International Journal of Scientific Engineering and Science
Volume 1, Issue 9, pp. 40-46, 2017. *ISSN (Online): 2456-7361*

Supporting IOT Low-Power Embedded Devices with Smart Gateway: Proof of Concept and Implementation

Dina Darwish[1], Adel Elzalabany[2], Osman Ibrahim[3]
[1]Multimedia and Internet Department, International academy for Engineering and Media Science, 6th October, Giza, Egypt
[2, 3]Informatics and Computer Science Faculty, The British University in Egypt, Cairo, Egypt

Abstract— *Internet of Things (IoT) is a concept that depends on enabling a variety of things existing in the environment to interact with each other and cooperate with other objects through wireless and wired connections and unique addressing schemes to create new applications and reach common goals. Internet of Things (IoT) is considered as a promising technology that can provide a boom of reforms, including the combination of wireless communication technologies, embedded systems, mobile applications and cloud technology. Most of the IOT technologies that have emerged recently, concentrate on a single intelligent product rather than the integration of the different IoT network components. There is a huge need nowadays to integrate all these technologies together in an internet connected world. In this paper, we focus on providing IoT-based devices with important communication services. These IoT-based devices represent a smart home scenario, along with embedded appliances, mobile devices and centralized home gateway.*

Keywords— *IoT, things, smart gateway, sensors, big data.*

I. Introduction

The aim of the Internet of Things is to allow communication between things at anytime, anyplace, with anything and anyone using any network and any service. Internet of Things can be considered as an evolution of the Internet. Objects have the ability to recognize themselves, and they obtain intelligence by taking context related decisions, due to the fact, that they can exchange information about themselves. They can reach information that has been generated by other things, and they can act as components of complex services. New types of applications can emerge, such as; the electric vehicle and the smart house, in which appliances and services that deliver notifications, security, energy-saving, automation, telecommunication, computers and entertainment, are merged into a single system with a shared user interface. But, everything has to take time to be in its ideal place. Smart environments can be realized in Europe nearer to 2020 [1], by developing the IoT technology through demonstrating, testing and deploying products.

Current Internet architecture as a design conception is limited, since it does not contain free information exchange between data and service levels; this fact triggers a race between academic and industry for designing the future Internet, in which ensuring information interoperability is a key challenge [2-4]. A fully coordinated course in terms of design and implementation has not happened, and this is considered as an undesired result of this race, due to many complex issues including especially deployment over particular diverse infrastructures. There is a lot of other research performed in the different areas of IoT, including networks types and devices types, as well as challenges facing the IoT.

In this paper we represent a whole description about the web application project. In section 2, related research work to this area is discussed. In section 3, the requirements and tools used in the web project application were provisioned. In section 4, we introduce the system architecture. Section 5 illustrates the system design supported by class and sequence diagrams. Eventually, section 6 discusses in details implementation issues. In section 7, future work is proposed. Finally, conclusion is explained in section 8, followed by references.

II. Related Work

In the following sections, we are going to discuss briefly some researches done in different areas to evolve the IoT.

Paper [5] illustrates the use of WLAN as an intermediate backbone connecting lots of sensors and devices. This creates miniature gateways, that allow the use of the multi-radio capability of the short-range radio chips, such as, chipsets supporting simultaneous use of Bluetooth low energy and WLAN. These miniature gateways exchange data through Bluetooth low energy down-stream and WLAN up-stream. In [6], we explore the IoT application space for the goal of determining two known challenges, that exist across this space; ultra-low power operation and system design using modular components. Then, a survey is done on recent low power techniques. Finally, a low power bus, that enables modular design, is proposed. Paper [7] provides a comprehensive model for the power consumption of wireless sensor nodes, which represents all the energy expenditures at system-level. The model depends only on parameters that can be empirically quantified, once the platform and the application are determined. This results in a new framework for the study and analysis of the energy live-cycle within the applications. In [8], the IoT gateway problem is discussed, because today's gateways conflate network connectivity, in-network processing, and user interface functions. Authors expect that solving these problems would enhance the connectivity potential for IoT devices, and that a worldwide deployment of IoT gateways could revolutionize application-agnostic connectivity and avoid the stove-piped architectures existing now.

