HDAA: High-speed Data Acquisition Algorithm of IoT

Zhijin Qiu, Suiping Qi, Zhengyu Zhou, Tong Hu, Jing Zou and Dongming Wang

November 12, 2018

HDAA: High-speed Data Acquisition Algorithm of IoT

Zhijin Qiu[1], Suiping Qi[1], Zhengyu Zhou[2], Tong Hu[1], Jing Zou[1], and Dongming Wang[1]

[1] Institute of Oceanographic Instrumentation, Qilu University of Technology(Shandong Academy of Sciences), Qingdao, 266000, China
[2] Shandong University of Technology, Zibo, 255012, China
qzjouc@163.com

Abstract. As the 'Industry 4.0' has been proposed, the Internet of Things (IoT) has been widely used in more and more fields, and the higher demand for data acquisition efficiency. Unfortunately, the format, type, and access methods of data sources in different areas are diverse, making low efficiency of data acquisition. To address this issue, based on the data acquisition middleware of IoT (DAQ-Middleware), a novel **H**igh-speed **D**ata **A**cquisition **A**lgorithm of IoT (HDAA) is proposed. The algorithm includes parallel data acquisition algorithm and acquisition efficiency optimization heuristic method. According to the characteristics of different data sources and data interfaces, the mapping relationship is established between data sources and data interfaces, which improves the data acquisition efficiency of IoT. In the practical application and simulation environment, the data acquisition efficiency of HDAA is analyzed and compared with other algorithms. The results show that HDAA greatly improves the efficiency of data acquisition. In particular, the advantage of the algorithm is more obvious in the case of more data sources and less data interfaces.

Keywords: Internet of Things · Data acquisition · Middleware · Parallel acquisition · Heuristic.

1 Introduction

With the development of pervasive computing, RFID, and sensor networks, the application and development of IoT technologies have been promoted. In different fields, the IoT realizes the acquisition, storage, analysis and display of global sensor data by connecting intelligent instruments, completes the analyzsis of big data, and guides enterprises to make strategic decisions and upgrades [1].
In order to realize data integration for existing IoT systems, the access data source also includes existing databases and data files, so the data source of the IoT is mainly divided into two parts: a) sensing instruments including smart sensors, RFID tags, computers and mobile devices; b) databases and data files. At present, no matter which kind of data source, there are different kinds of

data types, communication protocols and access methods, which make the development of the IoT system extremely complicated. Obviously, it is incredible to have a uniform communication protocol and data format for different instrument manufacturers.

In terms of IoT system design, many related studies have been conducted. Most of them mainly focus on the overall architecture design [2–5] and data display method [6–9] in IoT systems. There is no much study about the diversity of the data sources' format. In the research area of distributed systems, the relevant scholars have designed different architectures or algorithms for data acquisition system. Kovac [10] propose the use of virtual instrument technology and GPIB interface to achieve the acquisition of sensor data, which increases the convenience of access to the sensing instruments to a certain extent. Qiu et al. [11] propose a high performance data acquisition algorithm based on the analysis of dynamic delay characteristics of data acquisition. But this algorithm is only suitable for specific sensing instruments, and does not apply to multiple data sources. At the same time, the algorithm still using a serial data acquisition method for sequentially acquiring data from multiple data sources. Until recently, in [12] propose the concept of Complex Virtual Instrument System to handle multiple data sources. However, their architectures and application areas are still restricted. The Open GIS Consortium (OGC) [13] proposes to use Programmable Underwater Connector with Knowledge (PUCK) protocol to integrate the physical instruments automatically. Although, this has solved lots of problems in system integration and development, it must modify the ocean observing instruments and add PUCK model. Doing so increases the cost of instrument manufactures as well. In summary, none of the solutions above can settle comprehensively all the mentioned problems in data acquisition of IoT systems.

In this paper, according to the experiences of participation in the development of IEEE 1851, IEEE 2402 international standard and GB/T 33137-2016 national standard, we propose a data acquisition middleware based on IoT: DAQ-Middleware. DAQ-Middleware can provides sensing data for IoT system of the different areas through the standardized data interface.

