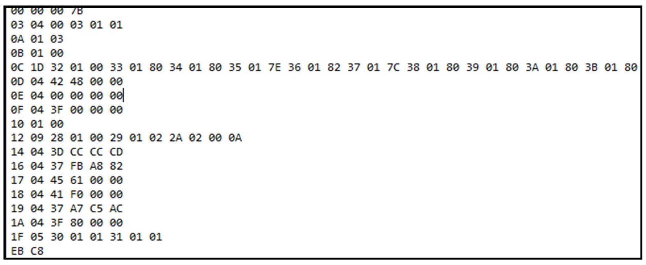
There are, however, three metadata tables that need direct user input: Variable, Source, and Site tables. The last two tables contain information common to all Transducer Channels attached to the TIM. There are three TEDS candidates to represent those tables: Meta-Identification TEDS, End User's Application Specific TEDS and Manufacturer-Defined TEDS. Based on the fact that the Meta-TEDS provides information common to all Transducer Channels, the Meta-Identification TEDS is the best candidate to represent the TEDS describing the Source and Site metadata tables in the CUAHSI ODM. This same reasoning holds for the Variable metadata table and the Transducer Channel Identification TEDS was selected to represent the information that will be loaded into the ODM Variable table. Our application implements the following TEDS.

6.1. Meta-TEDS

The Meta-TEDS ensures availability of information necessary to gain access to any Transducer Channel, plus the information common to all Transducer Channels. An essential Meta-TEDS consists of the key fields (with asterix) in figure 6. As an example, we show the geographic location of the Arduino Uno manufacturer in Turin, Italy together with a made-up manufacturing date and we also assume that a collection of seven sensors will be connected to the Arduino. Our application produces the TEDS as shown above where the first line indicates the total number of bytes in the TEDS, the second one is the TEDS identifier, the third one identifies the microcontroller manufacturer, the fourth to sixth lines show the communication-related time, the seventh is the number of sensors to be implemented, and the last one is the Checksum to ensure data integrity. To minimize the space occupation of the figures, we put the TEDS figure on the GUI.

6.2. Transducer Channel TEDS

The Transducer Channel TEDS describes and makes available all of the information concerning the Transducer Channel being addressed thus ensuring proper operation of the channel. We have used the MPX5050/MPXV5050G series pressure sensor [54] compatible with Arduino for testing, the results of which are shown below (Fig. 7). We do not show the GUI here because it is too large to fit into the text at a resolution high enough to discern the individual input field descriptions. Instead we show the TEDS content as a series of entries to facilitate its interpretation and understanding, but we could also group the whole content as a series of octets.



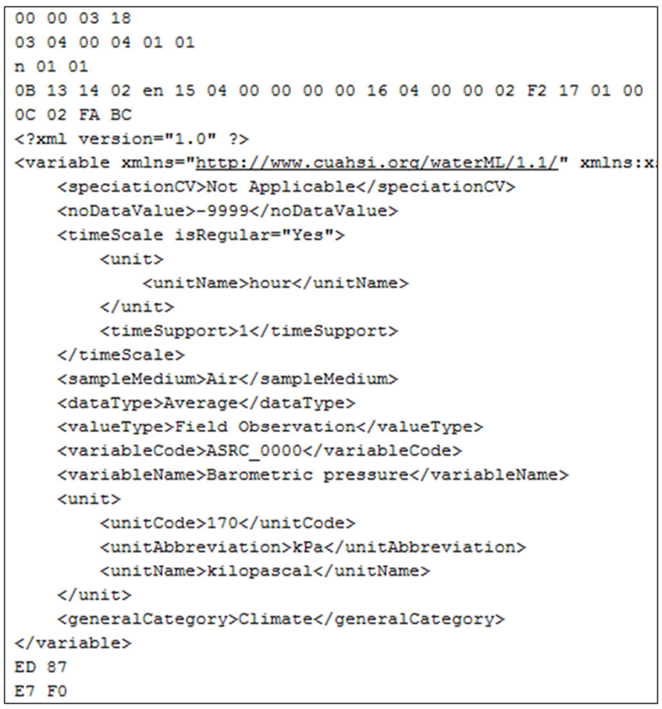
```
00 00 00 7B
03 04 00 03 01 01
0A 01 03
0B 01 00
0C 1D 32 01 00 33 01 80 34 01 80 35 01 7E 36 01 82 37 01 7C 38 01 80 39 01 80 3A 01 80 3B 01 80
0D 04 42 48 00 00
0E 04 00 00 00 00
0F 04 3F 00 00 00
10 01 00
12 09 28 01 00 29 01 02 2A 02 00 0A
14 04 3D CC CC CD
16 04 37 FB A8 82
17 04 45 61 00 00
18 04 41 F0 00 00
19 04 37 A7 C5 AC
1A 04 3F 80 00 00
1F 05 30 01 01 31 01 01
EB C8
```

