Proc. IEEE Region 8 EnergyCon 2020, Tunis, Tunisia, 28-30 September 2020, 6p.

Microservice-based Architecture of a Software as a Service (SaaS) Building Energy Management Platform

Ashraful Haque[1], Rasheq Rahman[2], Saifur Rahman[1]
[1]Dept. of Electrical and Computer Engineering, Virginia Tech, [2]BEM Controls, LLC
Email: hashraf@vt.edu

***Abstract*—The objective of this paper is to present a three layered microservice based approach to designing a Software as a Service (SaaS) based building energy management system (BEMS). The "Core layer" is comprised of microservices performing core functionalities of the system, the "API layer" serves as a gateway between the core system and front-end services. Finally the "Service layer" provides the SaaS functionalities and applications experienced by the end-user. This paper presents a microservice based energy management system that can be effectively used by an end user to monitor and control electrical equipment and can be integrated into a large power system network. The proposed architecture has been used to develop a cloud based web interface and an iOS/android application of a building energy management system which is presented as a case study demonstrating performance and development improvements for a BEMS system developed using the microservice approach.**

***Index Terms*—Building energy management system, micro service based architecture, REST API based communication, Software as a Service (SaaS).**

I. Introduction

Industry analysts expect that more than 30 billion interconnected Internet of Things (IoT) devices will be used globally by the end of 2022 [1]. IoT has had significant impact in various fields such as transportation [2], agriculture [3], health care [4] and supply chain management [5], [6]. Arguably the building energy management sector has seen the greatest impact of IoT due to the rapidly falling price and increased availability of both consumer-grade and industrial IoT energy devices such as smart thermostats, smart bulbs [7], [8], [9]. This is changing buildings into cyber-physical systems commonly known as "Smart Buildings" which can smartly interconnect with the grid in real-time. These smart buildings have IoT enabled sensors, actuators and controllers ensuring proper environmental control for building operations. Building operations can leverage technologies such as cloud computing, remote monitoring, edge analytics and context-aware work flows.

Building on IoT, the smart grid community has increasingly begun to think of an "Internet of Energy (IoE)", using smart devices connected to the smart grid to enable transactive energy applications. Authors in [10] describe a fog based IoE architecture for transactive energy management. Authors in [11] describe a web enabled system for IoE that is capable of communicating information over the Internet. Authors in [12] describe new energy business models enabled by IoE through such approaches as real time dynamic pricing, demand side management and optimal load distribution.

A building energy management system (BEMS) is a core component of the IoE. BEMS that are robust, support multiple device types and protocols and highly scalable are valuable for the IoE . Authors in [13], [14] describe BACnet based, authors in [15], [16] describe zigbee based and authors in [17] present Wireless LPWAN based BEMS. However, a microservice based BEMS which supports Software as a Service (SaaS) business model has not been much explored. The microservice based architecture improves the resiliency of BEMS by ensuring different components of BEMS work independently and SaaS ensures that the BEMS can scale up and down for the client. The contributions of this paper are:

- Presents a microservice based architecture for a resilient BEMS capable of IoT applications i.e. cloud computing, edge analytic, remote monitoring
- Presents applications of a SaaS based multi-functional BEMS capable of simultaneously being used by end users, utility, energy analytic platforms
- Presents REST API based communication structure for integration with microservices and applications
- Presents a case study of a BEMS software package rewritten from a modular to the proposed microservice approach and compares the performance improvement in terms of scalability of operation and software resource utilization

The paper is organized as follows: Section 2 describes the three layered microservices in details. Section 3 describes the services offered in a SaaS based BEMS. Section 4 presents a case study of microservice based architecture in terms of scalability and optimization of software resources.

II. Microservice based architecture of BEMS

In a monolithic software architecture, all code is executed from a single file. In order to increase the manageability of the code base, software developers have adopted more modular architectures in which different modules perform different functions and are often in different files. However, the modules are still part of the same application.Scaling up the functionalities of modules is therefore an issue. Since change in one module will require rigorous compatibility checking with all the other ones for the code to inter-operate. In a microservice based architecture, functionalities of an application are broken down into unique core functionalities and each core function is represented as a ”microservice”. These microservices work independently based on unique inputs and unique outputs. The main advantage of having a microservice based architecture is that if any microservice fails to operate, other services will continue operating. The paper proposes a three layered micro-service based architecture as shown in Figure 1. The layers are- Core layer, API layer and Service layer

