ATLANTIS PRESS

Advances in Computer Science Research, volume 87

3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2019)

Research on Dynamic Adaptation Technology of Multi-protocol in the IOT Gateway

Guoli Yang[a], Zhiqing Wei[b], Shuming Jiang[c], Xiangyang Liu[d], Huisong Wan[e], Hu Li[f]

Research and Development Center Information Research Institute of Shandong Academy of Sciences Jinan, China

[a] 835475760@qq.com, [b] weizhq@sdas.org, [c] jsm@sdas.org, [d] liuxiangyang@sdas.org, [e] wanhs@sdas.org, [f] lihu.911@163.com

Abstract. We propose a multi-protocol dynamic adaptation gateway technology to solve the multi-source heterogeneity of data in the Internet of Things. There are diversity of access devices and heterogeneous data in the Internet of Things. The major manufacturers of the data format don't have a uniform standard, resulting in a variety of data sources. The structure and semantics of these data sources are different, resulting in the collection of heterogeneous data. This paper designs an Internet of Things gateway based on the dynamic adaptation technology of multi-protocol. Multi-protocol dynamic adaptation is studying for the diversity of access equipment, including dynamic load and start-up, sensor and location addition, local display and processing. In addition, this paper also studies the interaction with the server, web data interface services based on RESTful and the interface service for drive.

Keywords: IOT gateway, Information collection and processing, Multi-protocol dynamic adaptation, Web services, Multi-source heterogeneous.

1. Introduction

Internet of things refers to the connection of objects, also known as M2M, the sensor network, the wisdom of the earth, ubiquitous computing, pervasive computing and so on. It is widely used, including intelligent home, transportation, medical, power grid, logistics, city, industry and agriculture. The number of sensors deployed in the world is growing rapidly, and market research shows that the sensor has had a significant growth over the past decade and will have a more significant growth rate in the future [1].

The system has accumulated a great deal of data that needs to be processed. In order to tap the value of these native data, they must become comprehensible. However, there are a wide range of sensors, and their data types vary. Even sensors that collect the same type of data may be unable to obtain unified perceptual data due to differences in manufacturer, device type and protocol. Generally speaking, on the one hand, different business applications, such as smart home furnishing, medical, security, logistics and city environmental monitoring, produced a variety of data sources. For example, structured relational databases and object-oriented data sources, semi-structured html and xml, unstructured text. On the other hand, there is no uniform standard for each manufacturer's protocol, which leads to heterogeneous data collection. If we solve these heterogeneous problems, we can easily use the data in a variety of applications.

2. Related Work

There are structural and semantic differences of these data sources. First, a variety of applications have collected a wide variety of data. When a large amount of data, manual to deal with these different types of heterogeneous data is difficult to discover and use comprehensive data contains information. Second, the data in the Internet of Things has a strong field and engineering characteristics, these standard specific format data usually choose to streamline storage, it is difficult to re-use the data to meet the data user requirements. Finally, the information must be based on a comprehensive analysis and judgments, excluding data uncertainty caused by hardware acquisition, software or other aspects

of interference, resulting in real and reliable information. How to solve the data heterogeneity has become a research hot spot at present.

The federated database is an early integration method, and its drawback is that the query is slow, not suitable for frequent queries and fast-growing data. An integrated approach architecture based on an intermediary or wrapper integrates structured data sources, unstructured or semi-structured data sources at the same time. The disadvantage is that the public model can describe more data, maintain the connection, resulting in inefficient query, complex matching data rules. The data warehouse extract data from heterogeneous data sources, and load it into the query mode. The disadvantage is that the timeliness of data in the warehouse cannot be guaranteed. After the semi-structured xml appeared, it quickly became popular because it solved the problem of converting and representing data in different systems [2]. For example, to solve the exchange of heterogeneous databases or focus on solving the data exchange process of security issues, or to solve the problem of web data exchange. Aiming at the problem of complex data exchange standard in Internet of Things, the researchers have established a data exchange research platform based on xml and ontology. The disadvantage is that the volume is too large when the collected key data is saved as an xml document [3]. The method of resolving data in xml document format cannot be suitable for different types of sensors. The global query request is more complicated, and its query efficiency is reduced, when the structured data in the database is exported to the xml global model.