Figure 7. *Transducer Channel TEDS for the MPXx5050 series.*

6.3. Transducer Channel Identification TEDS (Variable Metadata Table)

The Transducer Channel Identification TEDS falls into the category of Text-based TEDS that comprises two blocks: a binary which is a directory to allow a processor to locate and read the information block for a single language in the TEDS. Note that multiple information blocks in different languages are permitted. The second block, the Text (XML instance) based portion, should be created first as the binary portion contains sub-fields that depend on the memory size of the text portion. Text-based TEDS are much more cumbersome to be created manually than binary TEDS because byte counting, evaluation and subsequent checksum computations are labor intensive. They are also highly prone to error in addition to requiring multiple checksum computations for text based TEDS. This is even more difficult if the content is to be implemented in more than one language because the memory size of the text portion may vary from one language to another and the binary part has to contain the right information to allow the TEDS processor to accurately locate the text portion for each language.

Figure 8 shows the TEDS containing information about the ODM Variable table as a Water ML instance for the selected pressure sensor. Currently, our application supports only a single language (English) but there is the opportunity to extend it to support more than one language in the context of the "Internationalization of the CUAHSI Hydro Server" [42].



```
00 00 03 18
03 04 00 04 01 01
n 01 01
0B 13 14 02 en 15 04 00 00 00 00 16 04 00 00 02 F2 17 01 00
0C 02 FA BC
<?xml version="1.0" ?>
<variable xmlns="http://www.cuahsi.org/waterML/1.1/" xmlns:x
    <speciationCV>Not Applicable</speciationCV>
    <noDataValue>-9999</noDataValue>
    <timeScale isRegular="Yes">
        <unit>
            <unitName>hour</unitName>
        </unit>
        <timeSupport>1</timeSupport>
    </timeScale>
    <sampleMedium>Air</sampleMedium>
    <dataType>Average</dataType>
    <valueType>Field Observation</valueType>
    <variableCode>ASRC_0000</variableCode>
    <variableName>Barometric pressure</variableName>
    <unit>
        <unitCode>170</unitCode>
        <unitAbbreviation>kPa</unitAbbreviation>
        <unitName>kilopascal</unitName>
    </unit>
    <generalCategory>Climate</generalCategory>
</variable>
ED 87
E7 F0
```

Figure 8. *Text-based TEDS to describe the ODM variable table.*

For transmission purposes, the text information block (Water ML string) is compressed using the Gzip compression method natively available in Python and accepted by the IEEE 1451.0. This compression technique helps achieve a compression ratio of about 50%, which is important because data transmission/reception time is proportional to data packet size and a sensor node expends maximum energy in data communication [60]. We need to mention also that the compression has an impact on the checksum computation for the information block and subfields in the binary portion. To avoid users' mistakes in associating the Variable metadata to the appropriate sensor, the GUI generating this TEDS is integrated into the Transducer Channel TEDS GUI so they are implemented simultaneously.

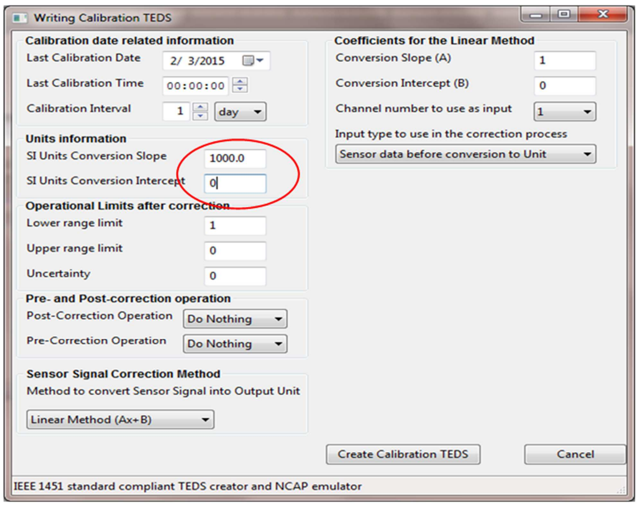
6.4. Meta-Identification TEDS (Source & Site Metadata Tables)

The Meta-Identification TEDS provides all information needed to identify the TIM plus any information common to all channels. Under this Text-based TEDS, we implement the ODM Source and Site Tables. The result is similar to the text shown Fig. 8 with the unique difference that this TEDS contains an information block with two XML elements "source" and "site". To minimize space, it is not presented here.

