Social Networking of Instrumentation – a Case Study in Telematics

Dan ROBU[1], Florin SANDU[1], Dorin PETREUS[2], Adrian NEDELCU[1], Alexandru BALICA[1]
[1]*Transilvania University of Brasov, 500036, Romania*
[2]*Technical University of Cluj-Napoca, 400020, Romania*
florin.sandu@unitbv.ro

***Abstract*—The research work contributes to the design and implementation of the communication part for integrating remote instruments and drives via social networks (SN) into instrumentation communities. It is used the virtual instrumentation (VI) to manage objects that tweet on popular SN platforms applying the concept of the Internet of Things (IoT). Local and remote resource aggregation is based on National Instruments (NI) data acquisition and distribution hardware in a NI software environment. NI LabVIEW-for-Twitter solutions (starting with simple authentication) are extended, integrated with various third party services and validated in a complete remote monitoring proof-of-concept workbench with a closed loop for alarming-compensation. Solutions are extendible to Machine-to-Machine communication (M2M) IoT scenarios for telematics, monitoring or control, and can be connected to intelligent systems based on powerful servers for cloud computing.**

Index Terms—application programming interfaces, computerized instrumentation, data acquisition, social network services, Twitter.

I. Introduction

IoT is a modern approach for the management of presence and availability, translating the basic data available in the SN model into the field of mixed hardware-software-netware objects (personal, domotics, enterprise etc.) [1-3].

Direct M2M communication should make use of solutions already consolidated on the Internet, in order to interconnect an estimated number of 16 billion objects until 2020.

IoT should perform discovery, identification, localization, monitoring and remote management of the M2M objects, supporting a constant Real-Time (RT) information stream between them.

IoT services can be built on-top of the information from sensors by algorithms for RT interpretation and decision [4].

Such solutions have real perspectives to integrate protocols like MQTT (Message Queue Telemetry Transport) [5] and XMPP (Extensible Messaging and Presence Protocol) [6]. These solutions could rely on Twitter but also on open source real time collaboration server solutions (e.g. OpenFire) for friendly user interface administration and notifications.

Many domains can benefit from these concepts, like distributed intelligent instrumentation for test & measurement (T&M) [7-8] using industrial communications – as well as domotic solutions (enterprise or residential) [9-10].

The telecom operators are interested, mainly, in IoT solutions for electrical Smart Grids and utility networks related to intelligent control and optimization [11-13].

Our *research goal* was to develop generic infrastructure-independent communications that would enable the creation of instrumentation communities.

Our *approach* is the use of popular SN platforms in order to support M2M communications.

Our *task* was the adaption of instrumentation control for social networking and the *objectives* of our work were:

- Leveraging the existing IP infrastructure;
- Reducing platform- and language- dependencies;
- Providing a generic abstraction layer available to different types of applications;
- Enabling real-time communication;
- Linking networked nodes based on social rules;
- Operating in a secure manner, providing features such as authentication, authorization and encryption.

Using the social media paradigm, our *research methodology* was structured as follows:

- *Survey & identification* of the appropriate communication protocols and open platforms, in order to support the interactions inside the distributed instrumentation communities on SN principles;
- *Design, develop, implement, integrate* and *validate* proof-of-concept solutions for telematic case-studies.

II. Implementing Instrumentation Communities via Resource Aggregation

There were identified the following two possible directions for our research.

1) The XMPP approach: in order to aggregate networked instrumentation based on social rules, our research explored the above-mentioned XMPP [6], also known as Jabber (the name of the company that launched it) – an open protocol used for instant messaging. In order to support the aggregation with XMPP of various and numerous instrumentation devices, we have used Openfire, a RT collaboration (RTC) server solution. Openfire incorporates a graphical interface based on XML (eXtensible Markup Language) – the same backbone markup language (used in all our research) for interlayer communication. Openfire also provides a database system and alarm control, making it an effective and flexible monitoring solution.

In order to support the remote network configuration it was taken into consideration the synergy of XMPP with SNMP (Simple Network Management Protocol) via a trapping mechanism. SNMP is based on client-server communication – alarms are interpreted using MIB files (Management Information Base), representing virtual databases of status information related to the nodes under surveillance; the result of this interpretation is wrapped (encapsulated) in XMPP messages.

Digital Object Identifier 10.4316/AECE.2014.02024

In our context of distributed instrumental communities, the advantage of XMPP as an Open Source solution (under Apache License) is the unrestricted possibility to interact with numerous and various sources or destinations, via social networks. The XMPP experience (e.g. Spark/Smack clients – human/instrumental – of the Ignite open source initiative) is enhancing the social networking paradigm with added value transfers that are transcending the borders of simple infotainment support into the industrial applications.

2) The VISA approach: our second approach started from the origin of the well-known plug-and-play (PnP) concept in modern operating systems, VXI Instrument Software Architecture (VISA) with VXI representing VME (Versa-Module Eurocard) eXtension for Instrumentation. Recently, VISA is more and more associated with Virtual Instruments (VI) software architecture. This 2nd approach was chosen to pursue the next phases of our research, on the principle of implementing instrumentation communities relying on VISA aggregation of resources into T&M configurations.

As the most suitable candidate for the generic layer available to different types of applications we have chosen the Representational State Transfer ("RESTful") Application Programming Interfaces (API) for the service-orientation towards our objective to reduce platform- and language- dependencies according to the SaaS (Software as a Service) paradigm.

As the best social network of choice, we have identified Twitter [23], reliable intermediary for message broadcasting with an advanced set of API and good level of security.

As VISA is service-oriented (based on event-driven control), we have identified as one of the best VI software development environment the National Instruments' LabVIEW [14], together with one of the best hypervisors of distributed configurations, the NI Measurement and Automation Explorer (MAX).

By these two modern and powerful means offered by NI, remote and local resources can be integrated in social communities where they expose T&M capabilities – this is the *push* side of an offer/bid scheme. Aggregation in social partnership can also benefit from the other side of this scheme – the *pull* side, used for polling and probing available resources.