The virtual database first forms a global view of the heterogeneous data source, which then maps the user's request to each local sub-request, and finally converts the result to the format of the user's request. Based on the Request Processing Center QPC study, all users and data are connected to the data center, and then the request is converted into sub-requests to generate xml formatted data. The development of a middle ware in a network analysis environment can shield relational databases in different locations and access them through a Web service interface. Data producers and users write or query distributed message intermediate systems without taking into account the diversity of intelligent sensor layer sensor nodes and the control of heterogeneous sensor networks [4].

According to the request to find out the results of the local storage, according to the maintenance of the table to the local query and storage. Because the cached result set is stored as a concrete table, the space swapping time does not require that the views be dynamically constructed with resources, so the query responds will be faster. The disadvantage is that the fact table update will lead to concurrent operation of the system, cannot quickly refresh, and the data is not accurate, increasing the demand for disk resources.

Ontology describes the semantics of local data sources and analyzes and matches them through conflict resolution algorithms. Next, the integration of the heterogeneous data source in conjunction with the schema integration approach requires a ontology library and a rule base. The disadvantage is that the matching algorithm is complex and the matching algorithms are different.

Traditional OSGI architecture is used to design an Internet of Things gateway. Users can start, update and uninstall services in the sensor network through OSGI life cycle. Each type of sensor corresponds to a corresponding Agent, and the messages generated by these agents are fixed xml format definition. The agent communicates with the upper layer using http. Through the xml mechanism to shield the intelligent perception layer format differences. The gateway utilizes the advantages of the OSGI framework to enhance the scalability of the gateway system [5]. But it has not yet reached the function of using plug-and-play, automatic identification, location addition and deletion, sensor addition and deletion, historical data change, etc. Although the gateway uses xml mechanism to shield the underlying sensor, to achieve the data acquisition and integration problems, but the gateway design is poor scalability, does not support a variety of manufacturers of hot-swappable sensors, but also do not consider the perception of different data acquisition [6].

We propose a multi-protocol dynamic loader gateway technology to solve the multi-source heterogeneity of data. Using Android dynamic loading apk package technology to realize the installation, operation and uninstall of hardware equipment of IOT without the case of updating the whole program. First, we study the local acquisition and processing systems, including sensor work, data acquisition and processing. Second, we study the multi-protocol dynamic adaptation, and aim at

the diversity of access devices, including drive dynamic loading and starting, sensor and location addition, local display and processing. Third, we study the problem of data display of the internal architecture of Internet of Things based on RESTful interface [7].

3. Overall System Design

3.1 Research Content

This paper studies multi-protocol dynamic loader gateway from the following seven aspects: fusion, dynamic loading drive, getting information, data storage and server interaction, embedded Web server access and packet transmission. Fusion means that gateway must have multi-standard interworking access. Able to integrate modern sensor network technology and shield protocol differences. Dynamic loading drive aspect s that module and decompose the data parser in the Internet of things. Each module can be installed without updating the entire program, uninstall, update. So, it is easier to access and expand the underlying sensor.

Assume the collection of information functions, including smart cards, RFID tags, identification code, sensors and so on. The method that data upload to the gateway is divided into two kinds, automatic upload and polling upload. For each upload mode, the gateway must have a corresponding data analysis module. The uploaded data can be distinguished by the hardware interface. However, it is necessary to send the data request command function to the acquisition module for the polling upload.

Data storage and server interaction aspect consists of two parts. The first part is the data in the iot gateway. The sending module will get the data from the working sensor. Then the data is encapsulated in a certain format and transferred to the data storage server like data storage server in the agricultural data platform via Tcp. The other part is that the data storage server database receiving module resolve the data, and then store data in the local database. At the same time, the iot gateway can get the required driver module from the remote drive server through this module, and driver module can be updated in the cloud server. When the corresponding driver module is downloaded, the local gateway can perform a local module update operation according to the corresponding version.

Embedded Web server access means the data information in the gateway can be accessed through the mobile or pc client, including viewing the current working sensor and historical data, adding or removing sensing, video monitoring, and operating and controlling the gateway.