6.5. Calibration TEDS

Finally, the calibration TEDS provides all of the information used by the correction software in charge of converting the sensor output (electrical signal) into engineering units in connection with the Transducer Channel being queried. Our application supports both the Linear and General (Polynomial form) sensor data Correction Method (LCM/LGM) of the IEEE 1451.0.

We referred to [64] to implement the general correction method in which provision to accommodate a large array of sensor outputs as inputs is made. However, due to the complexity of this method and even though we provide instructions to users about how to provide the information, it must be implemented by users with an adequate comprehension of the general calibration method of the IEEE 1451 to correctly supply the information to the application. We need to point out here that none of the previously studied TEDS creators has implemented the general correction method.



Writing Calibration TEDS

Calibration date related information

Last Calibration Date	2/ 3/2015
Last Calibration Time	00:00:00
Calibration Interval	1 day

Units information

SI Units Conversion Slope	1000.0
SI Units Conversion Intercept	0

Operational Limits after correction

Lower range limit	1
Upper range limit	0
Uncertainty	0

Pre- and Post-correction operation

Post-Correction Operation	Do Nothing
Pre-Correction Operation	Do Nothing

Sensor Signal Correction Method

Method to convert Sensor Signal into Output Unit: Linear Method (Ax+B)

Coefficients for the Linear Method

Conversion Slope (A)	1
Conversion Intercept (B)	0
Channel number to use as input	1

Input type to use in the correction process: Sensor data before conversion to Unit

IEEE 1451 standard compliant TEDS creator and NCAP emulator

Figure 9. *The Calibration TEDS GUI.*

For unit-to-unit conversion, the application provides the conversion slope and intercept to the user on the fly using the Physical unit selected in the Transducer Channel TEDS GUI once the related Transducer Channel TEDS has been created. Fig. 9 shows those two values for the selected sensor and figure 10 shows the calibration TEDS created (LCM).

```
00 00 00 55
03 04 00 04 01 01
0A 08 43 00 9F E0 00 00 00 00
0B 08 01 E1 33 80 00 00 00 00
0C 0B 1E 04 3F 80 00 00 1F 04 00 00 00 00
0D 04 00 00 00 00
0E 04 42 48 00 00
0F 04 3F 00 00 00
10 01 00
11 01 00
14 10 29 02 00 01 2A 01 00 33 08 3D B8 51 EC 00 00 00 00
F6 FC
```

Figure 10. *The Calibration TEDS for the pressure sensor used.*

7. Summary

A practical user-friendly and platform-independent application designed to generate Transducer Electronic Data Sheet (TEDS) according to the IEEE 1451.0 standard using the Python open source programming language has been presented. It has been designed to ease the end user's work with an on-the-fly help system provided and constrained the user to provide required information for compliance to the specification of the standard. It has been successful in producing the supported TEDS including Text-based TEDS smoothly. The Hierarchical Model-View-Controller software design architecture has been used to produce concise, maintainable and reusable high quality software. It also demonstrates the promising capability of the Python programming language to be used in sensor applications and more importantly when the aspect of sensor data management is considered.

The reusable and modular software development mind will allow us to easily decode and test the TEDS for compliance and perform tasks necessary to achieve an Auto-programmable Data Acquisition System. We have tested the results of our application with examples provided in the IEEE 1451.0 documentation, and both results show agreement.

References

[1] J., Higuera, J., Polo, & M., Gasulla (2009). A ZigBee wireless sensor network compliant with the IEEE 1451 standard. In Proceedings of the IEEE Sensors Applications Symposium.

[2] K., Lee (2000). IEEE 1451: A standard in support of smart transducer networking. In Instrumentation and Measurement Technology Conference, 2000. IMTC 2000. Proceedings of the 17th IEEE (Vol. 2, pp. 525-528). IEEE.

[3] Liu, W. (2006). Design of teds writer, reader and testing system for transducer interface modules based on the ieee 1451 standard. State University of New York at Buffalo.

[4] D., Wobschall (2007). IEEE 1451—a universal transducer protocol standard. In Autotestcon, 2007 IEEE (pp. 359-363). IEEE.

