A Lightweight Platform Implementation for Internet of Things

Young Ju Heo, Sung Min Oh, Won Sang Chin, Ju Wook Jang
Department of Electronic Engineering
Sogang University
Seoul, South Korea
{protoheo, dangkuk, mokey82, jjang}@sogang.ac.kr

***Abstract*— As the number of IoT devices explodes, we observe numerous IoT platforms which enables us to build applications, manage, and integrate things connected to the Internet. We identify key technologies embedded in contemporary IoT platforms as follows: virtualization, restful API, database for history, visualization, and mashup. Most IoT platforms are heavy-weight in terms of software or hardware and often require proprietary protocols. In this paper, we introduce a lightweight implementation of IoT platform which can be easily built from off-the-shelf components and windows OS, still realizing all the above key technologies as found in most heavy-weight IoT platforms. In addition, we make our IoT platform smart by adding context-awareness.**

***Keywords*— Internet of things; Node. js; lightweight; visualization; virtualization; Context**

I. INTRODUCTION

Iot platforms virtualize devices connected to the Internet and provide a growing number of services that can be shared among various IoT applications. Features may differ depending on various platforms most of them share a few key technologies or features: virtualization, restful API, database for history, visualization, and mashup. It is our observation that IoT platforms should include context-awareness to be more smart, involving as little human intervention as possible.

The virtualization of real world objects in the framework is realized through the creation of virtual objects which are semantically enriched with context related information. Additionally, virtual objects of different types can be combined in a more sophisticated way by forming composite virtual objects, which provide services to high-level applications and end-users.[1]

Restful API is a software architecture style consisting of guidelines for creating scalable web services. REST is a coordinated set of constraints applied to the design of components in a distributed hypermedia system that can lead to a more performant and maintainable architecture.[2] Xively, Thingworx and Axeda use RESTful API.

Platforms which are using RESTful API communicate over the Hypertext Transfer Protocol with the same HTTP verbs (GET, POST, PUT, DELETE, etc.) used by web browsers to retrieve web pages. [3]

Database for history: IoT network brings a series of challenges for data storage and processing. IoT data can be generated quite rapidly, the volume of data can be huge and the types of data can be various. So, IoT platforms are required to be equipped storage system for IoT data.[4] For example, Thingsquare provides smartphone application of IoT data's cloud service.

Visualization is technique for creating images, diagrams, or animations. Visualization is critical for an IoT application as this allows the interaction of the user with the environment. For example, Xively provides drawing chart API.

A mashup used in web development is a web page. It uses contents from sources to create a sole new service. For example, combing the addresses and photographs of library branches with a Google map creates a map mashup.

A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.[5] For providing context based service, context reasoning should be executed. Context reasoning has the following four models; Key-values model, Logic-based model, Ontology-based model and Probabilistic Graphical Model. Other platforms don't consider context, but we implement control algorithms by using Logic-based model.[6]

In this paper we address our lightweight platform which has above-mentioned key technologies implementation for IoT. In chapter 2, we introduce some related works and chapter 3, how we make the lightweight platforms, lastly chapter 4 we mentioned conclusion and future works.

II. RELATED WORK

A. Xively

Xively is IoT data service delivery platform. It collects real-time data from the connected sensors and devices as well as enable mashup service for the devices. Also it supports a variety of open-source hardware devices and provides a RESTful-based open API. Through this API, user could exploit such as management of connected devices and drawing a graph of stored data. Moreover Xively supports identifying registered devices by ID.[7]

978-1-4673-8103-1/15 $31.00 © 2015 IEEE
DOI 10.1109/FiCloud.2015.29

B. *Thingsquare*

Thingsquare supports open source Contiki OS-based devices and the cloud server platform. It is possible to access data in the cloud server through the smart phone application and monitor the status of the device. Thingsquare also provides a Web-based development environment to compile and install the software by online. It also supports the mesh routing, IPv6 addressing and connection to the cloud server via the gateway. [8]

C. *Thingworx*

Thingworx supports various protocol and provides a search engine for searching collected data and connected devices. Thingworx are focused on the service using the integration, modification, representation of the data. In addition, developers can create applications using the data from the platform via a user-friendly interface. Thingworx supports interlocking interface with business systems such as SAP, Oracle, and Salesforce. And it also supports cloud services and social services. [9]

D. *Axeda*

Axeda is IoT cloud service platform based business domain. It enables application services, integration framework, and data management. The Axeda platform provides connectivity between devices and objects. It also provides data acquisition, storage, device management, monitoring, provisioning and firmware update. Also message based data delivery is available when events has occurred between external service and Axeda platform.[9]

III. DESIGN OF HARDWARE AND SOFTWARE COMPONENT

Figure 1 shows the lightweight platform of a smart home sensor network system that we have developed. Open source hardware, Arduino, executes role of sensor/actuator nodes. We have run a stand-alone web server through Node.js without need that additional server like Apache. At this time, node can communicate with server for transfer data using WiFi or Ethernet.

