A proposal for an Internet of Things-based monitoring system composed by low capability, open source and open hardware devices

Jesús Rodríguez-Molina, José-Fernán Martínez, Gregorio Rubio, Vicente Hernández
Research Center on Software Technologies and Multimedia Systems for Sustainability (CITSEM—Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad)
Technical University of Madrid (*U.P.M.*)
La Arboleda, Campus Sur *UPM*. Ctra. Valencia, Km 7, 28031. Madrid, Spain

Abstract— **The Internet of Things makes use of a huge disparity of technologies at very different levels that help one to the other to accomplish goals that were previously regarded as unthinkable in terms of ubiquity or scalability. If the Internet of Things is expected to interconnect every day devices or appliances and enable communications between them, a broad range of new services, applications and products can be foreseen. For example, monitoring is a process where sensors have widespread use for measuring environmental parameters (temperature, light, chemical agents, etc.) but obtaining readings at the exact physical point they want to be obtained from, or about the exact wanted parameter can be a clumsy, time-consuming task that is not easily adaptable to new requirements. In order to tackle this challenge, a proposal on a system used to monitor any conceivable environment, which additionally is able to monitor the status of its own components and heal some of the most usual issues of a Wireless Sensor Network, is presented here in detail, covering all the layers that give it shape in terms of devices, communications or services.**

Keywords— Internet of Things; monitoring; application; middleware; architecture.

I. Introduction

The Internet of Things (or the IoT) offers an extensive cluster of possibilities unlike any other previously existing system. The number of functional electronic devices that are present in the world are but ever-increasing at a high rate along with the willingness to interconnect them, thus establishing communications where data ca be shared in an efficient and seamless manner. The number of applications or research projects related with the IoT has boomed, going from Near Field Communications in Peer-to-Peer transactions [1] to applications related with cloud computing [2], to name but a few. A significant amount of proposals revolve around the concepts of either providing services for human users so that their quality of life will be improved or supplying somewhat a sort of machine-to-machine communication (M2M) within a system. Incidentally, it is only natural the state of things turns out to be like that, as Mark Weiser, the forerunner of ubiquitous computing, claimed that machines would end up being so integrated within an environment that they would just recede to the background [3]. Therefore, electronic appliances must communicate one to the other requiring as little user intervention as possible. To accomplish this duty, many systems have been conceived and designed that are somewhat making features from the vision of the IoT (context awareness, ubiquitous computing, always-on connectivity, environment integration, etc.) to become an actual and genuine characteristic of a deployment.

Among the most usual IoT-related applications, the ones involving system monitoring and surveillance are rather common; unfortunately, they often share several flaws: these applications are restricted to a very specific area of usability (e-health, industrial monitoring, agricultural infrastructures, etc.) and if yanked out of it they do not seem to adapt with easiness to any other surroundings. What is more, even if their natural environment remains the same, should any other new service be included as part of a system update, they do not offer enough flexibility to make that service usable from the very first moment. Finally, despite many solutions offer an impressive performance as far as their actions are concerned, they tend to fare poorly whenever there is any kind of defect in the deployed system (faulty network nodes, damaged sensors). In this paper, we are presenting a proposal able to diagnose and self-heal most common issues that spring up in domains typical of an IoT system –as Wireless Sensor Networks, distributed middleware or embedded systems-. Plus, not only is our proposal able to provide services to human users as any other system, but also provides information of the most prominent characteristics to be taken into account from the system elements (battery level, transmission power, etc.). Additionally, rather than having a system tailored to work in only a specific area, we have designed one able to be adapted for different purposes, with very little effort required in order to do any adaptation to new circumstances.

II. An overview of the designed system

The major components that are present and their tasks are as follows:

Device layer. This layer is comprising all the hardware and all the appliances required to do the measuring and surveying the system. It will be responsible for gathering all the required context data whenever a request is taking place. In our suggested proposal, the system will be using Sun SPOT motes [4], MEMSIC Iris motes [5] and Arduino Uno boards [6] conveniently equipped with extra sensors, which may be equipped by the motes as well.