In summary, the major contributions of this paper can be summarized as follows: **a)** A scalable IoT data acquisition middleware is proposed, which not only supports access to the sensing instruments with interfaces of serial, network, GPIB and USB, but also supports access to various types of databases and files with interfaces of FTP, Web Service and MQ. **b)** HDAA is proposed, which include a parallel data acquisition algorithm and an acquisition efficiency optimization heuristic method. **c)** In the application scenario and simulation environment, the data acquisition efficiency of HDAA is compared with other algorithms. The purpose of this work is reduce the development cost and improve the data acquisition efficiency.

The rest of this article is organized as follows: Section 2 presents the architecture model of data acquisition system. Section 3 is an overview of DAQ-Middleware, and the HDAA is proposed in Section 4. Section 5 analyzes and compares the performance of HDAA. Concluding remarks are presented in Section 6.

2 Architecture model

2.1 Hierarchy model of IoT

Recently, many scholars put forward the object hierarchy model of IoT according to the different requirements, in order to facilitate the development and maintenance of IoT systems. According to [7], as shown in Figure 1, the object hierarchy model is divided into four layers (sensing layer, acquisition layer, management layer, and application layer). Each layer is independent of each other, and only through the data interfaces to interact between the adjacent layers. The output of a layer is the input of next level. In some cases, management and application layers are collectively called application system.

2.2 Structure model of DAQ-Middleware

As shown in Figure 2, DAQ-Middleware is located in the acquisition layer of the IoT hierarchical model. Its functions include receiving control commands from the application system, periodically obtaining the data of data sources, and transmitting the obtained data to the application system through the s-tandardized data interfaces. DAQ-Middleware uses Socket to communicate with application system. In different circumstances, DAQ-Middleware connects to the sensing instruments via interfaces, such as RS232, RS485, Ethernet and USB, and connects to the existing databases and data files through other interfaces, such as Web Service, MQ and FTP.

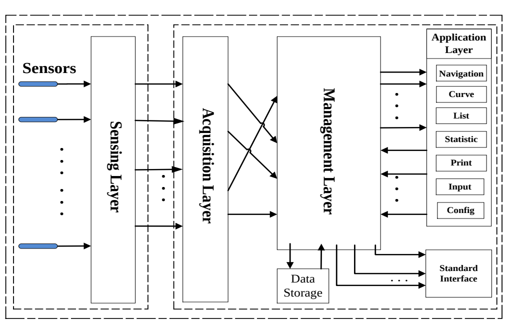


Fig. 1: Hierarchy model of IoT.

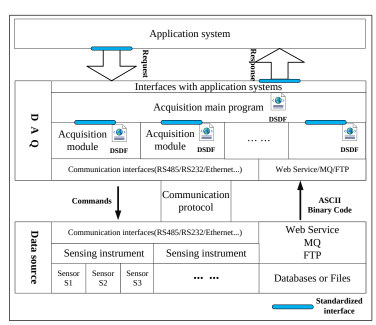


Fig. 2: Architecture of DAQ.

3 Overview of DAQ-Middleware

This section describes the data source description model, defines the interface between the application system and the middleware, and defines the interaction between the main program and the acquisition module. It can not only provide unified monitoring data for IoT applications in different fields, but also realize the addition, deletion and modification of data sources by modifying the data source description information.

3.1 Data source description file

The data source description file (DSDF) describes all the information of the data sources that are accessed. DAQ-Middleware can obtain various information, such as attributes, interfaces and sensor parameters of the data sources by analyzing DSDF. In order to facilitate the DSDF analysis, the format of DSDF is defined. DSDF is described in XML format. DSDF contains multiple data sources, and each data source includes information of attributes, interfaces and parameters. The node of attributes holds information such as serial number (GlbID), the name of data sources (Name), the name of acquisition module (Model), acquisition module's storage path (Path), the number of sensor parameters (ParameterNum), and data sources' provider (Manufacturer).
DSDF describes the ways of access, and the interface parameter's information of each data source, so that we can achieve the connection from DAQ-Middleware to the data sources. Different data sources are connected uses different interfaces, and the parameters of different data interfaces are not the same. The description of the interface information includes the RS-485, RS-232, Ethernet, GPIB, USB, Web Service, FTP and MQ. Different types of interfaces need to describe the parameters are different. For example, RS-232 needs to describe the information of serial number, baud rate, data bits, stop bits, and checksum, but RS-485 also includes the information of sensing device's address.

