Internet of Green Things with autonomous wireless wheel robots against green houses and farms

C Ramasamy Sankar Ram1, S Ravimaran2, R Santhana Krishnan3, E Golden Julie4, Y Harold Robinson5, Raghvendra Kumar6, Le Hoang Son7,8, Pham Huy Thong9,10, Nguyen Quang Thanh11 and Mahmoud Ismail12

Abstract
Nowadays, smart farming involves the integration of advanced technologies that incorporate low-cost robots to meet the required knowledge and maintain the health of plants in farming. Technologies like precision agriculture are also used to optimize resources based on the field condition. Internet of Green Things is also one of the technologies to integrate and share the information between people and healthy farm things. Internet of Green Things gives the information like soil moisture, temperature, humidity, and nutrient level by means of respective sensors. Monitoring and information gathering in green houses with the help of robots is a tedious and expensive process. In this connection, information is shared among low-cost robots encouraging data availability of the current state of a plant with other robots. This will emphasize the monitoring of green houses in a well-organized way. In this article, a Flask-based framework through Raspberry Pi is proposed for interoperability among the low-cost ESP8266 robots. Data gathering is performed by smart robots that are accessible through Message Queuing Telemetry Transport subscribes by means of Representational State Transfer Application Programming Interface. A cloud-like database server is provided to stock up the data. The integration of robotics with Internet of Green Things gains more advantage in gathering about spatial information data that are connected with the irrigation. Visualization techniques and perspectives based on Internet of Green Things for precision agriculture in the field of farming are highlighted.

Keywords
Message Queuing Telemetry Transport, ESP8266, Extensible Markup Language, JavaScript Object Notation, Application Programming Interface, Representational State Transfer, Internet of Green Things, robotics, Raspberry Pi, NodeMCU, gateway

Date received: 4 September 2019; accepted: 6 April 2020

Handling Editor: Rongbo Zhu

Corresponding author:
Pham Huy Thong, Informetrics Research Group, Ton Duc Thang University, Ho Chi Minh City 70000, Vietnam.
Email: phamhuythong@tdtu.edu.vn

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
**Introduction**

Recent advancement in the Internet and Web technologies has made availability of the pitch to users to create and use information as web services effectively. Web services incorporate much genuine data gathered by smart forms and provide the same to users. This stirring thought coined the term Internet of Green Things (IoGT). The Message Queuing Telemetry Transport (MQTT) protocol\(^1\) was invented by Andy Stanford-Clark and Arlen Nipper in 1999. The properties such as bandwidth efficient, light weight, data agnostic, simple implementation, and session awareness are much helpful to implement this framework. JavaScript Object Notation (JSON) is basically a syntax for serializing the objects such as strings, numbers, arrays, and so on and used for swapping data that are outlined to be perceived clearly by humans and parsed logically and effortlessly by machines. As JSON itself is entirely language independent, it is sought-after often for dispatching data through the RESTful Web Services.\(^2\)\(^-\)\(^9\) Since a public Web API is accessible, it provides a programmatic access to the developers. API is usually outlined as a collection of Hypertext Transfer Protocol (HTTP) for requested messages; it also receives alongside the response message structure in the context of Web development.

This article scrutinizes the MQTT's publish/subscribe services on low-cost smart robots, optimization of Representational State Transfer Application Programming Interface (REST API) and JSON data from sensors, and a simple Raspberry Pi–based cloud for green house. Furthermore, how far it is utilized to process the data from the Web Application Programming Interfaces (APIs) is analyzed. A standard data pattern should be determined while creating an application to communicate with private and warehouse systems. There are ranges of public and standardized options available and the supreme choice and the ultimate selection depends on the vital demands and functionality of their applications. The data in an Extensible Markup Language (XML) payload are encapsulated to some extent of a Simple Object Access Protocol (SOAP)-based Web services.\(^9\)\(^-\)\(^11\) In many application frameworks, when XML performs professionally, it has some downsides that make it less than ideal for others. As a result, the creation of interactive web applications creates petulant user experience by its innumerable access. Ultimately, using JavaScript, these asynchrome calls originate on the end user and integrate formatting information, transfer it to a web server, parsing and processing with the actual response information.\(^12\)\(^-\)\(^19\) JSON procures a standardized data exchange format, which is more appropriate for web applications, while many browsers are able to build, transfer, and parse XML.