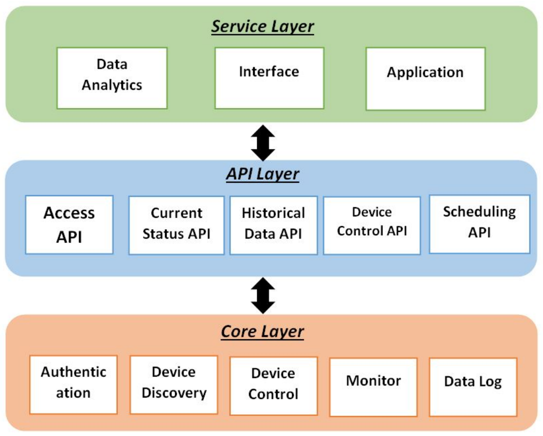


Fig. 1. Three layered microservice based architecture

A. Core layer microservices

The core layer microservice provides key functionalities needed for the BEMS to interact with smart devices as well as receive and transmit commands from/to user interfaces or other systems. These core functionalities are device status monitoring, device status controlling and device historical data logging.

1) Authentication microservice: The functionality of this microservice is to control access to building information according to user management hierarchy and store relevant information related to each building. After proper validation checks (i.e. username/password confirmation) this microservice issues an intermediate ”Token” with an expiration time tag to a particular user. This ”Token” is required as authentication token for further interaction with the BEMS within that time period. Once the session or token time expires, a new ”Token” needs to be fetched using same validation checking. Since the username,password are used only one time during the session, this reduces the security vulnerability related to cyber-attacks such as man in the middle attack, side channel analysis (SCA) attack.

Figure 2 shows different components of authentication microservice.Information that this microservice holds are:

- Token management
- User management hierarchy of each individual building
- List of devices of each individual building
- Static information of each building i.e. address, operational time.

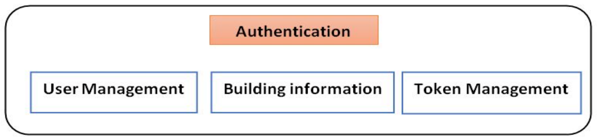


Fig. 2. Authentication microservice

2) Device discovery microservice: The functionality of this microservice is to discover and integrate new devices in a building. This microservice is capable of finding new devices through different communication protocols (i.e. WiFi, Zigbee, BACNet). After discovery is complete, this microservice associates the device with the relevant building and sends this device information to the authentication microservice as shown in Figure 3.

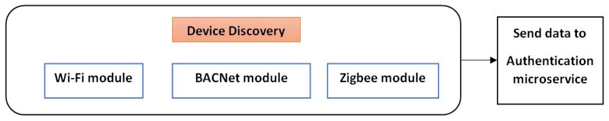


Fig. 3. Device discovery microservice

3) Device control microservice: The functionality of this microservice is to change the operations of electrical devices (i.e. changing HVAC set points, turning plug load on/off, increasing or decreasing brightness of light bulbs). Depending on the types of devices (i.e. HVAC, lights, smart plugs) this microservice has different components. The controlled device needs to have an API that can be accessed by a third-party developer through which it can be reached from this microservice. A valid ”Token” is required for successful control of the device. Once a device control command is received, this microservice will check the validity of ”Token” from authentication microservice and on successful execution will send the information to data logging microservice as shown in Figure 4.

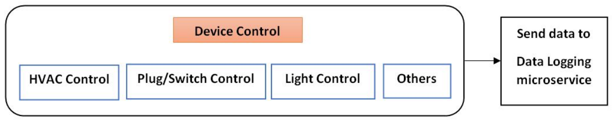


Fig. 4. Device control microservice

4) Monitoring microservice: The Monitoring microservice monitors the status of each device and takes a snapshot of the device status at pre-defined interval (i.e. every five minutes). The snapshot is then forwarded to data logging microservice which stores the snapshot in a database as shown in Figure 5

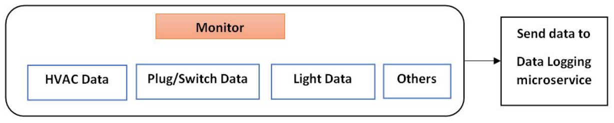
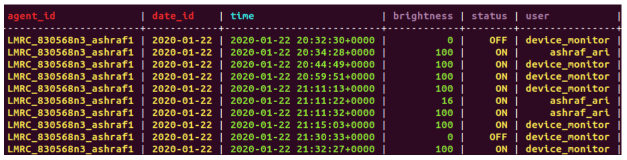