3.2 Interface Standardization

DAQ-Middleware is an independent system, so if you want to achieve no matter which the field of IoT can access to DAQ-Middleware, we need to standardize the interfaces between DAQ-Middleware and the application system. The communication parameter description file is used to describe the interface information, which includes four parameters: IPAddress, FilePort, ControlPort and DataPort. All these interfaces adopting Socket. For details of the interface, refer to the GB/T 33137-2016 national standard (China).

3.3 Acquisition Module

Figure 2 shows that DAQ-Middleware includes the acquisition main program and the acquisition module. DAQ-Middleware uses communication interfaces

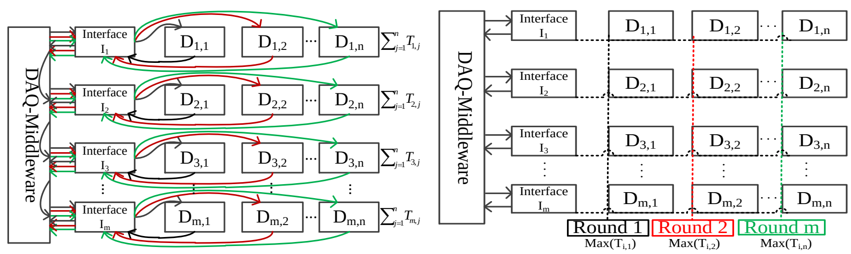
to acquire the data from the data sources based on specific protocol. We have developed a unique acquisition module for each data source based on different communication protocols.
The functions of acquisition main program: It accepts control commands and data request commands from the application system, and returns DSDF and the processing results. It is responsible for obtaining DSDF, and synthesizing-analyzing the information to implement the loading of the acquisition module. It instantiates the communication interface, interacts with the acquisition module, organizes the obtained sensor data in a standard format, and returns the sensor data to the application system.
The functions of acquisition module: It obtains sensor data from data sources via communication protocol, and the information of communication interfaces, which have been instantiated by the acquisition main program. It interacts with the acquisition main program through the standard interface, obtains the sensor list, and returns the sensor data to the acquisition main program.
The standardization of interface between the acquisition main program and the acquisition module enables the dynamic modification and deletion of the data source by modifying DSDF.

4 High-speed data acquisition algorithm for IoT

4.1 Parallel Data Acquisition

As Figure 2 shown, the main program receives data request commands from application system, which contain the sensor list of multiple sensing parameters for multiple data sources. In general (Serial Data Acquisition), the data acquisition middleware analyzes the sensor list, obtains the data sources to be acquired, and then performs the data acquisition *in turn*. However, as shown in Figure 3a, with the increasing number of data sources, this method leads to long acquisition cycle and low efficiency. Since the characteristics of data acquisition module are independently designed in the DAQ-Middleware, we propose a novel parallel data acquisition algorithm, aiming to improve the efficiency of data acquisition. According to the principle of computer interfaces, there is a situation in which the same interface can access multiple data sources. For example, an RS-485 interface can access multiple sensing instruments (with different addresses). But in order to accurately analyze the returned sensor data, every time the data acquisition middleware can only communicate with a single device on the same interface. As Figure 3b shown, since data acquisition between different data sources is independent of each other, we can perform them in parallel.