On the other side, JSON is an autonomous human-readable format with a broad range of implementation possibilities. Information will be intentionally formatted as per the JSON standard.\(^20\) It is now predominantly a format of data and is not limited to web applications virtually used in any framework where applications require the exchange or storage of structured information as text. Innumerable libraries for almost all programming languages grant simple parsing of JSON data, and most of the web frameworks currently reinforce generating JSON as response data.\(^14\)\(^,\)\(^16\)

In addition to the fact that JSON has a small-scale foot-print than XML, it is appropriate for situations where network speed is restricted. Evidently, mobile devices habitually have slower connections and hence welfare from a compact download.

API is a group of procedures, conventions, and mechanisms for creating software applications\(^21\)\(^-\)\(^28\). An API states how software elements communicate and are being exploited while programming the Graphical User Interface (GUI) components. In divergence to a private API, a public API is available widespread for all developers to fetch the green house data in gain access. They permit the developers to access data which can then be utilized to augment their own applications. Public APIs can remarkably enlarge income in the absence of the business to put money in recruiting brand new programmers who produce for them a very commercial software application. Nevertheless, it is essential to recall that opening the data from database to the public may result in a wide range of reliability and management challenges. For example, issuing public APIs can create rigidity for institutions to command experienced clients with their information benefits. Public API publishers cannot conclude that the client apps created on their APIs would provide a better user experience. The primary focus and usage of the MQTT protocol here are publishing and subscribing JSON-based data topics to open IoGT use cases.\(^2\)\(^,\)\(^4\)\(^-\)\(^9\)\(^,\)\(^20\)\(^-\)\(^40\)

Low-cost android application (app) contained a smart home system prepared to remotely turn on/off a group of 208 devices using only four digital pins of an Arduino UNO board–based ATmega328 microcontroller.\(^41\) The appliance-switching scheme requires approximately 70% lower cable costs for real-life fitting in home correlation to the standard devices linked. The smart home system includes repeated alerts by light if it detects humans very near at night and the “exterior mode” will shut it down for each single machine connected to the system with a touch from the Android app when the user leaves. A smart security system is able to make an alarm, send a message, and call...
the user by Global System for Mobile Communications (GSM) module to probe several intrusions in the house. Home energy management system (HEMS) is a decentralized structural design based on a multi-agent system.42 This intellectual energy management system is to be developed by IEEE- and FIPA-accommodating multi-agent platform. This enables agents to communicate, interact, as well as negotiate by energy sources plus devices of the smart home to give energy usage to reduce the cost of electricity bill. Multi-agent systems are used for implementing HEMSs to illustrate the communication of agents among devices for energy sources in sequence.43-50

Presently, no interacting low-cost and highly autonomous robots are found in the green houses. In particular, sensors are placed and data manipulation has been done manually or with the help of web interfaces. Besides, some tasks such as information gathering about current status and health of a plant have long been considered to be too complex to automate completely. In addition, green houses face the problem of labor shortages. As a consequence, the production process is progressively affected. In this context, we surveyed the above-said approaches and applications focused on improving the operational conditions, liveliness, effectiveness, protection, efficiency, and profitability of agricultural processes, in cases where physical labor will not be able to replaced by but can be complemented with proposed low-cost robots.

In this article, we propose a set of Flask-MQTT-based autonomous wheel robots working under the control of Raspberry Pi on which the broker is installed. These low-cost ESP8266 robots are interacting with each other about data gathering and updating with one another. The accumulated data from the robots are stored in the database. The integration of robotics with IoGT gains more advantage in acquiring spatial information data connected with irrigation. Visualization techniques and perspectives based on IoGT for precision agriculture in the field of farming are highlighted.

Table 1 describes the functionality metrics of MQTT over HTTP. From the observation of the table given, the key contribution of this proposed work can be viewed as the smart autonomous wireless robots controlling the green house from anywhere providing the green house’s data of flask server at the Raspberry Pi. The MQTT-based publishing/subscribing system for interoperability among smart robots and REST API has also been developed in Flask. In REST API, how the JSON data are manipulated with the MQTT protocol is described in the following section.

The organization of this work starts with various related works. From the observations, the significance of API and background technologies such as JSON and MQTT is explored. Then, implementation of IoGT—A Flask-based python framework—and REST API for accessing low-cost ESP8266 wheeled robots through MQTT publish–subscribe technique is discussed. The results from device interactions, round trip time (RTT), and performance are analyzed consequently.