Communications layer. This Layer is in charge of all the communications that may take place between the different devices. There are two domains where communication operations happen; initially, there is a regular domain, with connections based on Internet Protocol at the network layer and Transmission Control protocol or User Datagram Protocol at a higher level. On the other hand, there will be another domain where 802.15.4, an IEEE standard designed for low capability devices, will be used as the wireless protocol of choice. IEEE 802.15.4 is considered here to be used for the monitoring system domain, as well as for internode

communications, while the usual network architecture works on the application layer that will employ a web browser.

Middleware layer. Up until this point the presented layered model is a unity, despite having different objectives. However, since the applications that are going to be run are implying different areas of usefulness, it is advisory to split the higher levels of the architecture in order to better deal with challenges related with lower level communications. While information transfer will be made in usual terms as far as user web browser requests are concerned, management will take a very different approach. In the latter environment, requests and responses are done with an *ad hoc* middleware layer that has been named Request and Response Adapter Protocol (RRAP). This middleware level will establish a protocol –effectively standardizing communications under the scope of the management part of the system- used for data traffic aimed to get data related with the status of the system. Additionally, messages will be sent to the upper layer if any important event comes up, so that the human user will be aware of relevant changes in the system.

Application layer. This level is splitted in two parts of equal level but fulfilling different functionalities: a web browser that, regardless of the different ones available (Mozilla, Chrome, Opera, IE, Safari, etc.) will process the requests done by the human user that is controlling the device where requests are launched (a PC, a tablet, etc.), and a Graphical User Interface especially made for the management part of the system. This GUI will come in handy both for status requests and notifications.

The system has been portrayed as a layered architecture in Figure 1. The layers appearing at the generic system architecture have been particularized in the proposal that is being presented in this document. User part is made for user requests and responses, while management part is bent on monitoring the system itself.
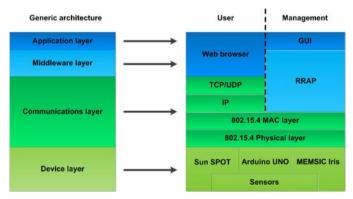

Figure 1: a holistic view of the whole architecture

After introducing the most important features of the proposal, each of the designed layers will be described in detail in the following sections.

III. Device Layer

Whenever a data request has to be fulfilled, device layer is the one with the suitable components to obtain the requested information. Being at the bottommost part of our proposal, it is therefore implied that this layer will deal with hardware, sensors and actuators more profusely than any other. There are three kinds of devices that are regarded as best-fitting for our proposal: two of them are motes –which are low capability devices frequently used as nodes in Wireless Sensor Networks with several built-in sensors and actuators- of different vendors –Oracle Sun SPOT and MEMSIC Iris- and the third one is the Arduino Uno board, a popular device for open hardware and software developments. The main features of these devices are displayed in Chart 1.

Device name	*Processor clock speed*	*RAM/Flash memory*	*Remarkable feature*
Oracle Sun SPOT	400 MHz	1Mbytes/ 8Mbytes	Capable of running HTTP
MEMSIC Iris	8 MHz	8 Kbytes/128 Kbytes	Java or nesC languages
Arduino Uno	16 MHz	2 Kbytes/32 Kbytes	Sensor flexibility

