Last-meter Smart Grid embedded in an Internet-of-Things platform

E. Spanò, *Student Member, IEEE*, L. Niccolini, S. Di Pascoli, and G. Iannaccone, *Senior Member, IEEE*

***Abstract*-- The Customer Domain of the Smart Grid naturally blends with smart home and smart building systems, but proposed approaches are "distributor-centric" rather than "customer-centric", undermining user acceptance, and often poorly scalable. To solve this problem, we propose a detailed architecture and an implementation of a "last-meter" smart grid - the portion of the smart grid on customer premises – embedded in an Internet-of-Things platform. Our approach has four aspects of novelty and advantages with respect to the state of the art: i) seamless integration of smart grid with smart home applications in the same infrastructure; ii) data gathering from heterogeneous sensor communication protocols; iii) secure and customized data access; iv) univocal sensor and actuator mapping to a common abstraction layer on which additional applications can be built. A demonstrator has been built and tested with purposely-developed ZigBee smart meters and gateways, a distributed Internet-of-Things server and a flexible user interface.**

***Index Terms*-- demand side management, internet of things, power meter, smart grid, telemetering.**

I. Introduction

The last-meter Smart Grid is the portion of the Smart Grid closer to the home, and the one with which customers interact. It allows a two-way information flow between customers and electric utilities, transforming the "*traditionally passive end-users into active players*" [1] in the energy market.

Considering the seven domains of the conceptual model of Smart Grids proposed by the National Institute of Standards and Technology [2], [3], the last-meter Smart Grid corresponds to the "Customer Domain". It enables residential, commercial, and industrial customers – based on their different energy needs – to optimize energy consumption and local generation, and to actively participate to demand-response policies [4], one of the most disrupting aspects of smart grids.

Non-technical customers need a simple way to control energy consumption and production, and to exchange power usage data at the proper level of granularity with energy providers or distributors.

From the point of view of market acceptance and penetration, the last-meter smart grid is just one aspect of the broader concept of smart home and smart buildings. The consequence of this consideration is that one can hardly imagine a situation in which the consumer side of the smart grid and other smart home applications rely on different and separate infrastructures or platforms.

However, smart-grid architectures proposed in the literature typically focus on the needs of power distributors to manage the complete power grid [5]. They reach customers' premises with an ad-hoc network of smart meters connected by GPRS or, sometimes, with a dedicated PLC technology [6]. They do not take into account the possibility that customers already have other smart home infrastructures [7]-[13]. On the other hand, some solutions proposed in the literature, based on a smart home infrastructure, are not designed to be seamlessly scalable to large deployments [14]-[23].

In this paper, we present an architecture for the last-meter smart grid that is embedded in a platform for the Internet of Things (IoT) [24]. Our architecture has four main advantages and elements of novelty with respect to the state of the art, each corresponding to the basic requirement of being "customer-centric" and scalable, in order to improve market acceptance and ease of deployment:

- **It seamlessly integrates smart grid with smart home applications.** We assume that the typical early adopter of a last-meter smart grid is also a user of smart home applications (dedicated to security, entertainment, home automation, et cetera). In order to avoid duplication and enable possible synergy, the platform must support both smart grid and other smart home applications.
- **It can gather data from heterogeneous sensor communication protocols.** The last-meter Smart Grid exploits existing infrastructure for in-home connection to smart meters. Therefore, its architecture allows different wireless or wired protocols to be used for communications between meters, users, and other parts of the system.
- **It provides secure and differentiated access to data.** Single customers have complete fine-grained access to their own data, and can enable access by third parties. On the other hand, distributors and energy utilities can receive coarse-grained and aggregated statistical data.
- **It allows to univocally map each sensor and actuator to a common abstraction layer.** To simplify interaction with non-technical users, sensors and actuators are also described at a higher abstraction level, independent of the physical

This work was supported in part by the ENIAC JU grants ERG (contract n. 270722-2) and E2SG (contract n. 296131), by the Italian Ministry of Economic Development through the Cleverhome Grant, by the Tuscany Region under the SED POR-FSE project, and by Quantavis. S.r.l. The authors wish to acknowledge support, services, and fruitful discussions from Dario Presti, Irene Bontà, and Gianluca Iannaccone, and Quantavis s.r.l.

E. Spanò, S. Di Pascoli, G. Iannaccone are with the University of Pisa, SEED Lab, PUSL, Via dei Pensieri 60, Livorno (email: elisa.spano@iet.unipi.it, s.dipascoli@iet.unipi.it, g.iannaccone@unipi.it). L. Niccolini was with the University of Pisa when the work was performed.

details and of the communication protocols. Developers and businesses can use this higher abstraction level to provide additional services.

In the following, we present the complete architecture of the IoT platform (Section II), a hardware and software implementation of the last-meter smart grid with experimental tests (Section III). In Section IV, we compare our proposal with related work, and in Section V we present our conclusion.

II. Platform for the Internet of Things

We have developed a platform for the Internet of Things (IoT) as a scalable distributed system that can seamlessly support an in-home Smart Grid and different concurrent applications for remote monitoring and control.

The platform architecture is illustrated in Fig. 1. It consists of three main parts: the sensor and actuator networks, the Internet-of-Things server and the user interfaces for visualization and management.

