Interfaces for Autarkic Wireless Sensors and Actuators in the Internet of Things

Hubert Zangl, Stephan Muehlbacher-Karrer and Raiyan Hamid

Abstract Wireless devices need to operate without connection to an energy source autarkically see below over a long time. As energy harvesting is limited this also implies that power and energy management are a major concern. Thus, the devices are often extremely reduced in terms of capabiltities and availability. Nevertheless, accessing such devices from the internet should still be easy and hassle-free. We show how the ISO/IEC/IEEE 21450-2010 smart transducer standard can be used for this purpose. In addition, we provide an overview of common concepts that might be used in a similar way. Furthermore, we discuss requirements with respect to various energy modes.

1 Introduction

In the Internet of Things (IoT) many devices covering a huge range of applications from home automation over automotive to industrial applications are interconnected. The scale ranges from a single constrained device up to massive cross-platform deployments of embedded technologies and cloud systems exchanging information in real-time.

Numerous legacy communications protocols exist and numerous emerging communication protocols are developed in order to tie everything together. With respect to standardization, many alliances and coalitions have been created in order to unify the somewhat unstructured organic growth of the IoT landscape. That limit the potential applications of the IoT. The fragmentation between the protocols utilized for communication within and across resource-constrained devices and resource-rich devices is not foreseen to change in the near future [1].

Sensors and actuators will play an important role in the IoT and embedded devices are expected to be dominant in the IoT [2]. Many of these devices may be

H. Zangl (✉) · S. Muehlbacher-Karrer · R. Hamid
Institute of Smart Systems Technologies, Alpen-Adria-Universitaet Klagenfurt, Klagenfurt, Austria
e-mail: hubert.zangl@aau.at

B. George et al. (eds.), *Advanced Interfacing Techniques for Sensors*, Smart Sensors, Measurement and Instrumentation 25,
DOI 10.1007/978-3-319-55369-6_5

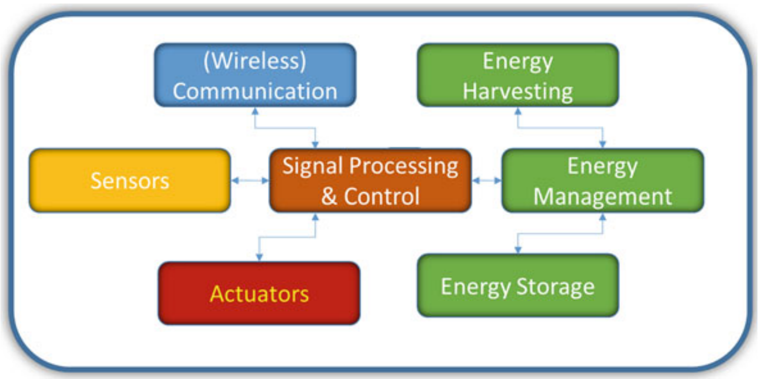


Fig. 1 A typical transducer node architecture consists of several main blocks: a power supply unit, a sensing or actuating subsystem, a processing and storage unit and a communication subsystem

wireless. True wireless sensors and actuators are either battery powered or come with an energy harvesting system. However, if the operation should be over a long time, power and energy considerations are a major concern. From the user perspective the access of the sensors should be as intuitive as possible to provide a high usability. This access will mainly be realized with software interfaces using certain protocols. A typical configuration of an autarkic, i.e. self-powered wireless sensor/actuator device is depicted in Fig. 1.

Even though the general structure is similar, the actual implementation of such a wireless device may be quite different depending on the field of application. For outdoor weather monitoring systems, large photo voltaic cells may be used and the available energy may be quite sufficient to power long range wireless communication using cellular networks. In other situations, e.g. indoor sensors for Heating, Ventilation and Air Conditioning (HVAC) systems, the size of the devices may be limited. Considering photo voltaic energy harvesting, indoor light intensities can also very low, such that we obtain a very energy constraint system. In this contribution we focus on such constrained situations, where energy management becomes a major concern. Such systems will be in sleep mode as much as possible and limit transmission of data as much as possible.

Some typical IoT configurations are shown in Fig. 2. As we are interested in wireless sensors (and actuators) that should operate over a long time or survive solely by energy harvesting, we may prefer designs, where the sensors are not directly accessible using IPv4 or IPv6 protocols. Instead, we use a gateway (no power constrains) that can be accessed directly from the internet. The communication between the gateway and the sensor does not have to be Internet Protocol (IP) based (yet it could). Gateway and server may be the same device (as in example (b)) or may be separated devices that would then also communicate using the IPv4/IPv6 network.

This chapter is structured as follows: In Sect. 2 we discuss energy management and energy saving mechanisms, in Sect. 3 we analyse requirements with respect to

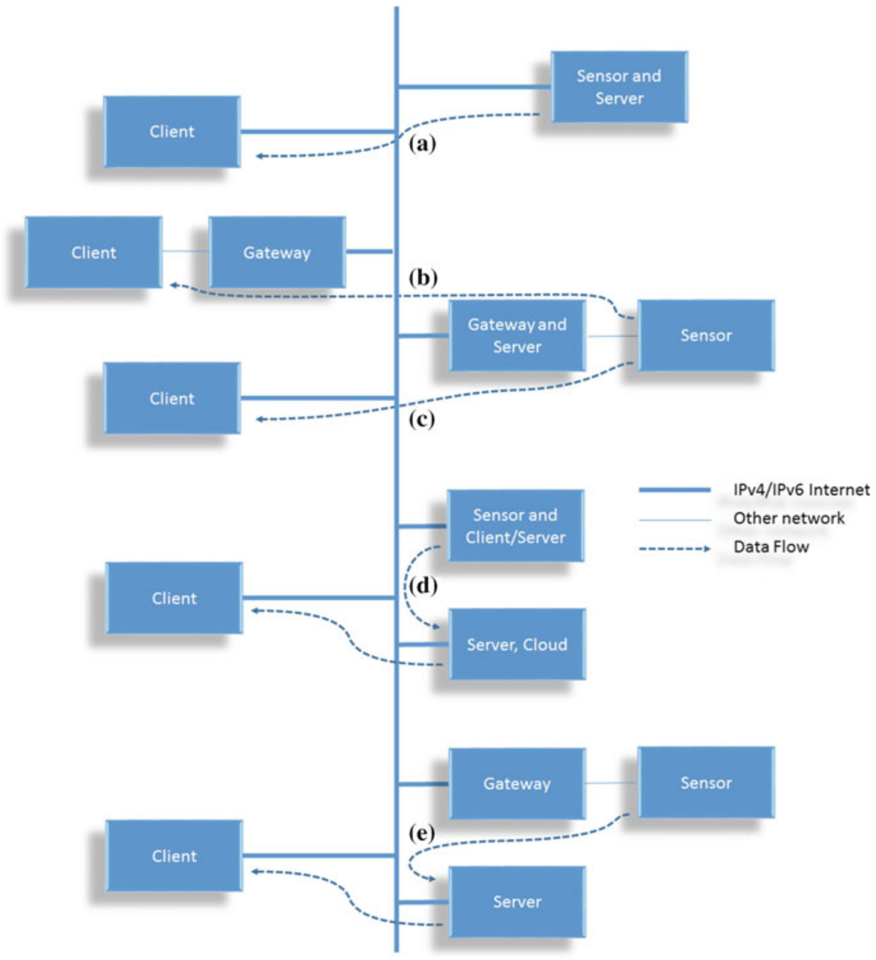


Fig. 2 Examples (non-exhaustive) for the communication between a sensor ("data source") and a client ("data sink") in the IoT. **a** Both client and sensor are linked to the IPv4/IPv6 internet, the sensor directly acts as a server. **b** Both client an sensor do not directly use IP but are linked to the internet by means of gateways. **c** As in (**b**) but here the client connects directly using IPv4/IPv6. **d** Both sensor and client communicate directly using IPv4/IPv6. However, an additional server, e.g. a cloud service or a message broker, relays the information. The server may actually store and preprocess the data of many sensors. As the processing is moved to a powerful server, the complexity of the sensor node can be reduced. **e** Same as (**d**) but with an additional gateway

autarkic wireless sensors and actuators, in Sect. 4 we provide an overview of current approaches used in the Internet of Things. Finally, in Sects. 5–7 we specifically discuss the use of ISO/IEC/IEEE 21450-2010 for autarkic wireless sensors.