In [9], the characteristics of different types of low power wireless networks are discussed and compared, such as, Bluetooth 4.2 that has a less range and bandwidth, and the IEEE 802.15.4 standard that defines the physical layer (PHY) and media access control (MAC) layer for low-power and low data rate personal area networks, and others. Also, in [10], the 6LowPAN standard was described in details. As its name implies, 6LowPAN, "IPv6 over Low-Power Wireless Personal Area Networks", is a networking technology that enables IPv6 packets to be transported efficiently within small link layer frames, such as those defined by IEEE 802.15.4. The paper [11] describes the unique benefits of Z-Wave technology for smart devices and the Internet of Things, such as, technical considerations, the role of certification and branding, and advanced development tools. Z-Wave is a low-power, wireless mesh network standard that is widely used for M2M and smart devices in the Internet of Things (IoT), home automation, and security markets, and, this makes it the best overall choice for the vast majority of smart device applications.

Paper [12] outlined that the lack of an IP-based network architecture prevented sensor networks from interoperating with the Internet, limiting their real-world impact. Also, this paper discussed the 6LowPAN and RoLL standards proposed by the IETF working groups with the goal of connecting low-power and lossy networks to the Internet, and described how the research community participated in this process of their design and their open source implementations. In [13], the authors described Constrained Application Protocol (CoAP) and other key enabling technologies together with an end-to-end IP and RESTful Web Services based architecture for integrating physical world devices in constrained environments with the Web. In the paper [14], the authors discussed two open problems. First, the set of existing solutions for configuration and service management are insufficiently studied in the framework of the IoT. Second, current approaches do not suit all the specific requirements and characteristics of constrained devices, in terms of memory usage and power consumption in particular. The paper [15] provides a realization of ultra-low voltage circuit design for Internet of Things. Semiconductor industries are investing in introducing power saving features by using ultra low voltage circuit design, because these devices depend on portable battery or external power source. Then, there is a need to reduce the power consumption by designing an ultra-low voltage circuit for IoT devices. The ultra-low voltage circuit design for Internet of Things is accomplished by simulating two input NAND gates in 120nm CMOS technology.

As mentioned before, several former efforts, aimed at providing a control interface to communicate with multiple low power devices and/or sensors, were performed. IoT goes further in opening new possibilities to attach with every smart device ranging from home appliances, communicating devices, ending with ultra-low power sensors and embedded systems.

While some smart devices such as home cinemas and mobile phones which have enough computing power to allow them to be attached directly to internet, they cannot actually host a service oriented interface. On the other hand, there exist ultra-low power devices such as light, temperature sensors, etc., as well as devices with restricted computing power. Also, automotive engine ECU (Electronic Control Unit) and industrial machine control units, are not applicable for providing a service oriented interface and may be interrupted by the overwhelming accompanied web service oriented interfaces.

III. Requirements and Tools

Leveraging the technology of integrating sensors (things) with web services, some of the relevant components from this technology tools have been incorporated as SOA-based Intelligence. We are adjusting the concepts of the design and implementation of Smart Gateway; that exposes an XML-RPC web service [16-19], SOAP web service and a RESTful interface. We embedded C, Python, Perl programming languages, Redis-server [20, 21]; Contains Redis Daemon and Redis-cli tool, Arduino studio [22, 23], and, also, Arduino Uno kit [24] (evaluation board with Atmel, AVR 8bit microcontroller).

Regarding the project features, and, to meet requirements that support our obtained intelligent web application, XML-RPC interface provides the following functions:

- Captures the current "instantly" readings from sensors. It sends commands to capture Temperature, Humidity, Smoke, Hydrogen sensor readings.
- Returns Readings from time based log (in SOAP).
- Returns Average values calculated from latest log recording (in REST).

The conduction of multi-intelligence sensor-based service web application can fulfill the following required tasks:

1. Proxies XML-RPC request from user application to a number of sensor Kits, through UDP based binary messages.
2. Monitor the sensors for new Temperature, Humidity, Smoke and Hydrogen, then, append the readings to Log according to a user specified time interval "in seconds".
3. The implementation includes emulated sensors; these adopt separate processes which listen to the incoming UDP messages in same format as a bare metal embedded evaluation board.
4. In addition, the implementation requires a deployment on a chip software that will send environment readings to the service or timer. See figure 1 for the illustrating use-case diagrams.

The system is implemented such that sensors "on the chip kits" and emulated ones will only react to the event of receiving UDP messages. It's allowed to the on-chip software to keep doing its dedicated task and only is being interrupted when needed, instead of having the on-chip software to broadcast its readings all time. Thus, the application keeps all configuration at the smart gateway side. In this respect, the smart gateway acts as a coordinator and a configuration management tool. For example, user-defined time intervals are handled by the smart gateway, so we do not need to configure each chip software to update the provisioned time interval. The chip software will only respond to events fired by the smart gateway timer or the web services.