Sensor and actuator nodes communicate in a reliable bidirectional way with the IoT server.

The communication between the nodes and the IoT server follows the TCP/IP client-server model. Sensors send messages in their native format to the IoT server (through a gateway, if needed), over an encrypted link.

The IoT server converts the raw payload, containing information from heterogeneous nodes, into a standard format, containing object identifier, object type, measurement unit, data field, geographical position, and timestamp. In this way, data can be easily represented, manipulated and aggregated without considering the communication protocol of the originating source.

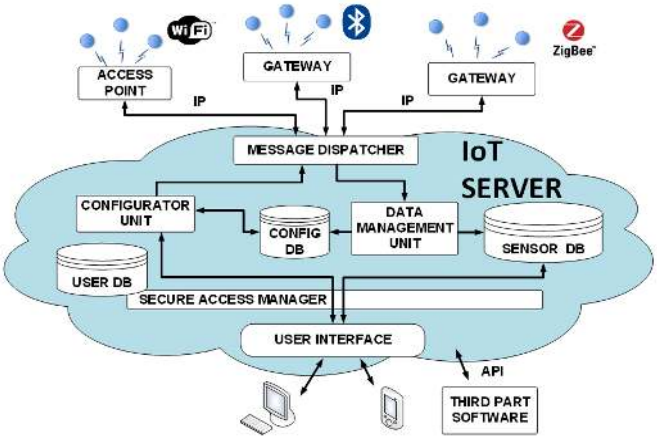

Fig. 1. Block diagram of the Internet of Things platform supporting the in-home Smart Grid.

A web-based graphical interface allows users to access real time and historical sensor data. The same interface allows users with administration privileges to manage networks and single nodes. Third-party software can access the platform using a REST API [25].

Due to the possibility of using the system to collect sensitive and confidential data, the platform ensures an adequate security level both to end-to-end communications and to data access. For this reason, users need to be authenticated before they can access the platform and can only access specific sets of sensor data through HTTPS. The IoT server supports multiple encryption protocols (AES-128, SSH).

At a finer level of detail, the Internet of Things platform consists of several hardware and software components, each described by its functions and by its interfaces with other components. In this way, the architecture is easily scalable and robust. Each component can be modified, redesigned, and extended with minimum impact on the rest of the system. The components are indicated in Table I and are described in more detail in the following subsections.

A. Sensor and actuator networks

1) Sensor and actuator nodes

The sensor and actuator nodes can be part of networks implemented with wired (e.g. CAN, power line communication) or wireless (e.g. ZigBee, Wi-Fi, Bluetooth) network protocols. The architecture is designed to accommodate different and heterogeneous sensor and actuator networks. The data management unit is responsible for translating information to the format required by the sensor database.

TABLE I
Main components of the Internet of Things platform

Internet of Things Platform	
Parts	**Main components**
Sensor and actuator networks	• Sensor and actuator nodes • IP gateways
IoT Server	• Message dispatcher • Data management unit and sensor DB • Configurator unit and database • Secure access manager
User interfaces	• Visualization interface • Configuration interface • Applications using the REST API

On the other hand, bidirectional communication channels to/from the nodes enable the IoT server to interrogate, configure, and program them. Configuration messages mainly carry node-specific information (for example measurement thresholds, alarm settings) or firmware updates.

Even if specific node characteristics depend on the network implementation, the proposed architecture supports the possibility to add or remove any network component in real time. Indeed, any node can join the system without requiring any change to the network implementation. For this reason, any new node that joins a network connected to the platform is automatically identified and immediately accessible from the network administration interface for registration and configuration. Similarly, updating or un-joining nodes are automatically referred to the IoT server. The interface between sensor networks and the platform is based on a communication protocol between the gateway and the IoT server defined by API specifications.

Each node has to be uniquely identified to ensure global device accessibility. However, node addresses in typical sensor networks may change over time and are often unique only within a single network. For this reason, the IoT server assigns a unique ID to each node of the network (for example

an octet string based on the EUI-64) and maintains the mapping between such ID and the network address provided by the local sensor network coordinator. When a node sends a message to the server, the gateway translates its network address into the unique ID, and vice versa for messages from server to nodes.

2) IP Gateway

The gateway is the element connecting a sensor/actuator network - if it has no direct IP capability - to the IoT server via an IP link. The gateway is bidirectional: for uplink communication it collects data received from the network nodes, performs reformatting/encapsulation if required, and sends them over a secure TCP/IP link to the message dispatcher. For downlink communication, it forwards to the receiver node(s) the commands received from the IoT server.

We propose a different gateway concept with respect to the one commonly used to integrate heterogeneous networks with an external network [26]-[28]. These systems use a gateway-based approach [29], where the gateway performs a conversion of data into a universal format.

In our architecture, instead, it is the IoT server that performs such operation. Therefore, the gateway sends network packets over TCP/IP in the *native* format and both the gateway and the message dispatcher are transparent at the logical communication level between nodes and IoT server.

This choice provides three meaningful advantages:

i) The gateway can have reduced hardware requirements and computational complexity. Our gateway has only to ensure an IP connection, to implement the encapsulation of the nodes' native protocol into TCP/IP packets, and to ensure the security level required by the specific application.

ii) Different applications and new functionalities can be developed and added without modifying the gateway.

iii) The user side of the platform can communicate at the application level directly with network nodes.