Workstations of the VISA configuration, besides their main role as web servers, can also be used as remote workbench servers (for resources grouping) with external Internet connection and local Intranet connections to instruments by Ethernet or directly by dedicated interfaces (e.g.GPIB), standard interfaces(e.g.USB) or proximity NFC. In our research, apart from NI-MAX and LabVIEW we have used a Data Acquisition (DAQ) system and accessed the Twitter services in order to integrate telematic configurations (for remote monitoring of meteorological stations or industrial weighing scales etc). According to our research methodology, in order to *validate* our findings we have integrated these simple proof-of-concept measurement solutions into transmission schemes based on Twitter and opened to third party tools for social media (e.g. SuperTweet). The bases of our ICT development were the NI VIs used to actuate and acquire (e.g. sensor data) and ensure proper LAN-WAN communications. The built-in NI Real-Time Engine meets also one of our specific objectives.

Publication of aggregated resources – that might be considered a tree-like manner representation of an instrumental *partnership* can be provided by NI MAX, having in this case the role of a *broker* (Fig.1).

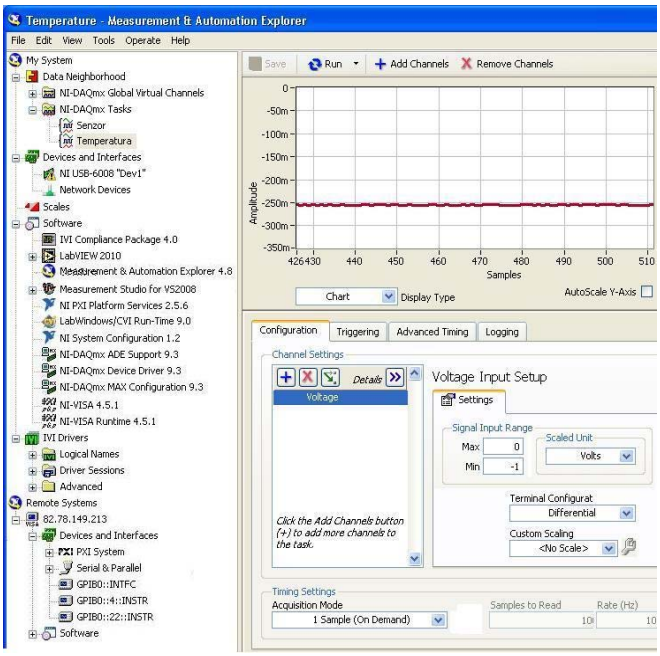

Figure 1. NI MAX hardware-software-netware community with details of the data acquisition system integration including association of logical in/out channels to the physical devices

For administration purpose, adding a new remote resource is possible in MAX (*Remote Systems* menu > *Create new* > *Remote VISA System* > configuration of properties in the displayed PnP window. IP port forwarding can be used to route Remote VISA data (coming through the dedicated port 3537) from the web-server to the workbench server. Security requirements can thus be fulfilled by configuring the access list for port 3537. Considering the first of our research objectives, this way any client (human or machine) of the system should only use the IP address of the web-server for remote use of instrumental resources that were "published" for open access.

Remote resource check uses the capability probing function available in MAX GUI (Graphical User Interface), by selecting the desired instrument (e.g. `GPIB::4::INSTR` – see Fig.1) and then selecting Open *VISA Test Panel*.

Write and Read commands are sent to remote instruments for instructions and for the display of measurement or status.

For M2M direct addressing of remote instruments, the control sequence should invoke *visa://ip_address_of_the_webserver/ instrument_ID*.

After opening a VISA session, remote instruments can be controlled using standard SCPI (Standard Commands for Programmable Instruments) or proprietary instrumental directives. LabVIEW modules *NI Datalogging & Supervisory Control* and *Shared Variables* can be used for efficient programming of distributed IoT applications, via NI-PSP (Publish & Subscribe Protocol).

III. Solutions for Applications Interaction

The overwhelming spread of ergonomic graphical user interfaces is underlining the importance of intuitive interaction design for the industrial applications.

Influenced by infotainment applications, the modern IoT proposed user friendly controls (even from "thin clients").

In the flow of operations of our application, different instrumentation-related tasks can be divided into subtasks which are assigned to the different members of the communities, according to their capabilities.

To fulfill our research goal with the chosen methodology, we worked on the synergy of instrumentation control with the Twitter API-s.

In order to meet our research objective related to security, one of the first tasks was to enable the Twitter encrypted authentication. Twitter interacts with client applications based on standard HTTP methods (corresponding to our research objective of leveraging the existing IP and to the research objective of minimizing dependencies – e.g. on dedicated industrial communication ports or protocols). Among the simplest HTTP methods, and the most used, including our implementations, are GET, POST and PUT.

Twitter allows recently only advanced OAuth (Open Authentication) that we will detail in paragraph VI.

M2M communication might yet accept – for an important range of non-critical use-cases – the simpler Basic Authentication (BA). There is the possibility to use third party services of intermediation for indirect access either via Internet, as the one mediated by the supertweet.net network or (partner of Twitter) that still allows Basic Authentication (BA). As one of the most important third party we have used the services of SuperTweet. Its methods (controls) can be invoked *remotely*, via API, being integrated in local programs (e.g. the main VIs, in our case-study). Another approach is the installation of a *local* SuperTweet Proxy that would translate BA data into OAuth requests sent remotely not via a SuperTweet server but *directly* to Twitter.

For our first BA approach, the APIs available in the SuperTweet network enable the running of simple "tweeting" applications that can be also implemented in smaller embedded systems like low-complexity appliances, e.g. based on the Lantronix XPort Embedded Web-Server [18], bringing minimal telecom functionality to intelligent sensors and transducers.

The generic specification of SuperTweet *functions* of the *server* api.supertweet.net is <*service classes*>/<*services*>**.** <*format*> . We used, mostly, /1/statuses/update.<*format*> and /1/statuses/home_timeline.<*format*>. The <*format*> field is replaced by the specification of a file in XML format or in JSON (JavaScript Object Notation – recently approached also by Google with various GSON converters [19]). This file records readings from GET commands and brings parameters for the POST command (main commands accessed via afore mentioned API-s).

