Design and Implementation of IoT Testbed with Improved Reliability using Conditional Probability Techniques

Kayalvizhi Jayavel
Assistant Professor, Department of IT, School of Computing
SRM Institute of Science and Technology
Kattankulathur, Chennai, India
kayalvij@srmist.edu.in

Kanagaraj Venusamy
Lecturer, Department of Engineering
University of Technology and Applied Sciences-Al musannah
Muladdha, Sultanate of Oman
kanagaraj@act.edu.om

Lavanya . G
Assistant Professor, Department of Mathematics
SRM Institute of Science and Technology
Kattankulathur, Chennai, India
lavanyag@srmist.edu.in

***Abstract*—Internet of Things (IoT) test beds are widely used by developers predominantly. Off late, Test beds are being used by data analysts, academicians, industrial persons and hardware ardent. The real purpose of test beds, is to achieve accurate testing results, mimicking the real time environment to the extent possible which is otherwise not possible to reproduce using simulators. As predicted by many industrial giants IoT based devices will reach the scale of billions by 2015. The applications and opportunities they create will also be innumerable. This has created a huge demand for such testing grounds, because a system deployed without proper testing may be vulnerable and sometimes disastrous. Thus our research aims to explore the qualities of test beds, the services they offer and how to enhance the performance of test beds. Our test bed framework is designed and developed with open-source boards to achieved heterogeneity, reusability, interoperability and scalability. This framework would like to be addressed as a utility, with "X" as service: data, sensor client, actuator client, and platform. To achieve this, APIs which is platform and language independent has been developed and provides third-party developer support. The APIs developed have shown considerable improvement in terms of data transfer rate, database upload and retrieval, and user responsiveness. Thus, our framework is capable of offering services through our API. And have demonstrated with the help of conditional probability techniques enhancement in performance and reusability, visualized the same in terms of graphs and datasets.**

***Keywords*—**Development framework; Internet of Things; Performance enhancement; Services; Testbed; Utility

I. INTRODUCTION

Internet of Things (IoT) is a technology capable of equipping the existing things, speak to each other anywhere, anytime. IoT-GSI has described IoT as the backbone of this era of information [1]. Communication at device level uses one of these protocols such as Ethernet [2], Wi-Fi among others including Bluetooth, Zigbee etc. Application-level communication protocols are HTTP, HTTPS, FTP, Telnet among others. The challenge in these protocols are, they are not developed with resource-constrained devices in mind and particularly not attractive for IoT based systems.

Having said this, it is not that, one needs to adjust with these existing protocols, many IoT protocols have already started to pitch in. Among them, two protocols namely CoAP (Constrained Application Protocol) [3] and MQTT (Message Queueing Telemetry and Transport) [4] wins most of the votes. CoAP is a request-response model and it is synchronous. MQTT is based on Hub-Spoke model (publish-subscribe) and supports asynchronous communication.

The primary objective of our work is to provide an enhanced utility-based architectural framework [5] which can serve both as testbed and development platform. There can be three types of users namely end-user, developer and data analyst. The challenge in designing and developing IoT system is it needs multi-stack knowledge base including hardware, software among other skill sets. But, the point is a software developer would mostly have little or no expertise on hardware. Thus, this paper provides a solution, proposing a utility, supporting development and testing needed for the developer. Similarly, an end-user or data analyst can use our APIs (Application Programming Interfaces) to access the sensor data to suits their needs. There exist many simulation tools (Fritzing/123dcircuits) as a simple solution for the above-mentioned problem but, they can only mimic, not the real code deployment happens. For instance, what if you want to test a real SMS from a GSM module to be sent. Thus, our paper provides a development and testing utility which offers sensor data, actuator, platform and API as services. Our major highlights are a zero learning curve for users/developers, socially reusable with zero development cost and 24x7 availability, teaching/testing/experimental platform usable by academicians, industrialists or analysts. This paper is formulated with section II comprising of state of the art. Section III is our proposed architecture. Section IV is our implementation and results. Section V is a conclusion and Results.