The smart gateway is implemented to accommodate high workloads, and should allow scaling to several machines. To achieve this goal, the system services do not keep limits between requests at all, and autonomously serve the requests regardless of the machine, that hosts the service in a cluster.

Besides that, the system should make use of a very light weight messaging style, to allow communication with resource constrained devices. Though, the messaging style used in the implementation does not need to reflect real world protocol due to complexity and testing requirements. Consequently, the design and development of this project as a scalable data integration platform interacts with the Python Redis database and makes use of Data technologies.

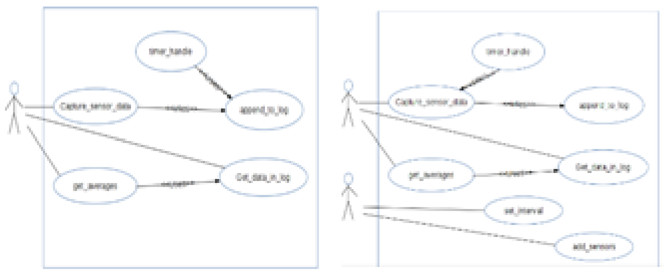

Fig. 1. Use-Case diagram for the sensor-based service web application.

IV. System Architecture

The system ingests intelligence [25-27] data from representative sensors provided in pre-specified formats, in order to perform the integration of a variety of intelligence data as used by intelligence analysts to conduct multi-intelligence all-source analysis. Structured data comes from sensors including track data reports, measurements to and from database. Concerning these processes in system architecture, proof of implementation concept uses a Reactor pattern. It provisions simpler design and faster development time. The current implementation consists of the following separate processes:

1) A none memory persistent CGI web-application scripts that act as XML-RPC client.
2) Three CGI XML-RPC services, used for capturing sensor data from the devices instantly, for returning Log of all previous readings, and returning average values of all previous readings.
3) Persistent timer process that sleeps for a user defined time interval, sends UDP binary messages to devices synchronously, and appends results to the log also synchronously "RedisDB".
4) Real and emulated devices that listen to UDP messages and send fresh sensor readings back.
5) Log server can be accessed using SOAP interface, returns all Log contents serialized in XML. while the analyzes server is RESTful "we implemented different service styles to demonstrate the interoperability"

While this architecture is simple and enables to distribute workload on CPU cores, it is not the recommended architecture for production as we recommend an event-driven Proactive pattern. A Proactive pattern will handle much more work-load with less resources, also readings to time accuracy will not be affected when handling a huge number of devices per event-based processes. It's still applicable to create a cluster of machines to handle workload in both architectures, as long as the service and its backing processes are stateless.

Figure 2 shows the whole application process of the system with the smart gateway architecture. The journey of the request of a web service is illustrated in figure 2 from a client web application towards the smart gateway.

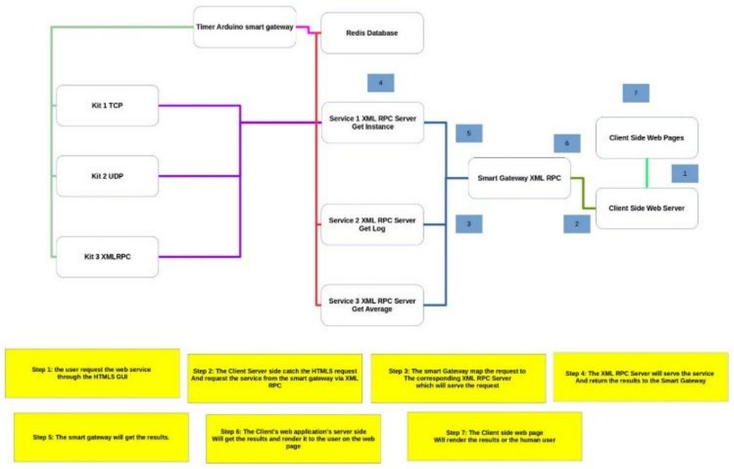
Timer Arduino smart gateway
Redis Database
Kit 1 TCP
Kit 2 UDP
Kit 3 XMLRPC
Service 1 XML RPC Server Get Instance
Service 2 XML RPC Server Get Log
Service 3 XML RPC Server Get Average
Smart Gateway XML RPC
Client Side Web Pages
Client Side Web Server
1
2
3
4
5
6
7
Step 1: the user request the web service through the HTML5 GUI
Step 2: The Client Server side catch the HTML5 request And request the service from the smart gateway via XML RPC
Step 3: The smart Gateway map the request to The corresponding XML RPC Server which will serve the request
Step 4: The XML RPC Server will serve the service And return the results to the Smart Gateway
Step 5: The smart gateway will get the results.
Step 6: The Client's web application's server side Will get the results and render it to the user on the web page
Step 7: The Client side web page Will render the results or the human user

Fig. 2. The whole application process of the system with a smart gateway architecture.