There are also <*service classes*> like *account, users, blocks, friendships, favorites*, *direct_messages, geo* and *saved_searches*.

Invoking of the respective APIs can be embedded in any program. For example, using *cURL,* a robust software package (that has to be pre-installed *locally*) dedicated to simple URL commands for transferring data:

```
curl -u user:password http://api.supertweet.net/1/statuses/update.xml
```

or

```
curl -u user:password http://api.supertweet.net/1/statuses/home_timeline.xml
```

– for the "tweets" that update the status with information taken from the *update.xml* file (as shown below), respectively for receiving a series of accumulated tweets. The history recorded in *home_timeline* will be written in the *home_timeline.xml* file (also detailed below).

In the main VIs *Update Status* and *Read Status* (that are driving Twitter social platform as measurement data broker) – presented in the next paragraphs – we have used the sub-VIs *HTTP POST* and *HTTP GET* having as string parameters:

```
IMVA_TWT:twt_avmi http://api.supertweet.net/1/statuses/update.xml
```

and, respectively,

```
IMVA_TWT:twt_avmi http://api.supertweet.net/1/statuses/home_timeline.xml
```

For the SuperTweet account, the username is the same as the Twitter account, but the SuperTweet password, e.g. `twt_avmi`, is different from the Twitter password, e.g. TWT_AVMI. The Twitter account password is introduced from the beginning, as the first phase of this indirect authentication is the login in SuperTweet where exact Twitter username and password are required. The proper SuperTweet password is needed only for opening the account and is always used by the external programs accessing SuperTweet APIs.

These HTTP calls are the most relevant for our implementation – as in the previous example where, `http://api.supertweet.net/1/statuses/update.xml` is used to transmit the status of the authenticated user, requesting a POST to the corresponding URL address, in order to update the status value.

These values are encoded in Base64 (see the presentation of the corresponding sub-VI in the next sections). This call is similar to `http://twitter.com/statuses/update.xml`. Type of the answer is `null`.

Another command of the previous example is `http://api.supertweet.net/1/statuses/home_timeline.xml` (see the previous example) that will provide chronologically the "tweets" sequence. This command is similar to `http://twitter.com/ statuses/user_timeline.xml`. These URLs do not need extra parameters and the type of answer is `status`.

IV. The Proof-of-Concept

For the first demonstrator of the proof-of-concept implementation, we have used the NI low-cost multifunction DAQ System NI USB-6008 [15] – that can be seen in Fig.2. It is spectacular that such a mini-system can tweet and close a complete telematic loop (computing and producing also actuator feedbacks). Its integration into NI-MAX is also flexible and forthcoming. As already visible in our particular NI MAX community in Fig.1, in the group Devices and Interfaces, it gets the (logical) identity ***Dev1***; this is a username (and an avatar associated with the role overtaken in the social network). *Dev1* capabilities are mediated by the NI-DAQmx set of drivers and middleware aggregated by NI MAX in the Software group (see Fig.1, center). This means the DAQ system can be probed in a push-pull configuration as advertised by the network.

We have created the *NI-DAQmx Tasks* group of tasks containing *Senzor* (sensor) and *Temperatura* (temperature). They are put by NI-MAX in the *Data Neighbourhood*

group, and are configured in relationship with the *Voltage* task, for the acquisition inputs of Dev1. *Voltages* can be checked prior to their use as logical channels in the higher level VI by probing DAQmx (the control is visible in Fig.1 – right). These VIs are parts of our software development for the case-study in telematics. They are implemented in LabVIEW that is also integrated in the NI MAX Software group – we have thus consolidated the community that would function with social networking rules, according to the specific SN-objective of our research.

Due to the *mediation* of NI MAX NI DAQmx as social network *facilitator*, the instrumentation service can be built by LabVIEW almost transparently – without detailed and very specific controls for DAQ hardware – as in former versions of LabVIEW [16]. This means that NI MAX & NI LabVIEW meet the functionality of a real SCE (Service Creation Environment) according to the IN paradigm – "Intelligent Networks" with liberalized access to service creation and deployment.

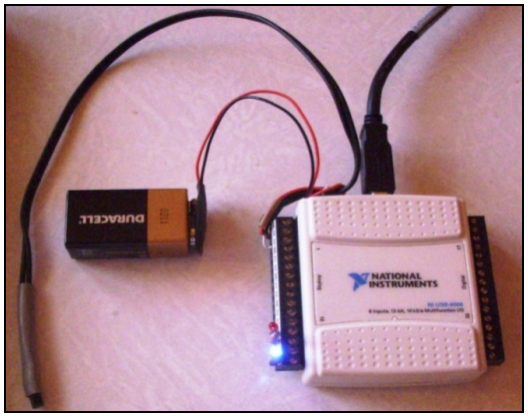

Figure 2. Simple and cost-effective proof-of-concept configuration: NI USB-6008 with red/blue LEDs (the second one lights, closing the Twitter loop by notifying an inferior temperature); LM35 sensor; 9V battery

For our case-study we benefited from the main system characteristics of the NI 6008 USB DAQ, using one of the 8 analog inputs (12b, 10kS/s) since the resolution is enough for the intrinsic noise of the temperature sensor LM35Z and the acquisition speed is enough given our thermal time constants. There are also available 2 analog outputs with lower speed, and 12 digital I/O ports.

The temperature sensor LM35DZ belongs to the National Semiconductor LM35 series of precision integrated-circuit sensors [17]. The output voltage is linearly proportional to the Celsius temperature – 0 to 1V output for 0 to 100 °C (operating temperatures from -55° to +150°C). The sensor does not require any external calibration or trimming to provide typical accuracies at room temperature or over a full temperature range making it ideal for accurate and manageable testing device. In addition, LM35 has low output impedance ideal for direct connection to DAQ systems like ours. It can be used with single power supplies, or with plus and minus supplies. We have also tested the use of NI USB-6008 own output for the single power supply of the sensors but, for greater accuracy, we chose separate ground for the measurement circuit of sensor outputs and DAQ inputs, using an extreme low noise dedicated supply – a 9V alkaline battery. This is possible since LM35 draws only 60 μA from its supply.

