Secure Gateway of Internet of Things Based on AppWeb and Secure Sockets Layer for Intelligent Granary Management System

Rong Tao, Senbin Yang, Wei Tan, Changqing Zhang

Secure Gateway of Internet of Things Based on AppWeb and Secure Sockets Layer for Intelligent Granary Management System

Rong Tao[1,a], Senbin Yang[1,b], Wei Tan[1,c], Changqing Zhang[1,d]

[1]Xi'an Communications Institute, Xi'an 710106, China
[a]firsttaorong@sohu.com, [b]ysb-007@163.com, [c]dzxxlab@163.com, [d]zhangcq1108@163.com

Abstract. To develop an intelligent granary management system (IGMS) based on Internet of Things (IOT), secure gateway of IOT (IOTGW), the key component in the system, is designed and achieved. According to the functional requirements of IOTGW in IGMS, embedded Web server is adopted as a lightweight approach for accessing perception devices and interacting with heterogeneous networks. The design scheme based on ARM chip and embedded Web server is given and the hardware and software architectures of IOTGW are investigated in detail. By comparing the major performances, AM3352, Linux, AppWeb, SQLite and C language are chosen for IOTGW implementation. In order to protect the sensitive information transmitted between the client and server, the secure sockets layer (SSL) protocol is added in AppWeb, and the compilation and transplantation of AppWeb with SSL are given detailedly. Experiment shows that the IOTGW can access different types of perception devices and actuators widely, exchange information between the perception layer and network layer safely, and control the perception nodes remotely, so the management of granary becomes convenient, efficient and intelligent.

Keywords: gateway, internet of things, intelligent granary management system, AppWeb, secure sockets layer

1 Introduction

Granary is a storehouse or room in a barn for threshed grain, and grain reserve has become a key issue concerning about people's livelihood and national defense. In order to reduce unnecessary losses caused by animals, floods, fires, etc in the process of grain storage, granary monitoring system is applied subsequently. The original method is manual inspection by sampling, which is time-consuming, inaccurate and inefficient. The second generation technique is electronic detection using computer and bus standard, that is, electronic monitoring equipments such as temperature and humidity sensors are installed to collect the temperature and humidity data of granary, and bus standard, take RS485, RS422 or fieldbus for example, is adopted to transmit these data to the monitoring computer [1]. This technique improves the accuracy of monitoring data and reduce the costs of granary management, however, there are some disadvantages including low transmission rate, weak system reliability, difficult for remote communication and big workload of wiring. With the rapid development of communication and network technology, the third generation monitoring system is

featured by networking, in other words, short-range wireless communication technologies such as Wireless Sensor Network (WSN) are employed to connect monitoring equipments and collect data, meanwhile, long-range wired and wireless communication technologies such as Internet and mobile communication system are used to convey sensor data and control commands remotely [2]. In this way, the management level of automation and information is improved greatly. Nevertheless, the system's scalability and intelligence are not good enough for it's not easy to increase the number or type of monitoring nodes, or to obtain, fuse and process monitoring information in real time and actively. To address these issues, Internet of Things (IOT) is introduced to granary monitoring as a new technique [3].

Internet of Things is a dynamic global network infrastructure with self configuring capabilities based on standard and interoperable communication protocols where physical and virtual "things" have identities, physical attributes, virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network. IOT is characterized by more thorough sense and measure, more comprehensive interoperability and intelligence, whose application areas cover modern agriculture, infrastructure construction, public security, environment protection, intelligent industry, urban management and other fields [4].

Restricts of time, space, region and human to real-time monitoring of granary can be broken through by using the technology of IOT, and the IOT-based granary monitoring system can collect, memorize and transmit the parameters including food temperature, humidity, gas concentration, insect and water content to the control center as well as execute the orders from the control center [5]. Baisen Xu et al. analyzed the information function model and architecture reference model of IOT-based grain monitoring system, which is composed of four layers: perception layer, transport layer, processing layer and application layer [3]. However, they only discussed the related key technical problems theoretically, the methods and steps of system implementation were not given. [6] proposed a wireless LAN monitoring system adopting intelligent granary monitoring, which is consisted of a central monitoring station, a wireless relay station and wireless collection points. The multi-function real time data is collected by collection points made by ARM11 embedded processor, and then they were sent to the wireless relay station by ZigBee wireless module. The wireless relay station was responsible for monitoring and preserving the information from the collection points, and sending the comprehensive information to the central monitoring station by WiFi module. The received data was processed by the central monitoring station to realize the monitoring of the real-time information of granary working status. However, the wireless relay station has many deficiencies. Firstly, it can only access these collection points supporting ZigBee protocol, and the collecting information is confined to temperature, humidity, oxygen and pest pictures; Secondly, it can't manage or control the collection equipments; Thirdly, it communicates with the central monitoring station only by WiFi, which has small network coverage, high cost and susceptible radio channel; Finally, its security is poor, for it lacks authentication, encryption and other information security measures. Furthermore, many granary monitoring systems based on IOT given in other literatures exist above defects more or less.

In order to maximize the advantages of IOT, achieve broad access and management of perception devices and secure information transmission, gateway of

Internet of Things (IOTGW) should be exploited to replace the wireless relay station. Therefore, this paper focuses on the design and implementation of secure IOTGW for the Intelligent Granary Management System (IGMS).

2 Intelligent Granary Management System Based on Internet of Things

2.1 IOT Architecture of Intelligent Granary Management System

The requirements of intelligent granary management system are as follows: First of all, the environment of grain storage should be monitored dynamically and adjusted automatically, i.e. temperature, humidity, gas conditions of granary and the extent of food mildew can be sensed in real time and storage conditions will be adjusted by automatic adjustment system accordingly; In the next place, dynamic perception of quantity and location of grain can be achieved for efficient inventory management; Last but not least, personnel access and pest invasions should also be apperceived to ensure the security of granary.

