Transformer Equipment Temperature Monitoring Based on the Network Framework of Django

Chunsheng Song
College of Electrical Engineering and Automation
Shandong University of Science and Technology
Qingdao,China
13287203460@163.com

Ruiping Huo
College of Electrical Engineering and Automation
Shandong University of Science and Technology
Qingdao,China
806420880@qq.com

Shuqing Wang
College of Electrical Engineering and Automation
Shandong University of Science and Technology
Qingdao,China
447866690@qq.com

Changzhi Lv
College of Electrical Engineering and Automation
Shandong University of Science and Technology
Qingdao,China
lvchangzhi@126. com

Abstract—**In order to improve the visualization and efficiency of transformer temperature monitoring, a real-time transformer temperature detection system based on Django network framework is designed, in which the temperature acquisition module is composed of temperature sensor and zigbee network module and is controlled by Raspberry Pi microcomputer. The server and the monitoring system are partly based on the B / S architecture design, and the python language is used to collect, store, send and display the data. Users through the web browser to view the data displayed in the form of charts to facilitate data analysis and warning of transformer equipment.**

Keywords—Temperature monitoring; Raspberry Pi; Zigbee; python; Django; Highcharts

I. INTRODUCTION

Temperature is an important indicator to determine whether the transformer electrical equipment is normal. Many equipment (transformer cabinets, high and low voltage cabinets, etc.) have certain latency and development time. Regular maintenance and inspection sometimes fail to detect sudden faults in time. In addition, the current transformer monitoring system is based on the C/S (Client/Server) architecture. It requires the installation of the client to operate. The compatibility is poor. If the client has problems, the client must be upgraded to solve the problem, trouble, affecting the daily monitoring of the transformer [1]. Aiming at the shortcomings of the traditional transformer temperature monitoring system, a transformer temperature monitoring system based on Django network framework is proposed. The server of the system uses the Raspberry Pi as the carrier, and the user enters the assigned address through the web browser on the client. Specify a monitoring chart for the device to visually analyze the temperature data of the device over a period of time. No need to install the client, it can be compatible with all kinds of operating systems that are common in the market, and it is easy to maintain and transplant.

II. SYSTEM OVERALL ARCHITECTURE DESIGN

The whole system is modular in design, including client, server and temperature acquisition module.

A. Client

The client is a web-side online monitoring system developed by the Django network framework based on the B/S architecture. The Django network framework follows the model-view-controller (MVC) control mode, which defines the code and the method of data access (model) separating from the request logic (controller) and the user interface (view) so that each component can be designed separately without affecting other components [2].

The model section is used to describe the data tables that we have collected and interacted with the database in the server for data. The view part is the interface that the user sees and interacts with, displays the collected temperature data to the user, and receives the user's input data [3]. At the same time, the view can also accept data update events from the model, so that our temperature monitoring data can be displayed in the user interface in real time. The controller is responsible for logic processing, temperature, time, equipment and other data displayed on the view, calling the model to process the business request.

B. Server

The server carrier is played by the microcomputer Raspberry Pi. The Raspberry Pi is equipped with an ARM11 core, with a 1.2Ghz frequency, strong performance, and excellent multi-tasking and multi-threaded tasks. The system uses the Raspbain system based on the official Linux kernel of the Raspberry Pi. The Raspberry Pi GPIO controling package with built-in Python language is convenient for development [4].

In data interaction, the Raspberry Pi is responsible for obtaining the data obtained from the data collection module and storing the data in the local Mysql database. At the same time, the Raspberry Pi is also the carrier of the Django web server, and the Django web server is the bridge between the database and the host computer.

C. Temperature acquisition module

The data acquisition module consists of a Zigbee terminal node and a Zigbee coordinator [5]. Zigbee terminal node is composed of temperature sensor and cc2530. The temperature sensor is made of DS18B20 produced by semiconductor company DALLAS. This temperature sensor has the characteristics of high precision and strong antiinterference ability, and can adapt to various non-extreme occasions. Zigbee Coordinator also uses TI's CC2591 as the RF front-end chip, CC2530 as the main control chip, Zigbee Coordinator establishes the entire Zigbee network, and each terminal node can access the network to realize data interaction [6]. The temperature monitoring system architecture is shown in Figure 1.