### Proposed system

MQTT is a lightweight messaging protocol used frequently in the message interactions of Internet of Things. JSON is a self-describing and hierarchal structure. The syntax of JSON is predefined by two basic structures as name-value pairs described as ordered set of values.9 The value in the said structures can be any of the types listed such as number, array, object, string (unicode), Boolean (true or false), and null. These structures can be nested; nested arrays are the mostly used values in the JSON data interchange format. The JSON values cannot be a function. Since this flexible structure is used in RESTful Web services, we bring this on MQTT as topics while publishing or subscribing.

### Example of green things in JSON

```
Listing 01

{    
    "Title": "Temperature", 
    "Description": "DS18B20", 
    "Type": "ONEWIRE", 
    "Properties": 
        { 
            "Values": 
                { 
                    "ID": "ESP-1", 
                    "temp": 34.3445 
                } 
        } 
}
```

The values and keys are description of temperature sensor and respective values enclosed in double quotes.
Double quotes are not necessary if the value is a number. If the JSON value is a number, it can be expressed as an integer or floating point. The file extension of the JSON data format is JSON.

Both the JSON and the XML shall be exploited to measure the local and flexible objects in human-readable and text-based, data proxy format. Besides, these data exchange formats are more identical with the given text in any preferable format, and the same another one is grasped in the other format. Therefore, when determining an information exchange format, choosing one to the other is not a simple occurrence; instead, preferably, the format has the characteristics making it the appropriate choice for a particular implementation. JSON has preferably roots in programming languages and hence it yields a more accepted and readily accessible mapping to exchange the structured data whereas XML has its own style preferably for mapping. Formally, both JSON and XML both support internationalization since they are supporting Unicode. Finally, this convenient data format is more practical on the transactions between the smart house and the data center.

**Analysis of web API**

GUI provides simpler access for users to utilize programs and APIs create it easier for programmers to use precise technologies in creating applications. By extracting hidden implementations and disclosing objects or actions that only the developers require, an API minimizes the cognitive work on them. While a graphical interface for an email user might supply a user along with a button which exhibits all the procedures for extracting and highlighting new emails, an API for file input/output will provide the programmer a function that duplicates a file from one location to another. Web APIs provide access for programmers to exploit remote data by underlying protocols and certain standards for interaction which allows various technologies to perform together, despite of platform or language. Consider an example of Java Database Connectivity API that permits developers to query various kinds of databases with the similar group of operations. Java Remote Procedure Call (RPC) API utilizes the protocol to permit invocation functions to operate remotely (Figure 1). Consequently, remote APIs are convenient in preserving the object encapsulation in object-oriented programming. Web APIs are the predefined interfaces across which communications occur between an enterprise and applications that utilize its benefits. An API perspective is an architectural outline which spins everywhere, assigning programmable interfaces to a group of services to a variety of applications serving widespread types of consumers.

![Diagram](Image)

**Figure 1.** Accessing data center via APIs.

The proposal of an API has outstanding influence on its performance. The concept of data encapsulation explains the task of programming interfaces by folding the implementation information of the modules by which the users of modules do not perceive the difficulties inside the modules. Hence, the design of an API tries to issue only the mechanisms a user would expect. The design of programming interfaces constitutes a significant part of software component. Public Web APIs are available for most of the data requirements for the developers and the programmers. APIs may include data of various areas like books, business, cloud storage and file sharing, health, music, news, science, sports/fitness, transportation, university, and so on.

The OpenWeatherMap is a public Web API which provides weather data for variety of mobile and web applications. This API provides free access to its subscribed customers. To access this API, users shall create an account and API key is provided for the subscribed user for accessing the Web API. Data for Weather Web API are collected from the weather stations in different format including current weather information, historical weather data, days, and an hour forecast information and daily forecast information. The current weather information format gives weather data for different geographical locations all over the world. This API is often updated from the weather stations established in different locations. The data are available in XML, HTML, and JSON formats. The user shall request the data in any format according to their need. The next format for retrieving weather data is historical weather data. This format provides historical data of the weather in hours for any location from 1 month and may be up to 5 years according to the user’s subscription. The days and hour forecast information contribute weather data for about 5 days. For these 5 days, the weather data for every 3 h are provided. The next format for this API is the daily forecast information. This format yields weather data for about 16 days. The weather data for any requested unique location or city can be retrieved by the name of the city, ID, coordinates, and by postal code. The city name and the country code of the city can be added to the request message. Then, the corresponding weather data for the given city can be obtained. The API provides a list of cities with corresponding IDs in their website. The weather data can be accessed using the city ID.
The weather data can also be accessed by the use of geographic coordinates like latitude and longitude and by the city’s zip code. Along with this, the API provides the weather data for multiple cities. The weather forecasting for the cities within a zone (specified by the latitude and longitude) and the cities within a circle defined by center point can be obtained.