Let I_i denote interface information, where i is the index of interface and $i = 1, 2, \cdots m$. The data source information is defined as R_j and j is the index of data source for each interface connection. The number of data sources connected to the I_i interface is $n_i^{'}$, i.e., $j = 1, 2, \cdots n_i^{'}$. As shown in Figure 3b, we define a matrix D_{m*n}, which represents the data source information of each accessed interface, obviously, $n = max(n_i^{'})$, $i = 1, 2, \cdots m$. Because different data interfaces are independent of each other, we divide the data acquisition into



(a) Serial (b) Parallel

Fig. 3: Serial/Parallel data acquisition algorithm.

n rounds, and each round we acquire data from a data source on m interfaces. Note that, there is a possible situation where no data source corresponding to the elements of matrix $D_{i,j}$, so we set that value 0. Otherwise, the data source's ID. The parallel data acquisition algorithm is described in detail as Algorithm 1. Algorithm 1 can in certain degree improve the efficiency of data acquisition. Ac-

---

Algorithm 1: Parallel Data Acquisition Algorithm

Input: Matrix D_{m*n} obtained from DSDF
Output: Acquired sensor data and acquisition time T_{m*n}

```
for round j = 1 to n do
    for interface i = 1 to m do
        if D_{i,j} ≠ 0 then
            Acquisition main program sends the sensor list to the acquisition
            module corresponding to data source D_{i,j};
        end
    end
    for interface i = 1 to m do
        if D_{i,j} ≠ 0 then
            Acquisition modules collect data according to the communication
            protocol and record the time T_{i,j} required for data acquisition;
        end
        if D_{i,j} = 0 then
            T_{i,j} = 0;
        end
    end
end
```

---

cording to the obtained acquisition time T_{m*n}, we can calculate the time required to complete one round of data acquisition. The time required for each round of data acquisition is the maximum value of each column of matrix T_{m*n}, namely

$t_j = \max_i T_{i,j}$. The time t required to complete a complete data acquisition is $t = \sum_{j=1}^{n}(t_j) = \sum_{j=1}^{n}(\max_i T_{i,j})$. The time $t^{'}$ of the Serial Data Acquisition is $t^{'} = \sum_{i=1}^{m}\sum_{j=1}^{n}(T_{i,j})$. Obviously, $t \leq t^{'}$, and the efficiency of data acquisition is improved significantly with the increasing of the number of data sources.

4.2 Acquisition efficiency optimization heuristic method

Since different types of interfaces have different characteristics and constraints, there are different acquisition methods when we allocate instruments and interfaces. In order to improve the efficiency of data acquisition, we can adjust the matrix D_{m*n} to reduce the total acquisition time t, while satisfying the constraints of interface attribute. This optimization problem can be described as follows.

$$
\begin{aligned}
\min \quad & t = \sum_{j=1}^{n}(t_j) = \sum_{j=1}^{n}(\max_i(T_{i,j})) \\
s.t. \quad & I_i \textit{ is satisfied.} \\
& \textit{Each data source only belongs to one interface.}
\end{aligned}
$$

We can find the optimal distribution using brute-force method. There are m^n allocations. Thus, brute-force method has an exponential time complexity. Instead, we now propose a heuristic method, which can obtain a reasonable data resource and interface matching efficiently. Each round of data acquisition time is decided by $t_j = \max_i(T_{i,j})$.

The basic idea of heuristic is to put the data sources of different interfaces with similar data acquisition time in the same round to save data acquisition time. The detailed algorithm is given in Algorithm 2. Firstly, there are often circumstances that the interfaces with different IDs belong to the same interface type, so we consolidate data sources of the same type of interface in T_{m*n} to obtain $G^{'}_{p*q}$. p is the number of interface types and q is the number of data sources owned by that type interfaces. Obviously, $p \leq m$ and $q \geq n$. The data source communication time for each interface type is then sorted in descending order to obtain G_{p*q}. Next, keep the columns of $D_{i,j}$ unchanged, and the rows gradually increasing. According to the interface I_i and matrix G_{p*q}, we assign data sources to the interface in turn, and then move to the next column. Repeat this until all the data sources are completed. Lastly, the relationship matrix $D_{i,j}$ is obtained between the data sources and the interfaces.

5 Performance comparison and analysis

In the practical application and simulation experiment scene, the performance of HDAA is analyzed and verified.