As a validation of this concept, the gateway is currently implemented with Cortex-M3/M4 microprocessors. It is designed for easy deployment in a typical home LAN, which uses private non-routable addresses and is connected to the Internet through a router able to perform Network Address Translation (NAT). The machines on such LAN cannot receive incoming TCP connection from a remote server without a manual configuration of the router. To avoid this user configuration, our gateway implements the client side of a TCP connection to the IoT server and always initiates the communication with the message dispatcher (Fig. 1).

B. IoT Server

1) Message dispatcher

The message dispatcher manages the bidirectional communication between each gateway and the rest of the system. It only deals with low-level communications from nodes (through the gateways) to the data management unit and from the configurator unit to the nodes.

It has the main task of listening to new connections from IP nodes that want to join the system. For every connection, it decrypts incoming packets and forwards them to the data management unit, for interpretation and storage. In the other direction (downlink), it encrypts and encapsulates messages from the configurator unit into a TCP message, and forwards them to the destination gateway. The packet structure is illustrated in Fig. 2. Each packet contains the following information: i) 64-bit gateway address (which uniquely identifies the local network), ii) opcode, iii) timestamp, with resolution of 1 second, iv) serial number, v) payload in raw format.

The opcode defines the function of each packet. Packets can be divided in two main classes: administration packets and data relay packets.

For every type of local sensor network protocol included in the platform two opcodes are defined: one used for the upstream and one for the downstream data transmission.

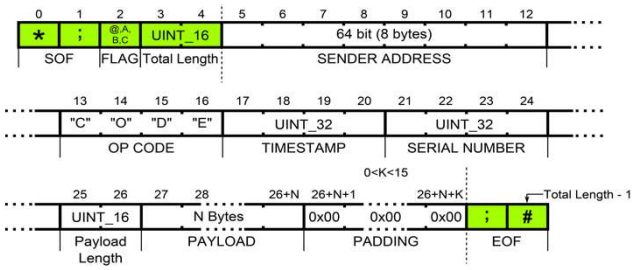

Fig. 2. Structure of the TCP/IP packet of the communication protocol between gateway and message dispatcher.

Administration packets are used for configuration and maintenance of the gateway. The addition of a different local sensor and actuator network requires only the addition of two new opcodes to the protocol.

2) Data management unit and database storage

The Data management unit is a collection of software modules, each able to manage the messages of a specific sensor network type. These components receive node packets in their native format and extract their payload. Depending on the payload, two different storing mechanisms are used:

- If the payload contains measurement data from a sensor or an event notification by an actuator, data are stored in a unique format in a streaming sensor database.
- If the payload contains specific network messages (configuration, management information, communication channel, node address, etc.), messages are stored in the original format into the configurator database.

The presence of the sensor database decouples data collection from data processing and visualization, so that users do not need to interrogate nodes directly. This approach is useful especially when sensor networks are heterogeneous. It is also very useful when nodes are battery-operated devices. Decoupling allows nodes to stay most of the time in sleep mode and periodically wakeup to receive commands and configurations and to send measurement and status data.

In the sensor database, sensor data are represented with a unique format, independent of the local sensor network protocol, and are univocally associated to the physical nodes through the unique node ID. In this way, data can then be easily accessed by performing a simple query to the database, and can be processed and visualized independently of the characteristics of the physical source.

Unlike sensor data, configuration variables and messages can be completely different for nodes of different type and network protocol. Storing configuration messages with no protocol conversion avoids possible loss of information.

Both sensor and configuration information are stored in remote databases. This makes the system easily scalable and does not impose limitations on the data volume a node can send to the server.

3) Configurator unit and database

This unit configures networks and nodes according to inputs from users and authorized applications and according to the system status stored in the configuration database. Also the configurator unit is a collection of software components, each dedicated to a specific type of sensor/actuator network. For any new added sensor network protocol, dedicated modules must be added to the configurator unit.

4) Secure access manager

A secure access manager that ensures privacy and data protection coordinates all communication between end-users and the IoT server. It provides access to stored information and network configuration only to authorized users or third-party applications, based on a database of users and their permission to each resource (networks, node). By default, network owners have administrator rights on their networks.

C. User interfaces

Users, service providers and application developers can interact with the platform through user interfaces (web-based or API). The user interface offers two main functionalities related to two main client profiles: standard users and administrators. Standard users can access sensor data and control actuators. Administrator users have superior access: they can also see the configuration and the status of the nodes and dynamically configure them.

The interaction between users and the platform through the web user interface can be modeled as a finite state machine. In this representation (illustrated in simplified form in Fig. 3). Transitions are triggered by IoT node events and client requests, and depend on user permissions.

The initial state is the login page. Non-registered users can only see public sensor data and cannot send commands or configure nodes and networks. Registered users have access to the user home page.

The web interface can be divided in:

1) Visualization interface

The visualization interface displays current and historical information from sensors and actuators is a series of pages. In addition, the visualization interface allows authorized users to send commands to actuators. Users can create custom data views and visualization pages, send commands, set rules and alarm notifications.