The responds of the API contain only the originally measured and calculated data. The responds comprise many parameters like coordinates (latitude and longitude), temperature, humidity, pressure in sea level and ground level, wind speed and direction, and rain and snow volume. If any parameter is not found in the respond, it implies that the particular phenomenon has not happened in the weather stations at the time of measurement. The value of the parameters is retrieved in the specific format defined in the API documentation. The formats have different units. For example, the value of the temperature shall be retrieved in Celsius, Fahrenheit, and in Kelvin units. The values can be retrieved in the users’ required formats. The values shall be retrieved in integer and float. The response of the API call shall be obtained in multiple languages supported by the Web API. The Web API supports numerous languages so as to draw attention from the users all over the world. The API produces the data in the user requested formats. Similar to the OpenWeatherMap API, we bring our own API developed by Flask framework to control and maintain the smart home data.

Sample API call and respond API request in JSON

Listing 02: API respond in JSON

```
{"coord":
{"lon":-3.74,"lat":5.2},
"weather": [{
"id":801,"main":"Clouds","description":"few clouds","icon":"02d"}],
... "id":2288873,
"name":"France",
"cod":200}
```

Listing 03: API request in XML

```
<city id="2288873" name="France">
<coord lon="-3.74" lat="5.2" />
<country>Cl</country>
<sun rise="2019-07-01T06:06:03" set="2019-07-01T18:31:08" />
<city>
...
<weather number="801" value="few clouds" icon="02d" />
<lastupdate value="2019-07-01T18:27:44" />
</current>
```

As JSON is simple to handle, it has become the typical format. The optimization of JSON files plays an important role while parsing the same. There are some requirements for the optimal performance and features of JSON files. Multiple JSON files are needed for better performance. Many of the JSON parsing libraries can read them from a stream when compared with reading from the string. As this is more efficient, preference shall be given for stream reading wherever it is possible. For achieving better optimization, the JSON files can be compressed up to 90%. The JSON files can be compressed in Zip format while communicating to the corresponding web services. Parsing of JSON can be avoided if it is not needed.

In case the web app receives the JSON and stores it to a database or any other storage, it is not necessary to parse them. The process of serialization should be given more importance for better JSON optimization. It is not necessary to serialize all data fields. Instead, the null values can be omitted. The use of shorter field names makes it more human-readable and gives 5–10 percentage boost in performance and parsing. It also allows smaller data packets to transfer along the web. The API requests will be handled by HTTP/GET and raw data in response can be serialized by means of any comfortable serialization methods available in some elegant scripting languages such as iOS swift, Python, Hypertext preprocessor (PHP), and so on. Obviously, the optimization may be sensed in between 3 and 5 Mbps by subtle variations with different service caching among XML (SOAP), REST XML, and REST JSON. Table 2 demonstrates the HTTP topics of Web API.

Implementation of Flask API for ESP8266-MQTT wireless robot

Flask is a lightweight python web framework to construct a flexible API. In order to build the API framework, there are two extensions in Flask that help build RESTful services. The clients of web service are low-cost NodeMCU robots that command requesting the service to add, remove, and modify tasks. This is what is needed to device a method to store independent tasks. An appropriate method is to build a flexible database. The HTTP topics are normally generated by accessing the smart robots as depicted in Table 2.

Flask Framework for green house is depicted in Figure 2 in which publish–subscribe is event-driven and readily pushes messages to smart robots. This involves an additional and essential communication point usually known as the MQTT broker, where the Raspberry Pi is in charge of transmitting all messages between senders and rightful robot clients. Raspberry Pi is responsible for monitoring the publishers as well as subscribers.
in the same way for the sake of knowledge about how many publishers are live. Every client robot that publishes a message to the Raspberry Pi includes a JSON URI as a topic into the message. The topic is normally the routing information for the Pi, and hence the robot that wants to receive messages, subscribes to a firm topic about holding the smart house data. The Pi brings all messages with the corresponding JSON URI or topic to the client who may or may not have to be familiar with each other. They insist to communicate the topic only. This construction facilitates highly scalable solutions without numerable dependencies between data producers and consumers.