An important consequent advantage is the very low self-heating (i.e. less than 0.1°C in still air) of LM35.

V. Data Collection and Tweeting via Virtual Instrumentation

The first VI developed for DAQ performs broadcasting temperature measurements via Twitter, using LabVIEW and is called *SuperTweet Update Status.vi*, presented in Fig.3 – the main Panel, (a) & (b), and the Diagram, (c).

It represents the *LabVIEW to Twitter* paradigm, with data collected and sent.

For the purpose of onscreen monitoring and control, the panel display contains the measured temperature, in analog and digital format, and the HTTP POST messages sent, that would produce the expected "tweets".

One can note some more important XML fields, e.g. *<text>Statia Meteo nr.1 Temperatura 26.1 C @ 6/7/2011 6:02 AM</text>* representing the actual "tweeted" text. Compared with the classical Twitter service, there are also more details between *<source>* and *</source>* markups.

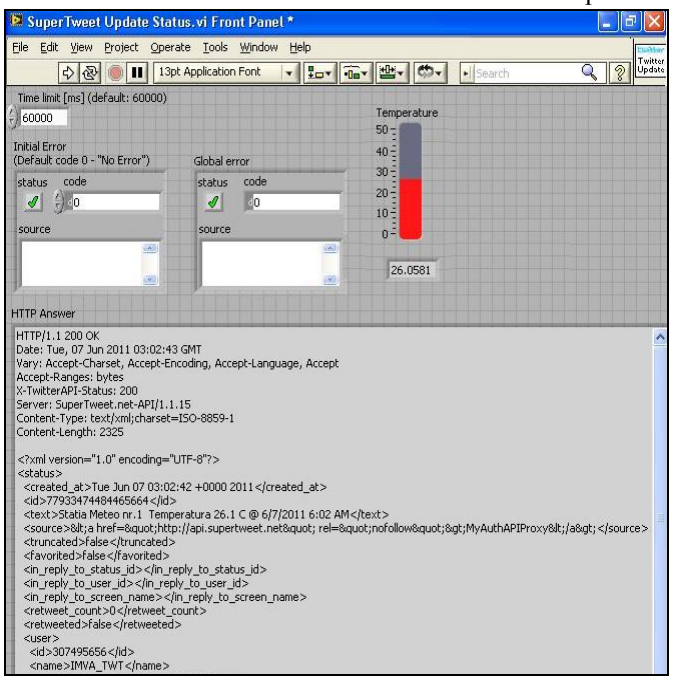

Figure 3 (a). SuperTweet Update Status.vi panel

For the `api.supertweet.net/1.1/statuses/update.json` frequently used, the HTTP Answer is like in Fig.3 (b). The "tweets" in fig.3(b) are from another case-study collecting and transmitting the measurement of a weighing scale sensor. The data, in this case, represents the quantity of 7.0 *grame* (grams) announced by *Cantar* (weighing-scale) with a local time-stamp 6/30/2013 9:33AM (GMT 6:33AM).

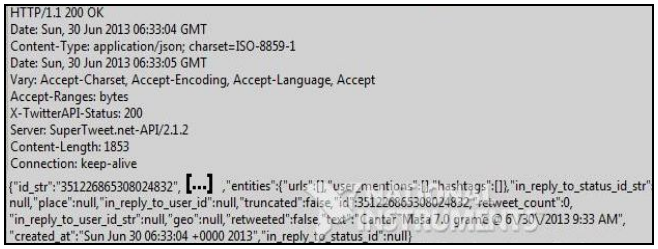

Figure 3 (b). HTTP Answer - fragments separated by **[...]** - in the case of SuperTweet Update Status.vi based on JSON

In the VI's diagram, Fig.3 (c), the following sequence is performed: acquisition of temperature signal from the LM35

sensor, its multiplication with the ºC/V factor and numeric to alphanumeric conversion of the result for temperature value in string format (from the matrix output, only the first string value is extracted, the one corresponding to measured status, temperature in this case).

The resulting string is prepared for transmission via a series of *parsing* and manipulation sequences. At first it is concatenated with a preamble, (i.e. the ID of a Meteorological Station in our case-study) and added a trailer with the time-stamp representing the local time (format including: date & hour) at the geographical location of the "tweet". Then the global assembly of the HTTP POST message is done (also by string-concatenation). Its final transmission is done in a TCP session via the appropriate HTTP port 80. This includes sending the specific HTTP fields associated with method identification (i.e. the specific API in use `api.supertweet.net/1/statuses/update.xml`). Finally, the error codes are managed via a pipeline, in order to allow their storage for later checking.

The main sub-VIs are:

– *DAQ Assistant.vi* – the specific set of LabVIEW functions & controls (a middleware that is *transparently exposing* only *logical* acquisition & distribution channels, whatever physical channels would be available via the DAQmx drivers – in our case, those of NI USB DAQ 6008);

– *Dynamic data converter.vi* – from the DAQ Assistant, in our case outputs of the temperature sensor LM35DZ – into numbers (multiplied afterwards with the ºC/V factor);

– *Trim Whitespace.vi* – eliminates of any blank characters (e.g. space, tab, line break), based on the `whitespace.ctl`. This sub-VI and the following are parts of the specific LabVIEW package that we have built-upon [21].

– Since Base64 is the encoding specific to Twitter, alphanumerical conversion into Base64 must be done, by concatenating the ASCII bytes of the input characters (i.e. the 17 characters in `IMVA_TWT:twt_avmi`), resulting a global string (in our case 17 x 8 = 136 bits), re-divided in characters of 6 bits each – in our case, the 136/6 (≈) 23 characters of a number in Base64 (=2^6) `SU1WQV9UV1Q6dHd0X2F2bWk`.

As in Fig.4, the "tweets" that reached Twitter servers from the instrumentation device can be displayed sequentially.