Algorithm 2: Acq-Efficiency Optimization Heuristic

Input: I_i, T_{m*n}
Output: $D_{i,j}$
Consolidate data sources of the same type interface and get G'_{p*q};
for *each number of interface types* **do**
  Sort data source communication time in descending order in G'_{p*q} to get G_{p*q};
end
for *each round j* **do**
  for *each interface i* **do**
    if *data source with interface type i of* G_{p*q} *has not been allocated* **then**
      $D_{i,j}$=ID of the data source;
    else
      $D_{i,j} = 0$;
    end
  end
end

5.1 Practical application

Experimental environment. The field of household appliances testing has a wide range of needs for IoT application systems. Table 1 summarizes the 9 data sources, which consist of sensing instruments, databases and data files (40 instruments and 410 parameters). Each data source corresponds to a data acquisition module, so we have developed a total of 9 acquisition modules, and configured each DSDF.

Result analysis. DAQ-Middleware adopts Algorithm 1 and Algorithm 2. The result show that the acquisition efficiency of HDAA is five times that of serial data acquisition algorithms.

We consider an IoT system where 40 sensing instruments (Table 1) are connected to the proposed middleware via 14 interfaces. The serial data acquisition algorithm sequentially obtains the data from 40 sensing instruments, while Algorithm 1 can obtain data in parallel and use Algorithm 2 to schedule the acquisition.

Particularly, we sort the acquisition time of different data sources to obtain matrix G_{7*10}. According to matrix G_{7*10}, and adopting the acquisition efficiency optimization heuristic, D_{14*5} is obtained. Then perform the parallel data acquisition algorithm based on D_{14*5}, T_{14*5} also is obtained. The time required t for HDAA to complete a data acquisition is only $t = \sum_{j=1}^{n}(t_j) = \sum_{j=1}^{n}(\max_i(T_{i,j})) = 387ms$, while the time required for the serial acquisition t' is $t' = \sum_{i=1}^{m}\sum_{j=1}^{n}(T_{i,j}) = 1,979ms$. It is obviously that t is only about 1/5 of t'. So the proposed HDAA (parallel data acquisition algorithm and acquisition efficiency optimization heuristic) can improve the efficiency of data acquisition. In addition, when the number of access instruments and interfaces increases, such enhancement is more obvious.

Table 1: Information of data sources.

Data sources: Sensor Devices (Interface types: RS-232, RS-485, USB, Ethernet)				
Name	Function	Parameters	Instruments	Total of Parameters
MX100	Acquisition Temperature	60	5	300
SR93	Temperature Controller	2	4	8
8775A	Power Meter	5	4	20
UT35A	Indicating Controller	4	4	16
Anemometer	Measuring Winds	2	5	10
Data sources: Database and Data File (Interface types: Web Service, FTP, MQ)				
Nmae	Function	Parameters	Instruments	Total of Parameters
Flowmeter	Measure Flow Rate	2	2	4
Manometer	Measure Liquid Pressure	2	2	4
Counter	Record Switching Doors	2	4	8
Vibrator	Measure Vibration	4	10	40

At the same time, under the condition of satisfying the interface attribute constraints, we compare the proposed Algorithm 2 with random allocation method in the acquisition of data sources. As shown in Figure 4, the random acquisition method is based on Algorithm 1, but use random scheduling. Its data acquisition time is lower than the serial data acquisition algorithm, but higher than HDAA.

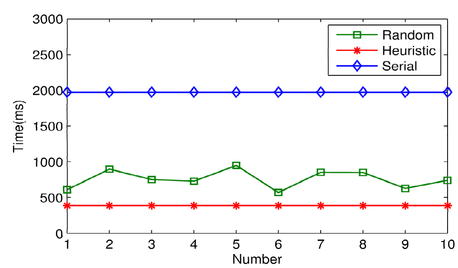


Fig. 4: Acquisition efficiency in practical application.

5.2 Simulation

Simulation environment. In order to further verify the acquisition efficiency of the algorithm in large-scale data sets, we simulated the data acquisition delay time and then compared it with different data acquisition algorithms. The article [14] analysis shows that the data acquisition delay time obeys the Gaussian distribution of $N(10, 4)$. Ten delay time data sets are randomly generated based on the distribution. For different algorithm, the average of the acquisition efficiency on the data set was compared and analyzed.