Chart 1: device layer components relevant data

All in all, the device layer must be understood as a Wireless Sensor Network with the following components:

a) *Base station/Sink*, which is directly connected to the device that has the web browser and the Graphical User Interface installed and running. Base station/Sink must be capable of managing information at the application layer, as HTTP requests about service information will have to be attended by it and sent to the devices that cannot handle information at layers as high as this. Since this node will behave as a bridge between the HTTP and the IEEE 802.15.4 domain, most of the petitions can involve obtaining data of different nature. Among the already mentioned devices, Sun SPOT Base station/Sink usage is mandatory here, for it is the only device present in our proposal with a HTTP client able to successfully get involved in HTTP communications. Besides, as it must be always attached to a computer to be powered, it remains unaffected by energy issues typical of Wireless Sensor Networks.
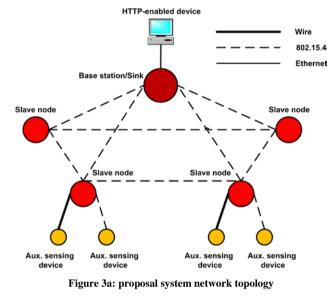
b) *Slave nodes*, which are connected to the Base station/Sink wirelessly by using communications standard IEEE 802.15.4. These nodes receive the requests that are meant to be answered by them; the requests will be sent by the Base station/Sink as soon as there is a petition at the web browser-enabled device. One very important feature of Slave nodes is that they can notify several issues that they may be suffering from; RRAP has a specific PDU that will be sent from a Slave mote to the most powerful-emitting node if it detects any performance strangling issue (as the most repeating, the battery is almost completely depleted). It must be taken into account that although nodes are physical devices, roles –at the end of the day, provided services- are purely made up by software, and their functionalities can be transferred from one node to another. According to the capabilities of the used devices, roles can be either activated if they are dormant (a more efficient option if energy consumption is taken into account) or being programmed Over-The-Air (OTA programming). In this case, either Sun SPOT motes or MEMSIC Iris ones can be used, as application layer features are not required at this part of the topology. Having equipment from different vendors communicating to each other at the same level can be a feature especially prone to interoperability issues: as Akribopoulos et al. mention in their research work [7], there may be incompatibilities due to payload sizes, addresses lengths, or some other derived from the particular platform of choice, regardless of claiming that

they are all using the same standard, as IEEE 802.15.4. Fortunately, any trouble that may be encountered should have been solved before by the RRAP implementation, and the work done at that point will be interesting to be considered for future interoperability challenges.

c) *Auxiliary sensing devices*. Nodes are made by actual devices that have several built-in sensors and actuators used for information provisioning; nevertheless, if they can be expanded so as to supply some information of different nature, then the system will be enriched and the end users will have more information at their disposal when the proposal is used. Interestingly enough, it is not difficult to augment the system capacities by using electronic devices with low capabilities, as the only requirement for them will be measuring environmental data and delivering it to its requester, without any other need of routing it anywhere. Consequently, available interfaces of the nodes can behave as ports for external data coming from other sensors and/or actuators alien to the node

More specifically, Arduino Uno boards are a very suitable option for this challenge; their capabilities are low enough to guarantee that they will not require a huge amount of power but, at the same time, will be able to store any small program – or *sketch*, as they are referred to- able to retrieve data. Provided that the needed elements –photoresistors, thermoresistors, etc- are available, obtaining readings from them can be done by plugging any element to a breadboard, mapping power references (power supply and ground) to the breadboard and getting the element reading as an analog or digital input for the Arduino Uno, provided that the pin mapping has been previously programmed. For example, as displayed in Figure 2, an Arduino board can be turned into a sensing/actuating device with ease; in the shown case, a switch is used to change from one sensor to another and to the actuator, thus having a LED, a photoresistor and a thermistor taking turns to execute their actions whenever the switch is pressed-. In the proposed system, either cabling to a port available at a node (for example, the Universal Synchronous/Asynchronous Receiver/Transmitter connections available in Sun SPOT motes [4]) or, as shown in Figure 2, adding a 802.15.4 XBee communication module can be

mounted either if a mote is wanted to be augmented with an Arduino board -as it may come in handy to test a service of similar nature in devices placed differently, and soldering may not be required- or the Arduino-built sensing/actuating device is preferred to run separately, as it will effectively turn into a low-cost mote.