Fig. 5. Device monitor microservice

5) Data logging microservice: The Data Logging microservice stores time-series device data in to a database (i.e. Cassandra, a NoSQL database platform). This microservice stores the data coming from the device monitoring microservice at a pre-specified interval. Additionally, whenever the device control microservice issues a command, the details of the command (execution time, command type, etc.) are stored by the data logging microservice. Figure 6 shows snapshot of Cassandra time series database storing historical data of a light bulb.



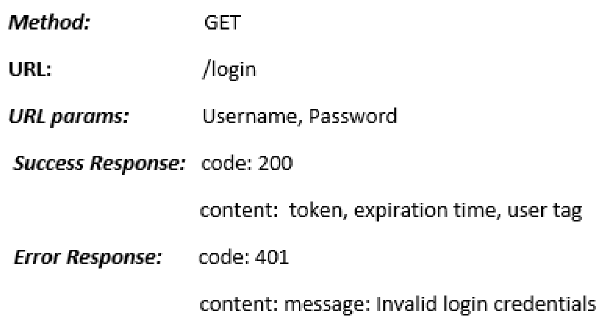
```
agent_id                 | date_id    | time                     | brightness | status | user           |
-------------------------+------------+--------------------------+------------+--------+----------------+
LMRC_830568n3_ashraf1 | 2020-01-22 | 2020-01-22 20:32:30+0000 |          0 |    OFF | device_monitor |
LMRC_830568n3_ashraf1 | 2020-01-22 | 2020-01-22 20:34:28+0000 |        100 |     ON |     ashraf_ari |
LMRC_830568n3_ashraf1 | 2020-01-22 | 2020-01-22 20:44:49+0000 |        100 |     ON | device_monitor |
LMRC_830568n3_ashraf1 | 2020-01-22 | 2020-01-22 20:59:51+0000 |        100 |     ON | device_monitor |
LMRC_830568n3_ashraf1 | 2020-01-22 | 2020-01-22 21:11:13+0000 |        100 |     ON | device_monitor |
LMRC_830568n3_ashraf1 | 2020-01-22 | 2020-01-22 21:11:22+0000 |         16 |     ON |     ashraf_ari |
LMRC_830568n3_ashraf1 | 2020-01-22 | 2020-01-22 21:11:32+0000 |        100 |     ON |     ashraf_ari |
LMRC_830568n3_ashraf1 | 2020-01-22 | 2020-01-22 21:15:03+0000 |        100 |     ON | device_monitor |
LMRC_830568n3_ashraf1 | 2020-01-22 | 2020-01-22 21:30:33+0000 |          0 |    OFF | device_monitor |
LMRC_830568n3_ashraf1 | 2020-01-22 | 2020-01-22 21:32:27+0000 |        100 |     ON | device_monitor |
```

Fig. 6. Data Log of a light bulb

B. API layer microservices

API layer microservices work as a bridge between the service layer and the core layer. Any request sent from the users is processed in this layer with the help of core microservices. The API layer adds a buffer layer ensuring core device-level microservices are being used only when a request from a user is valid and is properly authenticated. Each microservice in this layer serves as an API endpoint.

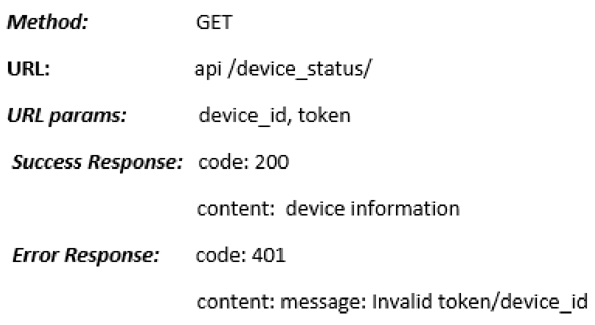
1) Access API: The Access API receives all login requests and provides the user with a ”Token” from the access control microservice when there is a valid request. This ”Token” is thereafter used for further execution. Since, ”Token” is required for any time of execution, every user will use this microservice to receive their ”Token”. Figure 7 shows the pseudo REST API structure for this microservice.