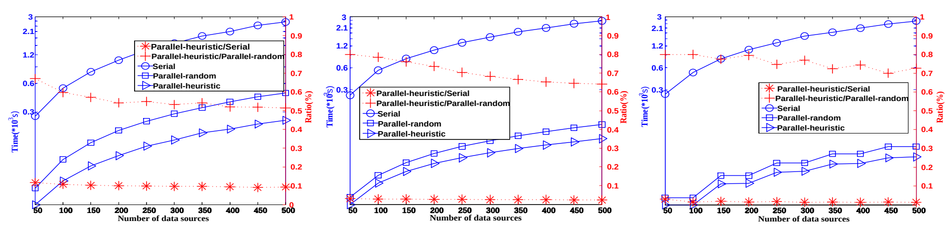
Comparison algorithm. Compare the acquisition efficiency of different algorithms.

- Serial data acquisition algorithm (Serial): Acquire all sensing instruments one by one.
- Randomly assigned parallel data acquisition algorithm (Parallel-random): Based on Algorithm 1, the acquisition order of sensing instruments on each interface is randomly assigned.
- HDAA (Parallel-heuristic): Combining Algorithm 1 and Algorithm 2, based on the parallel data acquisition algorithm, using the acquisition efficiency optimization heuristic method, the acquisition order of the sensing instruments on each interface is allocated.

Result analysis. The data acquisition efficiency of the three algorithms was compared and analyzed under different numbers of sensing instruments and different numbers of interfaces.

When the number of interfaces is 10, 50 and 100, the relationship between accessing different numbers $(50 - 500)$ of sensing instruments and the acquisition efficiency, and the ratio of data acquisition efficiency of different algorithms are analyzed and compared. The result as shown in Figure 5a, 5b and 5c, since Serial requires a long time, we use a log non-uniform interval for the ordinate.

In Figure 5a, 5b and 5c, it can be concluded that with the same number of interfaces, as the number of sensing instruments increases, the data acquisition time increases. The time of Serial is significantly higher than the Parallel-heuristic. When the number of interfaces is 100 and the number of sensing instruments is 50 and 100, the data acquisition time is the same. Because the number of interfaces is greater than or equal to the number of sensing instruments, only one sensing instrument is connected to each interface, and the data acquisition time is the same and the efficiency is the highest. As the number of sensing instruments increases, the acquisition efficiency ratio becomes smaller between the Parallel-heuristic and the other two algorithms.



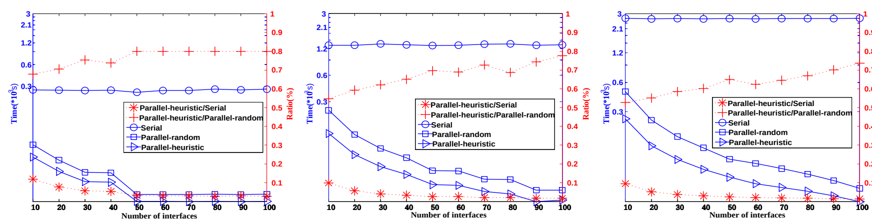
(a) #Interfaces 10 (b) #Interfaces 50 (c) #Interfaces 100

Fig. 5: Data acquisition efficiency of different number of data sources.

When the number of sensing instruments is 50, 250 and 500, the relationship between different numbers $(10-100)$ of interfaces and data acquisition time, and the ratio of data acquisition efficiency of different algorithms are analyzed and compared.

In Figure 6a, 6b and 6c, under the same number of sensing instruments, as the number of interfaces increases, the acquisition efficiency is gradually increased. Since the number of sensing instruments is constant for each experiment, the data acquisition time of Serial does not change as the number of interfaces increases. When the number of sensing instruments is 50, the number of interfaces is greater than 50, and the data acquisition time is no longer reduced. At the same time, with the increase of the number of interfaces, the ratio between Parallel-heuristic and Parallel-random is reduced, and the ratio is increased between Parallel-heuristic and Serial.

In summary, the data acquisition efficiency of the Parallel-heuristic is significantly higher than the other two algorithms, and as the number of sensing instruments increases and the number of interfaces decreases, the data acquisition efficiency is more significant.



(a) #Data sources 50 (b) #Data sources 250 (c) #Data sources 500

Fig. 6: Data acquisition efficiency of different number of interfaces.