In the end, the topology of the Wireless Sensor Network should look as how is displayed in Figure 3a.

Figure 3a: proposal system network topology

IV. Communications layer

In contrast with the particularity of the physical one, communications layer uses several standard and proven technologies, and has no *ad hoc* elements. As far as networking is concerned, there are two kinds of domains in the proposal to be taken care of: internode communications and web communications.

As already mentioned, the first domain is interconnected by using IEEE 802.15.4 standard. It has been found more useful than any other due to several advantages that other protocols lack: it consumes a low amount of energy and the
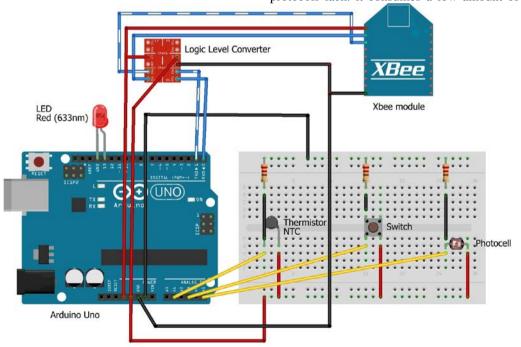

Figure 2: an Arduino-built, 802.15.4-enabled sensing/actuating device

available bandwidth, although somewhat scarce (only up to 250 KBps), is more than enough for what is expected to be done by low capability devices in Wireless Sensor Networks. Plus, many of these devices are already equipped with antennae enabled with the standard; Sun SPOT and Iris motes are 802.15.4 compliant, and almost any Arduino board can be equipped with a shield using a XBee module; there are even research products that are made of those components, as the Squidbee mote [8]. IEEE 802.15.4 standard is divided in two different layers: a physical one and a Medium Access Control (MAC) one. The former deals mostly with the channels available for transmission (usually, there are sixteen channels available in the 2450 MHz band, ten at the 915 MHz and one at the 868 MHz band, which is the one commonly use in Europe) and their frequencies, while the latter is involved in tasks typical of a MAC layer, as implementing a mechanism based on ARQ (Automatic Repeat Request) so as to guarantee error correction [9]. It must be mentioned at this point that, although frequently mentioned as the same concept, Zigbee is not IEEE 802.15.4; rather than that, it is a consortium devoted to the application layer services that may be able to be built upon the physical and MAC layers of IEEE 802.15.4, resembling in a way the similarities and differences between Wi-Fi Alliance and IEEE 802.11 standards.

The second domain that is present in our proposed system is a regular TCP/IP architecture. This domain has been placed higher in the layered architectural model as communications from the application layer will be transferred through an implementation supported on TCP/IP, while IEEE 802.15.4 communications are not expected to be routed as the TCP/IP-based ones are. Data transfer is done as usual in a network of these features: information regarding requests and responses is routed through a packet switched network and, depending on whether TCP or UDP has been chosen as the transport layer protocol, data transport will be done either at a slightly slower but more reliable way, enabling error correction and data retransmission (TCP) or at a more real time-like pace, risking the loss of information in the process (UDP). Judging from the data requirements of our system, it is considered that UDP is good enough, as it is important to get information quickly and chances of having data segments dropped should be fairly low. Considering how communications will be tackled, as well as which areas are using one architecture or another, network topology can be separated now in different areas involving different communication domains, as it has been portrayed in Figure 3b.