V. System Design

The smart gateway consists of multiple servers, each handles user requests, the advantages of this approach is that we always can deploy the servers on the same or on different machines, having a socket based services provides us with the flexibility to scale to multiple machines, each process can be then assigned more processing power or I/O rate using tools such as nice or Linux kernel name-spaces. Moreover, we can actually isolate these processes to ensure security, for example we can run each server under cheroot, this ensures that each server is locked into a specified directory and will never be able to make I/O other than already opened files and sockets. See class diagram and sequence diagrams for more vision explanation in figure 3, figure 4 and figure 5.

Another advantage is enabling easier monitoring of resources consumption, since we have one process handling every-request "as in pro-actor pattern", although it might perform better, but it will not be easy to investigate I/O bottle necks, unlike in reactor pattern in which we have a clear view of the status of each process, each process can be analyzed separately and we can discover bottle necks much easier.

A. Implementation Components

We have two parts of implementation. For the prototype implementation with Perl, the following components are used:

1) IO:Socket:INET, provides TCP/UDP sockets

2) XML:RPC, client & Server parser/deparser
3) CGI, CGI helper functions
4) Redis/hiredis, either a pure Perl or C based Redis connectivity
5) HTTP::Headers, HTTP headers helper functions

While the Python implementation, the following components have been leveraged:

1) socket, global socket interface
2) sys, base system functions
3) xmlrpc.client, xml-rpc client
4) xmlrpc.server
5) SOAP module
6) REST module
7) cgi, CGI helper functions
8) cgitb, CGI helper functions
9) redis: Redis connectivity

Here are three diagrams showing class diagram, and two sequence ones. The former shows user, device, interrupted ip-chip, log and Redis database classes. While the two later show the run-time processes between the web browser (client), CGI, sensors (device) and timer.

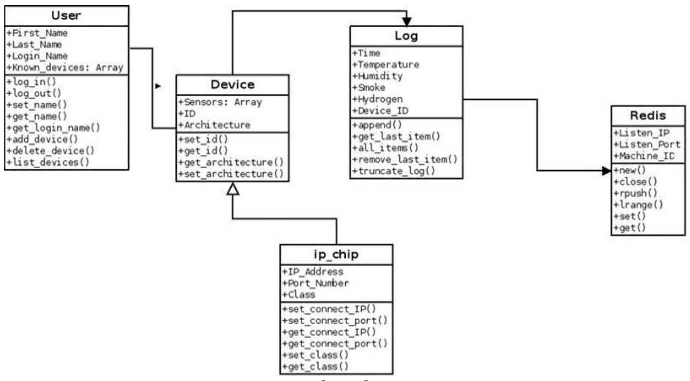

Fig. 3. Class diagram.

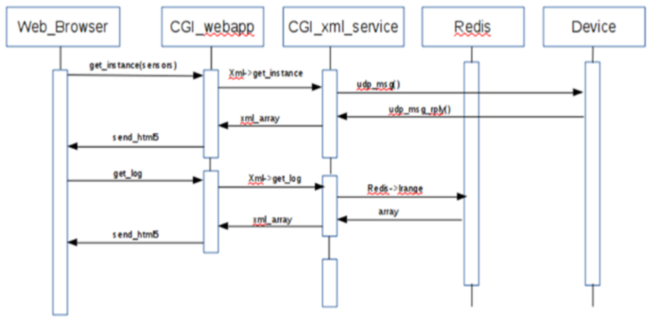

Fig. 4. Sequence diagram 1.

B. Interoperability

The smart gateway presents three types of services, XML-RPC, SOAP and REST, and this allows different clients to initiate requests from different platforms, all the mentioned interfaces use text based "XML" serialization of data structures, this also contribute to the interoperability measure, since textual data will not require byte ordering considerations, and hence clients running on different machine architectures should be able to make use of the smart gateway interface.