Method:	GET
URL:	/login
URL params:	Username, Password
Success Response:	code: 200
	content: token, expiration time, user tag
Error Response:	code: 401
	content: message: Invalid login credentials

Fig. 7. Access API pseudo structure

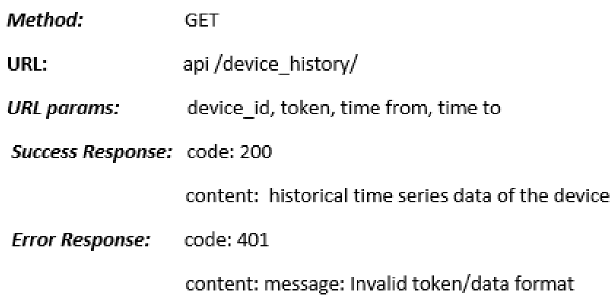
2) Current status API: Current status API pulls the data from the monitoring microservice. The access API token is used to validate the user’s ability to access this data. This API essentially provides real time data to the application layer and informs the current status of each individual device. Figure 8 shows the pseudo REST API structure for this microservice.



Method:	GET
URL:	api /device_status/
URL params:	device_id, token
Success Response:	code: 200
	content: device information
Error Response:	code: 401
	content: message: Invalid token/device_id

Fig. 8. Current status API pseudo structure

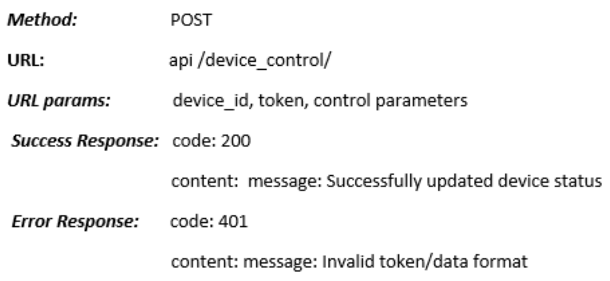
3) Historical data API: The historical data API provides historical information about a specific device from the data logging microservice. This API works as a source of historical data for data requests from the application layer. This historical time data can be used to train and validate machine learning models used for building energy analysis i.e. load forecasting. Figure 9 shows the pseudo REST API structure for this microservice.



Method:	GET
URL:	api /device_history/
URL params:	device_id, token, time from, time to
Success Response:	code: 200
	content: historical time series data of the device
Error Response:	code: 401
	content: message: Invalid token/data format

Fig. 9. Historical data API pseudo structure

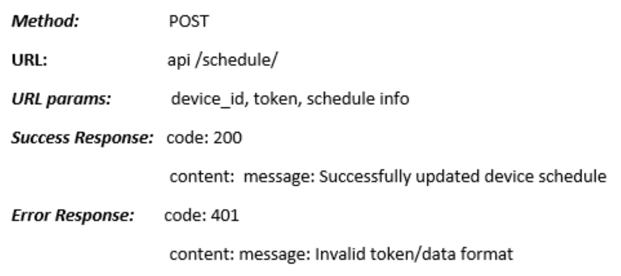
4) Device control API: The device control API controls the status of each individual device. The API receives the request to control a device and controls that particular device through the device control microservice in the core layer. Once the command is executed, the device control microservice sends the information to data logging microservice regarding what was done, who did it and when it was done for storage. This will help keep track of user activities. Figure 10 shows the pseudo REST API structure for this microservice.



Method:	POST
URL:	api /device_control/
URL params:	device_id, token, control parameters
Success Response:	code: 200
	content: message: Successfully updated device status
Error Response:	code: 401
	content: message: Invalid token/data format

Fig. 10. Device control API pseudo structure

5) Scheduling API: Scheduling API microservice performs three different functionalities related to scheduling. It stores schedule information of each device, checks whether any schedule becomes activated at a particular time and if any schedule is activated, then it sends the device control command to device control microservice to execute that schedule.Figure 11 shows the pseudo REST API structure for this microservice.