[5] L., Cámara, O., Ruiz, & J., Samitier (2000). Complete IEEE-1451 node, STIM and NCAP, implemented for a CAN network. In Instrumentation and Measurement Technology Conference, 2000. IMTC 2000. Proceedings of the 17th IEEE (Vol. 2, pp. 541-545). IEEE.

[6] S., Manda & D., Gurkan (2009). IEEE 1451.0 compatible TEDS creation using. NET framework. In Sensors Applications Symposium, 2009. SAS 2009. IEEE (pp. 281-286). IEEE.

[7] J., Guevara, F., Barrero, E., Vargas, J., Becerra, & S. Toral (2012). Environmental wireless sensor network for road traffic applications. IET Intelligent Transport Systems, 6(2), 177-186.

[8] D., Markovic, U., Pesovic, & S., Randic (2012)"TEDS specification for IEEE 1451.0 smart Transducer," Telecommunications Forum (TELFOR), 2012 20th, vol., no., pp.1532, 1535, 20-22 Nov. 2012.

[9] N., Kularatna & B. H., Sudantha (2008). An environmental air pollution monitoring system based on the IEEE 1451 standard for low cost requirements. Sensors Journal, IEEE, 8(4), 415-422.

[10] Jevtic, N., & Drndarevic, V. (2013). Design and implementation of plug-and-play analog resistance temperature sensor. Metrology and Measurement Systems, 20(4), 565-580.

[11] R., Rana, N., Bergmann, & J., Trevathan (2011). Towards plug-and-play functionality in low-cost sensor network. In Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2011 Seventh International Conference on (pp. 265-270). IEEE.

[12] IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Standard 1451.0-2007.

[13] E. Y., Song & K., Lee (2008). Understanding IEEE 1451-Networked smart transducer interface standard-What is a smart transducer? Instrumentation & Measurement Magazine, IEEE, 11(2), 11-17.

[14] W., Elmenreich, & S., Pitzek (2003). Smart transducers-principles, communications, and configuration. na.

[15] S., Woods et al. (1996). "IEEE-P1451.2 Smart Transducer Interface Module," Proceedings Sensors Expo Philadelphia, pp. 25-38, October 1996, Helmers Publishing.

[16] J., Wiczer & K., Lee (2005). 'A Unifying Standard for Interfacing Transducers to Networks–IEEE 1451.0. In Proceedings of ISA Conference, Chicago, IL.

[17] G. E., Krasner & S. T., Pope, "A cookbook for using the model-view-controller user interface paradigm in smalltalk-80," J. Object Oriented Program., vol. 1, no. 3, pp. 26–49, Aug. 1988.

[18] T., Reenskaug (2003). The Model-View-Controller (MVC): Its Past and Present. [Online] Draft of August 20, 2003. Accessed January 15th, 2015. http://heim.ifi.uio.no/~trygver/2003/javazone-jaoo/MVC_pattern.pdf

[19] T., Reenskaug (2007). The original MVC reports. [Online] February 12, 2007. Accessed January 15th, 2015. http://heim.ifi.uio.no/~trygver/2007/MVC_Originals.pdf

[20] J., Deacon (2009). Model-view-controller (mvc) architecture. Online] [Citado em: 10 de março de 2006.] http://www.jdl.co.uk/briefings/MVC.pdf

[21] A., Bower & B., McGlashan (2000). Twisting the triad: the evolution of the Dolphin Samlltalk MVP application framework. Tutorial Paper for ESUG, 2000.

[22] S., Burbeck (1987). Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC). Accessed January 15th, 2015.

[23] M., Fowler (2006). GUI Architectures. [online] martinfowler.com. Available at: http://martinfowler.com/eaaDev/uiArchs.html [Accessed 16 Jan. 2015].

[24] T., Reenskaug (1979). Models - Views - Controllers. Technical note, Xerox PARC.

[25] K. B., Lee (2014). Smart Transducer Interface Standard for Sensors and Actuators. Industrial Communication Technology Handbook, Second Edition. Aug. 2014, 1 -17.

[26] R. E., Johnson & B., Foote (1988). Designing reusable classes. Journal of object-oriented programming, 1(2), 22-35.

[27] D. G., Tarboton, J. S., Horsburgh & D. R., Maidment (2007). CUAHSI community Observations Data Model (ODM) version 1.1 design specifications.

[28] J. S., Horsburgh & D. G., Tarboton (2008). CUAHSI Community Observations Data Model (ODM) Version 1.1. 1 Design Specifications.