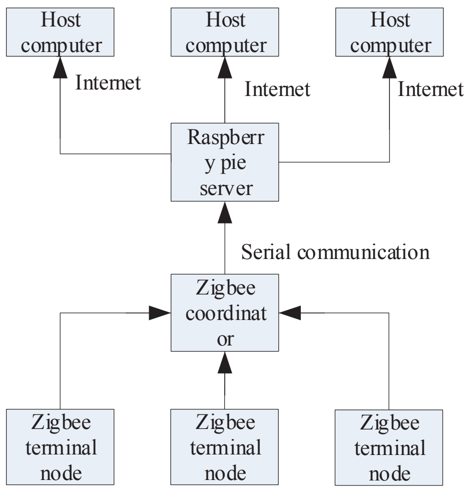


Fig. 1. Temperature monitoring system architecture diagram

III. Monitoring Module, Server and Host Computer Communication

A. *Temperature acquisition module and server communication*

The cc2530 in the terminal node can drive data to interact with the temperature sensor by driving the DS18B20 program on the protocol stack. The Zigbee coordinator in the temperature monitoring module is responsible for establishing the Zigbee network. When the coordinator is powered on and the network is established, the terminal node applies to join the network, and the coordinator will assign an address to each terminal. When the coordinator receives the data collected by the temperature sensor of the Zigbee terminal node, the data is packaged and sent to the Raspberry Pi server through the serial port communication for the next processing.

B. *Raspberry Pi server communicates with the host computer*

Hypertext Transfer Protocol (HTTP) encapsulates the communication flow of the entire web page. It consists of two parts: the request and the response. The core of the request is the URL, which allows a single address or URL to display multiple behaviors through parameterization of a series of methods. The response consists primarily of plain text, documents, or sound clips of the required data. Requests and responses can be represented as Python objects within the Django framework, with properties representing their data and can be called in the Highcharts integration library.

When the monitoring interface of the host computer sends a request for receiving data, Django uses a regular expression to match the URL mapping file corresponding to the corresponding time, device, etc. When the mapping file matches the URL-pattern, Django sends the request to the views function. After receiving the request, the database is accessed, the data is extracted and loaded onto the template page, and finally returned to the user's web browser, and the user can see the collected device temperature and other information [7].

IV. Implementation of Online Monitoring System

A. *Raspberry Pi server basic configuration*

Before the Raspberry Pi assumes the server function, it needs to perform some basic configuration on the software and hardware to facilitate the development, debugging and maintenance of the system. After the Raspberry Pi is inserted into the SD card of the Linux system, it can be booted after power-on. Since the Raspberry Pi has an onboard Wifi module, it can connect directly to the wireless network [8]. Use the apt-get command on the network to download the cron automatic task timing tool and Python third-party library modules such as Django as necessary components for system development.

In the hardware part, since the Raspberry Pi does not have an onboard RTC, in order to solve the problem that the time data corresponding to the device temperature in the system without the network is abnormal, we need to add an RTC module in Figure 2. In terms of reset, we have prepared two reset modes. The first one is to write a watchdog script that runs automatically at boot, and realizes soft start by monitoring the I/O status. In addition, the Raspberry Pi 3B+ version has a hardware reset interface reserved, and the access button can be used.

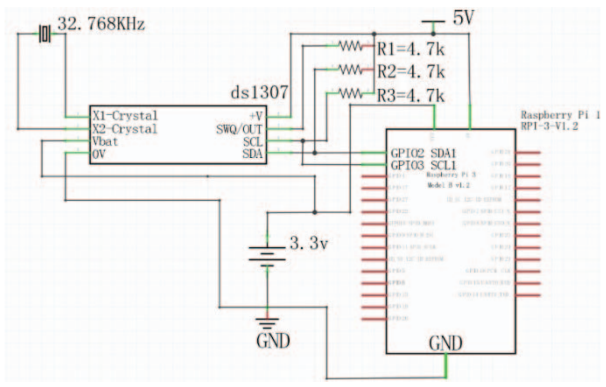


Fig. 2. RTC clock circuit diagram

B. *Raspberry Pi Python code optimization design*

The Raspberry Pi server needs to switch the priority of running programs in multiple modes. For example, when the host computer issues a request command, the server needs to respond to the request of the host computer user in addition to listening to the Raspberry Pi running status, acquiring the real-time data of the temperature sensor, and storing the data to the database, so the server needs to have a multi-thread running program ability.

Although the Raspberry Pi has a 4-core CPU, multithreading cannot execute code parallel on multiple cores because the Python language is limited by the Global Interpreter Lock (GIL). However, we can use the thread method in python's thread library to define the task class we need to execute as a thread object waiting to be called. Although GIL can't make Python code parallel, it will release

GIL when executing system calls, which will significantly reduce CPU processing time.

In addition, because we need to cycle through the data sent by different temperature monitoring modules, but if more advanced threads are suddenly inserted, staggered execution will cause data competition problems. So we use the Lock class in the threading module to protect the counting object.

C. Database design

The database uses the relational database Mysql. Before establishing the Django project, you need to establish the Databases and Tables for storing data. The id (serial number) is the primary key in the table, in addition to the device name, time and equipment data, etc. in Table I [9]. In order to solve the problem that Django project shows garbled characters in the view when calling JSON data, we set the data type of the corresponding temperature data in Mysql database to BLOB to ensure that the data is parsed into a dictionary format that Python can recognize.