2) Administration interface

The administration interface provides users with the possibility to remotely manage and configure their networks. In addition, users can set the data visibility of their own sensors and manage third-party access and privileges to their nodes. The layout and the fields included in the administration interface pages depend on the type of networks and on the corresponding protocols. The administrator interface is also used to easily and remotely register new gateways and configure new network

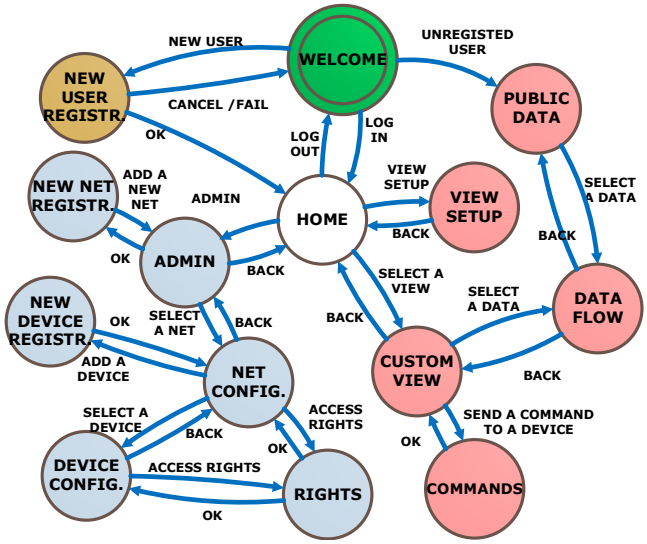

Fig. 3. User interface State machine diagram. Blue circles represent states of the administration interface; red circles represent states of the visualization interface.

connections.

To establish the connection, the gateway needs to know the network name and the IP address of the message dispatcher, the port number on which it accepts connections and the network AES security key. For this reason, it has to be registered and configured.

Using the web interface the administrator users can add a new network on their admin page, by inserting the gateway address, selecting the type of network and assigning a name, a description and a network location. The server will generate a network security key (ex. AES key) and will save it in the user database along with the network information.

After the configuration with this security key, the gateway connects to the server sending a request connection. The server processes the request, spawns a new process and lets it communicate directly with the gateway. This task acquires the network information from the configuration database and informs the gateway it can begin the encrypted communication. After the communication is setup, the network appears on the configuration page of the user.

3) Web service API

Web service APIs open the platform to service providers and new client applications (as for example an Android app as in Fig. 4). APIs offer an easy and unified way to retrieve information collected from heterogeneous sources. Service providers, utilities and third parties can use the API to obtain single, multiple or aggregated measurement data, useful to develop new services. To protect sensitive information, the sensor owner can define third-party accessibility of collected data. Only registered end-users and authorized third-party applications can retrieve sensor data from the sensor database through the API.

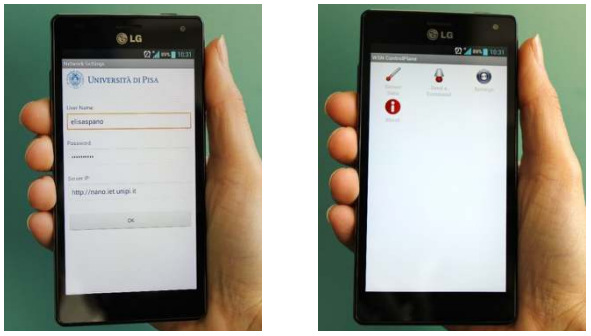

Fig. 4. Example of a sensor data visualization using a smartphone.

III. IN-HOME SMART GRID IMPLEMENTATION

We have implemented an in-home prototype on the IoT platform, building dedicated hardware and software. This first prototype only includes a ZigBee network connected to the IoT server through a ZigBee IP gateway. The sensors are smart plugs, placed between home appliances and a wall socket, and able to collect real-time power consumption data from the loads. Customers can have a visual feedback of their energy consumption and can remotely control each load. Let us consider in detail the elements of the system.

A. Smart plug

As shown in Fig. 5 (left), the smart plug is enclosed in a plastic case with a plug and a socket section and can be inserted in a standard wall socket.

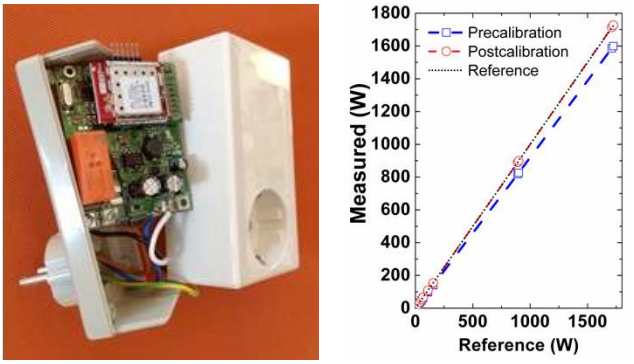

Fig. 5. Smart Plug prototype (left) and power calibration curves (right).

The smart plug collects load information from the attached electrical equipment. Information includes single-phase active, reactive, and apparent power; power factor; sampled waveforms; RMS current and voltage; on/off status. The smart plug is also an actuator, since it can turn the load on and off. Our smart plug has no buttons and can be completely configured and controlled through the user interface.