II. State of the Art

Our research is to design, develop, deploy an open IoT test bed cum development utility which offers sensor data, actuators, platforms and API as service. Thus, our research requires a literature review at two levels. There is a need to understand the existing test bed frameworks, their purpose, the services, challenges or issues. Secondly to investigate the way each service is offered, their methodology, merits, issues or challenges. Thus, our literature survey has been formulated in two genres, Test bed and Service respectively.

Way back in 2008, the idea of testbed was incorporated in the field of education. The authors have highlighted the importance of simulated and real experimental testbeds [6]. They were proactive in hinting on the importance of real testbeds over virtual for various reasons including lack of real data. They discuss how proprietary tools like LabVIEW or internet-ready languages like Java or Web services can be employed. Also, there is a huge complaint towards complexity and biasing towards large software vendors or integrators and not supporting open-source implementation. This inspired us to develop with more of an open-source model and make it truly public.

Kansei Genei Test bed exists since 2007 and still available for public use. It uses a 3-tier architecture comprising of wireless sensor nodes wired to a gateway. This architecture reduces the need for installing a costly network interface card in each node, which will enable them to communicate with remotely located testbed servers [7]. The Kansei Testbed was developed at Ohio State University and now federated with the NetEye Testbed at Wayne State University. There is scope for improvement on concurrency aspects of an IoT experimentation set up.

The authors have discussed the facilities needed for an experimental testbed. The authors have listed the challenges of IoT experimentation which have motivated them to identify the basic requirements that an IoT experimental test bed needs to support [8]. They have also mentioned that the majority of existing testbeds are “Intra-net” not real “Inter-net” of things. The challenges they have highlighted are scalability, heterogeneity, repeatability, concurrency, federation and experimental environment. They have emphasized that, though there are many test beds which provide useful features, still there are gaps in meeting various or all the challenges which open new research possibilities.

The author claims that many of the existing IoT technologies are tested in lab-based testbeds and they emphasize the need for realism in the external environment and the importance of real end-users in the loop [9]. They have created an infrastructure to support Smart building based experimentation with using TelosB motes.

To ensure the reliability of wireless communication in IoT based systems which are spread across distributed locations, enhanced lifetime through safe and proper maintenance of resource-constrained devices like energy, memory etc. of low power devices, authors have studied the existing experimental setups, various hardware they support and the flavor of services they offer. They have attempted to define the requirements of an IoT experimental testbed based on a survey of existing facilities, their purpose and functionality they offer [10].

FIT IoT-LAB provides a very large-scale infrastructure suitable for testing small wireless sensor devices and heterogeneous communicating objects. This project is an extension of SenseLab testbed set up. FIT is Future Internet of Testbeds was the abbreviation [11]. Recently changed to Future Internet of Things as an attempt to scale towards the Internet of Things requirements.

This paper attempts to propose the possible benefits of collaboration between simulation tools and real physical testbeds, where the simulation tools can favor in bridging the scalability issues. They have used Contiki operating system which is a lightweight protocol with µIP and µIPv6 [12]. This requires a manual reconfiguration of buffer size, data rate and traffic priority. They have provided QoS provisioning functions, which is one among our future improvements to consider but still, the interoperability of these virtual simulators with the real physical setup and their impact on performance on complete IoT based systems is pale. Moreover, ThingSpeak allows only one update every 15 seconds, when uploading data to an API [13], but there are few applications which need more frequent updates or on a specific change in the event in other words data. This has been overcome in our work by providing dynamic update intervals and updating the data only when there is a change in it.

Also, many of the existing open-source IoT platforms or API’s will demand a minimal coding knowledge, which has been eradicated in our work by providing the data needed to the end-user directly. And many of the open-source IoT frameworks often do not reveal how long the data is being stored in the servers. Transparency has been achieved in our work by providing the data from the point at which the system was deployed and provide a timestamp for every record so that the analyst knows when the data was added.