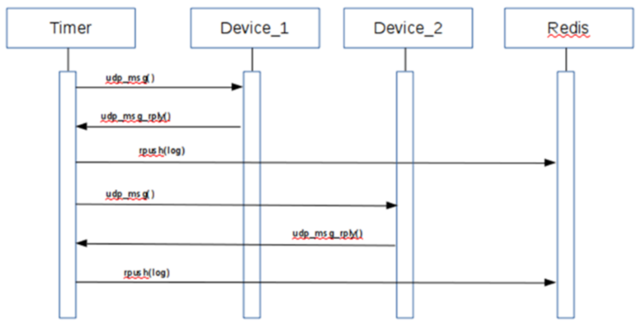

Fig. 5. Sequence diagram 2.

Another factor contributing to interoperability, is the fact that the smart gateway is fully POSIX compliant, POSIX is Portable Operating System Interface which is an IEEE standard for operating system APIs, it includes basic C standard library as well as a number of low level system-calls, having the smart-gateway POSIX compliant means, it can run on a number of Operating systems including: GNU-Linux, BSD variants, Solaris variants, HP-UX, MS-Windows using "services for Unix" or Cygwin.

C. Deployment

The software components of the smart-gateway are composed of three services, that can be deployed either on same server or on different servers, in figure 6 below each smart-gateway node has three services installed "multi-core machines are assumed", and an HTTP load balancer is used to distribute the load of requests among the smart-gateway cluster nodes. Open-source Load balancers provide better solution than commercial appliances, allow flexibility and fine tuning of both the proxy server and the underlying OS. Figure 7 shows vertical silos of IoT service deployment.

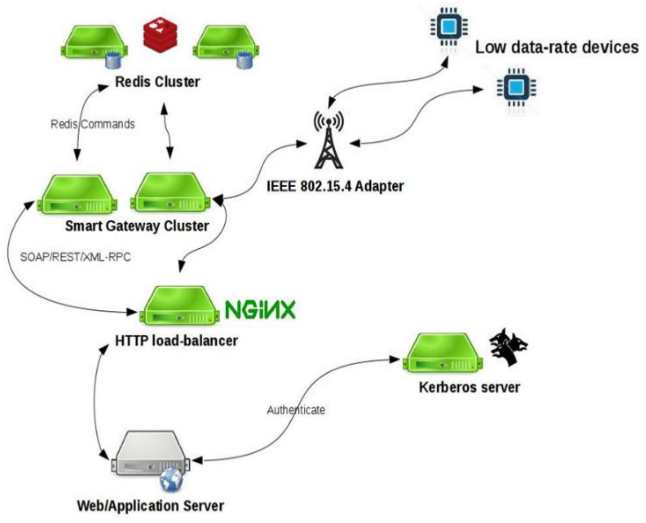

Fig. 6. Example of deployment model diagram.

The Load-balancer can be configured to make round robin balancing, or weighted load-balancing, where nodes with more computing resources get more requests than those with low lower capacity.

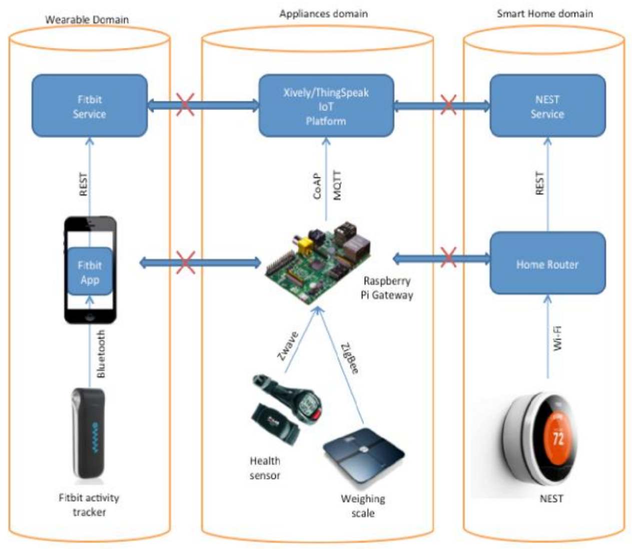

Fig. 7. Vertical silos of IoT service deployment.

The Smart gateway can connect to any IP enabled devices by default, either locally attached or using VPN, also it is able to use adapters to access low rate protocols such as RFID. The adapters can be also locally or remotely attached. Kerberos is used for authentication, since it provides best symmetric key authentication in practice for users. Figure 8 shows an example of Blocky script.