Web application is firmly developed using HTML, Flask (Python) web server platform, and JQuery (Figure 3). A call back function in the MQTT protocol is used here to send and receive data to green house. Web interface is made for controlling the smart robot and to monitor ONEWIRE (DS18B20) sensors data and each location of the green house. One of the benefits for the use of these sensors is that it allows digital input to make a network of sensors with a single outlet (GPIO).

Validation

In the assessment, temperature data are gathered from DS18B20 (mounted on ESP8266 robots) through Raspberry Pi via APIs of publish–subscribe within every second. The following steps have been discussed to acquire benefits of above-said Flask API server-based interface for a smart robot:

1. A web interface is implemented with a JSON endpoint. A separate JSON host is constructed in Flask to develop a responsive interface.
2. A web repository with Flask API with database is roughly used to build an interface for control application for implementation without hard-coding the technology of a device.
3. The experimental results have shown that it is better to adapt a lesser response time consumes MQTT than HTTP.
4. For the sake of convenience, the transmitted message is used as an effortless JSON object that can hold a single key-value pair along the publishes on smart robots.
5. The key alternate options for JSON are the standard methods of HTTP, usually GET or POST. However, they are not preserved due to the problems raised in content caching while gathering the sensor readings.

It is evident that when selecting MQTT over HTTP, the RTT as depicted in Figure 4 is mostly lesser than a standard HTTP. It is important to reclaim similar connection to some extent. The efficiency of communication against a smart robot gains is significant compared with the HTTP standards since MQTT is data agnostic and light weight. The ultimate guarantee of efficiency in MQTT protocol gives importance to density for each payload message. This technique ensures additional defiant ways to attain composition against gathered data from sensors. The connection status and delivery of messages to the smart robots are depicted in Figure 5. The response time per connection cycle is examined in Table 3. The critical approach is employed to decrease the payload size where enormous data could be transmitted by sensors eventually.

Figure 6 demonstrates payload size for different numbers of percentages. The simulation result proves that the proposed method has used less amount of

| HTTP method | Action | Topic |
|-------------|--------|-------|
| GET         | Acquire information about a resource | http://192.168.1.30/api/temp |
| POST        | Make a new resource | http://192.168.1.30/api/onewire |
| PUT         | Update resource | http://192.168.1.30/api/onewire |
| DELETE      | Delete a resource | http://192.168.1.30/api/onewire |

HTTP: hypertext transfer protocol; API: application programming interface.
payload size. Figure 7 illustrates the performance comparison in spite of average computation cost for the normal class and also the modified class for the proposed method MQTT.
Conclusion

This article proposed an MQTT-based Flask API framework in the IoT to monitor and control greenhouse system with Raspberry Pi cloud. The proposed architecture presents the frequent publish–subscribe messages among the low-cost ESP8266-attached smart robots. An access interface was developed as a Flask web page for users to access greenhouse from anywhere. JSON is a lightweight data interchange format when compared with the traditional XML. The response messages conspicuously retrieved in the JSON format are cached in effective when the public APIs are requested by the programmers rather than using

---

**Figure 5.** Connection status and messages delivery to robots.

![Figure 5](image.png)

**Figure 6.** Payload size comparison.

![Figure 6](image.png)

**Figure 7.** Performance comparison.

![Figure 7](image.png)
response in XML. This work determined the higher parsing efficiency of the JSON for accessing backend data through available public Flask Web API. An interface was developed in Python Flask to gather information from smart farm’s ambient data to manage people with various sensors connected in low-cost robots. The main contributions of this research are as follow:

1. IoGT: A Flask-based python framework and REST API was developed for accessing low-cost ESP8266 wheeled robots.
2. Data gathered from the robots through MQTT publish–subscribe technique and successfully transferred (also stored for future references) to REST API.
3. The data integrated by robots with IoGT gains more advantage in gathering about spatial information data connected with the irrigation.
4. Visualization techniques and perspectives based on IoGT helped for precision agriculture in the field of farming are highlighted.
5. Adapting a lower response time consumes MQTT than HTTP from the experimental results.
6. Implemented framework insisted that the incredibly lightweight JSON object hold data between the publishers and subscribers (ESP8266 robots).

