Context-aware Control Platform for Sensor Network Integration in IoT and Cloud

Daniel MEREZEANU, Gheorghe VASILESCU, Radu DOBRESCU

Faculty of Automatic Control and Computer Science,
University Politehnica of Bucharest
313, Splaiul Independentei, 060042, Romania
danmerezeanu@gmail.com

Abstract: The main goal of the paper is to integrate three emergent technologies (Wireless Sensor Networks, Internet of Things and Cloud Computing) in a framework supporting context-aware sensing, computing and communication capabilities into industrial applications. In this regard, the paper presents original solutions for a context-aware three-tier system architecture, that provides context information sensing and processing, an access mechanism for interfacing sensor networks with IoT and Cloud and a four-level architecture to perform industrial process control, that includes the modules of a context-aware control platform. These solutions are validated with a proof of concept application implemented on a IBM Bluemix IoT platform.

Keywords: Wireless Sensor Networks; Internet of Things Cloud Computing; context-aware systems; access mechanism; sensor-cloud interface; web services.

1. Introduction

Sensor networks have rapidly imposed usefulness in a variety of applications, such as in control of industrial processes, in environmental monitoring, or in active and assisted living. We find them in Supervisory Control and Data Acquisition (SCADA) systems, mostly as Wireless Sensor and Actor Networks (WSANs). However, because SCADAs used more and more programs with expensive proprietary hardware, software and communication protocols, new solutions that make use of the common Internet protocols have proven to be more effective. In this complex open service environment, the way to integrate heterogeneous sensor networks is very important. Some authors advanced the name of Ubiquitous Sensor Network Environment (USN) [10]. USN provides integration of data from various sensors, sensor data and context information processing, sensor network monitoring and intelligent events handling. USN enables the competitive Internet of Things (IoT), mediating interactions with the outside world in pervasive ways.

IoT consists of large-scale, distributed multi-agents placed in dynamically changing environments that present wireless connectivity. In the same time, with the development of wireless based Internet devices presenting ubiquitous availability, more and more entities included in IoT are becoming context-aware. These devices can perceive their surroundings based on the gathered context information.

If in its beginning IoT vision was to enable devices to transfer data from various objects into the web, today this vision aims at enabling smart and context-aware applications at global scale extension of pervasive computing paradigm. Therefore, context-aware architectures must be designed to respond to the various demands of users and applications by offering the possibility to acquire, analyze and interpret relevant context information, and to offer adequate feedback to contextual changes.

The purpose of this paper is to propose a competitive architecture of a context aware system that can allow the access of agents connected in sensor networks to the IoT, starting from the evidence that such objects have data and ubiquitous web services need to access, learn and interpret this data.

In the same time, the opportunities to use Cloud Computing facilities were analysed, as well as the distribution of platform modules of the proposed platform in different layers of the Sensor Network-IoT-Cloud global software architecture. The proposed architecture provides the technology to connect these three conceptual paradigms in process control applications. A proof of concept IoT application is described finally.

2. Related Work

Our research objectives are linked to two emerging areas of study, one related to the use of the context awareness concept in evaluation of dynamical systems evolution, the other describing the impact of the intensive use of IoT and Cloud Computing technologies in the management of industrial processes. In particular, our objective was to place sensor networks in a position that can take advantage of the facilities offered by the top achievements in these two areas.

Among the works on context-awareness we have chosen a few recent ones which are considering the possibility of IoT integration and the use of Web Services. Celikkan and Kurtel [4] propose a context-awareness based software platform modeled as a layered architecture, having extensibility as main facility. Boddhu *et al* analyse the integration of Human-centric sensing paradigm in a generic architecture able to enhance situational awareness [2].

In [8] the authors present a context aware assistance system using video information to avoid critical situations in industrial environment. A solution for integration in IoT of hundreds sensor nodes of wireless sensor networks sensor nodes is presented in [7]. The authors describe a middleware architecture that uses context information to provide resource management in heterogeneous sensor networks. The advantages of context-aware computing in providing situation specific services are presented in [11]. The authors present a solution to infer situations from contexts acquired from various sources based on a cloud service defined as INFerence-as-a-Service (INFaaS). Perera *et al* present in [16] a context-aware sensor model designed for efficiently selecting relevant sensor information from enormous amounts of data acquired with embedded sensors of IoT. The paper highlights the importance of combining semantic and quantitative reasoning in searching and selecting information.