[29] J. S., Horsburgh & D. G., Tarboton, D. R., Maidment & I., Zaslavsky (2008). "A relational model for environmental and water resources data." Water Resources Research, Vol. 44 (2008).

[30] M., Stonebraker & U., Cetintemel (2005). "One Size Fits All: An Idea Whose Time has Come and Gone". In Proceedings of the International Conference on Data Engineering (ICDE), 2005.

[31] Cai, J., Kapila, R. and Pal, G. (2000). HMVC: The layered pattern for developing strong client tiers. [online] Available at: http://www.javaworld.com/article/2076128/design-patterns/hmvc--the-layered-pattern-for-developing-strong-client-tiers.html [Accessed 17 Jan. 2015].

[32] B., Cogan (2010). HMVC: an Introduction and Application - Tuts+ Code Tutorial. [online] Code Tuts+. Available at: http://code.tutsplus.com/tutorials/hvmc-an-introduction-and-application--net-11850 [Accessed 17 Jan. 2015].

[33] W., Crow (2012). Hierarchical Model-View-Controller (HMVC): Planning for the Future. [online] Available at: http://somethingstatic.com/hierarchical-model-view-controller-planning-future/ [Accessed 17 Jan. 2015].

[34] B., Hamilton (1996). A compact representation of units. Hewlett-Packard Laboratories, Technical Publications Department.

[35] A., Thompson and B. N., Taylor (2008), Guide for the Use of the International System of Units (SI) NIST Special Publication 811, 2008 Edition (version 3.0). [Online] Available: http://physics.nist.gov/SP811 [Saturday, 24-Jan-2015 22:46:15 EST]. National Institute of Standards and Technology, Gaithersburg, MD.

[36] G. S., Novak (1995). Conversion of units of measurement. Software Engineering, IEEE Transactions on, 21(8), 651-661.

[37] Hu, P., Robinson, R., & Indulska, J. (2007). Sensor standards: Overview and experiences. In Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing ISSNIP'07.

[38] R. L., Oostdyk, C. T., Mata & J. M., Perotti (2006). A Kennedy Space Center implementation of IEEE 1451 networked smart sensors and lessons learned. In Aerospace Conference, 2006 IEEE (pp. 20-pp). IEEE.

[39] Zaslavsky, I., Valentine, D., Maidment, D., Tarboton, D. G., Whiteaker, T., Hooper, R., & Rodriguez, M. (2009). The evolution of the CUAHSI Water Markup Language (WaterML). In EGU General Assembly Conference Abstracts (Vol. 11, p. 6824).

[40] Valentine, D., Zaslavsky, I., Whitenack, T., & Maidment, D. R. (2007). Design and implementation of CUAHSI WATERML and WaterOneFlow Web services. In Proceedings of the Geoinformatics 2007 Conference, San Diego, California (pp. 5-3).

[41] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: elements of reusable object-oriented software. Pearson Education.

[42] J. Sadler, S. Bolster, D. Ames and E. Nelson (2013). Internationalizing HydroServer - Multilingual Support for Water Data Sharing. In: CUAHSI conference on Hydroinformatics and Modeling. [online] Available at: https://www.cuahsi.org/PageFiles/2013PosterAbstracts.pdf [Accessed 25 Jan. 2015].

[43] D. G. Tarboton, J. S. Horsburgh, D. R. Maidment, T. Whiteaker, I. Zaslavsky, M. Piasecki, J. Goodall, D. Valentine, and T. Whitenack, "Development of a community hydrologic information system," in 18th World IMACS Congress and MODSIM09 International Congress on Modelling and Simulation, 2009, pp. 988-994.

[44] Barrero, F., Toral, S., Vargas, M., & Becerra, J. (2012). Networked Electronic Equipments Using the IEEE 1451 Standard—VisioWay: A Case Study in the ITS Area. International Journal Of Distributed Sensor Networks, 2012, 1-12. doi:10.1155/2012/467124

[45] T. A. dos Santos Filho, A. C. R. da Silva, A. Luiz, A. Nogueira, S. R. Rossi, E. A. Batista (2010). Descricao Dos Teds Para O Controle De Motores De Passo Em Conformidade Com Do Padrao IEEE 1451. [online] http://www.eletrica.ufpr.br/anais/cba/2010/Artigos/65581_1.pdf [Accessed 01-29-2015]

[46] K. Lee, A Smart Transducer Interface Standard for Sensors and Actuators, The Industrial Information Technology Handbook, Zurawski R. (Ed.), CRC Press, Boca Raton, FL, 2004

[47] Ilyas, M., & Mahgoub, I. (Eds.). (2004). Handbook of sensor networks: compact wireless and wired sensing systems. CRC press.