In the current design (Fig. 6), the communication with the ZigBee network is provided by a Freescale MC13224 SoC, equipped with an AES128 encryption engine. The board includes an ARM7 processor with 128 kB of Flash, 96 kB of RAM and 80 kB of ROM memory. An Analog Devices ADE7953 is used for energy measurement.

Load control is implemented using a single pole bistable 12 V relay supporting loads up to 16 A.

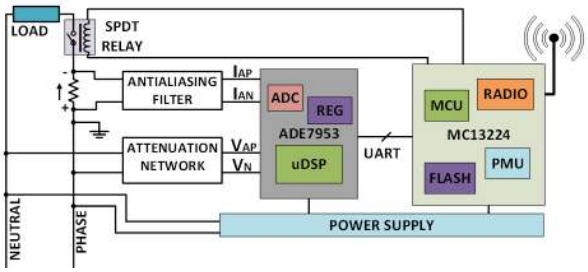

Fig. 6. Smart plug block diagram.

The board includes a power supply unit, which provides the supply voltages of 12 V for the relay and 3 V for the ADE7953 IC and the MC13224 SoC. The firmware running on the smart plug is implemented using the Freescale ZigBee stack, called BeeStack.

The ADE7953 can be calibrated by MCU through the serial link. The smart plug has been calibrated using as a reference meter tabletop power meter PCE-PA 6000 (Fig. 5 – right). Calibration coefficients can be remotely send to the node through the ZigBee radio. The ZigBee smart plug has an accuracy of 1.1% (post-calibration).

B. Gateway

As the power meter is a ZigBee device, a ZigBee/IP gateway is needed to allow communication with the IoT server. The gateway is composed by an Ethernet interface, a microcontroller, and a ZigBee RF transceiver.

Following what we wrote in Section IIA2, the gateway can have reduced hardware requirements. However, in order to reduce chip count, and hence cost, we have selected a microcontroller with an on-chip Ethernet controller, which slightly increases the processor hardware requirements.

Among the suitable microcontrollers (MCUs), we choose the Freescale Kinetis K60 MCU. It is based on an ARM Cortex M4 processor with hardware encryption (Fig. 7).

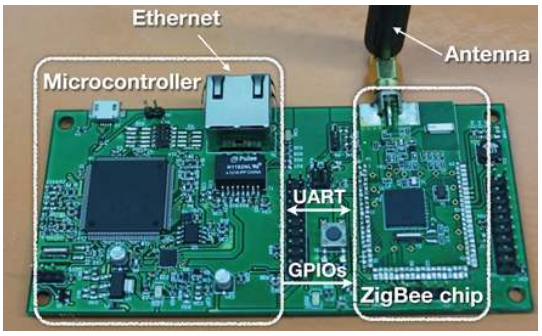

Fig. 7. ZigBee IP gateway prototype

The gateway firmware makes use of the lwIP TCP/IP stack [30]. When connected to a LAN equipped with a DHCP server (like most of home ADSL modem/routers) it can auto-configure its network interface. All the messages exchanged between the server and the gateway can be encrypted.

C. Message dispatcher

The message dispatcher is implemented as a multi process application running on a Linux machine. The main application task continuously listens to new possible connections from gateways or other IP nodes. Every time a

new TCP connection is established, a new process is created and remains active until the connection is closed by the gateway or a timeout occurs. This new process saves received packets in a UNIX-named pipe, which is read by the data management unit (II.B.2). Moreover, the process collects and delivers from the config database (II.B.3) downstream data addressed to the gateways.

D. Data collection and storage

The data collection unit is implemented using the CoMo platform software [31]. CoMo has been developed for the fast prototyping of network data mining applications and has been used in large testbed deployments, such as PlanetLab [32]. Hence it is scalable to very large systems and very high data rates.

The CoMo architecture presents an abstraction layer for the network interface and for the IoT server. Developers can implement new algorithms for processing sensor network data streams without any explicit knowledge of the internals of the monitoring system, transport media, memory and storage organization.

CoMo follows a classical modular approach that has proven to be successful in similar contexts [33]. The core system provides an API to enable the development of modules for each packet stream. Each module uses a common data model and specifies the information of interest (together with its resolution and accuracy). The system identifies if that information is available. CoMo converts all incoming data streams in a unified packet stream that is then delivered to the subsequent processing queries [31].

Fig. 9. Authorized uses can see possible commands that can be sent to each virtual device.

In the context of the IoT platform, each CoMo module interprets the packets of a specific type of sensor network and extracts data to be further included in the sensor database or in the config database.

E. User interface, Configurator unit, Secure access module

We have chosen a web interface implementation that: i) is easily scalable to many concurrent client connections, ii) implements user authentication, and iii) is friendly to inexperienced users on different form-factor devices.

The web interface, the configurator unit, and the secure access module are based on Tornado [34], an open-source scalable non-blocking web server and web application framework written in Python. The graphical interface is responsive and rests on Twitter Bootstrap [35], a set of ready-to-use graphical elements. Among the many services, which could be provided by our architecture, we have implemented five main user interface functionalities:

• **View sensor data:** As shown in Fig. 8, this page allows the user to visualize collected data of a sensor. The page provides a chart where data are plotted as a function of time and a time range selector. On the right, a timepicker allows an easy selection of measurement period.