2 Energy Management of Wireless Sensors

Energy management is essential for autarkic sensors, i.e. sensors that are powered solely by energy they can harvest from the environment and do not require exchange or recharge of batteries. For such devices, the maintenance effort is low, which is mandatory considering that the number of such devices is constantly increasing. Figure 3 sketches the power consumption for different subtasks of a typical autarkic wireless sensor as shown in Fig. 1. In order to determine the actual energy requirements, it is also important to consider the relative active times, as exemplarily illustrated in Fig. 4 [3].

Basically, according to these illustrations, reducing the average power consumption of a wireless transducer can only be achieved by two mechanisms:

1. Be in sleep mode as long and as often as possible (Reduction of active time).
2. Use low power modes for each task as much as possible.

This not only holds for autarkic sensors but can also be extended to autarkic actuators. However, actuation typically comes with very high power consumption and can thus only be active for very short periods. It is also possible that modules offer different power levels, e.g. transmitter power (see [4]), receiver sensitivity, accuracy of the acquisition system and strength of an actuation may be adjustable. In addition, power consumption can also be reduced on the receiver side by low power listening and clear channel assessments as implemented, e.g. in the B-Mac protocol [5].

The two mechanisms described above do come at a price. The reduction of the active time may lead to lower measurement rates and increased latency. Low power modes for transmitter and receiver may lead to shorter communication ranges, lower

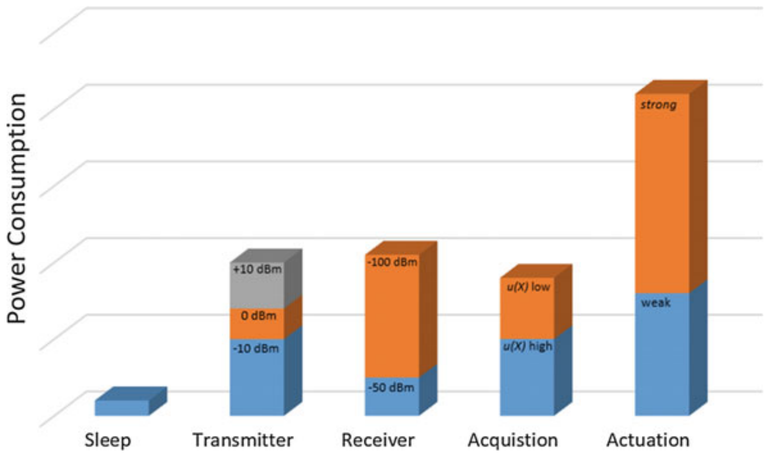


Fig. 3 Example of the power consumption of modules within a wireless transducer. The bar's annotations correspond to examples for transmission power (*transmitter*), receiver sensitivity (*receiver*), measurement uncertainty (*acquisition*) and actuation intensity (*actuation*)

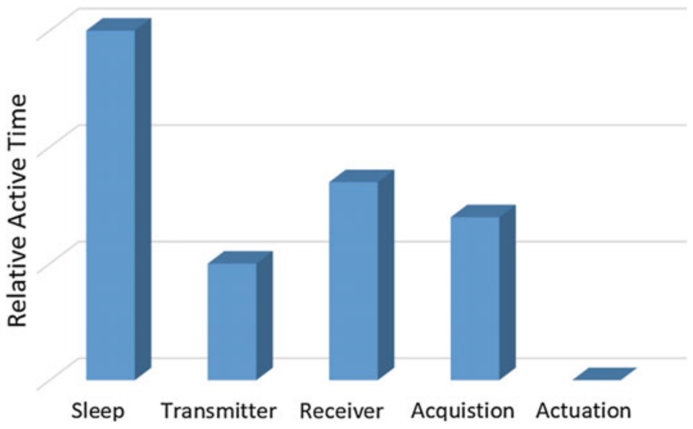


Fig. 4 Example of active time of different modules of a wireless transducer. As transmission is triggered by the transducer itself, it is only in active mode as long as needed. In contrast, if bi-directional, asynchronous communication the receiver needs to be in active mode for much longer times than for actual data transmission. The active time of sensors and actuators varies of large scales depending on the application and implementation but may be significant

data rates, higher probability of packet loss and higher latency. Low power modes for data acquisition modules may lead to higher uncertainty. Lowering power for an actuator may lead to slow responses or even to the failure to perform a desired action. For practical applications it is therefore important that the concepts are comparatively simple [6]. This has to be taken into account for the definition of the transducer interface.

If we use a networked sensor, the user (or client) should somehow be notified of these energy related properties of the device. While classically, only the state of charge of the battery is reported, there are many more aspects that could be considered. A universal sensor interface should provide mechanisms that this information can be conveyed.

As mentioned above, one of the most common means to reduce the average power consumption of wireless sensors is duty-cycling, i.e. to reduce the active time as much as possible, e.g. by a reduction of the measurement rate. In the following we present two other examples how reduction of average power consumption of wireless sensors can be reduced. These can be applied, e.g. when a very accurate wireless sensor is used in an application where high accuracy is not needed at all or at least only under certain circumstances. Then we may save energy and thus extend operation time by reducing the measurement accuracy.

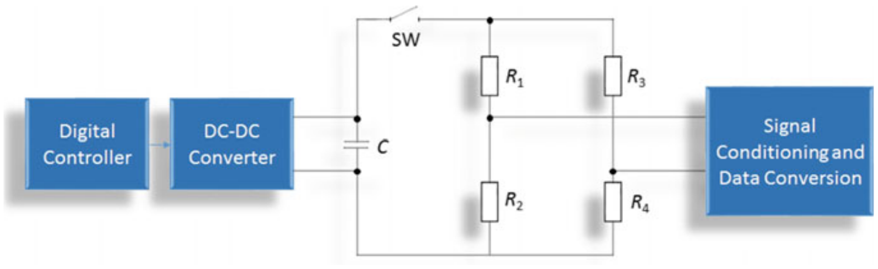


Fig. 5 Measurement block incorporating the bridge measurement circuitry (R_1, R_2, R_3 and R_4), a signal conditioning and data conversation block and a DC-DC converter which can be powered on and off by a digital controller. In addition, a buffer capacitor C and a switch SW is used to turn on and off the measurement bridge to reduce the power consumption

2.1 Energy Efficient Acquisition: Trading Energy Versus Accuracy

In this Section we illustrate by means of a simple example that different modes of the data acquisition submodule can be beneficial. Basically, we are trading energy and power consumption versus accuracy [3].

The concept is exemplarily depicted in Fig. 5 for a bridge measurement circuitry. Bridge resistors, e.g. for strain gauges or pressure sensors, typically have comparatively low impedances in the order of $k\Omega$.