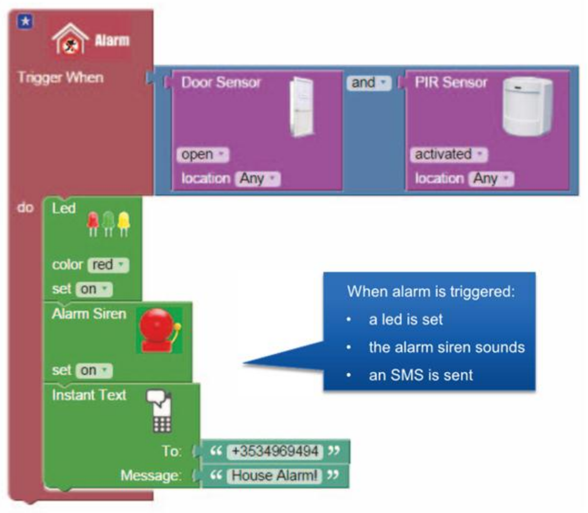

Fig. 8. Example blocky script.

Redis cluster can be deployed in two models:

- Transparent: the smart-gateway is not aware of any clustering in the back-end, and the cluster can be added transparently, and it offloads the programming overhead to a pre-packaged Redis proxy.
- Direct: the smart-gateway is aware of the cluster and it is responsible of mapping Redis keys to associated nodes. Though, this model provides no point of failure, it comes with much programming overhead.

VI. Implementation Issues

During the web application time, team faced a number of obstacles that changed decisions during the prototype implementation phase, the team found that Python modules are not always able to work with Perl's, due to the fact that some Python modules are written and tested only against the corresponding Python implementations. We had to make a decision of either re-write code in Python or complete everything in Perl, the first choice was taken because Python is more friendly for beginners and easier for everyone to contribute. Figure 9 shows a system workflow.

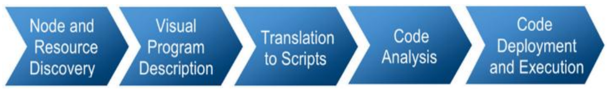
Node and Resource Discovery
Visual Program Description
Translation to Scripts
Code Analysis
Code Deployment and Execution

Fig. 9. System workflow.

In addition to that, code was written first to support sending binary messages to sensors using TCP, while this worked for simulated sensors, Arduino kits by default use UDP, and compiling TCP stack to it proved very tedious task, so we opted to change smart-gateway code to be able to send messages to the Kits using UDP, this proved that the smart-gateway can easily handle both protocols and that “TCP is still used with emulated sensors”. Figure 10 shows Python protocol stack at runtime in Contiki OS inside a WSN node.

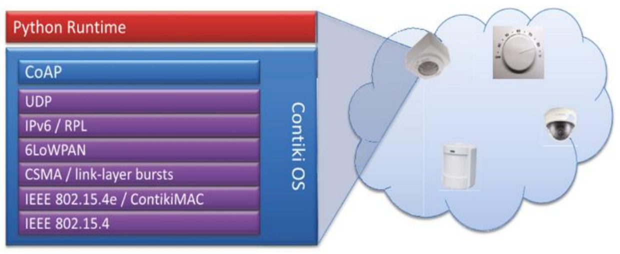

Fig. 10. WSN node embedded protocol stack.

Considering the huge amount of data produced every day in both the commercial and the defense areas, the Big Data paradigm promotes novel approaches and technologies for data capture, storage and analytics to deal with “massive volume of unstructured and structured data that cannot be managed and processed with traditional databases and software approaches”. Figure 11 shows data integration, analytics and management inside IoT.

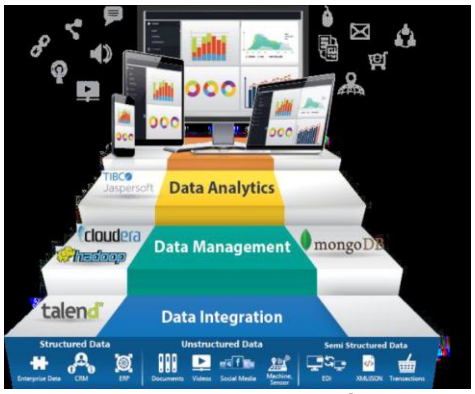

Fig. 11. Data representation inside IoT.

A. Utilized Tools

1. RedisDB.

We opted to use a NoSQL data store to store Log of sensor readings and to register new Sensors. The main advantage is that a NoSQL database will provide security, performance and scalability, which are described below.

- *Security:* It will eliminate all vulnerabilities that comes with SQL databases, and will also allow us to define application specific name-spaces to ensure controlling application multi-tenancy. This is possible due to the fact that RedisDB provides a Key-value interface, where the key can be partitioned within the application code, this can enhance security by creating a name space or prefix within the key.