TABLE 1. COMPARISON OF IOT PLATFORMS

	Xively	Thing worx	Thing square	Axeda	Proposed
Virtualization	O	O	O	O	O
Restful API	O	O	X	O	O
Storage	O	O	O	O	O
Mashup Service	O	O	X	X	O
Visualization	O	O	O	X	O
Context-aware	X	X	X	X	O

We focus on to design lightweight platform and it takes a short time. However, our platform has component of all IoT; Efficient data base, virtualization, visualization and mashup service. (Figure 2)

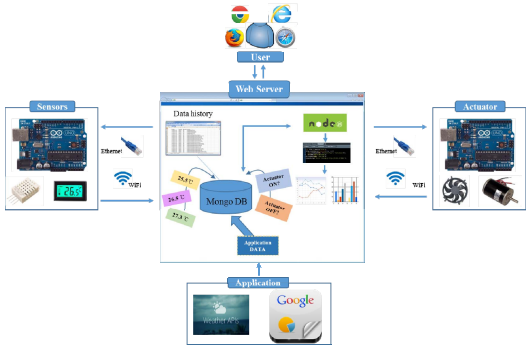

Figure 1. Overall platform architecture

A. *Sensor & Acturator*

First, the temperature & humidity sensor consists of Arduino Uno R3 combine with DHT11. DHT11 includes a resistive-type humidity measurement component and a NTC temperature measurement component, and connects to a high performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost-effectiveness [10]. (Table 2)

Arduino that connect with DHT11 continually transfer data which is temperature and humidity information. Temperature and humidity are displayed on a serial LED display.

To control the actuator, we connect LED/DC motor into Arduino. LEDs perform the role of the illumination and heater. DC motor perform the role of the fan. User can activate or deactivate these actuators through the web page.

Also, through the internet, Arduino needs to transfer sensor data to server or receive request of server. Therefore, we uses Ethernet/WiFi shield for using Internet.

TABLE 2. DHT11 SPECIFICATIONS

Measurement Range	Humidity Accuracy	Temperature Accuracy	Package
20-90% 0-50℃	±5%	±50℃	4Pin Single Row

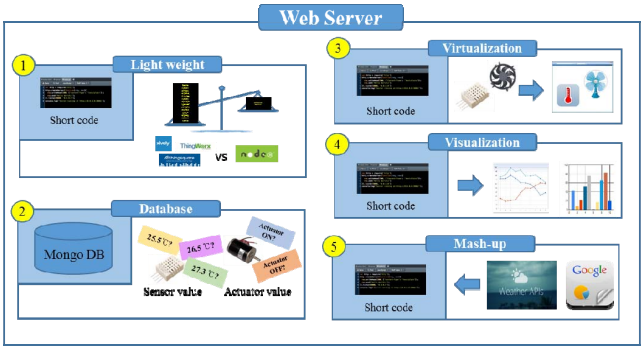

Figure 2. Feather of our IoT platform

B. Server using Node.js & Virtualization

Node.js is a server-side JavaScript environment. It's based on Google's V8 engine. V8 and Node are mostly implemented in C and C++, focusing on performance and low memory consumption.[11] Also, many libraries enable to easily run server. The example in Figure 3 shows how easily developers can build a simple asynchronous, event-driven network server. Through a web page, user can check status of sensor and actuator and virtualize them.

In a web page, virtualized sensor and actuator are displayed as an icon. Through a web page, user can activate/deactivate devices, but also it is possible to integrate them and increase manageability.[12] We have implemented a context-based control through virtualization (Chapter 3-F).

When a node transfers sensor data to server, server stores the data in a database, MongoDB. Explanation of a database is mentioned in next the chapter 3-C.

C. Database-MongoDB

MongoDB is a cross-platform document-oriented database. Classified as a NoSQL database, MongoDB is different from the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas, making the integration of data in certain types of applications easier and faster[13].