In order to achieve reasonable Signal to Noise Ratios (SNRs) it is thus necessary to use substantial currents. While the SNR increases with the current in the resistors, so does the power consumption

$$P = I_t^2 * R_t \tag{1}$$

where I_t represents the total current and R_t the total resistance. Reducing the current by a factor of 10 will decrease the SNR by 20 dB but also the power consumption is reduced to, actually by a factor of 100.

Calculating the power consumption from the resistors and the auxiliary bridge voltage we obtain

$$P = \left(\frac{1}{R_1 + R_2} + \frac{1}{R_3 + R_4}\right) * V_{in}^2 \tag{2}$$

This means that the power consumption is nonlinear in the voltage. If we assume an adjustable voltage, we obtain a nonlinear relation between the parameter voltage and power consumption. However, in order to predict the power consumption for a certain choice of the voltage, this nonlinear relation should be provided in an electronic data sheet of the device. Additionally, it should be described how the choice of the voltage affects the uncertainty of the measurement result, which can easily be determined for changing parameters using software tools [7]. An approach for such a

Table 1 Example for power consumption versus combined standard uncertainty for a bridge circuit as shown in Fig. 5 [3]. With less power used for the bridge supply voltage, the standard uncertainty of the measurement result for the unknown bridge resistor R_1 increases. However, when the higher uncertainty can be accepted in the application, significant power savings can be achieved

Supply voltage V_{in}	Standard uncertainty $u(R_1)$	Power consumption P
V	mΩ	mW
0.1	400.3	0.01
0.8	52.9	0.6
1.6	30.4	2.6
3.3	21.1	11

description based on ISO/IEC/IEEE 21450-2010 is provided in Sect. 5 [3]. Example data is provided in Table 1.

2.2 *Energy Efficient Data Transmission: Trading Energy Versus Accuracy*

Communication has often a major share of the total energy consumption of a wireless sensor. Consequently, reducing the number of transmissions can be a useful approach. Actually, the costs to transmit a single byte or a small payload may not be that much different, as we typically have a minimum costs due to start up period, synchronization, collision avoidance etc. So up to a certain packet size, the costs of a transmission do not significantly increase. Depending on the permitted latency, we may make use of a buffered transmission, i.e. collect several measurements and transmit it at once.

This idea can be extended if we can compress the data. Basically, any lossless data compression method could be used. However, in a resource constrained device, the algorithms must remain simple. A simple yet lossy approach is to only transmit data when a significant change has occurred or the elapsed time since the last transmission would otherwise exceed the maximum permitted latency time (this ensures a heartbeat signal). Again, it is necessary that the client and the sensor can exchange information, i.e. on the one hand what changes are significant for the client and on the other hand how much energy can be saved on the sensor side.

The concept is illustrated in Fig. 6. Note that data reduction not only helps to save power but also helps to significantly reduce the load in the communication channel. Similar approaches have been used in many applications ranging from data compression (e.g. [8]) to process control (e.g. [9]). Another (in terms of power consumption) similar approach suggested in [10] could also be treated in a similar way.

It should be noted that triggered transducer configuration may also have an influence on the acquisition system. For instance, the trigger may be implemented in the

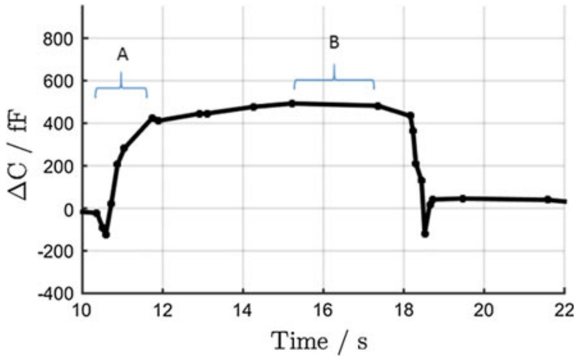


Fig. 6 Example for a simple data reduction approach [11]. Here, a capacitive hand detection sensor for a steering wheel only transmits data when the signal changes significantly (the level for significance can be adjusted) or the maximum latency (2 s) has expired. Consequently, during the transition of the signal (*region "A"*) many samples are transmitted, whereas in the steady phase ("B") only a heartbeat signal is transmitted. In comparison to the lossless compression in [12] (where every signal change is compressed and transmitted) this approach discards data which does not contain new information to the receiver. However, there is no information loss due to the fact that the receiver knows that the signal stays at a certain level as long as it does not get a new package from the transmitter. This approach trades energy savings against an increase of the uncertainty of the data

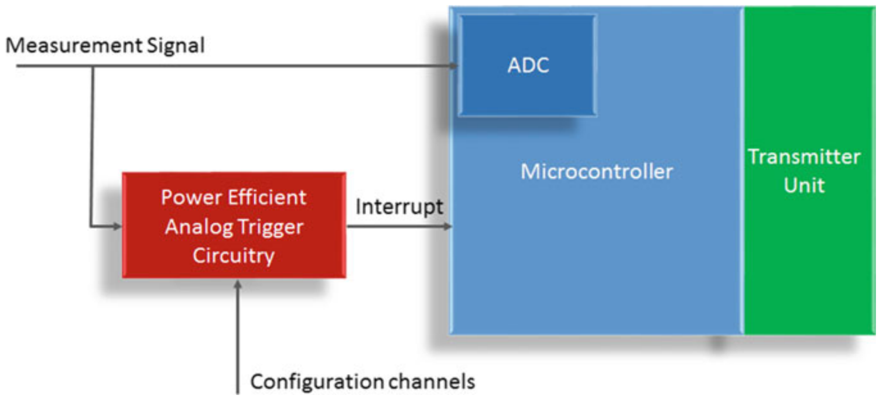


Fig. 7 Sketch of a trigger circuitry. The microcontroller including an ADC and transmitter unit are in sleep mode as long as the measurement signal is within a predefined range of 2 Δ (see Fig. 6). If the measurement signal is not in the range, the analog trigger circuitry fires a wake up interrupt to the microcontroller and the measurement data is transmitted via the transmitter unit. The configuration channels of the analog trigger circuitry are used to set the value of Δ

analog domain with highly power efficient analog comparators and Digital to Analog Converters (DACs). Such a configuration is shown in Fig. 7. Note that we can apply the same strategy as described in Sect. 2.1. Two embedded actuator channels may be used to assign the limits for the change identification. The power consumption for the acquisition module will now be different for the triggered and the non-triggered mode.

3 Requirements on Protocols and Standards with Respect to Autarkic Wireless Sensors

Our aim is to provide quick and easy access to autarkic wireless devices. This implies:

- No specific software should be required to access sensors and actuators, read and understand sensor data, write actuator data and set configurations.
- Sensor responses should be human readable and also machine readable.
- All necessary information (i.e. the electronic datasheet) should be stored within the devices, not just references to some server. This allows that such local devices can be used and configured without the need to have direct access to the internet. Additionally, no long term support (hosting of information) from the manufacturer or a vendor is needed.
- Energy and power related aspects should be covered, i.e. different situation dependent configurations (with varying performance and energy requirements) should be possible. Standardized descriptions of the capabilities should be provided.

Analyzing above requirements, we find that the ISO/IEC/IEEE 21450-2010 [13] standard covers several of these. It provides a Hypertext Transfer Protocol (HTTP) Application Programming Interface (API) that allows to access sensors solely by a web browser. It defines Transducer Electronic Datasheets (TEDS), suitable to be stored in the device itself. With different data formats provided (html, xml and text) the responses are both human and machine readable. Energy and power related aspects of wireless sensors are not been directly addressed in the standard but can be implemented, e.g. in the manufacturer specific part of the TEDS as proposed in [3]. More details on ISO/IEC/IEEE 21450-2010 are discussed in Sect. 5.

