ISSN 2039-7941
Anno 2016_Numero 336

rapporti tecnici INGV

Un metalinguaggio versatile per strumentazione con interfaccia di comunicazione seriale

336

Istituto Nazionale di Geofisica e Vulcanologia

Segreteria di Redazione
Francesca Di Stefano - Referente
Rossella Celi
Tel. +39 06 51860068
redazionecen@ingv.it

in collaborazione con:
Barbara Angioni (RM1)

REGISTRAZIONE AL TRIBUNALE DI ROMA N.173 | 2014, 23 LUGLIO

Sede legale: Via di Vigna Murata, 605 | Roma

ISSN 2039-7941
Anno 2016_Numero 336

rapporti tecnici INGV

UN METALINGUAGGIO VERSATILE PER STRUMENTAZIONE CON INTERFACCIA DI COMUNICAZIONE SERIALE

GianPaolo Donnarumma[1], Sergio Guardato[2], Giovanni Iannaccone[2]

[1]**SZN** (Stazione Zoologica "A. Dohrn", Napoli)
[2]**INGV** (Istituto Nazionale di Geofisica e Vulcanologia, Sezione di Napoli - Osservatorio Vesuviano)

336

Indice

Introduzione

La grande variabilità dei numerosi fattori fisici, chimici e biologici che caratterizzano in particolare l'ambiente marino, obbliga ad equipaggiare i sistemi progettati per il suo studio e monitoraggio con più tipologie di sensori relativi a differenti discipline, anche al fine di ottimizzare il costo di un'infrastruttura di osservazione sottomarina. Infatti, frequentemente tali sistemi sono equipaggiati con sensori oceanografici (correntometri, sensori CTD), geofisici (sensori di pressione, idrofoni a bassa frequenza, sismometri, tiltmetri, ecc), chimici (sensori di torbidità, ossigeno disciolto, ecc). Inoltre, in siffatti sistemi, sono anche integrati sensori per la misura dei parametri di stato, indispensabili per un corretto funzionamento dei sensori e, più in generale, per il controllo dello stato di tutta l'elettronica presente. Gran parte di questi sensori utilizza una comunicazione digitale di tipo seriale (TTL, RS-232, RS-422, RS-485, …) connessi a sistemi di gestione/acquisizione che hanno a disposizione diverse porte seriali per consentire l'integrazione della sensoristica necessaria allo svolgimento della propria missione.

In fase di progettazione del sistema globale, dopo aver identificato i sensori da utilizzare, si procede ad uno studio del protocollo di comunicazione relativo ad ognuno dei sensori componenti il sistema. In tale fase bisogna comprendere quali comandi inviare al particolare sensore (ad esempio, di lettura), e in che formato verrà restituita la misura desiderata. A seconda della configurazione scelta, poi, bisognerà implementare, tramite software/firmware, dei drivers specifici per consentire il corretto funzionamento del sensore all'interno del sistema (configurazione) e l'acquisizione dei dati di misura (funzionamento).

Durante il processo di integrazione di un sensore all'interno di un sistema di misura complesso, viene normalmente sviluppato un driver *ad-hoc* come modulo del software di controllo. Il driver ha la funzione di creare un'interfaccia in grado di mascherare il protocollo proprietario ed ottenere dei dati nel formato richiesto. Tale operazione richiede l'implementazione, all'interno del software, di driver specifici per ogni dispositivo.

Tutto questo limita notevolmente la flessibilità di un sistema di osservazione multi-parametrico in quanto l'eventuale aggiunta di un nuovo sensore o, semplicemente, il riposizionamento di un altro tipo di sensore già presente su una porta seriale differente, comporta l'aggiornamento di tutto il software/firmware di controllo. Inoltre, un approccio di questo tipo rende minima la portabilità di eventuali drivers già realizzati che dovrebbero comunque essere ricompilati ed adattati ad eventuali nuovi sistemi, in special modo quando si decide di passare ad un'altra piattaforma hardware.

Nell'ambito della gestione di osservatori da fondo marino, l'esigenza di semplificare l'uso di sensori di tipologia differente e giungere ad una standardizzazione è stata sempre molto sentita e sono state proposte varie possibili soluzioni.