For this reason, we used MongoDB as a database for store data. When actuator/sensor nodes transfer their data (status, temperature and humidity), server store the data with timestamp. (Figure 4)

MongoDB services Node.js API documents so that developers easily use the database with the help of modified CRUD (Create, Read, Update, and Delete). For easily edit or view data, we chose Robomongo program. Figure 5 shows that device, status and date in the database using Robomongo.

```
var MongoClient =
require('mongodb').MongoClient;
var mongoclient = new MongoClient(new
Server('localhost',27017,{'native_parser':true}));

mongoclient.open(function(err, mongoclient) {
        http.createServer(app).listen(app.get('port'
), function(){
                console.log(Node.js Server
listening on port ' + app.get('port'));
        });
});
```

Figure 3. Running a simple HTTP server

```
var db = mongoclient.db('db01');

app.post('/insert',function(req,res){
        var body = req.body;
        db.collection('wind01').insert({device:body.
name, status:body.status }, function(err,doc){
        res.redirect('/');
        });
});
```

Figure 4. Insert data into MongoDB

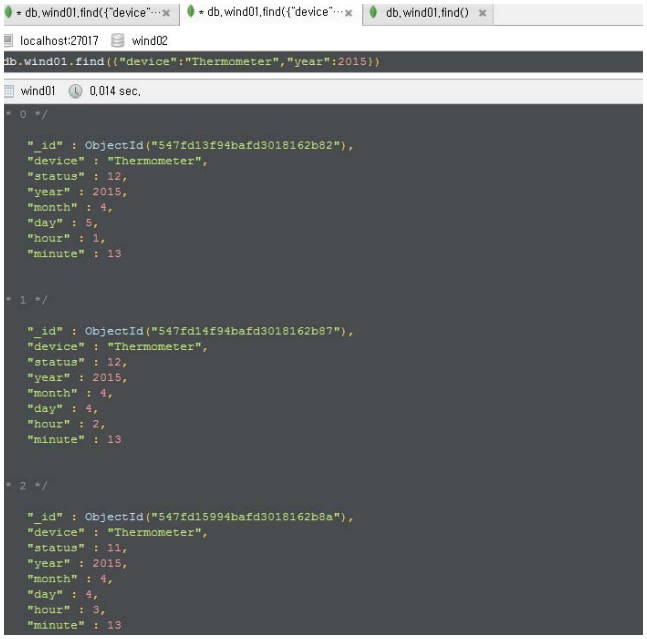

Figure 5. View database using Robomonogo

D. JSON data parsing

Parsing is the process of analyzing a string of symbols, in computer languages, conforming to the rules of a formal grammar.[14] To get the information which is outdoor temperature, we use external API, OpenWeatherMap. (Figure 7) The example in Figure x4 shows weather information (Temperature, weather, etc.) in Seoul, Korea. The weather information is JSON type. Using function JSONstream module, we can parse the JSON-type data (Figure 6).

```
{
"coord":{
        "lon":126.98,"lat":37.57
        },
 "sys":{
        "message":0.026,
        "country":"KR",
        "sunrise":1430944210,
        "sunset":1430994427
        },
"weather":[{
        "id":803,
        "main":"Clouds",
        "description":"broken clouds",
        "icon":"04d"
        }],
"base":"stations",
"main":{
        "temp":289.341,
        "temp_min":289.341,
        "temp_max":289.341,
        "pressure":1000.9,
        "sea_level":1024.13,
        "grnd_level":1000.9,
        "humidity":36
        },
"wind":{
        "speed":3.66,
        "deg":249.002
        },
"clouds":{"all":76},
"dt":1430991290,
"name":"Seoul",
}
```


Figure 6. Weather API in OpenWeatherMap

```
var request = require('request')
   , JSONStream = require('JSONStream')
   , es = require('event-stream');

request({url:'http://api.openweathermap.org/dat
a/2.5/weather?q=seoul,kr'})
.pipe(JSONStream.parse('main.temp'))
.pipe(es.mapSync(function (data){
console.log("temp:"+data);
}));
```

Figure 7. JSON data parsing

E. Sending data using RESTful API – GET/ POST Method

We can view a web page using REST. If we want to view a chart of thermometer per hour, GET method is used. When you access 'http://192.168.0.42:3000/chart/Thermometer/', you can see a thermometer chart.

Using HTTP POST method, transfer data between Sensor / Actuator nodes and web server. The POST request method is designed to request that a web server accepts the data enclosed in the request message's body for storage.[15]

When a web browser sends a POST request from a web form element, the default Internet media type is "application/x-www-form-urlencoded".[16] This is a format for encoding key-value pairs with possibly duplicate keys. Figure 8 and Figure 9 show header of POST request.