SensorBase [14] is a centralized data storage to log sensor network data in a blog style. SensorBase not only provides a way to store and access data consistently to users but also maintains. It is also referred to as “slogging data” to denote sensor log and to provide a blog like structure.

Semantic Sensor Web (SSW) [15] enables interoperability and advanced analytics for situation awareness and other advanced applications from heterogeneous sensors. The sensor networks these days can measure almost everything on earth. But the lack of integration and interconnection between these networks have left the most important data isolated and have aggravated the current problem "too much data but not enough knowledge" furthermore. SSW achieves interoperability by providing metadata along with the contextual information essential for situational knowledge. It achieves interoperability more suitable for heavyweight internet-based systems and its implications with resource-constrained devices is not explored to its depth.

All the above-mentioned frameworks provided on-demand resource sharing and facility for developing value-added services but the issue of interoperability was not addressed fully. Semantic Sensor Web is an exception which addresses interoperability but employs additional metadata to achieve, adding a bit more complexity to an already constrained system. Hence, it is our attempt to achieve interoperability in simpler terms by using the publishes-subscribe model where each device subscribes to a central broker if it is interested in a specific topic without worrying about details of its co-existing nodes. The interoperability is also enhanced by providing data in three data formats of csv, json and xml. Zhang J et al. developed a federated platform for mobile data-centric service development and sharing as an attempt to address interoperability through virtualization [16]. The issue in their proposal is, they have abstracted the virtual sensors or nodes either to represent an average or group of sensors as Temperature, Humidity and Noise. This approach has drawbacks like it would indulge more cost during sending, if the readings of any one of it change very rarely or if any one of the reading is faulty then the whole set must be resent. Also, this model requires the developer or user to write a script which can parse the received data to suit his/her needs.

[17] Silvia Santini and Daniel Rauch proposed Minos (Message Identification and Online Storage) a generic tool for sensor data acquisition and storage as part of Desthino (Distributed Embedded Systems Online) project. It is a java based software tool hence it is lightweight and portable. Though the tool is developed using java, it can interact only with sensor nodes running TinyOS which makes it platform-dependent. The user apart from mentioning the message type the tool should listen to, he or she is also expected to provide the java class representing the incoming messages. This java class generation again needs MGI (Message Generating Interface) a TinyOS utility.

Thus, our work attempts to address all the issues, as to reduce the learning curve the data analysts must undergo, to provide categorization based on sensors granulated at parameter, better visualization, and removing the dependency of any coding skills or hardware knowledge required to create the application that generates the data. Hence attempting to consolidate, testbed papers [18][19] of varied goals be it educational or industrial or testing or development, all of them had one issue in common, they all used proprietary hardware or software or both [20]. This left the major challenge of interoperability unattended. Thus, our work attempts to provide testbed as a utility with proof of concept to demonstrate that use of open-source boards and technologies comfortably takes care of the prime requirements of IoT (as stated by many authors) like heterogeneity, interoperability and reusability appreciably well.

III. PROPOSED ARCHITECTURE

An IoT architectural framework and infrastructural prototype have been proposed (Fig.1) which comprises of the Device layer, Communication layer, Aggregation layer, Control plane, Event Processing and Analytics, Web portal, Visualization, API, Device manager and Identity and Access management. A prototype has been developed which can be used for testing and development purpose.