Ad esempio, lo "*Smart Ocean Sensor Consortium*" (SOSC) raggruppa produttori ed utenti di sensori oceanografici al fine di migliorare l'affidabilità, l'utilità e la convenienza delle reti di sensori attraverso lo sviluppo e la promozione di interfacce e protocolli standard. In particolare il SOSC ha proposto un protocollo denominato PUCK (*Programmable Underwater Connector with Knowledge*), originariamente sviluppato dal *Monterey Bay Aquarium Research Institute* [O'Reilly & Reed, 2012].

PUCK definisce un protocollo di comunicazione seriale, di tipo RS-232, ed Ethernet per rendere standard le operazioni di comunicazione e memorizzazione dei dati ottenuti da strumenti digitali. Questo protocollo, memorizzando fisicamente informazioni dello strumento nello strumento stesso, consente di automatizzare le operazioni di installazione, configurazione e funzionamento del relativo sensore. In particolare, PUCK contiene informazioni standard sullo strumento, una sua descrizione (metadata), il codice del driver ed altre informazioni utili ad un sistema esterno. Quando un dispositivo "PUCK-Enabled" viene collegato ad un computer host questo può reperire informazioni sull'identità del sensore direttamente dal formato PUCK e predisporsi quindi al funzionamento selezionando il driver adatto al fine di interpretare ed eseguire il parsing dei dati da esso provenienti.

La seguente tabella descrive in modo schematico i vantaggi e gli inconvenienti relativi all'utilizzo del protocollo PUCK.

Vantaggi	Inconvenienti
Fornisce una descrizione del dispositivo	Deve essere implementato dai costruttori dei sensori "PUCK-Enabled"
Consente il download dei driver per la comunicazione	Non implementabile su dispositivi preesistenti
Permette di implementare sistemi «Plug & Work»	Non favorisce il riutilizzo di codice già sviluppato

Tabella 1. Vantaggi ed inconvenienti del protocollo PUCK.

Di fatto lo standard PUCK deve essere adottato dal costruttore del sensore il quale deve prevedere un'apposita memoria (EEPROM) all'interno del dispositivo stesso nella quale scrivere le informazioni del relativo strumento. Inoltre, tale protocollo non è implementabile sulla sensoristica precedentemente immessa sul mercato, sulla quale quindi non è possibile intervenire, rendendo inutile l'implementazione di un driver PUCK per i propri sistemi di lettura. Dal punto di vista del protocollo di trasmissione, inoltre, PUCK non sostituisce il protocollo di comunicazione originale, per il quale sarà necessario in ogni caso implementare un driver specifico.

Una più generale proposta di standardizzazione è stata effettuata tramite lo standard IEEE 1451 con l'introduzione della descrizione virtuale di un sensore e la definizione del concetto di *Transducer Electronic Data Sheets* (TEDS) [IEEE, 2004]. TEDS è uno standard nato per specificare informazioni su un sensore, quali quelle relative alla sua identificazione, calibrazione, correzione dei dati e casa costruttrice. Tali informazioni sono memorizzate all'interno dello strumento stesso in uno spazio di memoria noto; questo consente a sistemi esterni di prelevare i dati relativi al sensore e quindi predisporre un'interfaccia adatta alla sua integrazione nel sistema stesso.

Uno degli elementi chiave dello standard IEEE 1451 è prevedere la definizione di un TEDS per ogni sensore; il TEDS può essere contenuto in una memoria che verrà letta dal sistema esterno. Le possibili implementazioni sono due: la prima prevede che il TEDS risieda in una memoria interna al dispositivo (tipicamente una EEPROM) e che tale memoria sia accessibile al sistema a cui il sensore deve essere collegato; la seconda prevede che un TEDS "virtuale" possa esistere sotto forma di file disponibile al sistema di osservazione. Tale possibilità è molto più semplice da utilizzare in quanto prevede la creazione di TEDS anche per quei sensori/sistemi di acquisizione ai i quali non è possibile aggiungere una EEPROM interna.