6 Conclusion

Based on the DAQ-Middleware, this paper analyzes the characteristics between data source and interface, establishes the mapping relationship between data source and interface, and proposes a HDAA. In the practical application and simulation, the data acquisition efficiency of Serial, Parallel-random and HDAA are analyzed and compared. The results show that HDAA greatly improves the data acquisition efficiency. What's more, with the number of sensing instruments and the data interface decreases, the data acquisition efficiency is more significant.

Considering that the IoT application system is very widely used, future work in more IoT applications will verify, compare and analyze the performance of middleware and algorithm performance.

Acknowledgment: The study is supported by the National Natural Science Foundation of China (Nos. 41606112 and 41705046) and Key Research and Development Program of Shandong Province (No. 2016JMRH0538). Zhijin Qiu is the corresponding author.

References

1. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems **29**(7), 1645–1660 (2013)
2. Peng, M., Li, Y., Zhao, Z., Wang, C.: System architecture and key technologies for 5G heterogeneous cloud radio access networks. IEEE network **29**(2), 6–14 (2015)
3. Riedel, T., Fantana, N., Genaid, A., Yordanov, D., Schmidtke, H.R., Beigl, M.: Using web service gateways and code generation for sustainable IoT system development. In: 2010 Internet of Things (IOT), pp. 1–8. IEEE, Tokyo (2010)
4. Ning, H., Wang, Z.: Future internet of things architecture: like mankind neural system or social organization framework?. IEEE Communications Letters **15**(4), 461–463 (2011)
5. Qiu, Z., Hu, T., Qi, s., Liu, s., Wang, x., Guo, z.: Automatic Matching Method for Ocean Observation Elements Based on Edge Computing. In: International Conference on Big Data Computing and Communications, pp. 174–180. IEEE, Chicago (2018) https://doi.org/10.1109/BIGCOM.2018.00035
6. Gigli, M., Koo, S.G.: Internet of Things: Services and Applications Categorization. Internet of Things **1**(2), 27–31 (2011)
7. Qiu, Z., Guo, Z., Guo, S., Qiu, L., Wang, X., Liu, S., Liu, C.: IoTI: Internet of things instruments reconstruction model design. In: Instrumentation and Measurement Technology Conference Proceedings (I2MTC), pp. 1–6. IEEE, Taiwan (2016) https://doi.org/10.1109/I2MTC.2016.7520533
8. Kelly, S.D.T., Suryadevara, N.K., Mukhopadhyay, S.C.: Towards the implementation of IoT for environmental condition monitoring in homes. IEEE Sensors Journal **13**(10), 3846–3853 (2013)
9. Qiu, Z., Hu, N., Guo, Z., Qiu, L., Guo, S., Wang, X.: IoT sensing parameters adaptive matching algorithm. In: International Conference on Big Data Computing and Communications, pp. 198–211. Springer, Cham (2016) https://doi.org/10.1007/978-3-319-42553-5_17
10. Kovac, V.S.K.: Virtual instrumentation and distributed measurement systems. J. Electr. Eng **55**(1-2), 50–56 (2004)
11. Qiu, Z., Guo, Z., Liu, C.: Adaptive high-speed data acquisition algorithm in sensor network nodes. Journal of Southeast University. Natural Science Edition **42**(A02), 238–244 (2012)
12. Qiu, Z., Guo, Z., Wang, Y., Hu, T., Liu, C., Li, T.: Reconstruction Model of Ocean Observing Complex Virtual Instrument. Journal of Ocean University of China **17**(5), 1159–1170 (2018). https://doi.org/10.1007/s11802-018-3445-6
13. Toma, D.M., O'Reilly, T., del Rio, J., Headley, K., Manuel, A., Broring, A., Edgington, D.: Smart sensors for interoperable smart ocean environment. In: OCEANS, 2011 IEEE, pp. 1–4. IEEE, Spain (2011)
14. CHEN, Z., HUANG, Y., TANG, C., QU, L.: Estimation of transmission delay of time synchronization information for Internet of things of coal mine underground. Industry and Mine Automation **40**(12), 37–41 (2014)