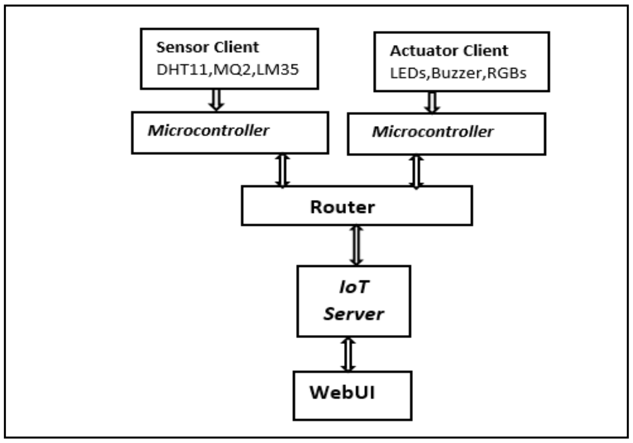

Fig. 1. Block Diagram Representation of Proposed Architecture

Device layer comprises of open source boards and sensors and actuators connected to it. Communication layer is useful in communicating the data collected from underlying sensors and command to actuators from users using any device to server protocol. Some examples are MQTT (Message Queueing Telemetry Transport), CoAP (Constrained Application Protocol) and XMPP (Extensible Messaging and Presence Protocol). MQTT has been deployed for its visible benefits towards event-driven scenarios. Aggregation layer is used to pack the data from various senor clients as need or application demands. This will help in scenarios to avoid redundant data, remove erroneous data, avoid sending unchanged data and many more. MQTT protocol broker takes the responsibility as an aggregator. The data received from sensors are updated to the database via message broker script. The analyst can use data available in the database server (Maria DB) and as the event processing depends on the application in hand, is left to the user or developer to handle. The visualization Web portal is our website which accepts requests from local or remote is investigated. API management helps in downloading the APIs for developers to actuate commands or utilize as libraries in their code development. Device management is responsible for monitoring the underlying hardware including sensors. Platform as service utilizes the service of the device manager. Identity and access management controls and authenticates the right users and developers to utilize our utility-based service. Authentication is having at two levels: Web portal at local access and session login in remote SSH. All of the above have been designed and successfully integrated to provide utility-based service to users and developers.

Our utility framework is designed with the goal of offering services at data plane, hardware plane and software plane. Sensor data is offered as service at the data plane. Our framework is capable of providing data in standard data exchange formats say JSON, XML and CSV. At the hardware level services can be categorized as sensor client and actuator client. The software plane services are offered at two dimensions namely remote code porting and API as service.

The framework is capable of visualizing the data via our dashboard. The users may be end-user or developer. End-user is one who is not necessarily tech-savvy and uses data for the knowledge base. A developer can be categorized as an application developer or product developer. The application developer will download our API package "iotdk.tar.gz" and use the system calls to use the services for his application development. The product developer is one who experiments his code, validates and modifies accordingly. Our WebUI provides a web SSH, remote shell to port code to the platform. The output is streamed to the user via our youtube channel for actuator results.

A feedback based ranking system has also been developed for efficient resource allocation for users. The feedback scores are used to decide the overall reliability and performance of the available services based on which scaled up our framework understanding the demand and supply needs.

IV. IMPLEMENTATION AND RESULTS

Our architectural framework have been implemented (Fig.2) to provide proof of concept. Our prototype has been developed with open-source boards.

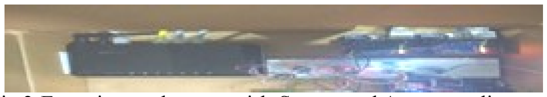

Fig 2 Experimental set up with Sensor and Actuator clients

The sensors namely DHT11, MQ-2, MQ-7, BMP-180, GPS, GSM, Buzzer, LEDs, RGBs, Color sensors, LM35, Potentiometer and LCD have been incorporated.

The sensors are connected to Sensor Client 1 and Sensor Client 2. Actuators are connected to Actuator client. And platform as service with sensors and actuators. The framework provides features for users and developers to exploit, according to their skillset and expertise.

The whole purpose of the test bed is achieved at its maximum usability. To achieve this and to offer the best performance-based service to a user a feedback based ranking algorithm has been designed. The parameters considered are sustainability/availability, reliability and performance.