L'implementazione dello standard IEEE 1451 per gli osservatori sottomarini è stata proposta con il progetto *Smart Sensor Interface* [Del Río, Auffret, Mihai Toma, & Shariat, 2010]. Tale progetto riguarda lo sviluppo di un modulo hardware nato per poter utilizzare tutti i sensori marini attualmente disponibili sul mercato. Il modulo viene collocato tra il sistema di osservazione ed il sensore da integrare e converte i dati provenienti da quest'ultimo in una Low Power Ethernet. Inoltre, questo modulo dispone anche funzionalità proprie come TimeStamp, ADC, GPIO ed eventuale memoria per consentire il datalogging.

Il modulo *Smart Sensor Interface* adotta, inoltre, un linguaggio software per la descrizione del sensore a cui verrà collegato che gli permette di adattarsi a diversi sistemi.

La seguente tabella descrive in modo schematico i vantaggi e gli inconvenienti relativi all'utilizzo del modulo *Smart Sensor Interface*.

Vantaggi	Inconvenienti
Sfrutta una descrizione del dispositivo/sensore	Rivolto espressamente a sistemi Ethernet
Modulo HW già sviluppato	il dispositivo deve essere riprogrammato
Aggiunge funzionalità al sistema collegato	Necessita di un modulo hardware aggiuntivo per ogni sensore utilizzato

Tabella 2. Vantaggi ed inconvenienti dello standard TEDS.

È da evidenziare che l'integrazione di eventuali nuovi sensori all'interno di un sistema di osservazione che adotta la soluzione fornita da *Smart Sensor Interface* prevede comunque la riprogrammazione di tale modulo, non essendo previsto un adattamento dinamico a nuovi dispositivi connessi. Inoltre, lo *Smart Sensor Interface* è pensato specificamente per realizzare un'interfaccia Ethernet standard per un unico sensore, diventa quindi necessario collegare un modulo hardware per ogni dispositivo che si intende integrare nel sistema di osservazione.

L'idea progettuale proposta nel presente lavoro è basata sull'astrazione del concetto di "sensore" concentrandosi sulla sua descrizione ad alto livello invece che sulla scrittura/riscrittura di firmware; in particolare, l'astrazione è focalizzata sulla parte di comunicazione seriale con il sensore di misura. In questo modo l'hardware di interfaccia dovrà essere unicamente in grado di "comprendere" la rappresentazione dei sensori ad esso collegati, ed avere risorse sufficienti - in termini di numero di porte seriali e velocità della CPU - per farli funzionare correttamente. Con tale approccio operativo, sia l'hardware di interfaccia che i dispositivi ad esso collegati diventano assolutamente intercambiabili e potenzialmente di tipo *PnP* rendendo possibile sia l'aggiunta di nuovi sensori al sistema di controllo, sia lo scambio delle porte seriali di comunicazione (rilocazione dei dispositivi), come anche la modifica delle operazioni da compiere o addirittura l'intercomunicazione tra gli stessi. Tutto questo senza dover ogni volta riscrivere il firmware ogni volta, ma "semplicemente" aggiornando la descrizione del sistema dall'esterno. In questo modo il sensore non avrà più un suo driver dedicato ma verrà rappresentato da un interprete mediante un metalinguaggio. L'interprete sarà quindi in grado di adattarsi dinamicamente alle comunicazioni necessarie al funzionamento dei vari sensori utilizzati. Esso potrà essere implementato in linguaggi di programmazione differenti e su varie tipologie di piattaforme hardware, mentre l'astrazione del protocollo può essere scritta in un formato "portabile" (tipo XML, *eXtensible Markup Language*) in modo da poter essere riutilizzata su sistemi differenti. Con tale "filosofia" implementativa, un "interprete" è in grado di comunicare con un qualunque sensore per il quale esista una descrizione correttamente espressa. Una volta che la descrizione è creata (magari proprio dal costruttore del sensore) può essere quindi utilizzata su piattaforme differenti riducendo notevolmente il lavoro necessario per la sua integrazione in un sistema di misura multi-parametrico.

Di seguito viene descritto il principio di funzionamento del metalinguaggio, una sua realizzazione nel linguaggio XML, il funzionamento dell'interprete del metalinguaggio ed, infine, un'applicazione per un sensore commerciale di largo impiego.

1. Il metalinguaggio

Il metalinguaggio denominato GAMbUSIA (*a GenerAl-purpose Metalanguage for instrUments with Serial InterfAce*) propone le regole di sintassi e la struttura per ottenere una descrizione ad alto livello delle comunicazioni necessarie alla corretta integrazione e funzionamento dei sensori all'interno del sistema di acquisizione, oltre che la descrizione delle informazioni fondamentali per l'identificazione del sensore stesso (marca, modello, versione firmware integrata, ecc.).