4 Overview of Current IoT Approaches

In the following we provide a brief overview of related protocols and standards, with a focus on sensors.

4.1 Infrastructure Internet

In Fig. 2 the IPv4/IPv6 internet connects all the devices. We consider this network as the backbone of the IoT. We refer to the "Internet" as the global network, where all connected devices could (if not prevented by firewalls) communicate with each other using IPv4/IPv6 by knowing the address or an associated name.

- *IPv4*: Internet Protocol version 4 (IPv4) is the fourth version of the IP and is described in Internet Engineering Task Force (IETF) publication RFC 791 (September 1981) [14]. As of 2016 it still routes most internet traffic today despite the ongoing deployment of a successor protocol, IPv6 [15]. IPv4 is a connectionless protocol for packet-switched networking on the network layer according to the Open Systems Interconnect (OSI) model [16]. The protocol does not ensure packet delivery, correct packet sequencing and packets may be duplicated. If needed, these aspects have to be handled on next upper transportation layer and their protocols, e.g. TCP, UDP etc.
  IPv4 uses a 32-bit address space and is thus theoretical limited to 4.29×10^9 addresses, which has been considered insufficient for the future internet. This has led to the development of IPv6.
- *IPv6*: [17] is intended to replace IPv4 on the OSI network layer. Besides other technical benefits, a main advantage is the larger address space. IPv6 uses 128-bits for device addressing, usually represented as eight groups of four hexadecimal digits, separated by colons, e.g. 2001:0db8:0000:0042:0000:8a2e:0370:7334. With about 3.4×10^{38} addresses it is believed that this will be sufficient for the future internet, even though not all of the addresses can be used as some of them are reserved for special use. Google statistics report that as of June 2016 about 12% of accesses to their servers are over IPv6.
- *6LoWPAN*: is an acronym of IPv6 over Low power Wireless Personal Area Networks. 6LoWPAN is an open standard defined in RFC 6282 [18] by the IETF. 6LoWPAN introduces an adaptation layer to be able to transmit IPv6 datagrams over IEEE 802.15.4-Based wireless networks [19]. It allows IPv6 packets to be transported within small link layer frames as those defined by IEEE 802.15.4. It has also been adapted and used over a variety of other networking media such as sub-1 GHz low power Wi-Fi, Bluetooth Smart, and Power Line Control (PLC) [20].
- *TCP*: The Transmission Control Protocol (TCP) provides reliable, ordered, and error-checked delivery of a stream between applications running on hosts communicating over an IP network [21] on the transportation layer. In order to achieve this, it uses acknowledgment and retransmission of lost packets.
- *UDP*: In contrast to TCP, the User Datagram Protocol (UDP) [22] is a connectionless transmission model on the transportation layer. With it, a device can send messages to other hosts using an IP network without the need of a prior connection setup. Thus it is simple, yet it does not guarantee delivery, ordering or duplicate removal. It avoids overhead of acknowledgment and retransmission and is thus favorable, e.g. for real-time or streaming applications, where retransmission of lost packets is not useful.

Table 2 IoT application protocols (adapted from [23])

	REST	Transport	Publish/ Subscribe	Request/ Response	Security	QoS
HTTP	x	TCP		x	SSL	
CoAP	x	UDP	x	x	DTLS	x
MQTT		TCP	x		SSL	x
MQTT-SN			x		SSL	x
AMQP		TCP	x		SSL	x
XMPP		TCP	x	x	SSL	x
DDS		TCP/UDP	x		SSL/DTLS	x
STOMP		TCP	x		SSL	

4.2 *Data Protocols*

While IPv4/IPv6 in combination with TCP or UDP ensure the data transmission between two end points (transportation-oriented), it is also necessary to define how this data communication is used, i.e. how requests and responses are described and how the data is transported (application-oriented). Table 2 provides a comparison of a selection of common protocols. Besides Message Queue Telemetry Transport for Sensor Networks (MQTT-SN) these have in common that they are based on TCP or UDP and thus using the IPv4/IPv6 internet. Most protocols provide a publish/subscribe approach, HTTP only provides a request/response approach. Security is provided using the standard Secure Socket Layer (SSL) or Datagram Transport Layer Security (DTLS) protocols.

- *REST HTTP*: REST stands for Representational State Transfer [24]. It constraints the client server interaction to be stateless, i.e. each request from client to server must contain all of the information necessary to understand the request by the server. The server does not store and context, states are entirely kept by the client. REST is usually used with HTTP and provides a simple approach to use HTTP request to read, write or delete data.
- *CoAP*: The Constrained Application Protocol (CoAP) [25] is an application layer protocol that is intended for use in resource-constrained internet devices, such as Wireless Sensor Networks (WSN) nodes. It realizes a subset of REST common with HTTP but optimized for Machine-to-Machine (M2M) applications. CoAP can be used for refashioning simple HTTP interfaces into a more compact protocol. It also offers features for M2M such as built-in discovery, multicast support, and asynchronous message exchanges.
- *MQTT*: It is a lightweight publish-subscribe-based messaging protocol for M2M communication on top of the TCP/IP protocol [26]. It requires a message broker

such as *Mosquito* [27], which is responsible for distributing messages to interested clients (compare Fig. 2d).

- *MQTT-SN*: It is a lightweight publish/subscribe middleware specifically designed for embedded devices on non-TCP/IP networks, such as Zigbee. MQTT-SN is close to MQTT, but redesigned to deal with the requirements of WSN such as high link failures, low bandwidth, etc. and resource limited devices in terms of energy, processing power, memory, etc. [28] (compare Fig. 2e).
- *XMPP*: Extensible Messaging and Presence Protocol represents an open technology for real-time communication, which aims to power a wide range of applications including instant messaging, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware, content syndication, and generalized routing of XML data. The core specifications for XMPP are developed at the IETF [29]. An extension of XMPP specifically addresses sensor data in the IoT [30]. As of 2016, the status of the extension is "experimental".
- *DDS*: Data-Distribution Service for Real-Time Systems represents a middleware standard developed by the Object Management Group (OMG). It directly addresses data centric publish-subscribe communications for real-time and embedded systems [31]. In particular it provides control of Quality of Service (QoS) parameters, including reliability, bandwidth, delivery deadlines, and resource limits. It uses a background discovery protocol to automatically find data. DDS systems are typically more contained, i.e. not spread across Wide-Area Network (WAN).
- *AMQP*: Advanced Messaging Queuing Protocol is an open internet protocol for business messaging [32], often used for server to server communication, yet it also finds applications in the IoT. Originating from banking industry, a main focus is on not losing messages, regardless of failures or reboots.
- *STOMP*: Simple (or Streaming) Text Oriented Message Protocol [33] is a simple text based interoperable protocol designed for asynchronous message passing between clients via mediating servers (brokers), similar to MQTT. STOMP is a frame based and assumes a reliable 2-way streaming network protocol (such as TCP) underneath. Being text-based, it is possible to directly communicate with a server, e.g. using a telnet connection.

Beside this selection of common protocols numerous other protocols have been developed and all of them come with certain advantages and disadvantages. The best choice will depend on the number of devices, required response times, etc. Servers often support multiple protocols so that clients using different protocols can access the same data.