Figure 3b: network topology separated by communication domains

V. Middleware layer

Middleware is envisioned as fulfilling an extremely important task as far as the IoT or Wireless Sensor Networks are concerned, for it will adapt all the heterogeneity of the device layer components, and all the hardware disparity, into a homogeneous-looking collection of operations and interfaces. Noha Ibrahim, which provides a taxonomy on middleware architectures, claims that "*They have evolved from simple beginnings - hiding network details from applications – into sophisticated systems that handle many important functionalities for distributed applications - providing support for distribution, heterogeneity and mobility*" [10]. In this case, middleware will provide the Graphical User Interface positioned at the application layer with several operations in terms of management and status report. The middleware layer has been deliberately left outside the user architecture part because the services and functionalities present at this side of the architecture stack are considered to be tackled by regular layered components, and it is in our interest designing a model where Web Services and Wireless Sensor Network-oriented ones can coexist under the same system. Nevertheless, since the management part is accessing to the Wireless Sensor Network nodes and, by proxy, to the data provided by the Arduino Uno boards, it would be possible to obtain data from the network regarding sensor readings.

The middleware layer that has been designed is named Request Response Adapter Protocol (RRAP). It is an accurate name to give because it is going to adapt all the requests that are made from the GUI to a specific Processing Data Unit (hereinafter, PDU) format flowing through the Wireless Sensor Network, and responses will be treated the same way, albeit on the opposite direction (from the Wireless Sensor Network to the GUI). There are several PDUs that have been designed, according to the different functionalities that are expected to be retrieved from the middleware architecture. While there are several different types of them, they are managed in a way that human operators do not perceive the disparity under any circumstance; their variety is due to the fact that the top design criterion was using as few data in radio transmission as possible, as radio messages are the most energy-demanding operation in a Wireless Sensor Network by far, as shown by Abdelmalik Bachir et al. [11].
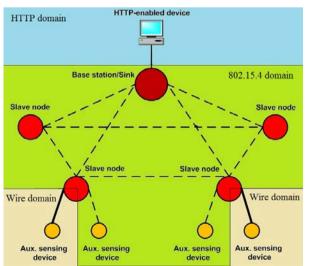
RRAP will be responsible for tackling several actions that must be performed, as they have been depicted in the use case diagram presented in Figure 4.
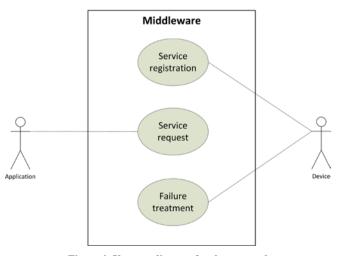

Figure 4: Use case diagram for the proposal

Service registration. In order to have functional, retrievable services, the Base station/Sink must be aware of them, so whenever a node is turned on, it will broadcast a PDU with all the available services that can be obtained either by its built-in sensors or from any Arduino Uno board (which, at the same time, will be connected by using wires to a mote port or work wirelessly by using a XBee module). This is the only PDU that must be transmitted in broadcast mode rather than unicast, as the node is unaware of where the Base station/Sink is. Its fields will consist of: a node identifier (that may be varying from one tailored for the system to a MAC address, as available in Sun SPOT motes [4]) and service identifiers for the services available at a node, along with their parameters. The different components of this packet have been presented in Figure 5.

way to request different information seems like a desirable feature of the system.

For example, if power transmission is requested from all the existing nodes, a PDU where the only feature that is necessary for the request to be made is the manager service identifier (e.g., in case a query made to learn the available services is executed) is sent. When the response is obtained, it will be done by providing the manager services and their parameters from each of the nodes. This interchange has been depicted in Figure 5 with a 1X-nX identifier, where 1-n acts as the node identifier and x as the service one, as it is likely that there are several different services running, along with their corresponding parameters.
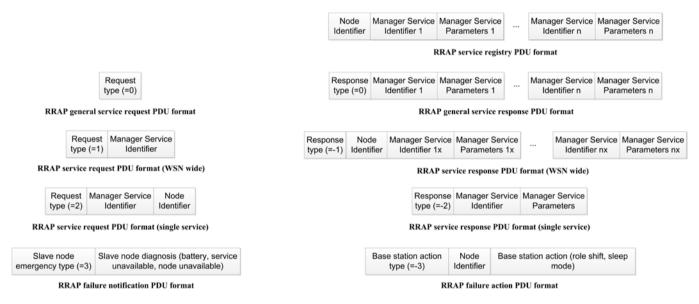