Per ogni sensore da integrare nel sistema di misura dovrà essere redatta, secondo lo schema proposto dal metalinguaggio, una descrizione generale contenente le informazioni sul sensore ed eventuali parametri utili per la comunicazione con esso. La descrizione generale dovrà includere, in apposite sezioni, la descrizione delle comunicazioni necessarie alla configurazione del sensore e quelle necessarie all'acquisizione dei dati da esso rilevati.

La singola comunicazione descritta mediante il metalinguaggio prende il nome di "Comando" e rappresenta i bytes da inviare al sensore e l'eventuale risposta attesa da quest'ultimo. Le sequenze di più comandi saranno racchiuse in una sezione denominata di SETUP (comandi necessari alla configurazione del sensore), ed in una sezione denominata TASK (comandi necessari all'acquisizione dei dati). Infatti, le attività eseguite da un sistema di acquisizione per la gestione di sensori su porta seriale possono essere schematizzate in due fasi: una di configurazione di tutta la sensoristica da acquisire (SETUP), ed una di colloquio con essi (TASK) per il recupero dei dati. Quest'ultima può avvenire sia con trasmissione continua dei dati che con invio dei dati su richiesta.

Per l'implementazione delle descrizioni è stato scelto il linguaggio XML. Tale scelta è dovuta al fatto che l'XML, essendo esso stesso un metalinguaggio, è molto utilizzato per la descrizione delle informazioni, è un linguaggio aperto e leggibile in chiaro, ed è di rapido utilizzo in ambito informatico (esistono diverse librerie per l'interpretazione di codice XML).

L'utilizzo del formato XML per la descrizione del protocollo di comunicazione dei sensori rende il metalinguaggio GAMbUSIA compatibile con altri standard già esistenti. Infatti la notazione XML è già utilizzata in altri standard come *SensorML* [Botts & Robin, 2014]. Una delle caratteristiche fondamentali di questi standard è quella di poter estendere liberamente le informazioni descritte a patto di rispettare la sintassi XML.

2. Descrizione di un sensore

Parametri informativi

Per identificare correttamente le descrizioni dei singoli sensori, alcune informazioni sono inserite all'interno del file scritto nel metalinguaggio GAMbUSIA in formato XML e residente nella memoria dell'architettura hardware del sistema di acquisizione dati. Le informazioni da specificare sono:

- **Nome**: costituisce il nome (etichetta) del sensore;
- **Marca**: il nome del fabbricante;
- **Modello**: il particolare modello.

Parametri di comunicazione

Al fine di consentire un corretto funzionamento dello strumento è necessario specificare alcuni parametri fondamentali:

- **ID Porta**: identifica la porta seriale di comunicazione da utilizzare per l'architettura hardware;
- **BaudRate**: specifica la velocità di comunicazione dati (in bps) tra sensore e sistema di acquisizione;
- **Tipo Porta**: specifica lo standard seriale di comunicazione da utilizzare (RS-232, RS-422, RS-485, ecc).

Nella successiva figura viene mostrato un esempio di descrizione in formato GAMbUSIA di un dispositivo generico.

```
<gambusia>
	<name>Sensor Name</name>
	<manufacturer>Sensor Manufacturer</manufacturer>
	<model>1.3.4</model>
	<port>
		<num>1</num> /* ex. 1 => COM1 */
		<baudrate>57600</baudrate>
		<type>RS232</type>
	</port>
	<setup_section>
		<setup_cmd_list>
			<command name="setup_cmd1"> ... </command>
			<command name="setup_cmdN"> ... </command>
		</setup_cmd_list>
	</setup_section>
	<task_section>
		<end_condition> ... </end_condition>
		<task_cmd_list>
			<command name="read_cmd"> ... </command>
		</task_cmd_list>
	</task_section>
</gambusia>
```

Figura 1. Esempio di descrizione.