The literature on integration of sensor networks in IoT and Cloud is much richer, so we will mention only a few recent works that emphasize also issues of Web Services utilization. A new methodology based on Timed Colored Petri Net was proposed in [6] with the aim to promote a standard solution for introducing and implementing sensor technologies in IoT. In [17] the aim of the authors is to add artificial intelligence techniques in the management of IoT systems. Another important theoretical issue discussed in [14] is the developing of mathematical models for the virtualization of sensor node resources. The characteristics of architectures based on Open Service Gateway and the advantages of using them for service management in IoT are discussed in [22].

A mixed CPS (Cyber Physical Systems) and SOA (Service- Oriented Architecture) for implementation of a layered architecture with five levels of abstraction is proposed in [13].

Another research direction is the approach of Artificial Intelligence techniques for communication in clusters of multi-agent systems, such in [3], [15].

3. Context-aware System Design

Context-aware systems (CAS) are "systems that are able to adapt their operations to context changes without explicit user intervention" [18]. They concentrate on modelling the context information, defined as information that can be used to characterize the situation of an entity. Context-awareness is the ability of a system to provide relevant information to users using context information, where relevance depends on the user's task.

In order to design CAS, one can use a model that explicitly supports the definition of the expected functionalities, context information and management actions and also supports the generation of the context-aware adaptive software implementations.

Figure 1 shows the model for a context-aware system, that includes several components of three main subsystems: Context Sensing, Context Information Management and Context-Aware Services Management, placed in a three-tier architecture.

This architectural model was implemented as software component of a Context-aware Control Platform which be presented in details in the next section.

The low-level layer (which includes the Context Sensing subsystem) interacts with the surrounding environment to learn the context

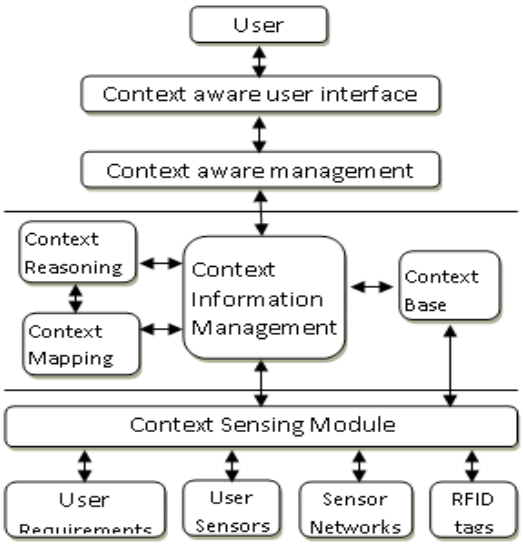

Figure 1. Context aware three-tiers architecture

information about a user or any entity and supports sensor communication, to provide the context data required by specific applications. It also retrieves user's requirements and preferences. In order to ensure update of context information it is able to make this either periodically (push mechanism) or on demand (pull mechanism).

The middle layer (which includes as kernel the Context Information Management module) contains also other three modules: Context Base, Context Reasoning and Context Mapping, all managed by the central module.

The Context Base stores acquired context data by from the low-level layer. It contains static, transient, or persistent contextual information. The static information is invariant and so never changes. The transient data content is updated at run-time. The persistent information represents historical data. The context base itself is managed and maintained by Context Information Management subsystem, which has similar functionalities to a database management system. In the particular case of industrial processes control an efficient solution is to structure this database according to the OPC-UA (OPC Unified Architecture) standard and so to offer an open library of complex reusable function blocks [5]. This approach offers the possibility to enrich our knowledge-base with relevant data elements coming from various sources and additionally to use historical data analysis, by means of data mining techniques, in order to identify hidden patterns in the parameters that control industrial processes.

The Context Reasoning module provides a decision making mechanism. It queries the Context Base and uses inference rules to generate a set of relevant information; then it interprets the information associated with an object in the system. In this respect it uses a classical technique of Artificial Intelligence, based on an inference engine which allows to infer the requested context from raw context data. The inference engine can use various reasoning technique to achieve this.

The Context Mapping is responsible for mapping the contextual information in an appropriate format, so that it can be shared or reused by other components.

The uppermost layer includes the Context-aware Services Management module and the Context-aware User Interface Management module.