In general, the object gateway system will handle with the processing of two kinds of packets of uplink and downlink packets. The uplink packet is the data collected by the sensing layer device, and the downstream packet is the control command for the sensor and the intelligent switch. As the Internet of Things is an embedded device, the sensing layer accesses a large number of intelligent sensing devices. With the increase in the number of access devices and the diversification of protocol types, the storage and processing capabilities of the gateway need to be upgraded to meet the needs of the load. These non-scheduled upgrades will lead to a shorter service life of the gateway system, which will also affect stability of the entire system and increase the cost of maintenance.

3.2 System Architecture

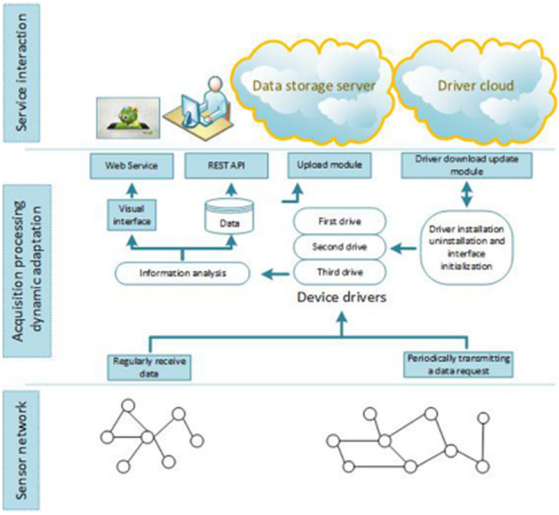

Fig. 1 Architecture

Fig.1 shows a detailed functional architecture diagram, which indicates the service interaction, acquisition processing dynamic adaptation and sensor networks. Acquisition processing dynamic adaptation includes multi-protocol dynamic adaptation loading module, visual operation interface module, device driver module, information analysis module, storage module, upload module and driver download update module.

The service interaction includes interaction with the data storage server, the driver server, the REST API, and the web service provided by the embedded web server. Data storage server section provides long-term and mass storage of information, while providing resource support for other applications. Further, the data storage server parses the received information and stores it in the database in a uniform format. Since each gateway can manually set the independent gateway ID, it is convenient to distinguish the source of the data. Driver server stores a variety of protocol drivers, providing download support. The driver server regularly updates the driver and maintains a list of driver version information. The client can download the required drivers by address and version number. Embedded Web server provides the ability to access and control the gateway through the Internet, including viewing the data stored on the local collection device, real-time monitoring, displaying sensor information, or performing operations related to the local acquisition system. Multi-protocol dynamic adaptation load module support download driver from the driver server through the manual way, dynamic search, load, uninstall, work and delete.

The sensor network includes a variety of types of sensors and smart switches from different manufacturers. Including smart cards, RFID tags, identification code, sensors, cameras and so on. Get the data from the sensor network through two ways, regularly receiving data and periodic transmission of data requests. The gateway must have a corresponding data receiving module to distinguish the data source through the hardware interface. However, the way of periodic transmission of data requests needs to achieve the function of sending the request data to the acquisition module.

Multi-protocol Dynamic Adaptation Loading Module: Acquisition processing dynamic adaptation layer supports multi-standard interworking access in addressing heterogeneous convergence of perceptual layers. It can fuse a variety of technical differences, including different data formats produced by the same protocol or different protocol. The resolution of the sensor in the gateway is modularized. Each module can be installed without updating the entire program, uninstall and update. This is a multi-protocol dynamic adaptation process.

In terms of control, the position graphical display interface and the sensor graphical interface are 1: N. Each sensor interface includes a number of sensor interfaces, each sensor interface contains real-time data information, such as air temperature and humidity, soil temperature and humidity, light intensity, switch status and other information. Users can configure, add or delete location interface,

and can dynamically configure, add and remove sensors in different location interface. This is the practicality of the gateway.

In the upload, the gateway will automatically upload the data server through Tcp periodically after the configuration of the ip and port parameters. This is the sharing of data. When the layer is configured with the corresponding data interface parameters, the user can also visually access the data stored in the gateway and control the sensor working state through the web. That is mobile accessibility.

Open Data Interface Function: The local gateway provides an open interface, which refers to two aspects. First, REST data interface, which can be accessed through the interface control information in the gateway, including the current work of the sensor and historical data, add or remove sensors, video surveillance. Second, the function interface, which is the main role of providing services to drive. So that when the driver needs the main program to do some work, the main program will provide this content to the need for the use of the driver, after the start of a program or loading a file.