Method:	POST
URL:	api /schedule/
URL params:	device_id, token, schedule info
Success Response:	code: 200
	content: message: Successfully updated device schedule
Error Response:	code: 401
	content: message: Invalid token/data format

Fig. 11. Scheduling API pseudo structure

C. Service layer microservices

Microservices in the service layer facilitate the insights, interfaces and applications that characterise a modern as-a-service business model where the end-user can consume the building energy management functionality in a specified amount for a specific duration (i.e. one device controlled over the course of one month). Each microservice enables a service that is being offered to end-user customers. In this paper we consider the following three services:

1) Interface microservice: The interface microservice provides the user experience for the SaaS energy management system. For example, a homeowner can control devices in his home through a home energy management portal, leveraging the interface service microservice. Moreover, hardware manufacturers who do not have their own software interfaces (i.e. a mobile app) can use this microservice to provide visualization and control of their devices. This microservice touches all three layers and all of the relevant APIs. Figure 12 shows how this microservice pulls data from other microservices.

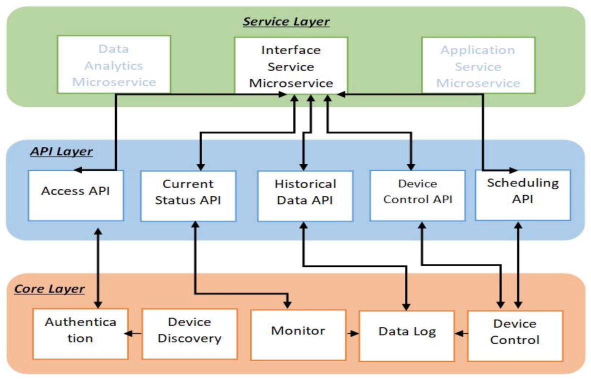


Fig. 12. User interface microservice

2) Data analytic microservice: Data analytic microservice provides load forecasting of each building. Using the historic load data, this microservice trains supervised machine learning models for each building to do load forecasting. An accurate load forecasting is required for optimal generation and distribution planning of power grid. Forecasted load is also often considered as an essential input to various power system analytical models.Figure 13 shows how this microservice uses components of other layers.

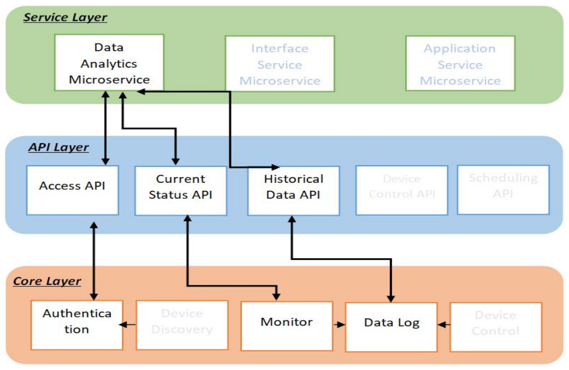


Fig. 13. Data analytic microservice

3) Application microservice: The application microservice offers combinations of multiple microservices working together to achieve an energy outcome such as reduced demand. One such type of application is- ”demand response”. The objective of a demand response application is to reduce load consumption during peak load hours. This microservice can be used to reduce electrical load by calling the device control API and or through the scheduling API during peak load hours.

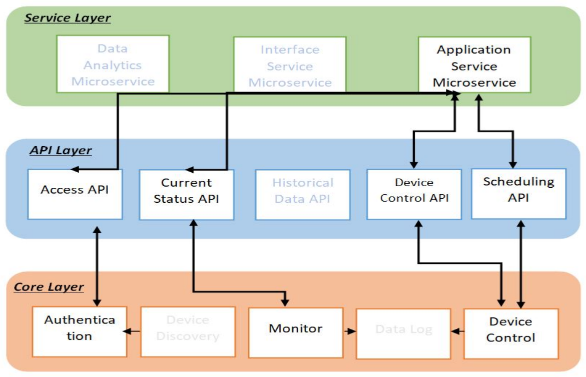


Fig. 14. Application microservice executing a demand response algorithm

Figure 14 shows how the application service microservices uses microservices of other layers to execute the application.

III. Software as a Service (SaaS) functionalities

Software as a Service (SaaS) is a business model where the end-user pays for access to software and/or services delivered via the cloud under a time-limited agreement. In a SaaS model, the service provider provides end to end software application service to the users based on the user requirements, service level agreement (SLA) and key performance indicator (KPI).