The Context-aware Services Management component is responsible for providing the services to the user. It incorporates information from the Data and Knowledge Bases of the middle layer and enables the user to model and generate the system implementations and validate its context-aware adaptive behaviour. It contains a model directory able to maintain model specifications of particular situations and events using contextual constraints. Depending on the context, a different set of candidate strategies can be selected and composed into a custom-generated global adaptation module matching the current environmental conditions. Like the already mentioned approach of using pre-determined functional blocks, this global adaptation module can evolve by dynamically composing reusable adaptation modules in response to context changes.

The Context-aware User Interface Management module includes, in addition to standard elements, a dedicated Interface Base containing building user interface blocks which can be selected according to user preferences to build the user interface. Of course, the interface corresponds to the type of device with which user interacts (eg. PDA device, smartphone, SCADA component, etc).

As one can easily observe, in the design of our CAS the integration and use of Artificial Intelligence (AI) techniques were taken into consideration. From this point of view, because the main objective of the paper is the use of sensor networks in the IoT is important to

judiciously choose the placement of the AI support in the IoT system information flow. Systems built using the concept of Internet of Things are increasingly complex and can make the decisions even in cases of uncertainty, hence the idea of the use of AI techniques, but also the need for appropriate placement of AI methods in the context of IoT architectures. The most natural places for AI methods are all kinds of servers because of their computing power. Such location has a positive effect on another aspect too. It is about the fact of reuse.

There are three modes of communication in the flow of information:

- external communication (data sent from the real world towards data processing systems).
- contextual communication (bidirectional transmission of data processed by systems that are associated or integrated in IoT systems and IoT devices which can respond after taking appropriate decisions).
- internal communication (data transmitted on direct communication channel between intelligent components of various IoT devices).

In our CAS system we tried to capitalize AI elements on all three levels, but the focus was on internal communication, especially at the top level. More exactly, we suggest to use contextual communication combined with additional AI elements contained in the intelligent devices having a high degree of autonomy. This approach is also an objective for future research, when we hope to transform our Context-aware System architecture in a Self-adaptive System (SAS) architecture, considering SAS as a system that modifies its own structure and behaviour in response to changes in its operating environment.

4. Access Mechanisms in IoT and Cloud

Our goal was to integrate the proposed context-aware architecture in applications based on IoT, as an effective service-based software platform following a middleware approach, named Context-aware Control Platform (CaCP). The platform separates context acquisition from the application code and handles many context data management issues on behalf of the applications. As middleware component the platform provides several services for applications such as query processing, sensor data management, sensor network monitoring, context-aware information processing, etc. Its role is to make available for applications data acquired from specific sensor networks and in order to provide web services. Therefore, middleware resources can be accessed by applications via web service interfaces.

The main advantages of using this layered architecture is that it allows reconfiguration and extensibility without modifying the applications. However, meeting the hard real-time requirements is difficult in layered architectures, as data flows through several layers before it reaches the application. Therefore, the proposed context-aware architecture is more suited to soft real-time requirements, because it can tolerate larger time latencies and can still operate adapting to environment changes.

4.1 IoT access modeling

The first objective in using CaCP is to realise the access in an IoT framework, as an entity which can interact with both IoT domain and ambient environment. This entity constitutes a *"thing"* in the Internet of Things, and is the main focus of interactions between software agents. It is represented by two components: the hardware and respectively the software one. The hardware component of the entity is named *"device"*. The device attaches the entity to its environment in order to monitor it. The software component that provides information on the entity and enables the control of the device is a *"resource"*. The resources exploitation is dependent on the actual hardware support of the device, so a *"service"* is necessary to provide all functionalities for interacting with entities and related processes, by the means of a standardised interface. The services ensure the functionality of a device by accessing its resources. The relations between entities and services are named *"associations"*.

Figure 2 illustrates the relations between the four elements of an IoT access model.

Let's note that the services provided by the access model in Figure 2 corresponds to the basic Semantic Web Services. According to OWL specifications [1], Semantic Web resources provide services which are described by their service profile, service model and service grounding.