Figure 5: RRAP PDU formats

Service requests. Whenever there is a query involving management services, it will be transmitted towards the Wireless Sensor Network from the Base Station/Sink in the simplest possible manner. Therefore, unambiguous identifiers will be used to do the request. To begin with, a request on the available services from the system can be done. As it will be the most generic and information abundant query, there is very little need to have many particularizing fields in the PDU that is transmitted towards the Wireless Sensor Network. In fact, if service registration has been done without anomalies, this request could not be mandatory, as data involving registered services can be stored at the host application running at the Base station/Sink. As it is displayed in Figure 5, this request PDU (labeled as type 0) will consist of just a field characterizing the petition, while the PDU containing the response results will be larger, as it must include service identifiers and parameters that are retrieved.

As the Base Station/Sink receives the available services that were offered by specific Wireless Sensor Network nodes, the service request message does not require a node identifier, although the response may vary depending on whether a reading from a single node was requested or an overall value that can be obtained from the whole Wireless Sensor Network. This has been conceived this way because having a flexible

On the contrary, if a management reading from a single service from a specific of the Wireless Sensor Network is requested, then the PDU will look as presented in Figure 5: a node identifier and a single manager service identifier are used to address the node. As the services and the entities providing them were registered before, the Base station/Sink is aware of where to find the node that will satisfy the request. Afterwards, when an answer is retrieved, only the service and the parameters the Base station/Sink is expected to fulfill from the single node are retrieved.

The entities that are involved in the already described information exchanges, along with the particular exchanges, have been depicted in Figure 6.

Failure treatment. The system is also taking into account whenever there is a failure in the Wireless Sensor Network. Without any unforeseen event, slave nodes may be faulty due to three different kinds of reasons: either their battery is about to run out of power, a service has become unavailable (for example, a sensor has been damaged or an Arduino board connection to a slave node has failed) or the node has become unavailable (it is no longer able to transmit/receive data). When one of these issues is taking place, the slave node sends a PDU as depicted in Figure 5 to the Base station/Sink announcing the problem. The next step will be taken by the Base station/Sink itself: either it will put the node in a sleep

mode in order to reduce energy consumption, or the role it is performing –that is to say, the parameters that are being collected- will be moved to another node. As it was done before, the entities involved in this use case are depicted in Figure 6.
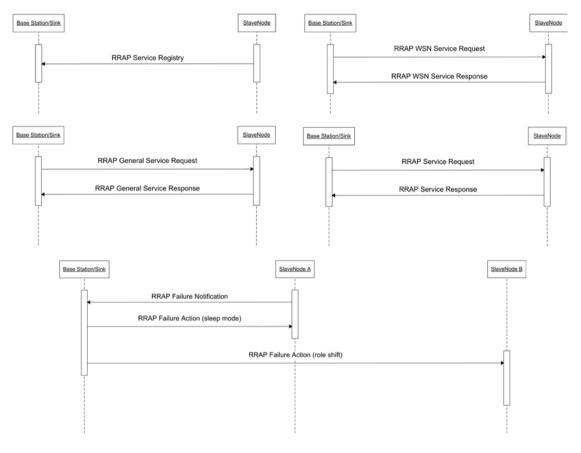

Figure 6: Entities involved in data transfers and failure treatment

VI. Application Layer

As firstly displayed on Figure 1, the application layer will be made by two different entities with very little actual relation: a web browser and a Graphical User Interface. The web browser is expected to be used from a PC, laptop or any device capable of having a Sun SPOT base station plugged to a USB port. It is mandatory that the appliance the base station is plugged to is at the same time connected to the Internet, as the appliance will be in charge of providing a reliable IP address to the Base station/Sink from where services can be requested; otherwise neither the Base station/Sink nor the services from the Wireless Sensor Network can be retrieved.