Further studies will create an adapted and responsive web application with the help of MQTT and machine learning methods51–61 to allow users to interact by topics against smart wireless robots with minimal efforts.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) received no financial support for the research, authorship, and/or publication of this article.

ORCID iDs
Y Harold Robinson https://orcid.org/0000-0002-4881-7103
Mahmoud Ismail https://orcid.org/0000-0003-2706-4621

References
1. Ara SS, Shamszaman ZU and Chong I. Web-of-objects based user-centric semantic service composition methodology in the Internet of Things. Int J Distrib Sens N 2014; 10: 482873.
2. Khan M, Silva BN and Han K. A web of things-based emerging sensor network architecture for smart control systems. Sensors 2017; 17(2): 332.
3. Shelby Z, Hartke K and Bormann C. The constrained application protocol (CoAP). Fremont, CA: Internet Engineering Task Force (IETF), 2014.
4. Davoli L, Belli L, Cilfone A, et al. Integration of Wi-Fi mobile nodes in a web of things testbed. ICT Express 2016; 2(3): 96–99.
5. Hernández-Rojas DL, Fernández-Caramés TM, Fraga-Lamas P, et al. A plug-and-play human-centered virtual TEDS architecture for the web of things. Sensors 2018; 18: 2052.
6. Stanford-Clark A and Truong HL. MQTT for sensor networks (MQTT-SN) protocol specification, version 1.2. Armonk, NY: International Business Machines (IBM), 2013.
7. Ramlee RA, Othman MA, Leong MH, et al. Smart home system using android application. In: Proceedings of the international conference of information and communication technology (ICoICT), Bandung, Indonesia, 20–22 March 2013, pp.277–280. New York: IEEE.
8. Teimourzadeh R, Ahmed SA, Chan KW, et al. Smart GSM based home automation system. In: Proceedings of the IEEE conference on systems, process & control (ICSPC), Kuala Lumpur, Malaysia, 13–15 December 2013, pp.306–309. New York: IEEE.
9. Meng J, Mei S and Yan Z. RESTful web services: a solution for distributed data integration. In: Proceedings of the international conference on computational intelligence and software engineering, Wuhan, China, 11–13 December 2009. New York: IEEE.
10. Piyare R and Tazil M. Bluetooth based home automation system using cell phone. In: Proceedings of the 15th international symposium on consumer electronics (ISCE), Singapore, 14–17 June 2011, pp.192–195. New York: IEEE.
11. Hamad H, Saad M and Abed R. Performance evaluation of RESTful web services for mobile devices. Int Arab J Inf Techn 2010; 1(3): 72–78.
12. Wenbo Y, Quanyu W and Zhenwei G. Smart home implementation based on Internet and WiFi technology. In: Proceedings of the 2015 34th Chinese control conference (CCC), Hangzhou, China, 28–30 July 2015, pp.9072–9077. New York: IEEE.
13. White M, Linares-Vásquez M, Johnson P, et al. Generating reproducible and replayable bug reports from android application crashes. In: Proceedings of the 23rd IEEE international conference on program comprehension (ICPC), Florence, 18–19 May 2015. New York: IEEE.
14. Bramantya W, Kusumo DS and Munajat B. Modularizing RESTful web service management with aspect oriented programming. In: Proceedings of the 3rd international conference on information and communication technology (ICoICT), Nusa Dua, Bali, 27–29 May 2015, pp.358–363. New York: IEEE.
15. Xanthopoulos S and Xinogalos S. A comparative analysis of crossplatform development approaches for mobile applications. In: Proceedings of the 6th Balkan conference in informatics, Thessaloniki, 19–21 September 2013. New York: ACM.
16. Tian M, Voigt T, Naumowicz T, et al. Performance considerations for mobile web services. *Comput Commun* 2004; 27(11): 1097–1105.

17. Santhi K, Zayaraz G and Vijayalakshmi V. Resolving aspect dependencies for composition of aspects. *Arab J Sci Eng* 2014; 40(2): 475–486.

18. Roh HG, Jeon M, Kim JS, et al. Replicated abstract data types: building blocks for collaborative applications. *J Parallel Distrib Com* 2011; 71(3): 354–368.