4.3 Semantics

Semantics in this context means that sensors are described in a standardized, typically machine-interpretable representation. As much of the IoT potentials is due to the capabilities of low-cost and energy-efficient sensors (and actuators) with mature wireless communication capacities and the interests in integrating the physical into the cyber worlds such semantics can help to handle the heterogeneity of things and to infer new knowledge together with other intelligent processing techniques [34]. Additionally, if the relation between performance and power consumption can be included in such a device description, an intelligent adaptive energy management becomes possible (compare Sect. 2).

Besides TEDS defined in [13] (compare Sect. 7), many other approaches to describe sensors have been suggested. In the following we briefly describe some prominent examples.

- *SensorML*: It provides standard models and an XML encoding for describing sensors and measurement processes [35]. It originates from geospatial sensing and thus includes location information but it is not limited to such applications. Definitions of metrological terms are partially different compared to the field of metrology as defined in [36, 37].
- *EDDL*: The Electronic Device Description Language is used in several industrial standards. An Electronic Device description (EDD) based on EDDL is usually provided by the manufacturer of a device. It is not stored in the device itself and it needs an interpreter to be executed. It is rather a programming language, e.g. for user interfaces in the Industrial Internet of Things (IIoT) and Industry 4.0 [38].
- *Semantic Sensor Net Ontology—W3C*: This ontology describes sensors and observations they make of the physical world using definitions of classes and properties (e.g. measurement range, latency, attached system) [39]. With respect to the 'quality of results' is uses the term measurement accuracy, which [36] defines as non-quantitative.
- *Wolfram Language*: Connected Devices provide symbolic representations of devices [40]. The devices can be accessed with a standard set of Wolfram Language functions like DeviceRead, DeviceExecute, DeviceReadBuffer and DeviceReadTimeSeries. It has a large number of devices in the database, yet not all device descriptions are very detailed. Currently, the support with, e.g. respect to measurement uncertainty appears limited.

4.4 Power Efficient Lower Layer Wireless Sensor Protocols

Numerous wireless technology standards such as WIFI, NFC, ANT, IEEE 802.15.4, ZigBee, WirelessHART, Bluetooth SMART, LoRaWAN etc. as well as proprietary solutions are used for the lower layer of wireless communication between sensors and the internet. From the perspective of the sensor interface as seen from the internet, it should not make a difference how the data is actually transferred. Consequently, all these technologies allow to cope with the requirements given in Sect. 3. The choice for a protocol will mainly depend on the requirements with respect to range, data rate, routing and multi-hop capabilities and of course power consumption.

4.5 Security

Security is a crucial topic for any communication network. In our approach we have three aspects:

- Security of the internet communication. This can be achieved by the usual means that are used for internet traffic.
- Security of the wireless link. This has to be addressed by the wireless communication protocol and depends on the respective standards.
- Security of the server.

The server in an architecture, where it, e.g. acts as a message broker may not just relay the information provided by the sensor but may also perform some processing, e.g. calibration or archiving for historical data access. Consequently, in this situation it must be able to read the data. Therefore, an approach using, e.g. message broker can not provide an end-to-end encryption.

As security is one of the major topics for IoT it actually is a topic on its own. A survey of existing protocols and open research issues can be found, e.g. [41].

5 Using ISO/IEC/IEEE 21450-2010 with Autarkic Wireless Sensors

The ISO/IEC/IEEE 21450-2010 [13] (prepared as IEEE 1451.0-2007) is part of a set of smart transducer interface standards developed by the Institute of Electrical and Electronics Engineers (IEEE) Instrumentation and Measurement Society's Sensor Technology Technical Committee describing a set of open, common, network-independent communication interfaces for connecting transducers (sensors or actuators) to microprocessors, instrumentation systems, and control/field networks. To go beyond the previous definitions, an ISO/IEC/IEEE 21450-2010 smart transducer is defined as a smart transducer that provides functions beyond those necessary for generating a correct representation of a sensed or controlled quantity. This functionality allows for simplifying the integration of the transducers into applications in a networked environment. One of the key elements of these standards is the definition of TEDS for each transducer. The TEDS is a memory *inside* the transducer, which stores transducer identification, sensor and actuator channel descriptions, optional calibration data, etc. The goal of this family of standards is to allow the access of transducer data through a common set of interfaces whether the transducers are connected to systems or networks via a wired or wireless means. This means ISO/IEC/IEEE 21450-2010 smart transducers have capabilities for self-identification, self-description, self-diagnosis, self-calibration, location-awareness, time-awareness, data processing, reasoning, data fusion, alert notification (report signal), standard-based data formats, and communication protocols [42].

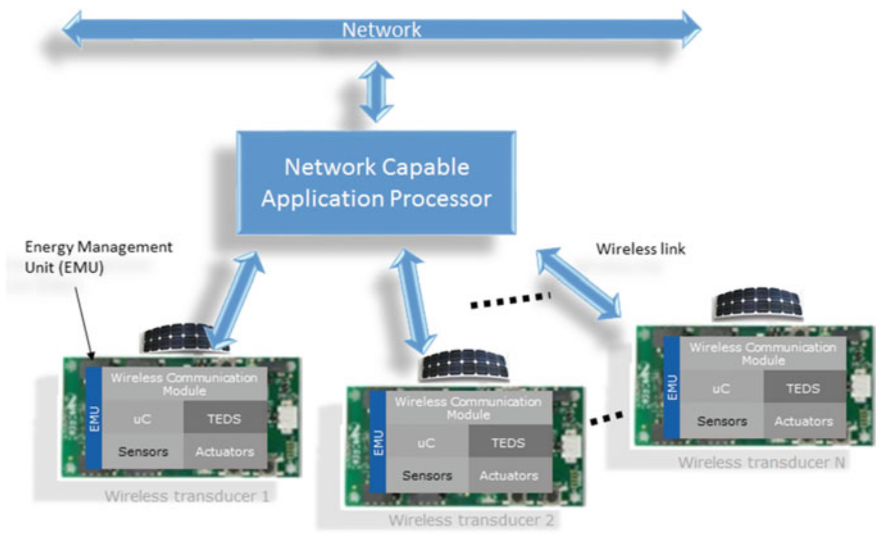


Fig. 8 Architecture in style of IEEE 1451-5. The network capable application processor, which is accessible from the network (e.g. IPv4) acts as an interface to the autarkic wireless devices. This corresponds to architecture (**c**) in Fig. 2. Energy and power management is important for both, the wireless link and the other components inside the transducer

Using ISO/IEC/IEEE 21450-2010 with wireless sensors, a so called Network Capable Application Processor (NCAP) acts as a wireless base station and connects the transducer to a network. Energy management of NCAPs is not considered here, as we assume that these devices are not limited in terms of power or energy. Therefore, the usual architecture is that the transducers connect in a star-like topology to a NCAP. Such topologies are fairly common, e.g. in wireless communication in vehicles or aircraft and also in industry [43]. Distances are usually short and the main backbone of the system is wired (at least for the power supply), e.g. based on Ethernet. An example configuration is illustrated in Fig. 8.

In this approach the NCAP acts both as a server and a gateway. Even when using 6LoWPAN for communication between the NCAP and the server, the sensors would not directly be accessed by their IPv6 address but only through the NCAP. From the client point of view it does not matter which wireless communication is used. It may follow one of the IEEE-1451-5 wireless standards such as ZigBee of Bluetooth or may make us of proprietary protocols of manufacturers or researchers. The different behaviour of the communication interface might find an abstraction in the PHY-TEDS, which is a mandatory part of the TEDS.