Instead of just hosting servers and databases, SaaS providers host and run the applications, maintain the physical hardware infrastructure and provides the software and service via internet accessible ”cloud”. The SaaS model is very efficient and cost effective because computational power and storage can be scaled up and down depending on business needs and often multiple customers can share the same hardware resources at the same time. BEMS have traditionally been on-premise solutions requiring on-site hardware and often a dedicated computer terminal to operate energy systems. With the advent of smart thermostats such as the Google Nest device, cloud-connected energy IoT devices became more affordable and eventually more commonplace in modern and existing buildings and homes. The microservice architecture proposed in this paper aims to show how the technical architecture of the underlying BEMS software enables a number of new services including:

A. Building energy management interfaces

The proposed architecture can be used to create a user interface for a building energy management system offered as a service to end customers. A sample iOS/android application using the software architecture outlined in the paper has been developed using the Python open source programming language.

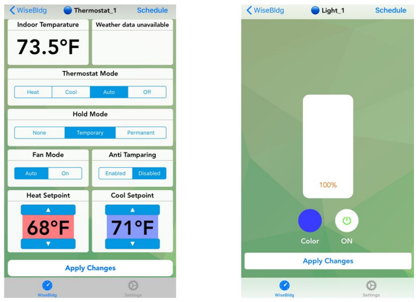


Fig. 15. iOS/android application

Figure 15 shows the iOS/android application interface of device control pages for HVAC and lighting controls respectively.

B. Energy analytics

Access to data is becoming essential, as more and more organizations seek to leverage ”Big Data” technologies such as machine learning, artificial intelligence to optimize the use of energy in their buildings. The proposed BEMS architecture is capable of enabling energy analytics algorithms such as:

- Demand response potential of a building
- Energy saving potential of a building
- Load forecasting under varying conditions
- Historical load profile
- Optimal schedule management under varying conditions
- Quality of building’s power sources
- Preventive maintenance status of electrical equipment

C. Participation in demand response program

While demand response (DR) programs continue to increase globally, many smaller commercial buildings and homes can not participate in these programs because they are not equipped to receive and act upon demand response signals from utilities. As mentioned above, the proposed BEMS architecture contains DR functionalities which can be deployed quickly and at low cost to small and medium sized buildings. Buildings enrolled in DR programs and running applications build on the proposed BEMS architecture will receive demand response signals from relevant authorities and execute necessary control actions to reduce energy usage during a DR event.

IV. Case study

To illustrate how the microservice based architecture described above impacts key performance indicators of a modern BEMS, a case study presented below compares a BEMs application written in the traditional modular approach versus the same application rewritten as a microservice. The application was tested in research setting using t2.medium cloud computing instance of AWS. The instance has up to 3.3 GHz Intel Scalable Processor, 2 vCPUs and 4 GiB of memory. Key metrics examined include:

1) Resilient and scalable architecture: As discussed above and demonstrated below, developing a BEM as a microservice greatly enhances resiliency and scalability. This is due to the component nature of microservices which breaks core functionalities of the BEMS into unique components enabling microservices to work independently. For example if a data logging microservice stops working, devices will still be controllable through device control microservices. This enhances resiliency by preventing cascading failures when one component microservice fails over. More resiliency means that systems can scale faster. As demonstrated in Table I below, the microservice-based BEM platform can concurrently control many more devices using the proposed microservice approach.

TABLE I
Scalability comparison

	Traditional modular	Proposed microservice
Concurrent controlled devices	400	3200

2) Optimization of software resources: A microservice based approach also frees up software resources as the number of devices scale-up, especially in dynamic environments such as during a demand response event. Microservices allow software and hardware resources to be optimized for current and future workloads. For example, a demand response program may only execute for several hours during selected days in a year and so it can be scaled up on demand. When a DR event is not taking place, the resources required to run a DR application can be redeployed to another activity. Table II below illustrates how using a microservice based approach, memory usage does not increase even as the number of controlled devices increases by more than 100%.

TABLE II
OPTIMIZATION OF SOFTWARE RESOURCES

No of devices	RAM usage GB (Traditional modular)	RAM usage GB (Proposed microservice)
0	1.6	1.84
15	2	1.84
30	2.2	1.84
45	2.35	1.84

TABLE III
SERVICE INTEGRATION PROCESS

Task	Traditional modular (man-days)	Proposed microservice (man-days)
New device integration	3	1
New application development	14	7

Capability of reusing core components for all the devices enables optimization of software resources.