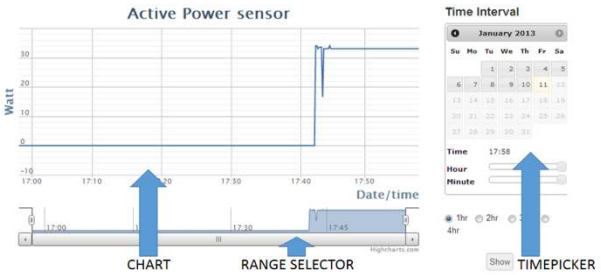

Fig. 8. Example of the visualization of data from a smart meter in the implemented last-meter Smart Grid.

• **Register and configure new networks:** The administrator users can register and configure new networks through dedicated pages. The network configuration is saved in the configurator database and the AES key for the gateway is generated. This is the only action required to an administrator user to make its network visible.

• **Network configuration and rights management:** The network configuration page allows the administrator to assign access rights with different privileges, to modify specific network options (netID, communication channel, security level, etc.) and to manage sensors visibility, grouping, and security.

• **Register new sensors:** The system automatically recognizes unregistered sensors that are sending data through a registered network/gateway and presents them to administrators, who can decide to register them.

• **Send commands:** Authorized users can send commands to actuators. All nodes are represented as one or more virtual devices (Fig. 9) that can possibly accept commands.

If this is the case, the command interface is visible to authorized users. When a command is sent through the user interface, a JSON message is created containing the recipient ID and the command. This message is then converted into the sequence of messages to be sent to the node. All messages are stored in a configuration database that is continually read by the message dispatcher.

The user database and the config database are implemented with MySQL. The Tornado backend implements direct access, after authentication, to the sensor data through HTTP. This makes it easy to ask for sensor data directly from the JavaScript frontend through Ajax requests. The query returns sensor data for a given network id, sensor id and time range as an array of <timestamp, measure> tuples.

F. Experimental demonstration

We have performed several extensive tests of the implementation to verify operation and reliability. Each element has been individually tested and validated. All sensors have been calibrated as described in Sec. III.A. A

complete demonstrator including all elements of the implementation and more than 10 sensors has run continuously for more than three months without loss of data in a laboratory setting, with quasi-daily addition and removal of sensors. As an example, data extracted from the demonstrator in a time span of about three hours from three smart meters are plotted in Fig. 10.

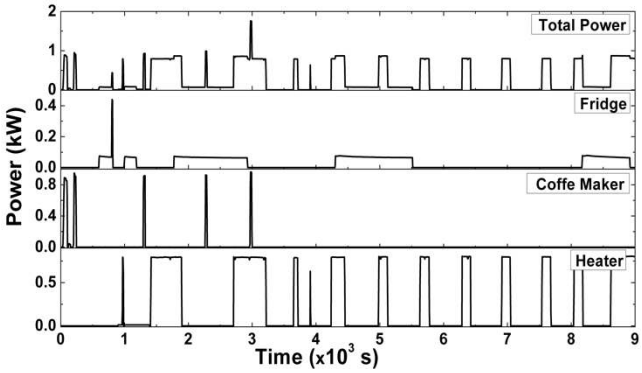

Fig. 10. Power consumption data collected from 3 common household loads and partial sum.

Data from a small refrigerator, an electric heater, and an expresso coffee maker show the typical features of the corresponding power loads.

IV. Comparison with Related Works

Comparison with related works must consider recent literature in the neighboring fields of distributed sensor networks, home automation and smart grids. We can loosely classify the large number of related papers in two groups.

A set of papers focuses on the automation of the complete power distribution grid, of which the "last-meter" smart grid is only a subsystem. In this case, the complete grid includes power generation plants, transmission and distribution networks, and "smart" consumers, with local generation capabilities, flexible usage and sometimes energy storage capacity. This large infrastructure is usually managed by a central server/data storage or SCADA system [7]-[9]. For obvious reasons, the proposed systems are described only at the architectural level, with an extensive discussion of the goals and objectives but with few details of the implementation. They also require substantial investments in infrastructure, especially for data transmission from the customer site to the last node of power distribution ("last meter"). Many transport options are typically proposed, such as the use of dedicated lines, to POTS/modem, power line communications (PLC), wireless links [10], [11].

Most of these projects include a "smart meter" used for both data collection and billing [9], which can be deployed only by the power distributor or in strict coordination with it. A pilot project deployed by a power distributor [12] required an investment of 10 M€ for the territory covered by a single primary distribution transformer (about 30 MVA). It is worth noting that deployed smart metering networks are usually based on PLC links [13].

With respect to this set of papers, the advantage and the uniqueness of our approach are apparent. Our proposal is "customer centric", as opposed to "distribution centric", in the sense that favors ease of deployment and user acceptance, leveraging the smart home trend to enable the merging of smart grid and smart home applications in customers' homes. Indeed, our proposal focuses on the customer domain of the smart grid, possibly leaving the domains more evidently controlled by the utilities, such as transmission and distribution, to a "distribution-centered" treatment.

There is also another set of papers presenting home automation systems for power metering and analysis, which are therefore closer to the present paper. While [14] and [15] provide mostly an architectural description, [16]-[19] and [23], provide implementation details and a demonstration.