As energy management is crucial for wireless devices, a standard on smart transducers should also consider this aspect. A TEDS may provide information on average power consumption in different operation modes. The idea is to simply provide the information on how much power each mode requires, no matter how it is implemented. Therefore, we are not looking into the implementation details of each protocol but look at it from the point of view of the application. Besides the measurement and update rate, the latency is an important factor. At the end, probability of failure is most important. The most common cause for a battery powered wireless device to fail is simply that it runs out of energy. This can be avoided as the system that utilizes a transducer has a reasonable prediction of the remaining lifetime of the device. In case that we run out of energy, the system can safely go into a safe system state.

6 ISO/IEC/IEEE 21450-2010 HTTP API

Besides the standard transducer service API and the module communications API, which may be used in programming languages such as Java or C++, the ISO/IEC/IEEE 21450-2010 standard also defines HTTP APIs corresponding to the transducer service API. With HTTP the request/response approach can be used. However, callbacks and a publish/subscribe approach are not supported in this API. The server may directly be implemented on the NCAP (compare Fig. 8) or may run on a separate device. In the following we assume that the NCAP also hosts the HTTP server.

The methods provided in the API can be classified into four groups:

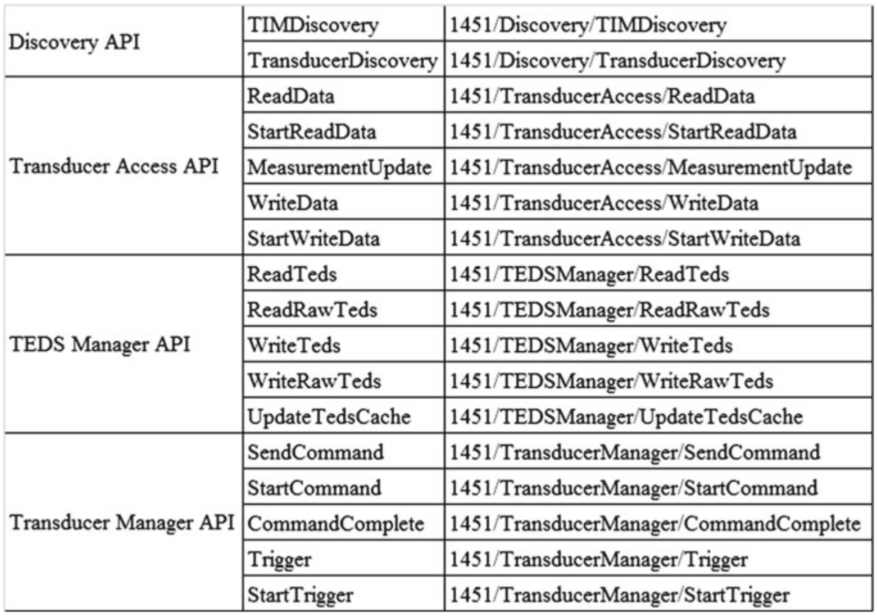
- Discovery: Methods for applications to discover available Tranducer Interface Modules (TIM) (i.e. wireless sensor nodes) and TransducerChannels organized in this interface.
- TransducerAccess: Access to sensor and actuator TransducerChannels will be by use of methods on this interface.
- TransducerManager: Applications that need more control over TIM access can use methods on this interface. An example are locks on TIMs for exclusive use.
- TEDSManager: Applications can use methods on this interface to read (and write) TEDS.

The HTTP APIs focuses mainly on accessing transducer data and TEDS using the HTTP 1.1 protocol. Users can send a HTTP request (see Table 3) to a server on the NCAP and get a response in the following way:

(a) A user or client sends a HTTP request to the HTTP server on the NCAP.
(b) The HTTP server on the NCAP receives the HTTP request, processes it, communicates with the transducers if necessary and gets the corresponding results.
(c) The HTTP server returns the corresponding HTTP response to the user.

In the standard it is suggested that internally in the NCAP the standard APIs of ISO/IEC/IEEE 21450-2010 should be used between the HTTP server and the NCAP. However, as this can not be seen from the client side it is also be possible that the HTTP server directly processes the request and communicates with the wireless devices. In our implementation we actually follow this approach.

Table 3 Commands in the HTTP API



Discovery API	TIMDiscovery	1451/Discovery/TIMDiscovery
	TransducerDiscovery	1451/Discovery/TransducerDiscovery
Transducer Access API	ReadData	1451/TransducerAccess/ReadData
	StartReadData	1451/TransducerAccess/StartReadData
	MeasurementUpdate	1451/TransducerAccess/MeasurementUpdate
	WriteData	1451/TransducerAccess/WriteData
	StartWriteData	1451/TransducerAccess/StartWriteData
TEDS Manager API	ReadTeds	1451/TEDSManager/ReadTeds
	ReadRawTeds	1451/TEDSManager/ReadRawTeds
	WriteTeds	1451/TEDSManager/WriteTeds
	WriteRawTeds	1451/TEDSManager/WriteRawTeds
	UpdateTedsCache	1451/TEDSManager/UpdateTedsCache
Transducer Manager API	SendCommand	1451/TransducerManager/SendCommand
	StartCommand	1451/TransducerManager/StartCommand
	CommandComplete	1451/TransducerManager/CommandComplete
	Trigger	1451/TransducerManager/Trigger
	StartTrigger	1451/TransducerManager/StartTrigger

A HTTP request can be made using any web browser by typing a command with the following syntax in the address field:

`http://<host>:<port>/<path>?<parameters>`

`<host>` represents the domain name or internet address of the NCAP, e.g. for IPv4 this could be "191.168.0.101". `<port>` is optional and only needed if the HTTP server on the NCAP does not use the standard port (80) for HTTP servers. `<path>` indicates the ISO/IEC/IEEE 21450-2010 path including the command (e.g. "1451/TransducerAccess/ReadData"), parameters associated with the command are passed using `<parameters>` e.g.

`timId=1\&channelId=2\&timeout=14\&samplingMode=`
`continuous\&format=xml`).

The available commands are provided in Table 3.

Using the format parameter, it can be selected if the response should come as plain text, HTML or XML. An example schema for a XML response on a ReadData request is shown in Table 4.