[48] National Instruments, (2006). Sensor Calibration with TEDS Technology. [online] Available at: http://www.ni.com/white-paper/4043/en/ [Accessed 29 Jan. 2015].

[49] N. Rappin and R. Dunn, wxPython in action. Greenwich, Conn.: Manning, 2006.

[50] M. Summerfield, Rapid GUI programming with Python and Qt. Upper Saddle River, NJ: Prentice Hall, 2008.

[51] Sasine, J. M., & Toal, R. J. (1995, November). Implementing the model-view-controller paradigm in Ada 95. In Proceedings of the conference on TRI-Ada'95: Ada's role in global markets: solutions for a changing complex world (pp. 202-211). ACM.

[52] A. Kumar, V. Srivastava, M. Singh and G. Hancke, 'Current Status of the IEEE 1451 Standard Based Sensor Applications', IEEE Sensors Journal, pp. 1-1, 2014.

[53] Fadzil, M. H., Abas, M. A., & Hakiim, A. K. (2010, April). Development of Environmental Monitoring Data Management System using OSS Python. InProceeding of the International Conference on Electrical and Computer Engineering.

[54] Freescale Semiconductor (2010). 'Integrated Silicon Pressure Sensor On-Chip Signal Conditioned, Temperature Compensated and Calibrated', 2010. [Online]. Available: http://cache.freescale.com/files/sensors/doc/data_sheet/MPX5050.pdf. [Accessed: 04- Feb- 2015].

[55] Kamala, J. and Umamaheswari, B. "IEEE 1451.0-2007 compatible smart sensor readout with error compensation using FPGA", International Journal of Sensors and Transducers, Vol. 102, Issue 3, pp. 10-21, 2009 (Elsevier Publishers).

[56] R. J. Costa, G. R. Alves, and M. Zenha-Rela, "Extending the IEEE 1451.0 Std. to serve distributed weblab architectures," in 1st Experiment@ International Conference (exp.at'11), Calouste Gulbenkian Foundation, Lisboa-Portugal, 2011.

[57] G. Giorgi and C. Narduzzi, "Instrumentation electronic data sheets: IEEE 1451-like extension to measuring systems, "Instrumentation and Measurement Technology Conference (I2MTC), 2012 IEEE International, 2012.

[58] Almoradie, A., Popescu, I., Jonoski, A., & Solomatine, D. (2013). Web Based Access to Water Related Data Using OGC Water ML 2.0. Specialissue, 3(3). doi:10.14569/specialissue.2013.030310

[59] P. Taylor, S. Cox, G. Walker, D. Valentine and P. Sheahan, 'WaterML2.0: development of an open standard for hydrological time-series data exchange', Journal of Hydroinformatics, vol. 16, no. 2, p. 425, 2014.

[60] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: a survey. Computer networks, 38(4), 393-422.

[61] D., Wobschall (2008). Networked sensor monitoring using the universal IEEE 1451 standard. Instrumentation & Measurement Magazine, IEEE, 11(2), 18-22.

[62] Ma, A. Cherian and D. Wobschall, 'IEEE 1451 Development Kit Description', Esensors Inc., 2013. [Online]. Available: http://eesensors.com/media/wysiwyg/pdf/1451_manual.pdf. [Accessed: 12- Feb- 2015].

[63] IEEE Standard 1451.4 (2004) IEEE Standard for a Smart Transducer Interface for Sensors and Actuators: Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats, IEEE Standard 1451.4-2004.

[64] Eccles, L. (1999). IEEE-1451.2 Engineering Units Conversion Algorithm. Sensors Magazine, [online] (Volume 16, No. 5). Available at: http://archives.sensorsmag.com/articles/0599/0599_p107/index.htm [Accessed 30 Jan. 2015].

[65] Botts, M. (2002). Sensor Model Language (SensorML) for In-situ and Remote Sensors IPR. OpenGIS Project Document.

[66] Reichardt, M. (2005). Sensor web enablement: An OGC white paper. Open Geospatial Consortium (OCG).

[67] M. Botts and L. McKee, 'A Sensor Model Language: Moving Sensor Data onto the Internet | Sensors', Sensors Magazine, 2003. [Online]. Available: http://www.sensorsmag.com/networking-communications/a-sensor-model-language-moving-sensor-data-internet-967. [Accessed: 14- Feb- 2015].