Most of the proposed implementations connect the home sensor or automation network to the wide area network or to a central server by means of a complex gateway, with large computational power (several MB of RAM and FLASH and a complete operating system) [19], [20]. Communication can occur for example via ZigBee and 6LoWPAN [18], [19], [23] but also dedicated point-to-point radio links are proposed in [16] and in [20]. The installation and configuration of this device makes the deployment of the system out of the reach of many end users. Ref. [19], [21] and [22] implement the data storage, analysis and user interface by means of a local server, assigning to the user the task to maintain and configure the system.

With respect to this other group of papers, the advantages of our proposed architecture and implementation consist in their intrinsic scalability to large-scale deployment. This is enabled by the choice of low-cost gateway and power meter (the bill of material of each is less than 15$), and by the accent on deployment by non-technical users.

V. Conclusion

We have presented an architecture, an implementation, and a demonstration of the Customer Domain of the Smart Grid, based on a platform for the Internet of Things that can host a broad range of smart home applications.

Novelty in this field must be found in the architectural concept, in the system integration, and in the prioritization of requirements. In this sense, our proposal has unique advantages and elements of novelty with respect to the state of the art: it is customer centric, it minimizes the deployment of specific smart grid infrastructure, and it leverages possibly available smart home applications, sensors, and networks. We believe this is key for a widespread acceptance of smart grid applications and equipment to be deployed at home.

VI. References

[1] V. Giordano, F. Gangale, and G. Fulli, "Smart Grid projects in Europe: lessons learned and current developments," JRC Scientific and Policy Reports, Eur. Comm., 2012 (Update).

[2] NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 1.0, NIST Special Publication, January 2010.

[3] R. Ma, H. H. Chen, Y. Huang and W. Meng, "Smart grid communication: Its challenges and opportunities", *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp.36-46, 2013.

[4] P. Palensky and D. Dietrich, "Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads", *IEEE Trans. Industrial Informatics*, vol. 7, pp. 381–388, no. 3, Aug. 2011.