F. Visualization

To implement visualization, we use Google Chart API. Google Charts provides a way to visualize data on a website. We can load some Google Chart libraries, list the data to be charted, select options to customize the chart. Also, we put the NoSQL data into the chart, so that history of database is displayed as a chart.

```
function PostCode(status) {
        var post_data = querystring.stringify({
                'LED' : status
        });
        var post_options = {
                        host: '192.168.0.11',
                        port: '3020',
                        path: '/',
                        method: 'POST',
                        headers: {
                                'Content-Type':
                                'application/x-
                                www-form-
                                urlencoded',
                                'Content-Length':
                                post_data.length
                        }
        };
        var post_req = http.request(post_options,
function(res) {
                res.setEncoding('utf8');
                res.on('data', function (chunk) {
                                console.log('Resp
                                onse: ' + chunk);
                });
        });
        post_req.write(post_data+'\n');
        post_req.end();
}
```

Figure 8. POST header on Node.js

G. *Context-based control*

Figure 9 shows an illustrative scenario of our context-based virtualization. In this example, the heater operates in three situations. In the first case, if the external temperature is less than 10 degrees and is the room temperature is less than 15 degrees than operates the heater. And if the room temperature is less than 15 degrees, then determine that the room temperature is not comfortable to the user, operates the heater. And, the user set the heater is switched ON, then heater will be operated regardless of any other conditions.

IV. EXPERIMENTAL RESULTS

We have developed a lightweight IoT platform and have tested sensor node and actuator. User can activate actuator. When user touch a virtualized actuator icon, then web server send POST data (key-value). For example, when user touch a bulb icon, server send LED-ON (or OFF) message. (Figure 10, 11)

There are history of sensor data (temperature and humidity) and in the database. User can access history or get the mean value of history. Also, History serves chart using Google Chart API (Figure 12).

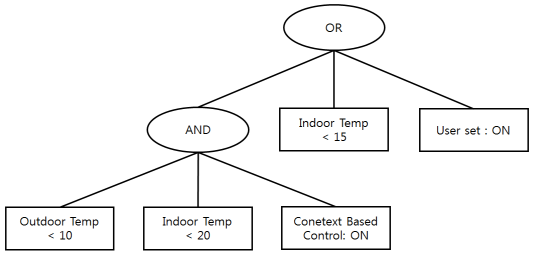

Figure 9. Context-based AND-OR tree

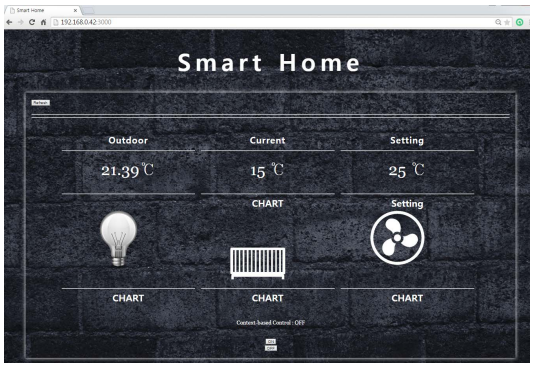

Figure 10. Index page of IoT platform

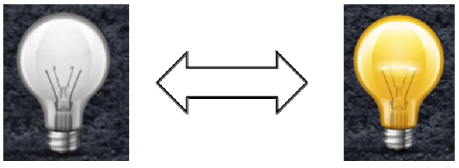

Figure 11. Virtualized LED icon

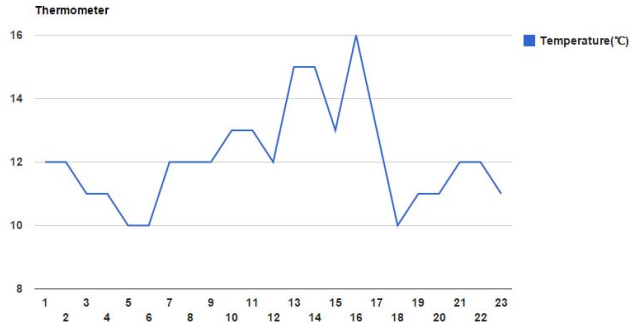

Figure 12. Chart using Google Chart API

In chapter III-F, we mentioned about context-based control. Figure 13 shows activation of context-based control. When three condition (outdoor temperature 9.56°C (under 10°C), indoor temperature 15(under 15°C) and context-based control ON) are satisfied, heater automatically activate.

V. CONCLUSION

In this paper, we have presented lightweight IoT platform developed by node.js. This platform has scalability for various sensors and actuators. The platform also supports virtualization, visualization and Restful API. In addition, without using the dedicated application, it can be used by only simple connection through web browsers. And it can be easily built and operated even in low-end hardware.