Another security advantage is that we are free to store different keys on different Redis nodes, and hence we can implement middle-ware that is able to authenticate requests from the application “smart-gateway in our case”. Redis runs on Socket connection, so the smart-gateway will never require any super user privilege.

- *Performance:* A NoSQL data-store doesn't require SQL, and hence no complex SQL query preprocessing is required, although some NoSQL implementations offer Stored procedure, which Redis capability doesn't provide, and it is not needed in our scenario. Another advantage is the ability to scale to multiple machines, either by partitioning keys or using a hash function to map keys to different nodes. In the smart-gateway scenario, no range searches are required, hence a hash function to map the keys to hosting nodes is preferred since it offers best performance. While Redis by default use Hash-map to internally map keys to values, some Redis forks use different key-value store engines, that support range searches if required. So Redis also contributes to lose coupling, as it became also an implementation and not only an interface.
- *Scalability:* As we mentioned before, Redis is able to run in clustered fashion, also there exists a number of middle-ware that functions as a coordinator, that will map keys to nodes without application interaction “transparent to the application”. The advantage of scaling to multiple machines is that we do not need a new infrastructure or complex migration when the business requirements change. Redis offer elasticity that will ensure having all components of the application running on multiple nodes. Another important factor is handling Denial of Service attacks if we need.

2. GNU-Linux OS.

Linux is the chosen development and test platform, and it is POSIX compliant, having packages available for all Python, Perl, C libraries and modules. Virtualization technologies are available and can be used in testing, Linux Containers (OS level virtualization), and KVM (full platform virtualization).

3. Nginx.

Used as a reverse proxy that perform HTTP load-balancing, and it is a single thread master-worker high performance web-server.

4. Python interpreter.

Available through Debian/Ubuntu GNU-Linux package managers.

5. Perl interpreter.

Already installed on almost all default Linux/Unix installations.

VII. Future Work

The implementation of our multi-intelligent data integration project system can leverage emerging Big Data and SOA technologies. While we aim at providing a comprehensive social beneficial media and exploitation application, further work is required to deal with embedding web services in this context. A listener from a specific country can interact with another one from different land and they can share their local weather statuses and climate differences. They can chat, discuss about the surrounding temperature and how it affects their mood, and exchange songs to listen. As most of us know, songs bring memories.

Intelligent social sound application may reveal interesting insights from the analysis of large predictive information by using appropriate techniques to choose a suitable sound track depending on the user mood and activity level. The user can provide a proposed rate of his status power to allow the application to play a fast-going song for doing workouts for example, or a slow song for a relaxing time. He/she can show the ready state for studying and, at this time, the application can provide a lecture which wasn’t played before to the user. Eventually, we cannot count the creative ideas which may be implemented to make this application lead the social media application all over the world.

Furthermore, potential capabilities, that can extend current features set in a production implementation include:

1) Add more devices to the smart gateway's control and monitoring domain dynamically.
2) Add different set of communication protocols for different devices with varied computing capacities.
3) The use of a more elegant event-driven approach where more workload can be handled with less resources.
4) Allow clustering the data store keys to multiple machines to enable scaling the backing data store “Redis”.
5) Implement a real time Ajax based web interface.
6) Add more meta-data related to devices controlled by the smart gateway such as location, provide processor/micro-controller architecture in safe operating environments, and demonstrate if the device is mobile or stationary, etc.
7) Based on previous point, a mash-up can be implemented with other web-based resources such as Google Maps.
8) Real time shipping of log data to a dedicated analytics service or time-series database for studying different behaviors based on the readings collected and other meta-data.

VIII. Conclusion

After experimenting how to build a proof of concept for smart gateway, the team is convinced that this architecture is highly efficient and effective, and through it, we are not only able to communicate to ultra-low power devices and resource constrained ones, but to create a management center that can store configuration of varied types of devices, and to extend to fulfill future requirements easily. We also demonstrate that a smart-gateway will not only contribute to having more resource constrained devices connected to the Internet, but also it can be a very useful reference of data sources for data scientists.

Big Data technologies represent a shift in terms of programming approach, and their promise produce an increasing interest within the data/information management community. But proposed solutions are still immature, and first experimentations show that they require incremental development and testing stages to improve performance. In our military intelligence context, Big Data performance is critical if these technologies are to be used in tactical environments.