3. I comandi

I comandi del metalinguaggio rappresentano le operazioni basilari di comunicazione che il sistema di acquisizione invia al sensore. A seconda del comando inviato, il sensore esegue una determinata funzione oppure semplicemente imposta un parametro al suo interno e quindi, a seconda del dispositivo, risponde con il valore richiesto o con una conferma (*acknowledge*). Quindi, è necessario, tramite metalinguaggio, fornire le informazioni relative all'implementazione dei comandi necessari al corretto funzionamento del dispositivo di misura utilizzato.

Struttura di un comando

Per consentire la corretta comprensione delle sequenze di comunicazione bisogna descrivere i comandi includendo le informazioni previste dal metalinguaggio GAMbUSIA.

Ogni comando è caratterizzato da un "nome" che ne riassume la funzionalità (es: READ_PRESSURE, SETUP_CONFIG1).

All'interno della descrizione devono essere specificati i dati comando da inviare, il formato dei dati della risposta attesa dal sensore, come formattare la risposta e, al termine, dove salvare od inviare il risultato. Queste quattro sezioni rappresentano la struttura fondamentale del comando:

- **CMD_DATA**: rappresenta la sequenza di bytes/caratteri da inviare al sensore sulla porta seriale prescelta;

- **RESPONSE**: descrive all'interprete in che forma il sensore risponderà e quindi dove, nella sequenza di dati ricevuti, deve reperire le informazioni di interesse;

- **OUTPUT**: consente di formattare l'output del comando (il valore ottenuto da RESPONSE);

- **DESTINATION**: specifica all'interprete dove inviare la sequenza dati generata nella sezione OUTPUT (es. console, memoria, altra porta com, ecc).

La Classe "data_pack"

La classe **data_pack** (Data Package) serve ad incapsulare e descrivere i dati che effettivamente vengono scambiati tra il sensore ed il sistema di acquisizione. La classe contiene al suo interno l'elemento "value" che consiste in una sequenza di byte/caratteri (denominati <data>) che rappresentano i dati da inviare e ricevere dal sensore descritto tramite metalinguaggio GAMbUSIA.

La natura dei dati presenti all'interno dell'elemento *data_pack* è identificata dall'attributo "type" che ne descrive il tipo (A: alfanumerico, B: binario, H: esadecimale, TS: timestamp).

Ogni *data_pack* può avere (o meno) al suo interno un elemento di "header" ed uno di "footer". Il primo rappresenta la sequenza di byte/caratteri che precedono l'elemento "value" nella trasmissione (esempio: ID Sensore, Indirizzo di memoria di un registro, ecc); l'ultimo rappresenta la sequenza di byte/caratteri successivi a "value" nella trasmissione (esempio. carattere di Carriage Return, separatore campo, ecc).

"header" e "footer" sono essi stessi dei "data_pack"; è quindi possibile descriverli dettagliatamente con un proprio "value" ed eventualmente con degli specifici "header" e "footer" in maniera ricorsiva.

Nella figura successiva viene mostrato un esempio di un elemento "header" formattato secondo la classe "data_pack":

```
<header class="data_pack">
	<!-- macro TIMESTAMP -->
	<value type="TS" />
	<!-- field separator -->
	<footer class="data_pack">
		<value type="A">
			<data>,</data>
		</value>
	</footer>
</header>
```

Figura 2. Esempio di elemento Header in cui è definito un Footer.

4. L'interprete del metalinguaggio

Per consentire l'integrazione del metalinguaggio all'interno di un sistema di acquisizione dati multi-parametrico, è necessario implementare un software in grado di leggere le descrizioni GAMbUSIA dei sensori, interpretarle, ed infine porre in essere tutte le comunicazioni necessarie al loro funzionamento. Tale software prende il nome di "interprete GAMbUSIA" e costituirà, di fatto, un processo aggiuntivo eseguibile sull'unità di elaborazione integrata a bordo della piattaforma hardware di acquisizione dati scelta per il sistema di misura.

L'interprete software può essere implementato utilizzando un qualsiasi linguaggio di programmazione e può essere compilato per essere eseguito su differenti sistemi operativi.

Durante il suo ciclo di funzionamento, l'interprete GAMbUSIA eseguirà le operazioni di comunicazione descritte all'interno delle sezioni SETUP e TASK, utilizzando i parametri di connessione del sensore.

SETUP

Durante tale fase quindi, l'interprete del metalinguaggio apprende la sequenza di comandi da inviare al sensore di misura.