3) Integration with other cloud web services and applications: With the rapid proliferation of new devices and APIs in the marketplace, a BEMS must be able to rapidly incorporate new and changed device APIs. The API layer in proposed approach allows easy integration with other web services and applications. For example, when a new device is to be integrated, a developer only needs to integrate device specific APIs to a device control microservice. Table III below shows how the time (man-days) to develop new applications and integrate new devices fell significantly when the BEMs software was transitioned to a microservice based approach.

V. CONCLUSION

A scalable, multi-functional and resilient energy management system is required to fully utilize the advancements made in the fields of IoT (i.e. cloud computing, edge analytics, etc). Additionally, ensuring that a cyber-physical system can integrate new technologies such as artificial intelligence, blockchain and 5G telecommunications requires that its architecture be microservice oriented. This paper presents a microservice based approach to building energy management system that can be used to provide multiple services to end-users. A working front end interface has been developed using the proposed microservice based approach. Future areas of studying include understanding how the architecture can be enhanced to increase data access for machine learning algorithms, new IoT device types and software driven transactive energy platforms.

REFERENCES

[1] E. M. Report, "Internet of Things forecast," https://https://www.ericsson.com/en/mobility-report/internet-of-things-forecast/, 2016, [Online; accessed 19-July-2016].

[2] W. He, G. Yan, and L. D. Xu, "Developing vehicular data cloud services in the iot environment," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1587–1595, May 2014.

[3] A. Kamilaris, F. Gao, F. X. Prenafeta-Boldu, and M. I. Ali, "Agri-iot: A semantic framework for internet of things-enabled smart farming applications," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, Dec 2016, pp. 442–447.

[4] B. Xu, L. D. Xu, H. Cai, C. Xie, J. Hu, and F. Bu, "Ubiquitous data accessing method in iot-based information system for emergency medical services," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1578–1586, May 2014.

[5] L. Li, "Application of the internet of thing in green agricultural products supply chain management," in *2011 Fourth International Conference on Intelligent Computation Technology and Automation*, vol. 1, March 2011, pp. 1022–1025.

[6] S. Yuvaraj and M. Sangeetha, "Smart supply chain management using internet of things(iot) and low power wireless communication systems," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, March 2016, pp. 555–558.

[7] K. Akkaya, I. Guvenc, R. Aygun, N. Pala, and A. Kadri, "Iot-based occupancy monitoring techniques for energy-efficient smart buildings," in *2015 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, March 2015, pp. 58–63.

[8] M. Wang, G. Zhang, C. Zhang, J. Zhang, and C. Li, "An iot-based appliance control system for smart homes," in *2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, June 2013, pp. 744–747.

[9] B. Sun, P. B. Luh, Q. Jia, Z. Jiang, F. Wang, and C. Song, "Building energy management: Integrated control of active and passive heating, cooling, lighting, shading, and ventilation systems," *IEEE Transactions on Automation Science and Engineering*, vol. 10, no. 3, pp. 588–602, July 2013.

[10] M. H. Yaghmaee Moghaddam and A. Leon-Garcia, "A fog-based internet of energy architecture for transactive energy management systems," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1055–1069, April 2018.

[11] N. Bui, A. P. Castellani, P. Casari, and M. Zorzi, "The internet of energy: a web-enabled smart grid system," *IEEE Network*, vol. 26, no. 4, pp. 39–45, July 2012.

[12] K. Zhou, S. Yang, and Z. Shao, "Energy internet: The business perspective," *Applied Energy*, vol. 178, pp. 212 – 222, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261916308273

[13] S. H. Hong, S. H. Kim, Jin Ho Kim, Yun Gi Kim, Gi Myung Kim, and Won Seok Song, "Integrated bacnet-zigbee communication for building energy management system," in *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, Nov 2013, pp. 5723–5728.

[14] T. Park and S. Hong, "Experimental case study of a bacnet-based lighting control system," *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 2, pp. 322–333, April 2009.

[15] D. Han and J. Lim, "Smart home energy management system using ieee 802.15.4 and zigbee," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 3, pp. 1403–1410, Aug 2010.

[16] J. Byun, B. Jeon, J. Noh, Y. Kim, and S. Park, "An intelligent self-adjusting sensor for smart home services based on zigbee communications," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 3, pp. 794–802, August 2012.

[17] Y. Song, J. Lin, M. Tang, and S. Dong, "An internet of energy things based on wireless lpwan," *Engineering*, vol. 3, no. 4, pp. 460 – 466, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2095809917306057