19. Nunes R, Rebouças M, Soares-Neto F, et al. Visualizing swift projects as cities. In: *Proceedings of the 39th international conference on software engineering companion*, Buenos Aires, Argentina, 20–28 May 2017, pp.368–370. New York: IEEE.

20. Pawlak R, Duchien L, Florin G, et al. Dynamic wrappers: handling the composition issue with JAC. In: *Proceedings of the 39th international conference and exhibition on technology of object-oriented languages and systems (TOOLS 39)*, 29 July–3 August 2001, pp.56–65. New York: IEEE.

21. Richardson L, Amundsen D and Ruby S. RESTful web APIs. Newton, MA: O'Reilly Media, Inc., 2013.

22. Al-Masri E and Hamdi M. RFID-based approach for monitoring patient’s health inside hospitals. In: Elleithy K and Sobh T (eds) *New trends in networking, computing, E-learning, systems sciences, and engineering*. Cham: Springer, 2015, pp.607–613.

23. Cook DJ, Youngblood M, Heieran EO, et al. MavHome: an agent-based smart home. In: *Proceedings of the first IEEE international conference on pervasive computing and communications 2003 (PerCom 2003)*, Fort Worth, TX, 26–26 March 2003, pp.521–524. New York: IEEE.

24. Das SK, Cook DJ, Battacharya A, et al. The role of prediction algorithms in the MavHome smart home architecture. *IEEE Wirel Commun* 2002; 9(6): 77–84.

25. Valtchev D and Frankov I. Service gateway architecture for a smart home. *IEEE Commun Mag* 2002; 40(4): 126–132.

26. Di Giorgio A and Pimpinella L. An event driven smart home controller enabling consumer economic saving and automated demand side management. *Appl Energ* 2012; 96: 92–103.

27. Byun J, Jeon B, Noh J, et al. An intelligent self-adjusting sensor for smart home services based on ZigBee communications. *IEEE T Consum Electr* 2012; 58(3): 794–802.

28. Lee N, Lee H, Lee H, et al. Smart home web of object architecture. In: *Proceedings of the international conference on information and communication technology convergence (ICTC)*, Jeju, South Korea, 28–30 October 2015, pp.1212–1214. New York: IEEE.

29. Hafidh B, Al Osman H, Arteaga Falconi JS, et al.SITE: the simple Internet of Things enabler for smart homes. *IEEE Access* 2017; 5: 2034–2049.

30. Wu S, Rendall J, Smith M, et al. Survey on prediction algorithms in smart homes. *IEEE Internet Things* 2017; 4: 636–644.

31. Alam MR, Reaz MBI and Mohd Ali MA. SPEED: an inhabitant activity prediction algorithm for smart homes. *IEEE T Syst Man Cy A* 2012; 42(4): 985–990.

32. Chen L, Thombre S, Järvinen K, et al. Robustness, security and privacy in location based services for future IoT: a survey. *IEEE Access* 2017; 5: 8956–8977.

33. Latif MA, Ullah F and Lee S. Extensible privacy framework for web of objects based ubiquitous services. In: *Proceedings of the 2015 international conference on information and communication technology convergence (ICTC)*, Jeju, South Korea, 28–30 October 2015, pp.1081–1084. New York: IEEE.

34. Wang X and Mu Y. Communication security and privacy support in 6LoWPAN. *J Inf Secur Appl* 2017; 34: 108–119.

35. Majeed A, Ullah F and Lee S. Vulnerability-and diversity aware anonymization of personally identifiable information for improving user privacy and utility of publishing data. *Sensors* 2017; 17(5): 1059.

36. Zur Muehlen M, Nickerson JV and Swenson KD. Developing web services choreography standards—the case of REST vs. SOAP. *Decis Support Syst* 2005; 40(1): 9–29.

37. Elappila M, Chinara S and Parhi DR. Survivable path routing in WSN for IoT applications. *Pervasive Mob Comput* 2018; 43: 49–63.

38. Han SN and Crespi N. Semantic service provisioning for smart objects: integrating IoT applications into the web. *Future Gener Comp Sy* 2017; 76: 180–197.

39. Evans D. *The internet of everything: how more relevant and valuable connections will change the world*. San Jose, CA: Cisco IBSG, 2012, pp.1–9.

40. Gubbi J, Buyya R, Marusic S, et al. Internet of Things (IoT): a vision, architectural elements, and future directions. *Future Gener Comp Sy* 2013; 29(7): 1645–1660.