L'elenco dei comandi contenuti nella sezione di SETUP viene eseguito in sequenza, dal primo all'ultimo e nell'ordine in cui sono stati specificati nel file di descrizione. La sequenza di comandi viene eseguita una sola volta; quando viene raggiunta la fine dell'elenco, l'interprete passa alla fase di TASK.

NB: *In caso di errori durante la fase di setup l'interprete segnala all'utente l'accaduto e quindi interrompe il processo.*

TASK

L'elenco dei comandi contenuti nella sezione TASK viene eseguito in sequenza, dal primo all'ultimo e nell'ordine in cui sono stati specificati nel file di descrizione; quando viene raggiunta la fine dell'elenco, la sequenza viene ripetuta ciclicamente.

È ipotizzabile l'inserimento di una condizione di termine della sezione TASK che può avvenire, ad esempio, quando viene raggiunto un certo numero di acquisizioni, oppure ad un certo tempo od intervallo temporale pre-impostato.

5. Esempio di applicazione per un sensore geofisico commerciale

In questo paragrafo è riportato un esempio di applicazione del metalinguaggio GAMbUSIA per il sensore di pressione di alta precisione, del tipo Digiquartz Pressure Transducer (Mod. 31K-101) della Paroscientific Inc.® (http://www.paroscientific.com/Depthsensors.htm). Questa tipologia di sensori è ampiamente utilizzata nella comunità oceanografica e geofisica per misure di pressione di alta precisione per applicazioni di allerta tsunami, di misura del livello del mare o per applicazioni di geodesia su fondo marino tramite la misura di movimenti verticali del fondo marino legate ad attività vulcanica o conseguenti all'occorrenza di forti terremoti.

```
<gambusia>
	<name>Digiquartz pressure transducer</name>
	<manufacturer>Paroscientific</manufacturer>
	<model>31K-101</model>
	<port>
		<num>1</num> /* ex. 1 => COM1 */
		<baudrate>9600</baudrate>
		<type>RS232</type>
	</port>
	<setup_section>
		<setup_cmd_list>
			<command name="PAROS_WRITE_TI">
			 <cmd_data class="data_pack">
			   <value type="A">
			     <data>*</data>
			     <data>0</data>
			     <data>1</data>
			     <data>0</data>
			     <data>0</data>
			     <data>E</data>
			     <data>W</data>
			     <data>*</data>
			     <data>0</data>
			     <data>1</data>
			     <data>0</data>
			     <data>0</data>
			     <data>T</data>
			     <data>I</data>
			     <data>=</data>
			     <data>6</data>
			     <data>6</data>
			     <data>6</data>
			     <data>\r</data>
			     <data>\n</data>
			   </value>
			 </cmd_data>
			</command>
			<command name="PAROS_WRITE_PI">
			  <cmd_data class="data_pack">
			    <value type="A">
			      <data>*</data>
			      <data>0</data>
			      <data>1</data>

			      <data>0</data>
			      <data>0</data>
			      <data>E</data>
			      <data>W</data>
			      <data>*</data>
			      <data>0</data>
			      <data>1</data>
			      <data>0</data>
			      <data>0</data>
			      <data>P</data>
			      <data>I</data>
			      <data>=</data>
			      <data>6</data>
			      <data>6</data>
			      <data>6</data>
			      <data>\r</data>
			      <data>\n</data>
			    </value>
			  </cmd_data>
			</command>
		</setup_cmd_list>
	</setup_section>
	<task_section>
```

```
            <task_cmd_list>
                    <command cmd_type="normal" name="READ_PRESSURE">
                          <cmd_data class="data_pack">
                                 <value type="A">
                                       <data>*</data>
                                       <data>0</data>
                                       <data>1</data>
                                       <data>0</data>
                                       <data>0</data>
                                       <data>P</data>
                                       <data>3</data>
                                       <data>\r</data>
                                       <data>\n</data>
                                 </value>
                          </cmd_data>
                       <!-- RESPONSE DESCRIPTION AND FORMAT -->
                          <response class="data_pack">
                                 <value type="A"/>
                                 <header class="data_pack">
                                        <value type="A">
                                               <data>*</data>
                                               <data>0</data>
                                               <data>0</data>
                                               <data>0</data>
                                               <data>1</data>
                                        </value>
                                 </header>



                                 <footer class="data_pack">
                                        <value type="A">
                                              <data>\r</data>
                                              <data>\n</data>
                                        </value>
                                 </footer>
                              </response>
<!-- OUTPUT DESCRIPTION: HOW TO FORMAT RESPONSE.VALUE: [TIMESTAMP],[VALUE][new_line]    -->
                              <output class="data_pack">
                                 <header class="data_pack">
                                 <!-- macro TIMESTAMP -->
                                 <value type="TS" />
                                   <footer class="data_pack">
                                          <value type="A">
                                                 <data>,</data>
                                          </value>
                                   </footer>
                                 </header>
                                 <footer class="data_pack">
                                        <value type="A">
                                               <data>\n</data>
                                        </value>
                                 </footer>
                              </output>
                     <!-- output destination -->
                          <destination>
                                 <file>
                                        <path>parosc.sta</path>
                                 </file>
                          </destination>
                    </command>
                </task_cmd_list>
            </task_section>
        </gambusia>
```