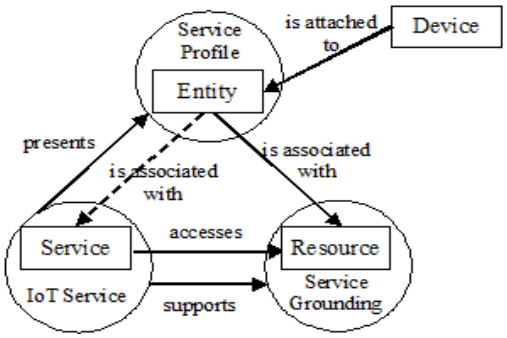

Figure 2. Interactions of components in a basic IoT access model

A service profile must contain information about the entity it is associated and on the link to the resource offering this service. The service profile describes services by their inputs, outputs, preconditions, and effects. A service which needs an input to be processed by a resource formulates this need by a property. According to this property the service is linked to an entity. By their action services change properties of entities from an initial state (specified as precondition) to a desired state (specified as effect).

IoT users have access to resources over the IoT through a suitable access interface which address a Web Service. The technical details that users need to access the service are specified in the service grounding. The service grounding specifies the mapping from domain specific entity attributes to properties observable by sensors. To each attribute of the entity assigned in the service profile is assigned a specific property (for example a measurement type).

4.2 Cloud access modeling

Harnessing the benefits of using the Internet in conducting effective services would be incomplete if it would not be used the advantages provided by Cloud Computing technology. The emerging Internet of Things usage also brings many applications. Cloud technology supports to process large scale data and enhances the rapid elasticity resulted from the increase of the number of network connected IoT devices and therefore of the huge amount of information to be processed. Our aim was to facilitate the access in IoT of sensor networks, so to transform the existing sensing infrastructure according to a cloud-centric IoT paradigm. According to Kantarci *et al*, the cloud-centric IoT paradigm [9] enables offering sensing resources and instruments as a service through cloud platforms. They use in this aim a cloud-inspired service model introduced by Sheng *et al.* under the name Sensing as a Service (S^2aaS) [19]. Since sensing is provided as a service over the cloud, sensors need to be virtualized to enable access of multiple users who can utilize the same sensing hardware concurrently without any interference. This opportunity was addressed by many researchers, offering solutions known as a CoS (Cloud of Sensors) [20], SCI (Sensor Cloud Infrastructure) [23] or simply SC (Sensor Cloud) [12]. All these solutions have sensors and a cloud as an integral part of their architecture. IoT itself contains a CoS solution, while enables pervasive and ubiquitous interconnection with different remote devices (mostly sensors) on a large scale.

In the particular case of integration of a context-aware middleware platform, its components can be distributed into different cloud levels of abstraction appropriate to the type of service they perform, as is depicted in Figure 3, where are included also the other hardware and software components of a context-aware control application, described in details in the following section.

1. The Sensor as a Service (S^2aaS) level allows the access from hardware components (individual sensors, sensor

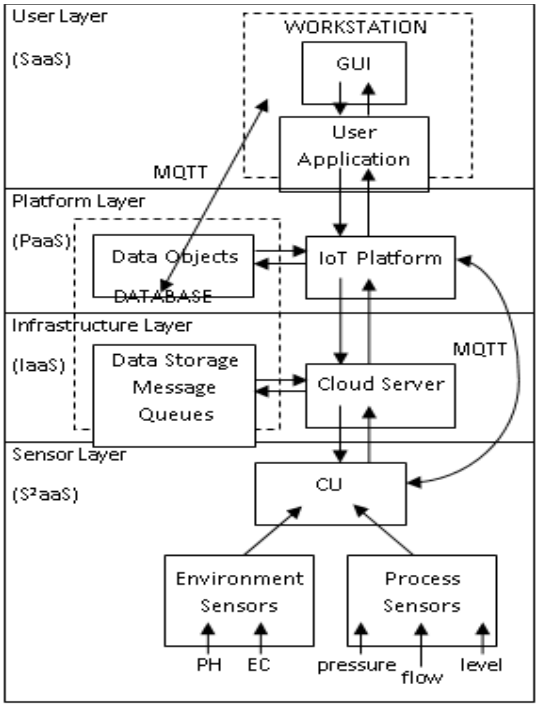

Figure 3. Distribution of a CoS structure associated with context-aware middleware

nodes of sensor networks, RFID tags) through a Sensor Network Access Interface. The physical sensors and the communication interface are delivered by sensor providers.