A. Test bed feedback based ranking system

A feedback based ranking algorithm has been designed to allocate the best services to the users. Score ranges in between High(1), Medium(0.5), Low(0.25). Initially, all platform is assigned with High Score(1). After every usage of the platform, the user gives feedback. The previous score of the platform gets replaced with the new value calculated from recent user feedback.

B. Algorithm

```
Step 1: Check Availability in lookup table entry based
on dynamic service discovery.
If (availability < requirement) quit
Else
Continue steps 2 to 6
Step 2: All the platforms score [P1, P2, P3,.., Pi,.. ,
Pn] are assigned with value 1
Step 3: Get the feedback score for Pi after its usage
from the user.
Step 4: Remove Pi from the array of platform score.
Step 5: Insert Pi into the sorted array at the appropriate
  position.
Step 6: Assign the highest valued platform to the next
  user.
      In the case of Tie, it is resolved using
  Decision Tree-based Ranking.
```

The user provides the feedback score based on user satisfaction of the test bed. The parameters adopted here are Reliability, Performance and Usability (API). These values are inserted as input to the Decision Tree. The Decision Tree (Fig.3) provides the best available platform to the next user based on the scorecard. Logging Score, Error data and additional Feedback is used for improving system state.

The feedback has been obtained from 500 users comprising of academicians (Ai), industrialists (Ii), data analysts (Di), researcher (Ri) and novice user (Ni).

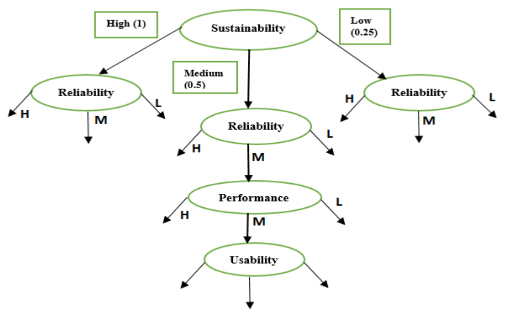

Fig. 3. Allocation decision tree

Based on log report, the job completion rate, signal strength and request/response rate is analyzed, the test bed was found reliable (score ~ 1) even when the user feedback scores were moderate. A utilization score based on the following rule is created: When scores are low, there may be 2 possibilities. One issue on the provider side or issue on the receiver/user side. Hence job completion, network and request/response time log are visited, to decide accordingly. If there was a network issue on the user side and job was not completed, the system was graded moderately reliable. If there was a network issue on the user side and job was completed with the delay, the system was graded justifiably reliable.

TABLE 1. CONFUSION MATRIX

Actually Low N=500	**User Score Low 40**	**User Score High 05**
Actually High N=500	**User Score Low 65**	**User Score High 390**

A confusion matrix have been constructed (Table 1) in which the following pattern is observed. The number of users was 500. True positive: 390, True Negative: 40, False positive: 5, False negative: 65. This was performed to predict how much time the test bed utilization (reliability) was detected low when the test bed was performing fairly moderate. Based on this, calculated the following:

Accuracy= TP+TN/Total=86%
Error rate=1-accuracy=14%
True positive rate=TP/Total positive=86%
False positive rate/Sensitivity/Recall=FP/Total negative=11%
Specificity: 1-FPR=TN/Total negative=89%
Precision=TP/total user score positive=98%
Prevalence=actual high/total=91%

Also to prove that our test bed was reliable or utilized well, joint probability function (Table 2) of the discrete random variables R(Reliability) and P'(Performance) is used.

$P(R=r_i,P^{'}=P_i^{'})$ denotes probability that R takes r_i and P takes $p_i^{'}$, it is the probability of intersection of events, namely

Reliability low and performance low will be $P(R=r_l, P^{'}=p_l^{'})$.

$P(r_i,p_i^{'})$ is the probability mass function, where i=l,m,h.