Figura 3. Esempio applicativo per il sensore di pressione Paroscientific.

6. I vantaggi dell'utilizzo di GAMbUSIA

L'interprete del metalinguaggio GAMbUSIA è quindi in grado di adattarsi dinamicamente alle comunicazioni con la variegata tipologia di sensoristica usata e necessarie al loro funzionamento. Esso può essere implementato utilizzando linguaggi di programmazione differenti e per piattaforme hardware differenti, mentre l'astrazione del protocollo di comunicazione è scritta con il metalinguaggio GAMbUSIA, in formato XML, in modo da poter essere riutilizzata, senza problemi, da sistemi differenti.

Con questa implementazione, l'interprete GAMbUSIA è in grado di comunicare con un qualunque sensore per il quale esista una descrizione correttamente espressa. Una volta che la descrizione viene creata (magari proprio dal costruttore del sensore) può essere quindi utilizzata su piattaforme differenti riducendo notevolmente, in tal modo, il lavoro necessario per l'integrazione.

Mediante il riutilizzo del codice l'interprete viene scritto un'unica volta per la piattaforma hardware che si utilizza nel proprio sistema di acquisizione, e quindi all'interno del proprio software si possono utilizzare diverse istanze dello stesso codice per interagire con i vari sensori collegati alla piattaforma utilizzata.

Un ulteriore riduzione dei tempi di sviluppo si ottiene considerando il possibile riutilizzo delle descrizioni, le quali possono essere caricate in forma identica su piattaforme hardware anche molto differenti tra loro, in quanto sarà compito dell'interprete GAMbUSIA comprenderne il significato.

Ottimizzando il processo di integrazione di nuovi sensori all'interno del sistema di acquisizione si ottiene, così, un notevole miglioramento nella flessibilità del sistema stesso, in quanto la struttura non resterà vincolata ai dispositivi previsti in fase di progettazione ma potrà essere modificata nel tempo senza compromettere l'intero sistema iniziale previsto.

L'output dei singoli sensori può essere adattato secondo le future necessità semplicemente aggiornandone le descrizioni.

Precaricando nella memoria integrata nella piattaforma utilizzata le diverse descrizioni dei sensori si può ottenere, quindi, un sistema *plug & work* in cui il software di sistema è in grado di comprendere quali sono i dispositivi collegati fisicamente alle varie porte seriali senza dover riscrivere il software di controllo.

Conclusioni

Con l'obiettivo di migliorare la flessibilità dei sistemi di acquisizione dati ed i tempi di integrazione di nuovi sensori su sistemi già funzionanti è stato sviluppato un metalinguaggio, denominato GAMbUSIA, per la descrizione astratta dei sensori. Tale metalinguaggio è focalizzato sul protocollo di comunicazione del sensore al fine di permetterne una descrizione "semplice" da comprendere anche ad alto livello (ad esempio letta direttamente dall'utente).

Un interprete GAMbUSIA è un processo software che gira sulla piattaforma hardware utilizzata dal sistema di misura multi-parametrico, ed è in grado di comprendere la descrizione del sensore fatta utilizzando il relativo metalinguaggio. L'interazione con il particolare sensore consisterà nell'inviargli i comandi necessari per il suo setup, avviare le procedure di lettura dei dati, e tradurre i dati ricevuti nel formato che l'utente deciderà di utilizzare.