2. The Infrastructure as a Service (IaaS) level offers for the users the cloud infrastructure including network, servers, operating systems, storage. Thus, consumers can deploy and run arbitrary software for specific applications.
3. The Platform as a Service (PaaS) level allows the users to deploy onto the cloud infrastructure new created or acquired applications using their familiar programming languages, libraries, services, and tools supported by the provider. At this level is placed the software component of the context-aware middleware platform.
4. The Software as a Service (SaaS) level provides all necessary software for applications, including messaging software, management software, development software, and so on. It contains also dedicated interfaces with human operators.

5. Proof of Concept IoT Application

As an example of using the context-aware control platform (CaCP) in an IoT assisted application we proposed a solution for automatic control of an irrigation system designed for small and medium farms. Irrigation equipment was constructed to operate automatically and ensure proper management of water, nutrients and pesticides. Besides the measurement of controlled output signals (pressure, level, flow) are measured environmental parameters (soil properties, cropping patterns and unexpected events) too, making the system to be context-aware of surrounding changes. The context information is evaluated considering soil parameters: PH, humidity and conductivity, and meteo parameters: temperature, air pressure and wind speed. System requirements for control and monitoring are provided with reduced human operator intervention. Such a system is built around a central module (controller) which implement control functions, sensor data acquisition and data transmission to a supervisor, which will be named in the following Command Unit (CU). This unit embeds a part of the hardware component of CaCP. CU is implemented around a 32-bit microcontroller. The program running on the microcontroller controls the hardware resources of equipment, being able to provide both autonomously and via a IoT platform. Thus, pressure, flow and level are the controlled process variables, which must be maintained in a recommended working zone within acceptable limits, while soil PH, soil conductivity (EC) and air temperature are considered context information. Crop irrigation is done in accordance with user-defined schemes. These scenarios can be adjusted depending the context changes by the means of the middleware component of CaCP. Figure 3. illustrates the distribution of the hardware and software components in the general framework of cloud computing support.

The control programs for regulating physical parameters, running on CU, control the actuators of the irrigation system (pumps, solenoids). To monitor the entire system was used a solution that realizes real-time process supervision by integration of the controller on the IoT Bluemix platform developed by IBM [21]. This logistic support of PaaS type was launched in 2014 and allows, in addition to the previously mentioned features, the development of applications in various programming languages (Java, Python, SQL, PHP, etc.) and integration and interconnection of several IoT devices. The platform performs also the functions of Cloud server being based on "open source" Cloud Foundry technology, so in particular it can be used for data storage. In order to monitor parameters of interest in agriculture, module control and data acquisition will transmit the acquired data to the platform Bluemix, following that, through an application, they are stored both in the cloud and in a database of the beneficiary, with the purpose of preparing regular reports on process evolution indicators.

Regarding application development, the user has two possibilities for: writing apps directly on the platform (this is not available for all applications) or writing them in a specific programming environment that is compatible with the chosen platform. Bluemix specify where applications can be developed to easily make contact with IoT environment, as well as useful links to download the installation kits of those programs. The platform has a menu where the customer is informed of the

steps to follow when loading an application in this IoT environment.

An essential aspect to take into account when using Bluemix to develop monitoring applications is the configuration of MQTT (Message Queue Telemetry Transport) protocol, which provides a client – server communication. This standard was developed by IBM over TCP / IP in order to achieve efficient real-time exchange of data from sensors and other mobile devices and rely on messages switching. MQTT defines five methods for client – server communication: "connect" (client request to connect to server), "disconnect" (client is disconnecting from the server), "subscribe" (client subscribe request), "unsubscribe" (unsubscribe request) and "publish" (publish the message). When one of these methods is chosen, its action is to be done on an identified resource. The resource can be either pre-existing data or dynamically generated data, depending on the way the server was implemented. Usually, the resource can be a file or an executable of the server. When developing an application, the programmer has to implement these methods, so the communication between client and server works.

Bluemix IBM platform uses version 3.1 of the Protocol, which is the one that is best suited to applications developed in the cloud and provides secure transfer of data. To develop an application. a programmer should consider MQTT standard adaptation for embedded devices, applications developed on the platform and gateways.