In terms of accuracy and safety. Each gateway can be configured with a separate ID. The data server can distinguish multiple gateways according to the ID. Since each sensor also has a separate Sensor_ID, the corresponding sensor can be quickly found by location or Sensor_ID. At the same time, the gateway has intelligent data real-time display and alarm function.

When adding a new protocol sensor or other device, simply add a new resolution driver module. Therefore, the gateway supports a variety of manufacturers protocol driver. The main program does not need to develop many times.

3.3 Key Technology and Solution

We chose two kinds of sensor network devices based on ZigBee protocol and GPRS protocol. ZigBee is a short-range, low-power wireless networking technology based on IEEE802.15.4 standard low-power LAN protocol. It is characterized by close proximity, low complexity, self-organization, low power consumption and low data rate. It is mainly suitable for automatic and remote-control applications and can be embedded in various devices. GPRS packet switching communication technology is also known as 2.5G. Compared with the ZigBee protocol, its power consumption and cost are high, but its transmission rate is high, the distance is far. Device applications based on these two protocols are fairly common, but vendors typically develop specific network protocols for their own devices.

The platform uses Android system, with 3G features, USB interface and Consol interface. The Dalvik virtual machine on the Android system is the same as the Java virtual machine. First, you need to load the corresponding class into memory when the program is running. The Android system derives two classes from ClassLoader. DexClassLoader and PathClassLoader. DexClassLoader can load files on the file system jar, dex, apk and so on. PathClassLoader can load the "/data/app" directory under apk that has been installed.

Modular Loading Method: According to the function of the above modules, the gateway program is designed in a modular way, reducing the complexity of programming and making maintenance and operation easier. The local acquisition and processing system are divided into four parts: the main program part, the multi-protocol dynamic loading library part, the drive part and the interface part.

Main Program Part: The visual interface module is made a corresponding main program. The program solves the following problems. Database creation and operation, real-time interface display, historical interface, location addition and gateway configuration. The goal is to separate the analytic drivers associated with the IntelliSense layer into the public part of all drivers. For example, when you add a sensor, you can select an interface that is appropriate for its type from the interface library. This design minimizes the dependency between programs.

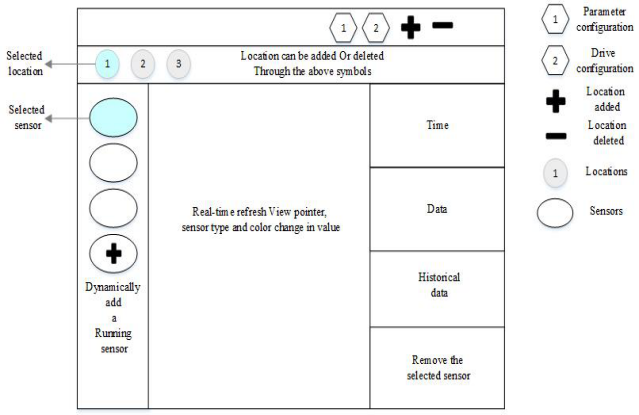

Fig. 2 Main program interface

Fig. 2 shows the main program interface and functional design diagram. The middle is real-time refresh interface pointer, its dial scale corresponding to different color changes. On the left, you can add a running sensor with the plus button. The sensor add box includes the node channel, type, sensor name, add time and confirmation cancel button. The node channel and type are stored when the sensor is working properly and can be selected in the drop-down box. There is a correspondence between the two parameters. Because when the node and the channel is determined, you can determine the kind of sensor by received data. There is a place to note here. The sensor corresponding to one of the node channels measures two or more types of data. When you add a type of sensor, there are additional types of sensors on this channel. After adding a certain type, this type is excluded from this channel, to prevent over-loading.

The right side is some of details of the sensor, including time, data, historical data and delete the sensor button. Historical data is also an interface. The data alarm section checks the legitimacy of the data in real time. Deleting a sensor means that it is not only necessary to delete the sensor interface in the current position and the sensor display icon that has been added, but also to delete the information recorded in the database. Deleting the sensor interface requires this container to be empty. Deleting an already added sensor display icon you need to delete the sensor record in the list of this location and then notify the adapter that the icon of the sensor has been deleted.