[5] K. Samarakoon, J. Ekanayake, N. Jenkins, "Reporting Available Demand Response," *IEEE Trans. Smart Grid*, vol.4, no.4, pp.1842,1851, Dec. 2013.
[6] Energy Community Regulatory Board, European Union, "A review of smart meters rollout for electricity in the energy community," 2010.
[7] A. A. Khan and H. T. Mouftah, "Web services for indoor energy management in a smart grid environment," *in Proc. 2011 IEEE 22nd International Symposium on PIMRC*, pp.1036-1040.
[8] J. Byun, I. Hong, B. Kang, and S. Park, "A smart energy distribution and management system for renewable energy distribution and context-aware services based on user patterns and load forecasting," *IEEE Trans. Consumer Electronics*, vol. 57, no. 2, pp. 436–444, May 2011.
[9] A. Zaballos, A. Vallejo, and J. Selga, "Heterogeneous communication architecture for the smart grid," *IEEE Network*, vol. 25, no. 5, pp. 30–37, Sep. 2011.
[10] T. Sauter and M. Lobashov, "End-to-end communication architecture for smart grids," *IEEE Trans. Industrial Electronics*, vol. 58, no. 4, pp. 1218–1228, Apr. 2011.
[11] F. Benzi, N. Anglani, E. Bassi, L. Frosini, "Electricity Smart Meters Interfacing the Households," *IEEE Trans. Industrial Electronics*, vol.58, no. 10, , pp.4487-4494 Oct. 2011.
[12] Enel Press Release: "Italy's first smart grid in Isernia", 4 Nov. 2011, [Online]. Available: http://goo.gl/RsY8F
[13] B. Botte, V. Cannatelli, S. Rogai, "The Telegestore project in Enel's metering system," in *Proc. 2005 18th International Conference and Exhibition on Electricity Distribution 2005. CIRED 2005*, pp.1-4.
[14] Y. Yang, Z. Wei, D. Jia, Y. Cong, and R. Shan, "A Cloud Architecture Based on Smart Home," in *Proc. 2010 Second International Workshop on ETCS*, vol.2, pp.440-443.
[15] Hu, Q.; Li, F., "Hardware Design of Smart Home Energy Management System With Dynamic Price Response," *IEEE Trans. Smart Grid*, vol.4, no.4, pp.1878-1887, Dec. 2013.
[16] Q. Liu, G. Cooper, N. Linge, H. Takruri, and R. Sowden, "DEHEMS: creating a digital environment for large-scale energy management at homes," *IEEE Trans. Consumer Electronics*, vol. 59, no. 1, pp. 62–69, Feb. 2013.
[17] J. Park, I. Han, J. Kwon, J. Hwang, and H. Kim, "Development of a residential gateway and a service server for home automation," in *Lecture Notes in Computer Science, Vol. 2402*, 2002, pp. 137–149.
[18] I. Choi, J. Lee, and S.-H. Hong, "Implementation and evaluation of the apparatus for intelligent energy management to apply to the smart grid at home," in *Proc. 2011 IEEE International Instrumentation and Measurement Technology Conf.*, pp. 1–5.
[19] F. Viani, F. Robol, A. Polo, P. Rocca, G. Oliveri, A. Massa, "Wireless Architectures for Heterogeneous Sensing in Smart Home Applications: Concepts and Real Implementation," *Proceedings of the IEEE*, vol.101, no.11, pp.2381-2396, Nov. 2013.
[20] C. Borean, A. Ricci, G. Merlonchi, "Energy@home: a "User-Centric" Energy Management System", in *Metering International,* no. 3, pp. 52-56, 2011.
[21] F. Salvadori, C.S. Gehrke, A.C. de Oliveira, M. de Campos, P.S. Sausen, "Smart Grid Infrastructure Using a Hybrid Network Architecture," *IEEE Trans. Smart Grid*, vol.4, no.3, pp.1630-1639, Sept. 2013.
[22] N. Morimoto, Y. Fujita, M. Yoshida, H. Yoshimizu, M. Takiyamada, T. Akehi, M. Tanaka, "Smart Outlet Network for Energy-Aware Services Utilizing Various Sensor Information," in *Proc. 2013 27th International Conference on WAINA*, pp.1630-1635.
[23] B. Becker, A. Kellerer, and H. Schmeck, "User interaction interface for Energy Management in Smart Homes," in *Proc. 2012 IEEE PES Innovative Smart Grid Technologies (ISGT) Conf.*, pp. 1–8.
[24] E. Spanò, S. Di Pascoli, G. Iannaccone, "An Intragrid implementation embedded in an Internet of Things platform", in *Proc. 2013 IEEE 18th International Workshop on CAMAD,* Germany, pp. 134-138.
[25] Roy T. Fielding. Architectural styles and the design of network-based software architectures. PhD Thesis, University of California, 2000.
[26] N. Meratnia, P. Havinga, J. Muller, P. Spiess, S. Haller, T. Riedel, C. Decker, and G. Stromberg, "Decentralized enterprise systems: a multiplatform wireless sensor network approach," *IEEE Wireless Communications*, vol. 14, no. 6, pp. 57–66, Dec. 2007.
[27] C. Pastrone, M. A. Spirito, R. Tomasi and F. Rizzo "A Jabber-Based Management Framework for Heterogeneous Sensor Network Applications," *International Journal of Software Engineering and Its Applications*, vol. 2, no. 3, pp. 9–24, 2008.
[28] A. Kansal, S. Nath, J. Liu, and F. Zhao, "SenseWeb: An Infrastructure for Shared Sensing," *IEEE Multimedia*, vol. 14, no. 4, pp. 8–13, Oct. 2007.
[29] S. Lei, W. Xiaoling, X. Hui, Y. Jie, J. Cho, and S. Lee, "Connecting Heterogeneous Sensor Networks with IP Based Wire/Wireless Networks," in *Proc. 2006 The Fourth IEEE Workshop on SEUS-WCCIA*, pp. 127–132.
[30] A. Dunkels "Design and Implementation of the lwIP TCP/IP Stack." Technical Report, *Swedish Institute of Computer Science*, 2001.
[31] G. Iannaccone, "Fast prototyping of network data mining applications," in *Proc.* 2006 *Passive and Active Measurement Conf.*, Adelaide, Australia, pp. 41-50.
[32] L. Peterson, S. Muir, T. Roscoe, A. Klingaman, "PlanetLab architecture: An overview". Tech. Rep. PDN-06-031, PlanetLab Consortium, Apr 2006.
[33] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman and S. Zdonik, "Monitoring streams: a new class of data management applications", in *Proc. 2002 28th international conference on VLDB*, pp. 215-226.
[34] *Tornado Documentation Release 3.1.1*, Nov. 2013, [Online]. Available: www.tornadoweb.org
[35] M.Otto, J.Thornton, *Twitter Bootstrap. (2011)*. [Online]. Available: http://getbootstrap.com/

Elisa Spanò (StM'2013) was born in Polistena (Italy), on September 14, 1981. She received the M.S. degree in electrical engineering from the University of Messina, Italy, in 2006. She is currently working toward the Ph.D. degree at the University of Pisa. Her fields of interest include Wireless Sensor Networks (WSN), Internet of Things (IoT) platforms and smart grid technologies.

Luca Niccolini was born in Empoli (Italy) in 1983, He received his MS and his PhD in Computer Engineering from the University of Pisa in 2008 and 2012, respectively, with a research activity focused on energy efficiency in networked systems. During his PhD, he has been a Research Intern at Intel Research Berkeley and a visiting student at the University of California, Berkeley. In 2012, he joined Riverbed Technology in San Francisco.

Stefano Di Pascoli was born in Cremona (Italy) in 1967. He received the MSEE from the University of Pisa in 1992, and the Ph.D. degree from Scuola Superiore S.Anna, Pisa in 1997. He joined the University of Pisa in 1996, where he is now Associate Professor of Electronics. His research interests include the design of wireless sensors for home automation and bioengineering applications, and embedded systems. He is author or co-author of more than 35 articles.

Giuseppe Iannaccone (M'98–SM'10) is professor of electronics at the University of Pisa. His research interests include electron device modeling, nanoelectronics, analog design, and smart systems. He has authored and coauthored more than 160 papers peer-reviewed journals and 90 papers in proceedings of international conferences. He has coordinated a few European and national research projects. Visit him at http://www.iannaccone.org.