In our application, on the Bluemix platform was first integrated the controller of the irrigation system. As unique identifier code was assigned its MAC address by the means of the devices configuration menu provided by Bluemix. The platform enables sensor data storage in the cloud, and the graphical display of the evolution of measured parameters for all devices embedded by the user. For monitoring were developed two applications:

1. an application in the Java language for the acquisition in cloud of data from sensors and their storage in a database, and
2. an application in C# language for manipulating stored data and generating reports on the numerical evolution of the measured parameters.

5.1 Java application for the acquisition and storage of data in the cloud

For the development of this application was used Kepler Eclipse programming environment, after installing Oracle "javac" compiler on the workstation. By the configuration of MQTT protocol were defined the five mentioned methods for client – server communication. To simplify work, the program included several Java archive (.jar executable) for protocol configuration. To ensure the link with the database, was created with the support of the library *java.sql.** a JDBC (Java DataBase Connection) procedure which includes both the string for the connection to the database and the string for SQL Server driver. The drivers for databases development environments are selected according to manufacturers' specifications and the compatibility between programs. It was also envisaged the setting up of SQL Server environment by activation of the TCP/IP port and setting its default number in the SQL Server Configuration Manager menu.

Since the events transmitted by integrated device platform IoT are represented in JSON (*JavaScript Object Notation*), in the application has been added *java.json* library. With *messageArrived_PH* and *messageArrived_EC JSON* methods are processed strings of events to retrieve the numerical values of measurements made by sensors connected to the irrigation system controller. For loading the developed Eclipse application on the IoT platform is necessary installing Cloud Foundry command line interface (CLI CF). After installation, CF CLI can be called from the command line of the operating system.

For loading the application, the following steps are:

Step 1: >*cf login –a https://api.ng.bluemix.net*

When executing the above command, the user will have the possibility to authenticate using his Bluemix account. If authentication data are correct, the customer will have to choose one of its workspaces (when there is only one workspace created on the platform, the connection will be established directly with that space).

Step 2: > *cd ~ work Path folder*

To know the exact name of the file to be loaded, run the command line:

Step 3: > *ls -l*

If the returned list is found a file with the extension .war (Web Archive), run the following command:

Step 4: > *cf push Name_project Name_file.war*

After this final command, the application is loaded on the Bluemix platform. The URL returned in the command line of the operating system will offer independent access to the application for the client.

5.2 C # application for manipulating stored data

For storing data in the cloud, it was first created a database using SQL Server 2012 environment. The tables and the relationships between them were defined using SQ and other tools provided by the development environment. Database tables are: "Users" (for creating access accounts to data manipulation application), "Parameters" (definition of measured parameters), "Crops" (to measure parameters of interest of irrigated crops and to define locations where the measurements are taken), "Measurements "(values for measured parameters)," and "Resources" (typical assets for performance improvement such as fertilizers, pesticides, etc. But also software assets in the form of block functions). The tables are presented in Figure 4.

The database thus created was populated with appropriate records. For this was used the Visual Studio programming environment. This program provides programmers the ability to manipulate data stored in a database. Applications can be written in different programming languages (C + +, C #, Java, etc.) that allow the use of specific tools to work with databases.

The application for manipulating data acquired from Cloud contains eight *Windows Forms Application* (Form1 - Form8) whose functions were written in C#. When creating windows, Visual Studio automatically generates code skeleton. The latter includes those resources used by the keyword *"using"* and a structure of *"public partial class"* type in which one can define the desired objects and methods.