By rule, $\sum S_i=\sum T_i$, where i=l,m,h

The marginal probability function of Reliability R

$P_R(r_i)=P(R=r_i)$

$=P(R=r_i,P'=P_l')+P(R=r_i,P'=P_m')+P(R=r_i,P'=P_h')\Rightarrow p_{io}$

$R=r_i$, where i=l,m,h

The set (r_i,p_{io}) is the marginal distribution of reliability.

Similarly, the marginal distribution of performance P'

$P_{P'}(p_i^{'})=P(P'=p_i')$

$= P(R=r_l,P'=P_i')+ P(R=r_m,P'=P_i')+ P(R=r_h,P'=P_i')\Rightarrow p_{io}$

Having found the marginal distribution of R and P', the conditional probability distribution are as follows:

$F(P'/R)=f(R,P')/f(R)$ and

$F(R/P')=f(R,P')/f(P')$

TABLE 2. JOINT PROBABILITY DISTRIBUTION

P' / R	Low	Medium	High	$\sum a_{ij=1\text{ to }3}$
Low	a_{11}	a_{12}	a_{13}	S_l
Medium	a_{21}	a_{22}	a_{23}	S_m
High	a_{31}	a_{32}	a_{33}	S_h
$\sum a_{ij=1\text{ to }3}$	T_l	T_m	T_h	$\sum S_i=\sum T_i$

The conditional distribution of reliability when performance is low is given as

$F(R/P')=f(R,P')/f(P'=low)$

where $f(P'=low)=T_l$

The conditional distribution of reliability when performance is moderate is given as

$F(R/P')= f(R,P')/f(P'=medium)$

where $f(P'=medium)=T_m$

Thus, the conditional distribution when performance is low is given as follows:

$P(R=l/P^{'}=l)=a_{11}/T_l$

$P(R=m/P^{'}=l)=a_{21}/T_l$

$P(R=h/P^{'}=l)=a_{31}/T_l$

Similarly, the conditional distribution of reliability when performance is low is given as follows:

$P(R=l/P^{'}=m)=a_{12}/T_m$

$P(R=m/P^{'}=m)=a_{22}/T_m$

$P(R=h/P^{'}=m)=a_{32}/T_m$

On calculation, it was observed that the performance of the test bed was considerably moderate even the scores happened to be low or moderate. The reason for it was because of the network delay on the user side. Based on our network log, concluded that our test bed performed well and proved reliable when the scores were otherwise.

V. CONCLUSION AND FUTURE WORK

Our IoT architectural framework has been designed, developed and implemented through our open-source infrastructural setup. The services offered are collectively qualified as a utility, offering it as on-demand or use as you want. The algorithms and API's have been developed at the sensor data level, actuator, platform level which has shown good improvement in performance in terms of re-usability and utilization measured in temporal plane. Usability and performance were measured with the help of user feedback scores. Based on the user feedback score, a confusion matrix and Joint probability distribution have been derived which has shown considerable performance even the scores happened to be moderate or low. The reason for this was observed due to network latency at the user side. Thus our utility-based IoT model has performed well in terms of request-response ratio, database update and retrieval interval and utilization factor. Our model has been demonstrated using Message Queuing Telemetry Transport and attempted to develop a similar model using MQTT-SN. The security [21] aspect to our model would like to be added and demonstrate this architecture is fool-proof as well. Also planned to extend our work towards green computing [22] and energy harvesting models to achieve optimized energy model.

REFERENCES

[1] A. Gluhak S. Krco M. Nati D. Pfisterer N. Mitton T. Razafindralambo. (2011). A survey on facilities for experimental Internet of Things research. IEEE Communication , 49 (11), 58-67.