Meanwhile, at the Wireless Sensor Network, Sun SPOT motes will have an HTTP server installed that will be listening to any request done from the web, and whenever there is an invocation it will be sent to the suitable node. For example, if luminosity from a node placed in a room numbered as 45 at the second floor in an industrial facility, then the service could be requested as:

```
http://192.168.10.25:1267/spot-
79E3/luminosity/industrial/2nd/45
```

In this example, the fields present at the Uniform Resource Identifier (URI) are:

1. `192.168.10.25` as the IP address of the device the Sun SPOT base station is connected to (which in fact behaves as a gateway from/to the Wireless Sensor Network).

2. `1267` is the port used for the communications.

3. `luminosity` is the name of the requested service.

4. `spot-79E3` is a predictable part; `spot` simply represents the name of the devices manufactured by the vendor, while `79E3` are the last four digits of the MAC address that is displayed at the motes on their translucent plastic radio antenna protector [4].

5. The last part of the URI (`industrial/2nd/45`) is the path that has been established to reach the specific device, which will be defined at the implementation stage

Responses can be watched at the device the Base station/Sink is plugged to in a variety of formats. If data is to be given any sort of hierarchy, XML or JSON formats suit fine for this purpose. Iris motes not executing HTTP petitions will be communicating to Sun SPOTs via 802.15.4 data interchange whenever a service only the former are able to provide is queried.

At the same time, a Graphical User Interface must be enabled for the monitoring of the current capabilities of the Wireless Sensor Network. Using a Java-based Base station/Sink that is able to run Java applications as if it was a communications host, a GUI can be developed where all the important parameters that ought to be born in mind for the correct performance of the system can be requested. One example of that GUI is illustrated in Figure 7, with the available options and fields to be filled up with information.

Figure 7: example of a system Graphical User interface

VII.   Use case scenarios

There are many environments where this monitoring system can be used, provided that devices at the Wireless Sensor Network that has been designed are guaranteed certain acceptable environmental conditions. Usually, whenever there are data to be collected or monitored the proposed system is likely to come in handy. The main scenarios where the system can be applied are:

Agricultural facilities. In this field, several parameters that can be easily measured by the proposed system (Sun luminosity, environmental temperature, humidity) are of critical importance for crops development or cattle care. Additionally, chemical sensors can be added to the Arduino Uno boards so as to measure parameters that are not available by default in the measuring motes.

Infrastructure monitoring. Material stress or infrastructural wobbling can be surveyed by this proposal as well by making use of either built-in mote sensors or any other that may have to be added to the Arudino Uno boards.

Industrial processes. Monitoring is an extremely desirable functionality for any production process or supply chain, for Quality Assurance in manufactures is almost mandatory for industry-related products. For an application in this area, thresholds may be more varying than somewhere else, as manufacturing a metal-derived product requires facilities widely varying in their context characteristics (pressure, temperature, etc.).

Tertiary and domestic environments. Our proposal can be used to improve control on how energy is spent for more efficient heating or lighting. Storage that has to be done under special temperature conditions may benefit from the usage of the proposed system as well.

Mineral exploitations. Gas sensors are at its finest here; firedamp deposits are a major concern in places where mineral extraction is prominently made by human miners instead of mining machines, and tunnel tilts can be measured as well for collapse prevention (actually, Sun SPOT motes have a built-in 3-Axis accelerometer that may fit in for this purpose).

All these scenarios, along with several parameters and useful services, have been summarized in Chart 2.