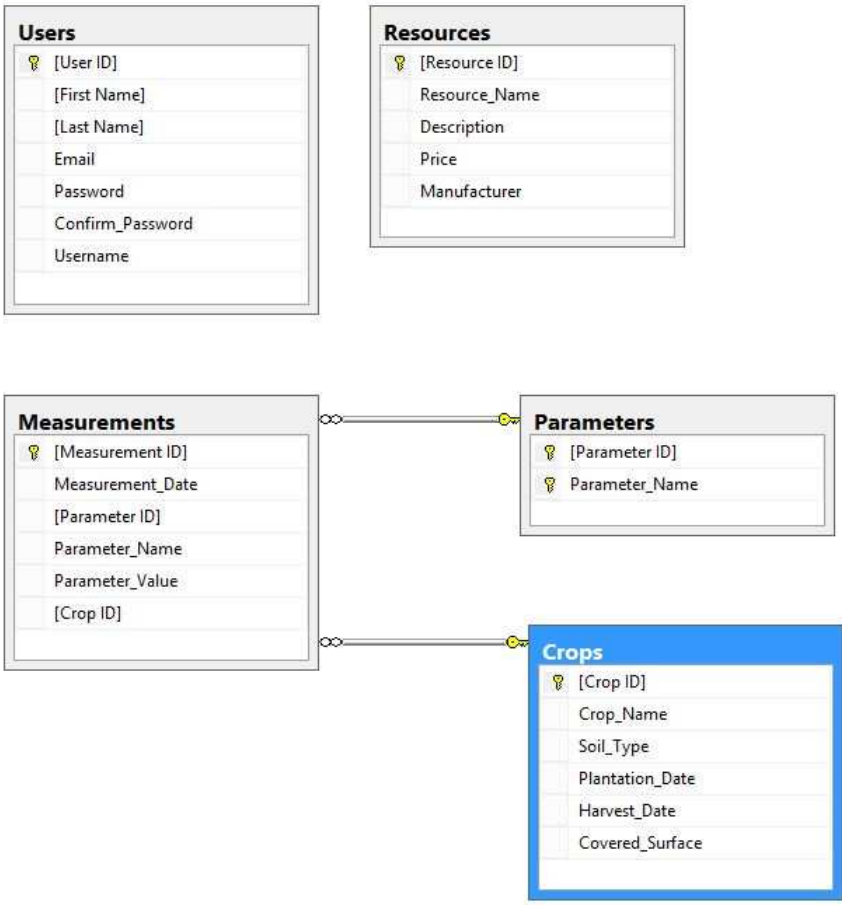

Figure 4. Database structure

6. Conclusions

The main conclusions derived from the analysis of experimental results of the application which has been utilized as a proof of concept, for controlling and monitoring an irrigation system connected to an IoT platform. For those tasks were elaborated a hardware Command Unit (CU) and a software application running on IoT platform.

UCCSI was built on the IBM Bluemix to monitor data from sensors. For storing the measured values of the IoT platform, it was developed a Java application that takes data from the process and the surrounding and transfers them in a database. Using a C # application running on a workstation (PC) and providing access to the database, the user can draw on the progress of the measured parameters and gives customers the opportunity to make a qualitative assessment of soil tillage and hence the development of crops.

We believe that the proposed solution, validated by the case study compelling results, has the merit that it can be easily adapted for applications no matter how complex, based on the integration of sensor networks in IoT and Cloud. This work can be seen as a network enabler who abstracts from the technical underlying sensor network layer and provided easy access to it. Higher Layers allow connect to middleware the context-aware information processing algorithms and respectively service provision from other more powerful component. In future work we will focus on two directions: to extend the use of previous stored software assets to provide various other applications and services and to transform our context-aware architecture in a self-adaptive architecture.

Acknowledgements

This work was partially supported by the Romanian National Research Program PNII, project 257/2014 – CALCULOS.

REFERENCES

1. BARNAGHI, DE, S., P., BAUER, M. AND MEISSNER, S, **Service Modelling for the Internet of Things**, Proc. of Federated Conference on Computer Science and Information Systems, 2011. pp. 949-955.
2. BODDHU, S., R. FLAGG, P. GRZEBALA, R. BODDULURI, R. WILLIAMS, **A Generic Sensor Fusion Architecture for enhancing Situational Awareness**, IEEE National Aerospace and Electronics Conference, 2014, pp. 143-148.
3. CAMARIHNA-MATOS, L. M., et al. **A Comprehensive Research Roadmap for ICT and Ageing**. Studies in Informatics and Control, 2013, vol. 22 (3) pp. 233-254.
4. CELIKKAN, U., K. KURTEL, **A Platform for Context-Aware Application Development: PCAD**, Proc. of the Federated Conference on Computer Science and Information Systems, 2015, pp. 1481-1488.
5. CHENARU, O., A. STANCIU, V. SIMA, D. POPESCU, G. FLOREA, R. DOBRESCU, **Modeling Complex Industrial Systems Using Cloud Services**, 20th International Conference on Control Systems and Computer Science (CSCS), 2015, pp. 565-571.
6. DAVOUDPOUR, M., A. SADEGHIAN, H. RAHNAMA. **CANthings: Context-Aware Networks for the Design of Connected Things**, IEEE International Conference on Semantic Computing (ICSC), 2015, pp. 127-130.
7. GANZ, F., P. BARNAGHI, F. CARREZ, AND K. MOESSNER, **Context-Aware Management for Sensor Networks**, Proceedings of the 5th International Conference on Communication System Software and Middleware, 2011, pp. 1-6.
8. IMTIAZ, J., N. KOCH, H. FLATT, AND J. JASPERNEITE, **A Flexible Context-Aware Assistance System for Industrial Applications Using Camera based Localization**, IEEE Emerging Technology and Factory Automation, 2014, pp. 1-4.
9. KANTARCI, B., H. T. MOUFTAH, **Sensing Services in Cloud-Centric Internet of Things: A Survey, taxonomy and challenges**, IEEE International Conference on Communications, Workshop on Cloud Computing Systems, Networks and Applications, 2015, pp. 1865-1870.
10. KIM, M., H. BANG, H. J. GAK, C. S. PYO, **The Access Control Model in**