Table 4 Example schema for a response of the HTTP server in XML format (command: ReadData)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:stml=http://grouper.ieee.org/groups/1451/0/1451HTTPAPI
<xs:complexType name="ReadDataHTTPResponse">
<xs:sequence>
<xs:element name="errorCode" type="stml:UInt16"/>
<xs:element name="timId" type="stml:UInt16"/>
<xs:element name="channelId" type="stml:UInt16"/>
<xs:element name="transducerData" type="stml:ArgumentArrayType"/>
</xs:sequence>
</xs:complexType>
</xs:schema>
```

Table 5 General format of ISO/IEC/IEEE 21450-2010 TEDS

Field	Description	Type
-	TEDS length	UInt32
1 to N	Data block	Variable
-	Checksum	UInt16

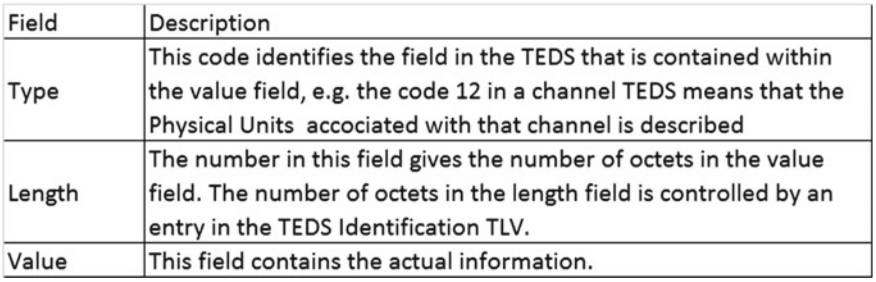
7 TEDS Structure and Energy Related Extensions

In contrast to most description languages described in Sect. 4.3, which are text based, the ISO/IEC/IEEE 21450-2010 TEDS represents a comparatively compact binary description. However, with its clear structure, it can easily be translated into other formats, e.g. into XML. There is not just a single TEDS for one transducer. Actually, four TEDS are mandatory. The Meta-TEDS for all information needed to gain access to any TransducerChannel, plus information common to all TransducerChannels, the TransducerChannel TEDSs for all information concerning the TransducerChannels, the Transducer Name TEDS to provides a place to store the name by which the system or the end user will know this transducer and the PHY TEDS for all information needed to with respect to communication (this depends on the type of communication and is thus not described in ISO/IEC/IEEE 21450-2010).

The general structure of a TEDS is shown in Table 5. Every TEDS starts with the length information, continues with a variable number of data blocks and ends with a check sum.

The individual data blocks are described using a Type/Length/Value approach as illustrated in Table 6. The standard provides a number of predefined types. Some of them, e.g. channel identifiers for physical units are mandatory, others are optional. Furthermore, the manufacturer can define additional types. We use this possibility to define additional types to include energy related information as described in the following. It would be beneficial if future standards would natively include such energy related extensions.

Table 6 Description of fields in ISO/IEC/IEEE 21450-2010 TEDS



Field	Description
Type	This code identifies the field in the TEDS that is contained within the value field, e.g. the code 12 in a channel TEDS means that the Physical Units accociated with that channel is described
Length	The number in this field gives the number of octets in the value field. The number of octets in the length field is controlled by an entry in the TEDS Identification TLV.
Value	This field contains the actual information.

Following the discussions in Sect. 2 the data section of an extended datasheet may start with the number of (energy-)modes, as each mode obtains an individual datasheet. The first section for each mode is then the common data sheet, e.g. according to the current standard or including modifications as suggested, e.g. in [44] for PHY TEDS. This is followed by a parametric description of the average power consumption and the combined standard uncertainty for a transducer channel or the latency for the PHY TEDS, again using newly defined types. A possible general form of a parametric model is given by

$$X = \begin{cases} (c_0 + c_p p)^\lambda, & \lambda \neq 0 \\ ln(c_0 + c_p p), & \lambda = 0 \end{cases} \tag{3}$$

where c_0, c_p and λ are the coefficients provided in the data sheet using additional types. The parameter p is set using an embedded actuator channel, i.e. an actuator channel that is internally used in the transducer. If this channel is set to zero, then no parameter is used. X may stand for average power, latency, or combined standard uncertainty depending on the data sheet. We believe that (3) provides better fits than a second order polynomial approximation, which would require the same number of coefficients. However, both descriptions could be included using corresponding types.

The section as described above repeats for each mode and thus N-times, where N is the number of modes. The mode is also selected by an embedded transducer channel. If the channel is set to zero, than there is only one mode. The total average power consumption can thus be calculated as the sum of the power consumption of all $N_{submodule}$ modules, i.e. the transducers and the communication unit:

$$P_{total} = \sum_{i=1}^{N_{submodule}} P_i \tag{4}$$

An example for an extended TEDS is given in Table 7.

An energy storage device attached to a TIM is treated as a sensor channel, reporting the state of charge of the device, optionally using a calibration TEDS. Whereas the static information about the nominal capacity of the battery is also stored in the

Table 7 Energy management extension for TransducerChannel TEDS [3]

	Field name	Description	Data type	# octets	
	NbrModes	Number of modes	uint8	1	Extension
Mode 1 (selected by dedicated transducer channel)	Original datafields according to IEEE 21451				
	ModeID	ID of the mode	uint8	1	Extension
	ParameterID	ID of the parameter	uint8	1	
	ParamMin	Minimum value of parameter	float32	4	
	ParamMax	Maximum value of parameter	float32	4	
	Cop	Coefficient for power	float32	4	
	Cpp	Coefficient for power	float32	4	
	λp	Coefficient for power	float32	4	
	Cou	Coefficient for uncertainty	float32	4	
	Cpu	Coefficient for uncertainty	float32	4	
	λpu	Coefficient for uncertainty	float32	4	
Mode 2 (slt. by d. trans. ch.)	Original datafields according to IEEE 21451				
	ModeID	ID of the mode	uint8	1	Extension
	⋮	⋮	⋮	⋮	
⋮	⋮				
Mode N	⋮	⋮	⋮	⋮	

TransducerChannel TEDS (self defined type). Furthermore, a harvesting device is also treated as a sensor channel, reporting actual energy harvesting, optionally using a calibration TEDS. Information on maximum harvesting power is also stored in the corresponding TransducerChannel TEDS. Consequently, only minor extensions of the current standard are needed to implement an energy management information exchange. Please note that the approach to configure the TIM by means of transducer channels also permits the use of virtual TEDS and TEDS caching (compare [13]), as the TEDS information can be read only. However, as explained in Sect. 3, TEDS that are directly stored in the devices offers some distinct advantages.

8 Steps for Using an Autarkic Wireless Sensors Example with IEEE 21450-2010 HTTP API

Accessing a sensor will typically involve the following steps:

- *Find the relevant NCAP*: The wireless device will connect to a NCAP nearby, thus we need to know the address of this NCAP. Usually, this will be known from the installation fo the NCAP to the network.
- *Discovery Request*: Obtain the TIM IDs connected to that NCAP.
- *MetaTEDS Request*: Obtain the META TEDS for the TIMs. The meta TEDS contains the information, how many channels a certain TIM provides.
- *Obtain the channel TEDS*: The channel TEDS is a description of the sensor or actuator channel including, e.g. the physical dimension of the quantity being reported and maximum standard uncertainty linked to this measurement.
- *Obtain additional TEDS (optional)*: More information such as calibration data may be required.
- *Configure Device (optional)*: Use, e.g. embedded actuator channels to configure physical channels.
- *Get the Data*: Read from a sensor channel (Write to an actuator channel).

Considering that we just want to get the data from a sensor, it may appear that above procedure is complex. However, the advantage is that we do not just get numeric values but also interpretation of the data and means to adjust the device to the specific needs in an application. Indeed, several steps can be done in an automated fashion, i.e. getting calibration TEDS and carry out calibration in the background. Thus, the practical application is still simple.