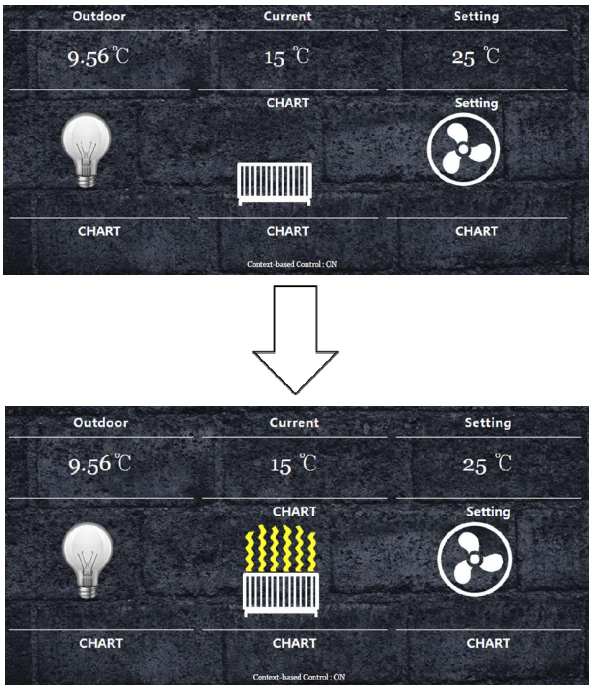

Figure 13. Context-based control result

As future work, the platform, in this paper, can be expanded in a number of different aspects. For example, various protocols such as Bluetooth Low Energy and IEEE 802.15.4 can be used. Also, not only ipv4 but also lightweight IPv6 addressing scheme such as 6LoWPAN can be considered for addressing. And for low power consumption, low-energy communication protocol such as COAP and MQTT also should be considered.

ACKNOWLEDGMENT

This Work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. B0126-15-1051, A Study on Hyper Connected Self-Organizing Network Infrastructure Technologies for IoT Service).

REFERENCES

[1] Kelaidonis, Dimitris, "Virtualization and cognitive management of real world objects in the internet of things.", 2012 IEEE International Conference on, 2012.

[2] RT Fielding, RN Taylor, " Principled design of the modern Web architecture", ACM Transactions on Internet Technology, 2002.

[3] L Richardson, S Ruby, “RESTful web services”, O’REILLY, 2008, pp. 17.

[4] Lihong Jiang, Li Da Xu, Hongming Cai, Zuhai Jiang, Fenglin Bu, and Boyi Xu, "An IoT oriented data storage framework in cloud computing platform," IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 10, NO. 2, pp.1443-1451, MAY 2014.

[5] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, “Towards a better understanding of context and context-awareness,” in Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, ser. HUC ’99. London, UK: Springer-Verlag, 1999, pp. 304–307. [Online]. Available: http://dl.acm.org/citation.cfm?id=647985.743843

[6] G. Nalepa and S. Bobek, “Rule-based solution for context-aware reasoning on mobile devices,” Computer Science and Information Systems, vol. 11, no. 1, pp. 171.193, 2014.

[7] Köhler, Marcus, Dominic Wörner, and Felix Wortmann, "Platforms for the Internet of Things–An Analysis of Existing Solutions," (Forthcoming).

[8] Mazhelis, Oleksiy, and Pasi Tyrvainen, "A framework for evaluating Internet-of-Things platforms: Application provider viewpoint," Internet of Things (WF-IoT), 2014 IEEE World Forum on. IEEE, 2014.

[9] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, “Contemporary Internet-of-Things platforms” http://arxiv.org/abs/1501.07438, Jan 2015, technical report.

[10] D-Robotics UK, "DHT11 Humidity & Temperature Sensor", DHT11 datasheet, July. 2010.

[11] S Tilkov, S Vinoski, “Node. js: Using JavaScript to build high-performance network programs” IEEE Internet Computing, 2010.

[12] Md. Motaharul Islam, Mohammad Mehedi Hassan, Ga-Won Lee and Eui-Nam Huh, "A Survey on Virtualization of Wireless Sensor Networks", Sensors, vol. 12, pp.2175-2207, Feb. 2012.

[13] P Membrey, E Plugge, D Hawkins, “The definitive guide to MongoDB: the noSQL database for cloud and desktop computing”, Apress, 2010, pp. 25.

[14] parse. (n.d.). Collins English Dictionary - Complete & Unabridged 10th Edition, 2015, Available: American Psychological Association online, http://dictionary.reference.com

[15] Berners-Lee, Tim; Connolly, Dan, "Hypertext Markup Language - 2.0 - Forms", 1995

[16] R. Fielding, J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", 2014