[2] Andrew Banks, Rahul Gupta. (2015, December 10). OASIS Standard. Retrieved from MQTT Version 3.1.1 : http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html

[3] Angelopoulos, C. M., Evangelatos, O., Nikoletseas, S., Raptis, T. P., Rolim, J. D., Veroutis, K. (2015). A user-enabled testbed architecture with mobile crowdsensing support for smart, green buildings. 2015 IEEE International Conference in In Communications (ICC). IEEE

[4] Chang, Kevin, Nathan Yau, Mark Hansen, and Deborah Estrin. (2006). Sensorbase.org-a centralized repository to slog sensor network data. Los Angels: National Science Foundation.[5] FIT consortium. (2014). IoT-Lab. Retrieved from FIT IoT-Lab: https://www.iot-lab.info/what-is-iot-lab/

[6] IEEE Standard for Ethernet. (2016, March 4). IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012) , pp. 1-4017.

[7] IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks--Specific requirements - Part 11: Wireless LAN Medium AcceIEss Control (MAC) and Physical Layer (PHY) Specifications. (2016, December 4). IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012) , pp. 1-3534.

[8] International Telecommunication Union. (2017, August 16). ITU .Retrieved from JCA-IoT and SC&C: http://www.itu.int/en/itu-t/jca/iot/Pages/default.aspx

[9] Kolias, V., Anagnostopoulos, I., & Kayafas, E. (2008). Remote experiments in education: A survey over different platforms and application fields. 11th International Conference in Optimization of Electrical and Electronic Equipment,. OPTIM 2008 (pp. 181-188). IEEE.

[10] Maureira, G. A. M., Daan Oldenhof, Livia Teernstra. (2015). ThingSpeak–an API and Web Service for the Internet of Things. mediatechnology.leiden.ed.

[11] Mukundan Sridharan, W. Z. (2010). Kanseigenie: Software Infrastructure for Resource Management and Programmability of Wireless Sensor Network Fabrics. In Next Generation Internet Architectures and Protocols.

[12] Nati, M., Gluhak, A., Abangar, H., & Headley, W. (2013). Smartcampus: A user-centric testbed for internet of things experimentation. 16th International Symposium in Wireless Personal Multimedia Communications (WPMC) (pp. 1-6). IEEE.

[13] Paul Fremantle. (2014). A Reference Architecture for the Internet of Things. WSO2 White Paper.

[14] Santini, Silvia, and Daniel Rauch. (2008). Minos: A generic tool for sensor data acquisition and storage. In Proceedings of 19th IEEE International Conference on Scientific and Statistical Database Management. IEEE.

[15] Shelby, Zach, Klaus Hartke, Carsten Bormann. (2014). The constrained application protocol (CoAP).

[16] Sheth, Amit, Cory Henson, and Satya S. Sahoo. (2008). Semantic sensor web. IEEE Internet computing. IEEE.

[17] Tonneau, A. S., Mitton, N., & Vandaele, J. . (2014). A survey on (mobile) wireless sensor network experimentation testbeds. . IEEE International Conference on in Distributed Computing in Sensor Systems (DCOSS) (pp. 263-268). IEEE.

[18] Zhang, Jia, Bob Iannucci, Mark Hennessy, Kaushik Gopal, Sean Xiao, Sumeet Kumar, David Pfeffer et al. . (2013). Sensor data as a service--a federated platform for mobile data-centric service development and sharing. IEEE International conference on In Services Computing (SCC). IEEE.

[19]GeorgiosZ.,Papadopoulos,Antoine,Gallais,Guillaume,Schreiner,EmeryJou,ThomasNoel. (2017). Thorough IoT testbed characterization: From proof-of-concept to repeatable experimentations. Computer Networks , 119, 86-101.

[20] M. Saravanan, M. Aramudhan, S. Sundara Pandiyan, T. Avudaiappan. (2018). Priority based prediction mechanism for ranking providers in federated cloud architecture. Cluster Computing , 1-9.

[21]Haoxiang, Wang. "Trust management of communication architectures of internet of things." Journal of trends in Computer Science and Smart technology (TCSST) 1, no. 02 (2019): 121-130.

[22] Bashar, D. A. "Review on sustainable green Internet of Things and its application." J. Sustain. Wireless Syst. 1, no. 4 (2020): 256-264.