The second row shows the location that has been added, and the top line contains function buttons such as gateway configuration, add, delete, download, and close. After adding the location, the location information interface regenerates a corresponding interface container, which can also dynamically add the sensor that is working and remove the sensor that is no longer needed.

Interface switching is divided into three cases. The first case is for the interface switch between different locations. The second case is for the different sensor interface in the same location switch. The third switch is about the sensor interface and C historical data interface. First of all, the idea of the first case is to generate a corresponding interface for each additional position. The location and the corresponding interface are stored in the form of key-value pairs. First, set page indicator to indicate location of the page switch. This type of GuidePageChangeListener implements OnPageChangeListener interface. The value added in the interface is 0, 1, 2. Interface class LocationFragment class is inherited Fragment. It implements the OnItemClickListener interface. When adding a location, just type the instance once. Switching adapter MyPageAdapter class is inherited from the FragmentStatePagerAdapter class. The main function is storage location, interface and save status.

Second, for the second and third cases, the historical data and the interface corresponding to each sensor contain a large amount of data. If the design of the fast cross-switching process is not appropriate, it will be very difficult to achieve the desired effect. Location interface, sensor interface and historical interface, treat them as the same Fragment class. Differences are only nested between different Fragments. They are all displayed in the same content container. Although the historical interface is started on the sensor interface, this does not affect the switching between the three interfaces.

1. Main Program Part

Implement a common interface that is loaded to implement the interface in the main program. When you need to call the main program resource, the driver implements this functionality through the interface. The purpose of providing the interface part is that the driver may need the main program to load the external library functions and provide some other functionality. This is a separate interaction between the main program and the drive.

Interface part has two main classes HostInterfaceManager and HostInterface. HostInterface interface class implements the usb port and baudrate method and HostInterfaceManager class implements the functionality to get and set the interface. The function of HostInterface is that the driver can set the baud rate of the port on the hardware device. As long as the interface HostInterface to be achieve in the main program, the driver can call this function. Main program provides the function of loading the interface by implementing HostInterfaceManager. setHostInterface (new hardwareImp ()). Driver implements JNI's local call and sets the interface parameters by using the following method.

HostInterfaceManager.getHostInterface().OpenSerialPort(device.getAbsolutePath(),baudrate).

2. Multi - protocol Dynamic Adaptation Part.

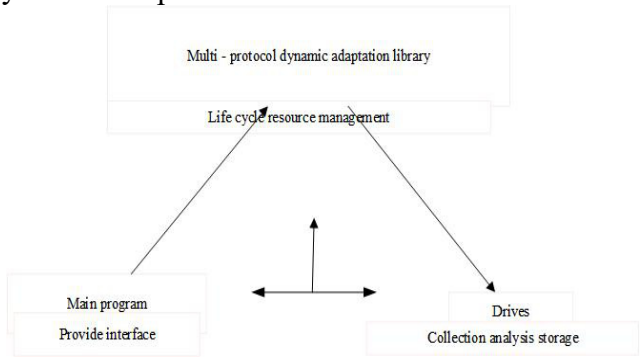

Fig. 3 Working mode

Fig. 3 Shows working mode of the three parts. Research questions include library loading time selection, startup or shutdown timing, and update drivers. Involving the following four directions. Driver and the main program work mode, drive the life cycle management, drive the required resource calls, the drive requires the main program to operate the interface call.

Driver information class QUPackage member variable information. Its role is to save the package name and path read out from the loaded driver package, and select the class loader and the resource manager. The following are adaptation principles and workflows. First of all, according to the driver package and class information, the user can put it in the list or can also use the option to start the driver. Second, the main program will load the class file into the virtual machine through the reflection method, and then select the drive to start. Finally, build the desired class name, package name, and context information, and start the thread in the driver by dynamically loading the method in the library.

```
DexClassLoader classLoader = quPackage.classLoader;
String className = dlIntent.getPluginClass();
className = (className == null ? quPackage.getDefaultActivity() : className);
className = packageName + className…..;
```

The multi-protocol dynamic loading library implements the lifecycle management of the driver module. Most of the lifecycle methods of Activity are extracted as an interface to DLPlugin. And then the lifecycle approach is implemented through the agent Activity DLProxyActivity calling the plug-in Activity. This completes the plug-in Activity lifecycle management, and does not use the reflection method.