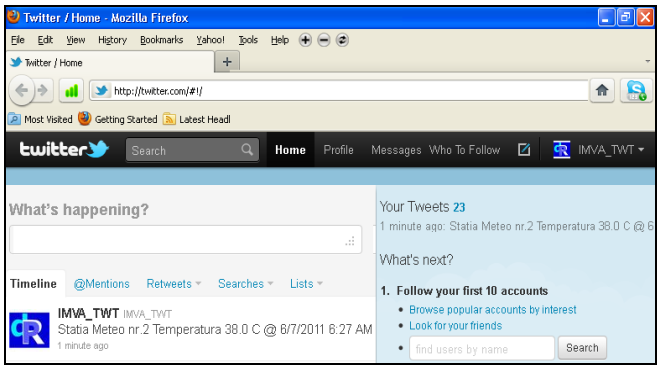

Figure 4. Display of tweets (sent by the VI) in Twitter timeline

The Twitter messages are completely customized, including the avatar image for the emulated sending account IMVA_TWT. The logo file *DPR_normal.JPG* is specified in the sending and formatting part of the main-VI, the HTTP POST message definition sections, visible in Fig.5 (a).

As shown above, the first VI was an extension of the NI libraries [21], with our solutions to the problem of authentication and of the integration with a DAQ instrumental configuration. Thus, the second VI developed represents a complete original solution.

This complete solution provides *Twitter to LabVIEW* side of a complete bilateral and direct M2M communication suitable for IoT via Twitter, without human mediation.

The *Read Status.vi* Panel and Diagram are presented in Fig.5 (a) and (b).

The main version containing fixed authentication parameters and fixed preamble can be programmed for periodical "tweets", as described before.

The VI allows not only direct M2M communication for measurement, status and commands for a broad range of control scenarios in IoT, but can also close the acquisition-drives loop in automation, engaging actuators [10].

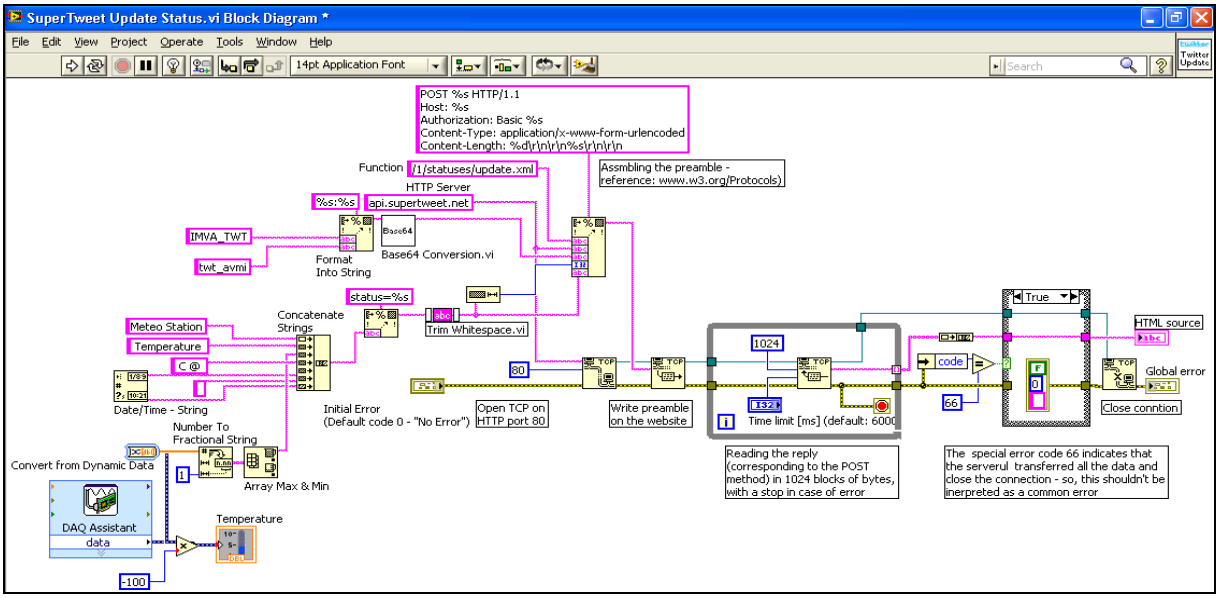

Figure 3 (c). SuperTweet Update Status.vi diagram

In our case-study, we have programmed fixed alarm thresholds for temperature, producing binary on/off controls of potential heating or cooling systems. These controls are emulated by red and blue LED-s that don't switch on in the programmed range of temperatures, ensuring also a hysteresis – a margin that should prevent switching oscillations, but switch on in alarm conditions (e.g. received by a remote central heating system) – too warm or too cold. The main versions of this VI easily allow modification of these thresholds via dedicated numeric controls available in the panel, with direct input or increment/decrement, as can be seen in Fig.5 (a), making it viable for domotics. A numerical display would indicate the remotely transmitted ("tweeted") actual value representing the actual Twitter status – in our case-study a temperature broadcasted by a Weather Station – without needing the periodical access of a human operator to read thermometers or other equipment like barometers or hygrometers. In other versions, thresholds can be fixed (not accessible in the VI panel).

The *Read Status.vi* includes authentication controls and parameters like in the *SuperTweet Update Status.VI*. It is invoked by the other specific API, `http://api.supertweet.net/1/statuses/home_timeline.xml` mentioned in the previous paragraph.

The main sub-VIs used (functions and controls from LabVIEW libraries) are:

– *HTTP Request (GET).VI,* that receives the HTTP message of the "tweet". It has a structure very similar with the above-detailed HTTP POST; – *Twitter Extract XML from HTTP Response.vi* and *Twitter Parse Statuses from XML.vi* allow the string formatting and particular string processing, besides the control display on VIs panel. The parser is searching in the XML file the preamble *<?xml* to check XML validity, then the > character that is typical ending XML labels, extracting line by line, in an internal shift register, the statuses, respectively the code between the *<status>* and *</status>* labels that corresponds to a tweet. When the length of such a sequence becomes 0, the reading of statuses came to end, the repetitive run is stopped.

– Formatting and listing (in Reading Tables on the panel) of sub-strings extracted from tweeted status info, grouping alphanumeric variables *user_name*, *user_screen_name*, *text*, *created_at* with separator characters.

It can be noticed again that time-stamping of the "tweets" is using GMT. The variables are separated (filtered) from the extracted XML file as in Fig.5(a).