REFERENCES

[1] *Internet of Things - converging technologies for smart environments and integrated ecosystems*, River Publishers Series in Communications, O. Vermesan and P. Friess, 2013.
[2] D. Clark, "NewArch: Future Generation Internet Architecture", NewArch Final Technical Report, [online] http://www.isi.edu/newarch/
[3] M. Blumenthal and D. Clark, "Rethinking the design of the internet: the end to end arguments vs. the brave new world," *ACM Transactions on Internet Technology*, vol. 1, issue 1, pp. 70-109, Aug. 2001.
[4] A. Feldmann, "Internet clean-slate design: what and why?", *ACM SIGCOM Computer Communication Review*, vol. 37, issue 3, pp. 59-64, 2007.
[5] M. Andersson, "Short-range low power wireless devices and internet of things (IoT)," connectBlue, February 2014. [online] http:// www.connectblue.com
[6] D. Blaauw, D. Sylvester, P. Dutta, Y. Lee, I. Lee, S. Bang, Y. Kim, G. Kim, P. Pannuto, Y.-S. Kuo, D. Yoon, W. Jung, Z. Foo, Y.-P. Chen, S. Oh, S. Jeong, and M. Choi, "IoT design space challenges: circuits and systems," *Symposium on VLSI Technology (VLSI-Technology)*, 2014.
[7] B. Martinez, M. Monton, and J. Daniel Prades, "The power of models: modeling power consumption for IoT devices," *IEEE Sensors Journal*, vol. 15, issue 10, pp. 5777–5789, 2015.
[8] T. Zachariah, N. Klugman, B. Campbell, J. Adkins, N. Jackson, and P. Dutta, "The internet of things has a gateway problem," *HotMobile'15 Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, New Mexico, USA, February 12–13, 2015.
[9] *Low power networks hold the key to IoT*, Rethink Technology Research Ltd, Published June 2015.
[10] *6LoWPAN demystified*, J. Olsson, Texas, USA, October 2014.
[11] *Z- Z-Wave Wireless Communications for Smart Devices and IoT*, Sigma Designs inc., October 2014.
[12] J. Ko, A. Terzis, S. Dawson-Haggerty and D. E. Culler, "Connecting low-power and lossy networks to the internet," *IEEE Communications Magazine*, April 2011.
[13] *RESTful Web Services for the Internet of Things*, M. Laine, Finland, 2011.
[14] H. Petersen, E. Baccelli, and M. Wahlisch, "Interoperable services on constrained devices in the internet of things," *W3C Workshop on the Web of Things*, Berlin, Germany, June 2014.
[15] M. Kumar, "Realization of ultra low voltage circuit design for internet of things," *Journal of Electron Devices*, Vol. 21, pp. 1801-1805, 2015.
[16] M. Zorrilla, A. Martin, J. R. Sanchez, I. Tamayo, and I. G. Olaizola, "HTML5-based system for interoperable 3D digital home applications," *Fourth International Conference on Digital Home*, 2012.
[17] X. Liu, K. Tang, J. Hancock, J. Han, M. Song, R. Xu, V. Manikonda, and B. Pokorny, "SocialCube: A text cube framework for analyzing social media data," *International Conference on Social Informatics*, 2012.
[18] J. E. Stewart, "Implementing an XML authoring project in a new media course," *IEEE International Professional Communication Conference (IPCC)*, 2014.
[19] J. Q. Zhang and K. W. Xie, "Development and application of streaming media courseware based on XML," *Ninth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2013.
[20] *The little Redis Book*, K. Seguin, 2012. [online] http://openmymind.net/redis.pdf.
[21] *Redis Cookbook*, O'Reilly Media, T. Macedo and F. Oliveira, USA, 2011.
[22] Arduino website. [online] https://www.arduino.cc/en/main/software.
[23] *Arduino Programming Notebook*, First Edition, B. W. Evans, August 2007.
[24] M. Schwartz, "5 Arduino Starter Kits Reviewed," 29 March 2016. [online] https://openhomeautomation.net/best-arduino-starter-kit
[25] J. Roy and A. Auger, "Collective C2 in multinational civil-military operations," *the Multi-Intelligence Tools Suite – Supporting Research and Development in Information and Knowledge Exploitation, 16th International Command and Control Research and Technology Symposium (ICCRTS)*, Québec City, Canada, June 21-23, 2011.
[26] M. Á. Serna, C. J. Sreenan, and S. Fedor, "A visual programming framework for wireless sensor networks in smart home applications," *IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2015.
[27] T. Kovácsházy, G. Wacha, T. Dabóczi, C. Erdős, and A. Szarvas, "System architecture for Internet of Things with the extensive use of embedded virtualization," *IEEE 4th International Conference on Cognitive Infocommunications (CogInfoCom)*, 2013.