Create the class loader and load the driver module into the virtual machine. Using DexClassLoader loader to load the file system on the jar, dex, apk and other files, do not need the program has been installed to the device. As shown in the following function, dexPath refers to the path of the APK or Jar file that needs to be loaded. DexOutputPath refers to the optimized dex file storage directory, and mContext.getClassLoader () calls the parent loader of the class loader.

3. The drive sections

It is designed for device and gateway interfaces in the architecture. The main function is to achieve data collection and processing, and store information to the gateway database. The driver part is oriented to the intelligent sensing device, so the driver part is divided into multiple independent packages according to the intelligent sensing device difference.

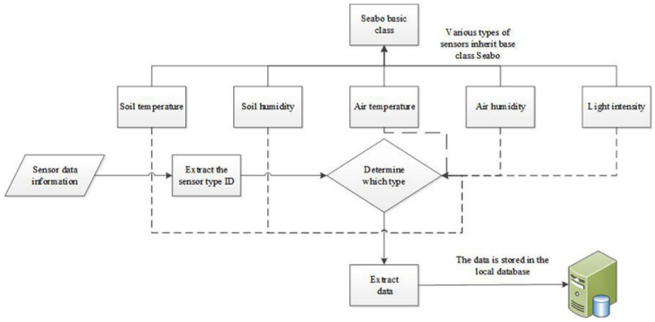

Fig. 4 Regularly receiving data

Fig. 4 shows that the program is about the design process of regularly receiving data. The selected automatic upload driver is based on the GPRS protocol of the sensor device. The device has a separate message for each access sensor, and each sensor device transmits the collected message to the upper layer device at every fixed time. The main program looks for the load driver and starts the default Activity. The role of this activity is used to start the thread. When the relevant thread is started, close the Activity interface and do not display this interface. After the configuration parameter thread is started, you can start the program that reads data from the USB port. Here you need to start a thread set port, baud rate and local address. Then the received data will be distinguished and the format definition will be resolved. Finally, the data is stored in the local database.

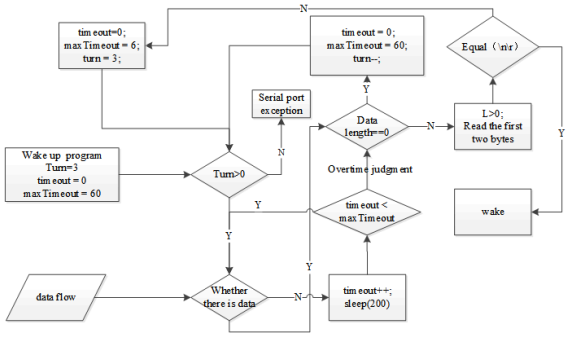

Fig. 5 Periodic transmission of data requests

Fig. 5 shows that the program is about the design process of the periodic transmission of data requests. Periodic transmission of data requests Equipment is based on ZigBee protocol weather station sensor equipment. The basic working principle is that after receiving the wake-up message '\ n', devices will enter the wake-up state and reply to the '\ n \ r' command. The program receives the uploaded 99byte LOOP package by sending the 'LOOP num' request command. The initialization is slightly different from the above. On one hand, in addition to setting port and baud rate parameters, the normal work of the device needs to load the external so library. On the other hand, only the driver sends a data query request, it can receive the information collected by the sensor.

Fig. 5 is the program wakeup algorithm. When the driver is initialized, it enters the wake-up process of the program. The overall idea of the wake-up process is that the sensor receives the command sent by the driver and confirms it. However, other situations must be considered, such as hot swapping of sensors, overtime connections, and data errors. When the wakeup program starts, initialize 3 variables. The loop number variable Turn is 3, the timeout variable timeout is 0 and the maxTimeout variable is 60. Then the program goes into the loop of Turn. If Turn is not greater than 0, the serial port is abnormal. When Turn is greater than 0, check if there is data. If there is no data,

start the timeout timer and make it automatically increment. After each increase and dormancy, compare the timer and the maximum time-out to determine if the timeout has occurred. The next data is judged. If there is data or has timed out, have to enter the data length of the judge. When the data length is 0, the Turn parameter decrements and sets the timeout timer to zero at the same time. When the data length is greater than 2, the first two bytes are intercepted to verify whether the message is a confirmation message. Finally, start the process of data processing, or reduce the maximum time-out, and re-enter the Turn of the cycle.