9 Summary

In modern systems, sensors and actuators are frequently used in a network and nowadays the network is often directly connected to the Internet. Consequently, the sensors and actuators become parts of the IoT. Therefore, it will be important to have standardized interfaces on how to connect to the devices. Currently, this field is quite fragmented with many different protocols and interfaces in use. An interface for Internet access to autarkic wireless sensors and actuators has to provide more than just access to sensor data and actuator settings. With the huge number of deployed devices it is important to identify the devices and also get an interpretation of the results. Furthermore, energy management is important and should also be accessible by the interface. By this, the remaining battery lifetime or the sufficiency of energy harvesting of a device can be predicted and safe system states can be achieved before a wireless transducer runs out of power. The ISO/IEC/IEEE 21450-2010 HTTP API allows to access sensors and actuators from the Internet using NCAPs and provides

TEDS that can be extended to include information relevant for energy management. Such extension could also be implemented for other protocols and device descriptions.

References

1. A. Al-Fuqaha, A. Khreishah, M. Guizani, A. Rayes, M. Mohammadi, Toward better horizontal integration among iot services. IEEE Commun. Mag. **53**(9), 72–79 (2015)
2. H. Zhou, *The Internet of Things in the Cloud: A Middleware Perspective*, 1st edn. (CRC Press Inc., Boca Raton, FL, USA, 2012)
3. H. Zangl, M. Zine-Zine, S. Mühlbacher-Karrer, TEDS extensions toward energy management of wireless transducers. IEEE Sens. J. 15(5):2587–2594 (2015), http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7009985
4. B. SIG, *Bluetooth Core Specification 4.1*, Bluetooth SIG, 09 (2013)
5. J. Polastre, J. Hill, D. Culler, Versatile low power media access for wireless sensor networks, in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, ser. SenSys '04, (ACM, New York, 2004), pp. 95–107. doi:10.1145/1031495.1031508
6. S. Hussain, D. Gurkan, Management and plug and play of sensor networks using snmp. IEEE Trans. Instrum. Meas. 60(5):1830–1837 (2011), http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5740597
7. H. Zangl, K. Hoermaier, Educational aspects of uncertainty calculation with software tools. Measurement (2015), http://www.sciencedirect.com/science/article/pii/S0263224115005916
8. E. Bristol, Swinging door trending: adaptive trend recording, in *ISA National Conference Proceedings* (1990), pp. 749–753
9. J. Lange, F. Iwanitz, T. Burke, *OPC? From Data Access to Unified Architecture*, 4th edn. VDE VERLAG GMBH (2010)
10. G. Monte, V. Huang, P. Liscovsky, D. Marasco, Standard of things, first step: understanding and normalizing sensor signals, in *39th Annual Conference of the IEEE Industrial Electronics Society, IECON 2013* (2013), pp. 118–123, http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6699121
11. S. Mühlbacher-Karrer, L.-M. Faller, R. Hamid, H. Zangl, A wireless steering wheel gripping sensor for hands on/off detection, in *2016 IEEE Sensors Applications Symposium (SAS)*, Apr 2016, pp. 1–5
12. F. Marcelloni, M. Vecchio, A simple algorithm for data compression in wireless sensor networks. IEEE Commun. Lett. **12**(6), 411–413 (2008)
13. ISO/IEC/IEEE information technology—smart transducer interface for sensors and actuators—common functions, communication protocols, and transducer electronic data sheet (TEDS) formats, *ISO/IEC/IEEE 21450:2010(E)*, pp. 1–350, May 2010
14. *RFC 791 Internet Protocol - DARPA Inernet Programm, Protocol Specification*, Internet Engineering Task Force, Sept 1981, http://tools.ietf.org/html/rfc791
15. https://www.google.com/intl/en/ipv6/statistics.html, Aug 2008
16. I. Standardization, Iso/iec 7498–1: 1994 information technology-open systems interconnection-basic reference model: the basic model. Int. Stand. ISOIEC **74981**, 59 (1996)
17. *RFC 2460 Internet Protocol, Version 6 (IPv6) Specification*, Internet Engineering Task Force, Dec 1998, http://tools.ietf.org/html/rfc2460
18. J. Hui, P. Thubert, Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks, RFC 6282 (Proposed Standard), Internet Engineering Task Force, Sept 2011, http://www.ietf.org/rfc/rfc6282.txt

19. IEEE standard for local and metropolitan area networks—Part 15.4: low-rate wireless personal area networks (lr-WPANs), in *IEEE Std 802.15.4-2011 (Revision of IEEE Std 802.15.4-2006)*, pp. 1–314, Sept 2011
20. J. Olson, 6lowpan demystified, Texas Intruments, Technical Report, 2014, http://www.ti.com/lit/wp/swry013/swry013.pdf
21. V. Cerf, R. Kahn, A protocol for packet network intercommunication. IEEE Trans. Commun. **22**(5), 637–648 (1974)
22. J. Postel, User Datagram Protocol, RFC 768 (INTERNET STANDARD), Internet Engineering Task Force, Aug 1980, http://www.ietf.org/rfc/rfc768.txt
23. A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: a survey on enabling technologies, protocols, and applications. IEEE Commun. Surv. Tutor. **17**(4), 2347–2376 (2015)
24. R.T. Fielding, Architectural styles and the design of network-based software architectures, Ph.D. Dissertation, 2000, aAI9980887
25. Z. Shelby, K. Hartke, C. Bormann, The Constrained Application Protocol (CoAP), RFC 7252 (Proposed Standard), Internet Engineering Task Force, June 2014, updated by RFC 7959, http://www.ietf.org/rfc/rfc7252.txt
26. A. Banks, R. Gupta (eds.), *MQTT Version 3.1.1*, 10 April 2014, http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf
27. *An Open Source MQTT v3.1/v3.1.1 Broker*, Aug 2016
28. A. Stanford-Clark, H.L. Truong, Mqtt for sensor networks (mqtt-sn) protocol specification (2013)
29. P. Saint-Andre, Extensible Messaging and Presence Protocol (XMPP): Core, RFC 6120 (Proposed Standard), Internet Engineering Task Force, Mar 2011, updated by RFC 7590, http://www.ietf.org/rfc/rfc6120.txt
30. P. Waher, XEP-0323: Internet of Things - Sensor Data, XMPP Standards Foundation (XSF), 11 2015, http://xmpp.org/extensions/xep-0323.pdf
31. *Data Distribution Servics (DDS)*, Aug 2016, http://portals.omg.org/dds/
32. I. Standard, ISO/IEC 19464:2014 Information technology – Advanced Message Queuing Protocol (AMQP) v1.0 specification, May 2014
33. *STOMP Protocol Specification, Version 1.2*, accessed Aug 2016, https://stomp.github.io/stomp-specification-1.2.html
34. P.M. Barnaghi, W. Wang, C.A. Henson, K. Taylor, Semantics for the internet of things: early progress and back to the future. Int. J. Semant. Web Inf. Syst. 8(1):1–21 (2012), http://dblp.uni-trier.de/db/journals/ijswis/ijswis8.html
35. http://www.sensorml.com, Aug 2016
36. BIPM, JCGM 200:2012 : International Vocabulary of Metrology - Basic and General Concepts and Associated Terms (VIM 3rd edn)
37. BIPM, JCGM 100:2008: Guide to the expression of uncertainty in measurement
38. www.eddl.org, Aug 2016
39. *Semantic Sensor Network Ontology*, Mar 2016, http://www.w3.org/TR/2016/WD-vocab-ssn-20160531
40. *Wolfram Connected Devices Project*, Mar 2016, http://devices.wolfram.com/
41. J. Granjal, E. Monteiro, J.S. Silva, Security for the internet of things: a survey of existing protocols and open research issues. IEEE Commun. Surv. Tutor. **17**(3), 1294–1312 (2015)
42. E. Song, K. Lee, Understanding IEEE 1451—networked smart transducer interface standard—what is a smart transducer? IEEE Instrum. Meas. Mag. 11(2):11–17 (2008), http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4483728
43. R. Matischek, T. Herndl, C. Grimm, J. Haase, Real-time wireless communication in automotive applications, in *Automation and Test in Europe, DATE* (2011), pp. 1036–1041
44. J. Higuera, J. Polo, IEEE 1451 standard in 6LoWPAN sensor networks using a compact physical-layer transducer electronic datasheet. IEEE Trans. Instrum. Meas. **60**(8), 2751–2758 (2011)