Una volta creata una descrizione GAMbUSIA del dispositivo di misura, questa può essere utilizzata su diversi sistemi e diverse piattaforme software (ad esempio: Linux Embedded, Arduino, RTOS, …) nonché distribuita per un utilizzo comunitario.

Gli sviluppatori non dovranno quindi scrivere drivers specifici per ognuno dei sensori collegati al proprio sistema di acquisizione ma semplicemente basterà implementare, nel proprio software, un interprete del metalinguaggio o utilizzarne uno già esistente.

Infine, è possibile la creazione di un repository comunitario dal quale scaricare liberamente le descrizioni già esistenti dei sensori così come scaricare il codice per l'implementazione degli interpreti per le diverse piattaforme software.

Sito WEB

Relativamente al progetto GAMbUSIA è stato attivato un sito web raggiungibile all'URL http://www.gambusia.it

Il sito ha lo scopo di fornire maggiori informazioni relative al metalinguaggio GAMbUSIA e promuoverne l'utilizzo. Esso ospiterà un repository delle descrizioni di dispositivi e sensori pronte per essere utilizzate in sistemi di monitoraggio già dotate di interprete oppure come template per crearne di nuove.

Tramite il sito verrà anche fornito un tool per la generazione guidata delle descrizioni realizzate con il metalinguaggio GAMbUSIA.

Bibliografia

Botts M. & Robin A., (2014). *OGC® SensorML:Model and XML Encoding Standard*.

Del Río J., Auffret Y., Mihai Toma D. & Shariat S., (2010). *Smart Sensor interface for sea bottom observatories*.

IEEE, (2004). *Digital Communication and Transducer Electronic Data Sheet (TEDS) Formats*.

O'Reilly T. & Reed C., (2012). *OGC® PUCK Protocol Standard.*

ISSN 1590-2595

Quaderni di Geofisica

http://istituto.ingv.it/l-ingv/produzione-scientifica/quaderni-di-geofisica/

I Quaderni di Geofisica coprono tutti i campi disciplinari sviluppati all'interno dell'INGV, dando particolare risalto alla pubblicazione di dati, misure, osservazioni e loro elaborazioni anche preliminari, che per tipologia e dettaglio necessitano di una rapida diffusione nella comunità scientifica nazionale ed internazionale. La pubblicazione on-line fornisce accesso immediato a tutti i possibili utenti. L'Editorial Board multidisciplinare garantisce i requisiti di qualità per la pubblicazione dei contributi.

ISSN 2039-7941

rapporti tecnici INGV

http://istituto.ingv.it/l-ingv/produzione-scientifica/rapporti-tecnici-ingv/

I Rapporti Tecnici INGV pubblicano contributi, sia in italiano che in inglese, di tipo tecnologico e di rilevante interesse tecnico-scientifico per gli ambiti disciplinari propri dell'INGV. La collana Rapporti Tecnici INGV pubblica esclusivamente on-line per garantire agli autori rapidità di diffusione e agli utenti accesso immediato ai dati pubblicati. L'Editorial Board multidisciplinare garantisce i requisiti di qualità per la pubblicazione dei contributi.

ISSN 2039-6651

Miscellanea INGV

http://istituto.ingv.it/l-ingv/produzione-scientifica/miscellanea-ingv/

La collana Miscellanea INGV nasce con l'intento di favorire la pubblicazione di contributi scientifici riguardanti le attività svolte dall'INGV (sismologia, vulcanologia, geologia, geomagnetismo, geochimica, aeronomia e innovazione tecnologica). In particolare, la collana Miscellanea INGV raccoglie reports di progetti scientifici, proceedings di convegni, manuali, monografie di rilevante interesse, raccolte di articoli ecc..

Coordinamento editoriale e impaginazione
Centro Editoriale Nazionale | INGV

Progetto grafico e redazionale
Daniela Riposati | Laboratorio Grafica e Immagini | INGV

Via di Vigna Murata, 605
00143 Roma
Tel. +39 06518601 Fax +39 065041181

http://www.ingv.it

Istituto Nazionale di Geofisica e Vulcanologia