4. Web Services

System designs a REST data interface, which can be accessed through the interface control information in the gateway, including the current work of the sensor and historical data, add or remove sensors, video surveillance. That means every resource can be addressed by URI and can directly get, create, modify, and delete resources. Get is equivalent to read-only operation. Post is equivalent to read and write operations, including create, update, delete and so on. so, through GET and POST method can achieve the following table functions.

Embedded browser work mode is divided into three main levels, the client browser, Web server and database part. A client browser refers to an intelligent mobile device or desktop computer that a user can operate with a browser. The user obtains the gateway information and controls the gateway through the browser. When the user accesses the URL through the browser, the browser generates a data request message. It uses jQuery Ajax technology to implement CRUD request. The browser forwards the user request to the server. After receiving the user's request message of HttpServlet instructions, the server uses the servlet to parse the user JSON request. Then the Web browser sends a command for the data request to the database. SQLite database calls CRUD manipulation statements to complete the operation on the data. Finally, the success of the command or data will be returned to the browser.

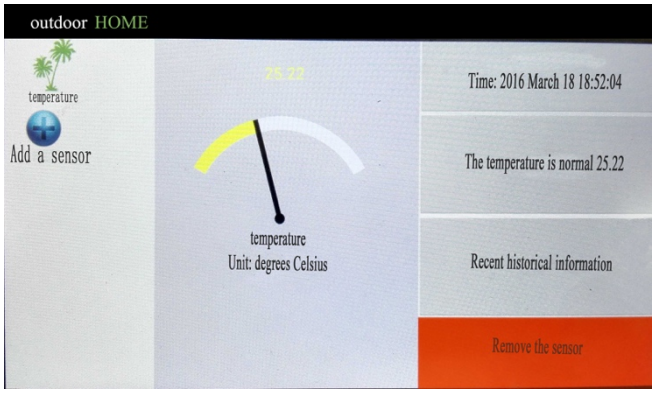

Fig. 6 Real-time acquisition and historical data

Fig. 6 shows the real-time acquisition and historical data of the on-site test of precision agricultural networking base. The scenario includes several parts, the Internet of Things platform, the service layer of the interactive cloud server, the agricultural information technology platform data server, the Web emded server and the smart sensor layer contains two kinds of factory sensors. The collection equipment includes air temperature and humidity sensors, soil temperature and humidity sensors, light intensity sensors and weather stations of air temperature and humidity sensors and soil temperature and humidity sensors.

4. Conclusion

This paper designs an Internet of Things gateway based on the dynamic adaptation technology of multi-protocol. Solved the multi-source heterogeneity of data in the Internet of Things. Multi-protocol dynamic adaptation is studying for the diversity of access equipment, including dynamic load and start-up, sensor and location addition, local display and processing. In addition, this paper also studies the interaction with the server, web data interface services based on RESTful and the interface service

for drive. This paper solves the problem of high cost, complicated maintenance, short life of the gateway, poor system stability and poor scalability in the upgrade process.

References

[1]. Sarma Sanjay, Brock David L, etc. MIT whitepaper: The Networked Physical World, Proposals for Engineering the Next Generation of Computing, Commerce & Automatic-Identification[J]. Mit Press. 2000.

[2]. Wang Hongrong, Wu Baoguo. Design and implementation of a dataexchange tool among heterogeneous databases[J]. Journal of BeijingForestry University, VoU, Nov.2009, pp:102-106.

[3]. GUAN Jian. Research on the data format standard of Internet of Things based on XML and ontology[J]. Internet of Things Technologies. 2012.

[4]. WENG Zu-quan, GAO Zhe. Design of data processing gateway for Internet of Things. Internet of Things Technologies. 1 March 2013.

[5]. Martin Stusek, Jiri Hosek. Performance analysis of the OSGi-based IoT frameworks on restricted devices as enablers for connected-home. International conference on ultra modern telecommunications. 2015.

[6]. Giuseppe Di Modica, Francesco Pantano, Orazio Tomarchio. An OSGi middleware to enable the Sensor as a Service paradigm. Scalable Computing:Practice and Experience. 2014.

[7]. Li Li, Wu Chou. Design and Describe REST API without Violating REST: A Petri Net Based Approach[J], Web Services (ICWS), IEEE International Conference on Web Services-ICWS, 2011.