41. Ahmadi-Karvigh M, Becerik-Gerber B and Soibelman L. Intelligent adaptive automation: a framework for an activity-driven and user-centered building automation. *Energ Buildings* 2019; 188–189: 184–199.

42. Janssen CP, Donkera SF, Brumby DP, et al. History and future of human-automation interaction. *Int J Hum Comput Int* 2019; 131: 99–107.

43. Amersh S, Weld D, Vovorereanu M, et al. Guidelines for human-AI interaction. In: *Proceedings of the SIGCHI conference on human factors in computing systems*, Glasgow, May 2019. New York: ACM.

44. Borojeni SS, Weber L, Heuten W, et al. From reading to driving: priming mobile users for take-over situations in highly automated driving. In: *Proceedings of the 20th international conference on human-computer interaction with mobile devices and services*, Barcelona, September 2018. New York: ACM Press.

45. Chien SY, Lin YL, Lee PJ, et al. Attention allocation for human multi-robot control: cognitive analysis based on behavior data and hidden states. *Int J Hum Comput St* 2018; 117: 30–44.

46. Gould SJ, Cox AL and Brumby DP. Influencing and measuring behaviour in crowd sourced activities. In: Filinowicz M and Tzankova V (eds) *New directions in third wave human-computer interaction*, vol. 2. Cham: Springer, 2018, pp.103–130.

47. Janssen CP, Boyle L, Kun AL, et al. A hidden Markov framework to capture human–machine interaction in automated vehicles. *Int J Hum Comput Int* 2019; 35: 947–955.
48. Jha K, Doshi A, Patel P, et al. A comprehensive review on automation in agriculture using artificial intelligence. *Artif Intell Agric* 2019; 2: 1–12.

49. Post MA, Bianco A and Yan XT. Autonomous navigation with open software platform for field robots. In: Gusikhin O and Madani K (eds) *Informatics in control, automation and robotics*. ICINCO 2017 (Lecture notes in electrical engineering), vol. 495. Cham: Springer, 2020, pp.425–450.

50. Tao B, Zhao XW and Ding H. Mobile-robotic machining for large complex components: a review study. *Sci China Technol Sc* 2019; 62: 1388–1400.

51. Garg R, Mittal M and Son HL. Reliability and energy efficient workflow scheduling in cloud environment. *Cluster Comput* 2019; 22: 1283–1297.

52. Jha S, Kumar R, Son HL, et al. Deep learning approach for software maintainability metrics prediction. *IEEE Access* 2019; 7: 61840–61855.

53. Kapoor R, Gupta R, Kumar R, et al. New scheme for underwater acoustically wireless transmission using direct sequence code division multiple access in MIMO systems. *Wirel Netw* 2019; 25: 4541–4553.

54. Khan T, Singh K, Son HL, et al. A novel and comprehensive trust estimation clustering based approach for large scale wireless sensor networks. *IEEE Access* 2019; 7: 58221–58240.

55. Krishnan RS, Julie EG, Robinson YH, et al. A new algorithm for high power node multicasting in wireless sensor networks. *IEEE Access* 2019; 7: 38584–38592.

56. Pal S, Kumar R, Son HL, et al. Novel probabilistic resource migration algorithm for cross-cloud live migration of virtual machines in public cloud. *J Supercomput* 2019; 75: 5848–5865.

57. Popa A, Hnatiuc M, Paun M, et al. An intelligent IoT-based food quality monitoring approach using low-cost sensors. *Symmetry* 2019; 11(3): 374.

58. Robinson YH, Krishnan RS, Julie EG, et al. Neighbor knowledge-based rebroadcast algorithm for minimizing the routing overhead in mobile ad-hoc networks. *Ad Hoc Netw* 2019; 93: 101896–101909.

59. Robinson YH, Julie E, Kumar R, et al. Probability-based cluster head selection and fuzzy multipath routing for prolonging lifetime of wireless sensor networks. *Peer Peer Netw Appl* 2019; 12(5): 1061–1075.

60. Sahu DP, Singh K, Manju M, et al. Heuristic search based localization in mobile computational grid. *IEEE Access* 2019; 7: 78652–78664.

61. Tam NT, Hai DT, Son HL, et al. Improving lifetime and network connections of 3D wireless sensor networks based on fuzzy clustering and particle swarm optimization. *Wirel Netw* 2018; 24(5): 1477–1490.