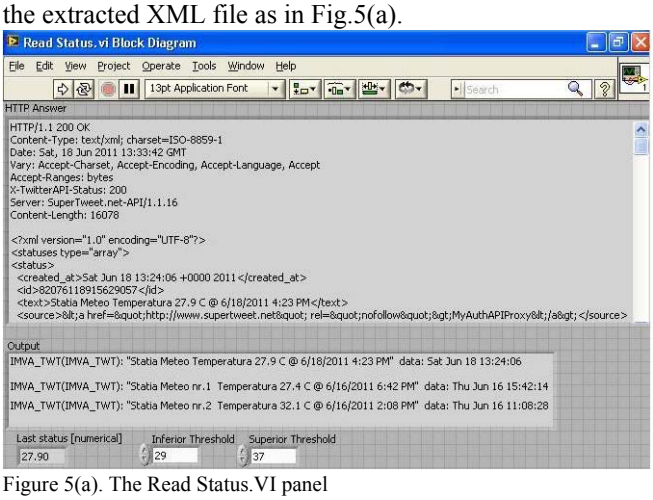

Figure 5(a). The Read Status.VI panel

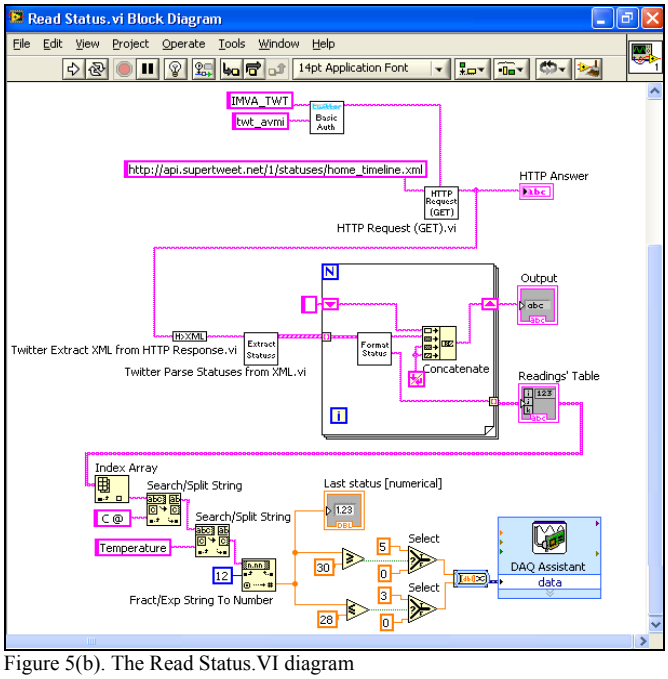

Figure 5(b). The Read Status.VI diagram

They are referred, as *<name>* and *<screen_name>* for the *<user>* data, and as *<text>* and *<created_at>*. This filtering process is enabled by searching based on labels (keys) specific to the XML, eg. *created_at*, *id*, *text*, *source*, *truncated*.

– DAQ Assistant is used this time for signal distribution and commands in analogic format, having pre-configuration of output channels using DAQmx. The dynamic data converter sends alarms voltage (e.g. 0/5V quasi-binary values), particularly by the analog outputs. The software is adjusting LED luminosity via a series resistor.

A solution based on the above-mentioned `curl` tools that is apparently simpler but indirectly using supplementary software is presented in Fig.6, where *M:\curl>* is, in our example, the path of the `curl` executable file.

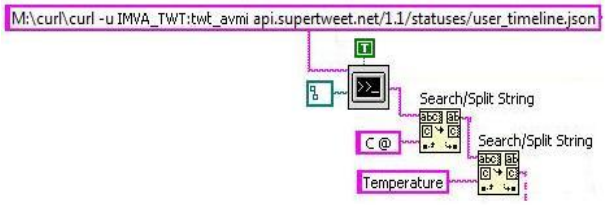

Figure 6. Modification of Read Status.VI based on third party curl software

For easy to use IoT applications dedicated to things that *dweet* (*direct tweeting*, by simply open broadcasting), Twitter or SuperTweet methods can also be replaced by recent third party online solutions like Dweet [22]. To demonstrate this, we used IMVA_DWT as the name of the {thing}, in a simple HTTP POST like: `https://dweet.io/dweet/for/IMVA_DWT?Temperatura=27.4C&foo=bar`

As immediately visible with *json viewer* or other freeware tools, the *dweet* has the JSON structure: `{"this":"succeeded", "by":"dweeting", "the": "dweet", "with": {"thing": "IMVA_DWT", "created": "2014-04-20T 15:59:42", "content":{"Temperatura": "27.4C", "foo":"bar"}}}`

If the name of the {thing} isn't trivial and should be known only be corresponding nodes in the IoT (with simple precautions like periodical rename), individual *dweets* can be retrieved with HTTP GET commands like `https://dweet.io/get/latest/dweet/for/IMVA_DWT`

The *json viewer* can display, at the corresponding client, a similar *dweet* – the difference is the field `"by":"getting"`, instead of `"by":"dweeting"` in the post.

We can use, for instance, cURL with a syntax like:

```
D:\> curl -k
https://dweet.io/get/latest/dweet/
for/IMVA_DWT{"this":"succeeded","by":"getti
ng","the":"dweets","with":[{"thing":"IMVA_D
WT","created":"2014-05-
20T15:59:42","content": {"Temperatura":
"27.4C", "foo":"bar"}}]}
```

Then, in a way similar to the details of Fig.6, this can be easily embedded in an adapted LabVIEW virtual instrument.

VI. Implementation of OAuth

For the more advanced OAuth method, a confidential `oauth_consumer_key` and an `oauth_token` (as can be seen in the following example) are generated by the Twitter management service and made available only to the user (for his/her exclusive use).