TABLE I. DATABASE PART FILTER DATA DIAGRAM

id	device	time	temp
1	transA_temp	2017-10-11 09:00:00	44.1
2	transA_temp	2017-10-11 09:10:00	43.6
3	transA_temp	2017-10-11 09:20:00	44.3
4	transA_temp	2017-10-11 09:30:00	44.6
5	transA_temp	2017-10-11 09:40:00	44.2
6	transA_temp	2017-10-11 09:50:00	45.5

D. Server design based on Django-web framework

The closest thing to the user is the HTTP communication protocol. Through the URL, the user can send a request to the Django web application. While the client responds, the user can also use the partial refresh technology to use JavaScript to load the data asynchronously. At the other end, the model and the Django ORM manage the database layer, communicating with the database through the Python interface.

Both are based on the MVC framework. After the web server receives an HTTP request and forwards it to Django, the Django engineering system framework diagram is shown in Figure 3. Django receives it at the requesting middleware layer, and then assigns it to the appropriate view according to the URLconf pattern matching. The view performs is the core of the required work, and generates a response as needed with the model or template. The response is returned to the web server through the middle tier and forwarded to the user.

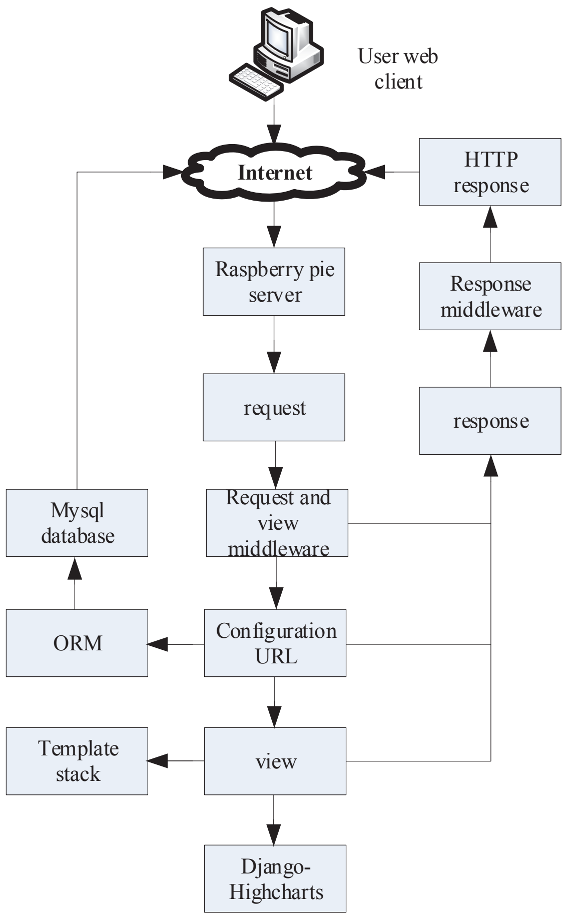


Fig. 3. Django engineering system framework diagram

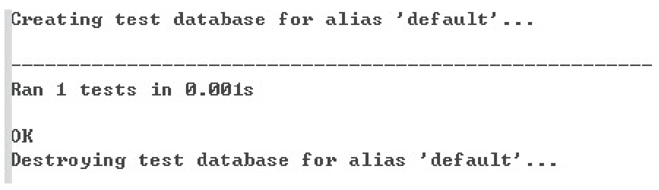
In order to make the operation and maintenance of transformer equipment more convenient, use Django-Highcharts integrated library to make our data more intuitive [10]. The html file under the Temgplates template is the page that the user finally accesses. Considering the real-time display of monitoring data, avoiding the problem of reloading the entire page, integrating local refresh technology (ajax) to send data request parameters, database data is transmitted in JSON format. Then, combined with the views function, the data is rendered in html to realize data graphing [11].

V. MONITORING SYSTEM TESTING

A. Functional testing of the monitoring system

Simulating the transformer monitoring environment in the laboratory environment, powering on each module, waiting for the Raspberry Pi to start, the test communication is normal in the Raspberry Pi server, and the database data is normal.

Then you need to unit test the migration and storage of database data in Django project, user request and access, and data graphing. The test will temporarily generate a database and automatically destroy it after the test, without affecting the original data sheet. The test results show that the system can run various functions normally, and some tests are shown in Figure 4.



```
Creating test database for alias 'default'...

----------------------------------------------------------------------
Ran 1 tests in 0.001s

OK
Destroying test database for alias 'default'...
```

Fig. 4. Unit test chart

B. Monitoring system use test

After the python manage.py runserver command is run in the command prompt (cmd) in the Windows emulation environment, the system runs, and any requests and responses received by the system will be displayed here.

After the system runs, enter the address of the system and the system login interface, as shown in Figure 5.