Scenario	*Input parameters*	*Services*
Agricultural facilities	Temperature, luminosity, air quality	Parameter real-time measurement and/or monitoring
Infrastructure monitoring	Vibration, 24-hour temperature difference	Historic statistics, parameter real-time measurement
Industrial processes	Temperature, pressure, chemical agents	Warning and real-time alarms, monitoring
Tertiary and domestic environment	Power consumption, air quality	Monitoring, consumed power cost
Mineral exploitations	Gas concentration, rafter tilt, chemical agents	Explosion prevention, real-time monitoring

Chart 2: scenario parameters and obtainable services

There are many other developments on how to use systems that somewhat resemble the one described here; for example, Weimei Zhang puts forward an Internet of Things-based platform for digital agriculture [12], or Florian Broekaert et al. suggests an energy harvesting Wireless Sensor Network Application [13]. However, these proposals, compelling as they are, do not usually become as flexible or user-friendly as the one that has been presented in this document.

VIII.   Conclusion and future works

In this document a proposal for a holistic architecture, which has as objectives attending requests related with measuring services provided by a Wireless Sensor Network – capable of being easily augmented by means of sensor addition and service registration- and is able to self-monitor and self-heal itself from critical conditions that make a node unavailable has been displayed. The different elements it has been designed with have been described and information about how it is expected to tackle interoperability and interconnectivity issues, among other challenges, has been provided too. The idea of facing difficulties for establishing connections among devices expected to implement the same radio standard in case of IEEE 802.15.4 is somewhat shocking, and it only points out the urgent need to further standardize the technologies expected to run a leading paper in the development on Wireless Sensor Networks or the Internet of Things in general.

On the other hand, the design presented here has the potential of being upgraded to an extent. Semantic capabilities would an interesting add-on, and would further guarantee the interoperability and scalability of the system, as they would

provide an accurate description of how services should be described. Alas, an implementation of a Graphical User Interface powerful enough to show topology information and node pictures for accurate description purposes would be welcomed if real-time information belonging to the Wireless Sensor Network is wanted to be retrieved.

References

[1] Urien, P. *LLCPS: A new security framework based on TLS for NFC P2P applications in the Internet of Things*. in *Consumer Communications and Networking Conference (CCNC), 2013 IEEE*. 2013.

[2] Pereira, P.P., et al. *Enabling Cloud Connectivity for Mobile Internet of Things Applications*. in *Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*. 2013.

[3] Weiser, M., *The computer for the 21st century*. Scientific American, 1991. **265(3)**: p. 94-104.

[4] Oracle/Sun, *Sun™ SPOT Main Board Technical Datasheet*, Oracle, Editor. October 2010.

[5] MEMSIC, *http://www.memsic.com/userfiles/files/Datasheets/WSN/IRIS_ Datasheet.pdf*, M. Inc., Editor. 2013.

[6] Arduino. *Arduino Uno web site*. 2013; Available from: http://arduino.cc/en/Main/arduinoBoardUno.

[7] Akribopoulos, O., et al. *Building a Platform-Agnostic Wireless Network of Interconnected Smart Objects*. in *Informatics (PCI), 2011 15th Panhellenic Conference on*. 2011.

[8] Libelium. *Squidbee Main Page*. 2013; Available from: http://www.libelium.com/squidbee/index.php?title=Main_Pag e.

[9] Singh, J. and D. Pesch. *Enhancement of IEEE 802.15.4 MAC layer to combat correlated channel errors*. in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*. 2011.

[10] Ibrahim, N. *Orthogonal Classification of Middleware Technologies*. in *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2009. UBICOMM '09. Third International Conference on*. 2009.

[11] Bachir, A., et al., *MAC Essentials for Wireless Sensor Networks*. Communications Surveys & Tutorials, IEEE, 2010. **12**(2): p. 222-248.

[12] Weimei, Z. *Study about IOT's application in "Digital Agriculture" construction*. in *Electrical and Control Engineering (ICECE), 2011 International Conference on*. 2011.

[13] Broekaert, F., et al. *Prototyping an Energy Harvesting Wireless Sensor Network Application Using HarvWSNet*. in *Architecture of Computing Systems (ARCS), Proceedings of 2013 26th International Conference on*. 2013.