The first approach of this *direct* solution (without involving third parties that still accept BA methods) is the use of curl directives. Compared with the previous Read Status directives, in this case we should use:

```
curl --get
'https://api.twitter.com/1.1/statuses/
user_timeline.json' --header
'Authorization: OAuth
oauth_consumer_key="...",
oauth_nonce="...", oauth_signature="...",
oauth_signature_method= "HMAC-SHA1",
oauth_timestamp="...", oauth_token="...",
oauth_version="1.0"' -verbose
```

where `oauth_nonce` parameter is the unique, encoding in Base64 of a sequence with 32 bytes of random data generated once for each request; the `oauth_signature` is produced with the `oauth_signature_method` Hash Message Authentication Coding (HMAC) – of all these previous parameters by the secure hash algorithm (SHA) number 1; `oauth_timestamp` represents the number of seconds since January 1, 1970 00:00:00 GMT.

In order to integrate these directives with the social networking of instrumentation powered by LabVIEW, our first solution is the straight adaption of the VI set presented in Fig.6 – with the SysExec sub-VI for using programs like curl that can be invoked directly from a DOS command line.

The recently released "i3 Twitter Toolkit for LabVIEW" [23] is using only OAuth (version 1.0a) by dedicated "i3 OAuth Toolkit for LabVIEW". It supports mainly JSON data format (by an "i3 JSON Toolkit for LabVIEW"). It can invoke any Twitter API v.1.1 and provides already built VIs functionality for posting statuses, reading user's tweets and mentions or search Twitter based on queries. These solutions belong to the i3 "Interactive Internet Interfaces" of recent LabVIEW versions.

After the simple and direct integration (via curl and SysExec sub-VI) of OAuth with our telematic demonstrator, our next approach was based on the same principle of text processing (segmentation, parsing etc) using the available LabVIEV library "i3-twitter.lvlib".

In fig.7 and 8 we present how the tweets of our proof-of-concept weighing scale can be brought into the telematic system, in a bilateral way.

VII. Conclusions and Further Development

The present research paper describes our contribution to the integration of distributed instrumental systems in the IoT using consolidated social networking platforms like Twitter for asynchronous communication.

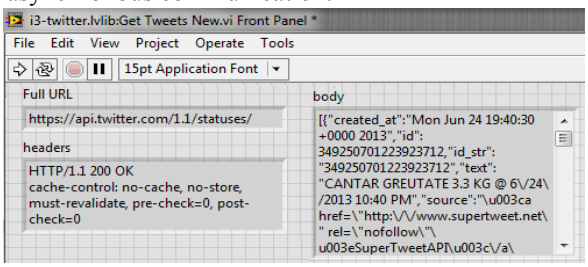

Figure 7. Panel of the Get Tweets New.VI – one can see the JSON format of the Tweet published by the instrument "cantar" – Weighing scale

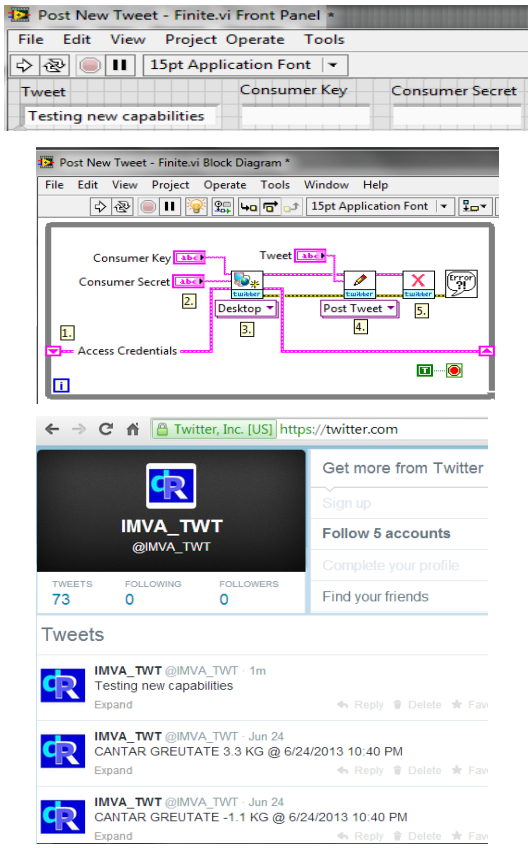

Figure 8. Panel and Diagram of the Post New Tweet - Finite.VI, followed by a proofing image of the Twitter timeline updated with the "Testing new capabilities" tweet

Our case studies in telematics have included proof-of-concept demonstrators using sensors and actuators driven by local embedded intelligence close to Smart Dust paradigm.

We have demonstrated that Twitter RESTful APIs can build an application interconnect layer compliant with cloud SaaS paradigm, meeting most of the research objectives for implementing social networks of instrumentation. Leveraging existing IP-based technologies, these APIs advantage was the reducing of platform dependencies as they can be invoked using generic HTTP methods, via the common TCP port 80 opened in most of the networks.

Any web-connected machine can tweet in order to broadcast its process-specific messages. As the APIs are hosted in the cloud, it is not need anymore to implement a server-proxy infrastructure for dedicated web-services.

Our objective of supporting real-time communications was met partially by the LabVIEW built-in Real-Time Engine but mostly for local transfer.

As Twitter is asynchronous, we look forward to its further developments enabling streaming features (or, at least, its

permanent *push* capabilities) enabling integration of DSTP (DataSocket Transfer Protocol) into faster social networking of instrumentation. Other versions of our VIs, transforming monitoring into alarms, might implement periodic measurement by the LabVIEW *Continuous Run* function;

they should transmit (tweet) only for exceeding threshold values (e.g. temperature limits in our case-study). In case of reliable Internet communications and greater time-constants accepted in industrial processes, distributed on/off actuators (e.g. electro-valves) can be engaged for closed-loop control. Besides project-specific solutions for various authentication mechanisms, a main contribution was the bilateral extension of instrumental communications in the social network.

Our objective of securing the entire communication chain was met by various solutions of BA and OAuth (combined with SSL encryption). Our case studies used the BA of the service provider SuperTweet, just as the current third party of choice, in order to validate the instrumentation communities' concept via a complete loop for collection, transmission and presentation. Consolidated solutions would need either a contractual SLA (Service Level Agreement) or a good Risk-Management plan for any business-case based on such intermediation – considering backup solutions in case of interruption such of third party services. Local round-about solutions like curl and/or BA-to-OAuth translation proxy were presented. Further customization of our telematic solution could easily pass from fixed authentication parameters to variable ones, quite usual in M2M where security and access issues remain as relevant as the critical information transmitted.