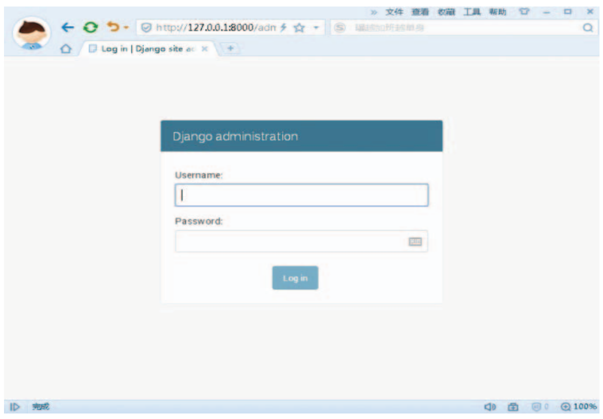
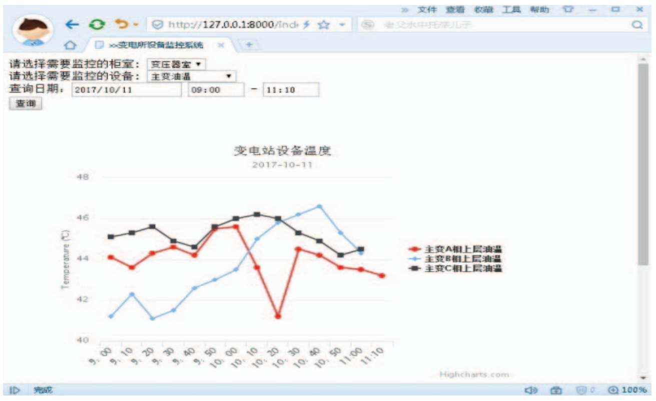


Fig. 5. System login interface

Type the account and password of the corresponding device. After logging in to the monitoring interface, you can select the monitored equipment room, indoor equipment, and historical data, as shown in Figure 6.



请选择需要监控的柜室: 变压器室
请选择需要监控的设备: 主变油温
查询日期: 2017/10/11 09:00 - 11:10
查询

Fig. 6. Monitoring system renderings

VI. CONCLUSION

Based on the Django network framework, this paper uses Zigbee technology to realize data communication of temperature acquisition module, and design a transformer equipment temperature monitoring system with Raspberry Pi as the server carrier. The client of this system adopts MVC design mode and has high cohesion. Sexuality and low coupling improve the portability of the system and reduce the difficulty of development and maintenance. At the same time, Highcharts chart integration library has the advantages of strong compatibility, powerful function and simple calling method, which greatly shortens the difficulty of web development. In addition, due to the low coupling of the system, the interactive interface and data transmission method have reference significance for data monitoring of other devices.

ACKNOWLEDGMENT

We are very grateful to the editors and anonymous reviewers for their insights and suggestions, which have greatly improved the quality of our papers. We are also very grateful to the anonymous reviewers for their valuable time for the peer review process.

REFERENCES

[1] Zhang Qiaofen. Design and implementation of wireless monitoring system for substation equipment temperature [D]. Dalian: Dalian Maritime University, 2011.

[2] Xue Yaowei. Design and implementation of automatic generation module based on Django framework management interface [D]. Harbin: Harbin Institute of Technology, 2014.

[3] Li Yanhui. Design and implementation of online evaluation system in Weifang Vocational College [D]. Jinan: Shandong University, 2010.

[4] Gu Yanhua, Wang Chuang. Design and implementation of HD video player based on Raspberry Pi[J].Science & Technology Vision,2015,(29):30+20.

[5] Yang Bingliang, Chen Yulong, Yao Yingwei. Design of infrared remote monitoring system based on ZigBee[J]. Electronic Design Engineering, 2014, 22(02): 115-117.

[6] Song Yan. ZigBee-based substation primary equipment condition monitoring system [D]. Huainan: Anhui University of Science and Technology, 2014.

[7] Ma Shizhen, Bai Lixin, Zhang Haichun, et al. Visualized seismic catalog service system based on Django framework[J]. Earthquake Disaster Prevention Technology, 2015, 10(03): 695-699.

[8] Kong Demin. Design and research of detection system for early warning aircraft based on Python [D]. Harbin: Harbin University of Science and Technology, 2017.

[9] Yao Tengqi. Research on real-time construction and management technology of high-speed ship automatic driving control simulation system [D]. Harbin: Harbin Engineering University, 2012.

[10] Zhong Wenguang, Wan Qiang, Xu Changfu. Application of Highcharts Chart Library in Web Development[J]. Science and Technology Plaza, 2017, (04): 55-58.

[11] Hao Jie. Application of ASP.NET AJAX Framework in Web Development[J]. Electronic Technology and Software Engineering, 2017, (17): 55.