Ubiquitous Sensor Network Environment, Fifth International Joint Conference on INC, IMS and IDC, 2009, pp. 739-744.

11. KIM, M. K., S. D. KIM, **Inference-as-a-Service: A Situation Inference Service for Context-Aware Computing**, International Conference on Smart Computing, 2014, pp. 318-324.
12. KOTHARI, A., V. BODDULA, L. RAMASWAMY, N. ABOLHASSANI, **DQS-Cloud: A Data Quality-aware Autonomic Cloud for Sensor Services**, International Conference on Collaborative Computing: Networking, Applications and Work Sharing, 2014, pp. 295-303.
13. LEE, J., B. BAGHERI, H. KAO, **A Cyber-physical Systems Architecture for Industry 4.0-based Manufacturing Systems**, Manufacturing Letters, vol. 3, no. 1, 2015, pp. 18-23.
14. MISRA, S., S. CHATTERJEE, M. OBAIDAT, **On Theoretical Modelling of Sensor Cloud: A Paradigm Shift from Wireless Sensor Network**, Systems Journal, IEEE, vol. PP, no. 99, 2014, pp. 1-10.
15. NOROUZI, A.; F. S. BABAMIR; A. H. ZAIM, **An Interactive Genetic Algorithm for Mobile Sensor Networks.** Studies in Informatics and Control, vol. 22(2), 2013, pp. 213-218.
16. PERERA, C., A. ZASLAVSKY, C. H. LIU, M. COMPTON, P. CHRISTEN, D. GEORGAKOPOULOS, **Sensor Search Techniques for Sensing as a Service Architecture for the Internet of Things**, IEEE Sensors Journal, vol. 14, No. 2, 2014, pp. 406-420.
17. PONISZEWSKA-MARANDA, A., D. KACZMAREK. **Selected Methods of Artificial Intelligence for Internet of Things Conception**, Proc. of the Federated Conference on Computer Science and Information Systems, ACSIS, Vol. 5, 2015, pp. 1343-1348.
18. SAREMI, A., **CaDSS: A Framework for Enhancing Decision Making in Pervasive Computing Environment**, Third European Symposium on Computer Modeling and Simulation, 2009, pp. 17-22.
19. SHENG, X., J. TANG, X. XIAO, G. XUE, **Sensing as a Service: Challenges, Solutions and Future Directions**, IEEE Sensors Journal, vol. 13(10), 2013, pp. 3733-3741.
20. SKOWRONSKI, A., J. WEREWKA. **A Quality Attributes Approach to Defining Reactive Systems Solution Applied to Cloud of Sensors**, Proc. of the Federated Conference on Computer Science and Information Systems, ACSIS, Vol. 5., 2015, pp. 789-796.
21. TANG, C. B., **Explore MQTT and the Internet of Things Service on IBM Bluemix**, IBM tutorial, available on site: https://www.ibm.com/developerworks/cloud/library
22. WILUSZ, D., J. RYKOWSKI, **Comparison of Architectures for Service Management in IoT and Sensor Networks by Means of OSGI and Rest Services**, Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, ACSIS, vol. 2., IEEE, 2014, pp. 1207-1214.
23. YURIYAMA, M., T. KUSHIDA, **Sensor-Cloud Infrastructure – Physical Sensor Management with Virtualized Sensors on Cloud Computing**, 13th International Conference on Network-Based Information Systems, 2010. pp. 1-8.