The development of future M2M communications in IoT will enable the successful integration of industrial and household (domotics) applications in our daily life. As the current paradigm of social interaction is shifting towards mobility and permanent availability of information, the IoT must meet the same level of demand and ergonomics.

The main guidelines for implementation represent the unrestricted access to information and the desire for a sound economic model. The Smart Grids and other utility networks can even be equipped with monetizing layers, adapting the communication to the recent *prosumer* (producer-consumer) capabilities. Information exchanged between aggregated devices can also contain counters for monetary information – it is obvious that such "tweets" should be even more secured. It is expected that the next years will bring also recommendations for pre-standard *interaction* in the IoT on behavioural models based on event-driven control. We have shown how collecting "tweets" by Virtual Instrumentation can close such control loops based on embedded processing, local interpretation and decisions, an important step that was emulated in our case-study.

Our study also contributes to the trend of future aggregation of virtual colonies as IaaS (Infrastructure as a Service) [24-26], including transactional (negotiation) schemes like those recently introduced to cooperating robots running ROS – Robots' Operating Systems. Using solutions like the ones we have proposed, implemented, tested and validated, publish/subscribe procedures become possible for broadcasting status, service requests, capability offers etc. thus enabling full remote control.

Remarkable Twitter-based services were proposed by Neoformix [27] enabling models of presence/availability in social networks to be extended for IoT by with various capabilities to detect similar behaviours in the cyberspace and to suggest partnerships.

REFERENCES

[1] S. Sean Dodson, R. Van Kranenburg, The Internet of Things - A critique of ambient technology and the all-seeing network of RFID, Institute of Network Cultures, 2009.

[2] D. Uckelmann, M. Harrisson, F. Michahelles (editors), Architecting the Internet of Things, Springer, 2011.

[3] K. Ashton, "That 'Internet of Things' Thing" – RFID Journal, July 2009 [Online]. Available: www.rfidjournal.com/article/view/4986

[4] A. Broering, T. Foerster, S. Jirka, C. Priess, "An intermediary layer for linking sensor networks and the sensor web," Proceedings of the 1st International Conference on Computing for Geospatial Research & Application, "COM.Geo 2010", ICPS - ACM International Conference Proceeding Series, Article 12, pp. 12:1-12:8, 2010

[5] Message Queue Telemetry Transport, [Online]. Available: http://mqtt.org

[6] P. Saint-Andre, K. Smith, R. Troncon, XMPP - the definitive guide, O'Reilly Media, 2009

[7] P. Ogrutan, L. E. Aciu, "Microcontroller-based system for accelerated reliability tests of electronic equipment," Proceedings of the International Conference AFASES 2013, pp. 4.1.1-6, 2013

[8] E. Coca, V. Popa, "A practical solution for time synchronization in wireless sensor networks," Advances in Electrical and Computer Engineering, 12(4), pp. 57-62, 2012.

[9] B. Li, J. Yu, "Research and application on the smart home based on component technologies and Internet of Things," Volume 15 of the Procedia Environmental Sciences Journal, pp. 2087 – 2092, 2011.

[10] A.C. Stanca, V. Sandu, R. Vaduva, O. Nemeth, "Distributed system for indoor temperature control," Proceedings of the IEEE International Conference on Applied and Theoretical Electricity - 11th edition - ICATE 2012, pp.6.3.1-6, 2012.

[11] Commission of the European Communities (2009-06-18), Internet of Things – An action plan for Europe, COM(2009) 278

[12] Smart Grid Leadership Report: Global Smart Grid Implementation Assessment, oct.2010, document no. 1021417, [Online]. Available: http://www.smartgrid.epri.com

[13] K.C. Budka, J.G. Deshpande, T.L. Doumi, M.Madden, T.Mew – "Communication network architecture and design principles for smart grids," Bell Labs Technical Journal 15(2), pp. 205-227, 2010.

[14] National Instruments Corp. - LabVIEW - www.ni.com/labview

[15] National Instruments Corp. - Multifunction DAQ for USB [Online]. Available: www.ni.com/pdf/products/us/20043762301101dlr.pdf

[16] National Instruments Corp. - Data acquisition basics manual – 2000, [Online]. Available: http://www.ni.com/pdf/manuals/320997e.pdf

[17] National Semiconductor – Precision centigrade temperature sensor - [Online]. Available: www.national.com/mpf/LM/LM35.html

[18] The Lantronix XPort embedded web-server – [Online]. Available: http://www. lantronix.com/pdf/XPort_DS.pdf

[19] Google GSON [Online]. Available: https://code.google.com/p/google-gson/

[20] cURL – Command line tool for transferring data with URL syntax [Online]. Available: http://curl.haxx.se

[21] C. Loew, "Update Twitter status from LabVIEW", National Instruments Corp., Document # 3284 of the NI Developer Community 2009 [Online]. Available: https://decibel.ni.com/content/docs/DOC-3284

[22] Simple messaging for the Internet of Things - Bug Labs Inc., 2014 [Online]. Available: https://dweet.io

[23] National Instruments Corp. – i3 Twitter Toolkit for Labview, 2014 [Online]. Available: https://decibel.ni.com/content/groups/interactive-internet-interface-twitter-toolkit-for-labview

[24] T. Berners-Lee, J. Hendler, O. Lassila, "The Semantic Web," Scientific American Magazine, 284(5), pp. 34-43, 2001.

[25] M. Kranz, L. Roalter, F. Michahelles, "Things that Twitter: social networks and the Internet of Things," Proceedings of the 8th International Conference on Pervasive Computing, 17-20 May 2010, Helsinki, Finland – " Lecture Notes in Computer Science", Volume 6030, pp. 51-60, 2010, DOI: 10.1007/978-3-642-12654-3

[26] J.B. Waldner, Nanocomputers and swarm intelligence. London, Wiley-ISTE. pp. 227-231, 2008.

[27] J. Clark, "Discovering and illustrating patterns of data", [Online]. Available: http://neoformix.com/