According to the above requirements, the IOT-based architecture of IGMS is shown in Fig. 1, which can be divided into three layers: perception layer, network layer and application layer.

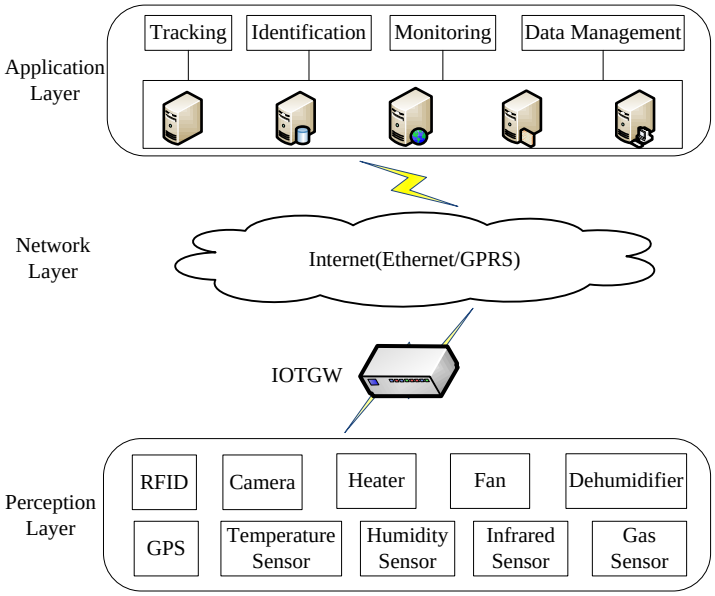

Fig. 1. IOT architecture of IGMS

The perception layer aims to acquire, collect and process the data from perception devices including RFID label, GPS, camera, temperature sensor, humidity sensor, infrared sensor and gas sensor, and control the actuators such as heater, fan and dehumidifier. The major communication protocols between these equipment and

IOTGW are ZigBee, Bluetooth, WiFi and USB. The network layer aims to transfer the data collected from perception layer and commands issued by application layer in a large area or long distance, which is constructed based on the Internet (via Ethernet and GPRS). Data processing and services providing are two major purposes of the application layer. The data from network layer is handled by corresponding management systems and then various services including tracking, identification, monitoring and data management will be provided to granary manager.

2.2 Analysis of IOTGW in Intelligent Granary Management System

As shown in Fig. 1, it's difficult to connect different devices in perception layer and communicate with Internet, for it lacks of uniform standardization in communication protocols and sensing technologies [7]. Therefore, IOTGW is introduced to carry out data communication between the two kinds of networks by coping with both the protocol transformation and the device heterogeneity throughout IOT network. In other words, IOTGW is the bridge to connect the perception layer with network layer, which has the following functional requirements: 1) Broad access. The perception devices and actuators should be integrated through IOTGW, namely, the IOTGW can access devices with different protocols including ZigBee, Bluetooth, WiFi and USB. 2) Protocol conversion. The core protocol of Internet is HTTP/IP, which is different from the protocols of perception layer in many aspects such as data formats, data rate and data meaning. In order to exchange information between these two layers, the IOTGW should fuse heterogeneous networks and support protocol interworking seamlessly including conversion of data messages, events, commands and time synchronization. 3) Manageable capability. The management of IOTGW contains perception nodes management and gateway device management. The former aims to acquire the node's identification, status and properties, and to realize remote startup, shutdown, control and diagnosis, the latter aims to realize the gateway device's configuration, supervision, upgrade and maintenance. 4) Information security. The information transmitted in the network may affect the granary security, and they are transmitted by wireless channel, therefore, safety precautions should be taken to ensure information security.

3 Design of IOTGW Based on ARM and Embedded Web Server

3.1 Design Elements and Principles

There are already some researches on the design and implementation of IOTGW system. Qian Zhu et al. proposed an IOTGW system based on Zigbee and GPRS protocols according to the typical IOT application scenarios and requirements from telecom operators, and the implementation of prototyping system based on ARM9 and Python were given [8]. Till Riedel et al. used Web service based interface descriptions paired with a model driven approach to achieve a high flexibility at a low runtime overhead when designing message based communication within an Internet

of Things. The experiments with an industrial servicing use case showed that Web Services and standard HTTP communication based gateways for sensor nodes can integrate multiple concurrent IOT systems [9]. In a word, IOTGW plays a leading role in the IOT systems, and the technologies of ARM and Web service are helpful for its development.

An embedded Web server is a component of a software system that implements the HTTP protocol. The embedded Web server technology is the combination of embedded device and Internet technology, which provides a flexible remote device monitoring and management function based on Internet browser and it has become an advanced development trend of embedded technology. Through this embedded Web server user can access their equipments remotely [10]. There are some advantages to using embedded Web server to design IOTGW: 1) HTTP is a well studied cross-platform protocol and there are mature implementations freely available; 2) Web browsers are readily available with all computers and mobile phones, and other utility software are needless in the application layer; 3) With the usage of Web service, interacting with a perception device becomes as easy as typing a URI in a Web browser. Consequently, embedded Web server can be adopted as a lightweight approach for accessing perception devices and interacting with heterogeneous networks. Moreover, the general problem of using embedded Web services is that even light-weight implementations are too resource heavy for many IOT systems.

The IOTGW is designed with three main principles in mind: simplicity, extensibility and modularity. Simplicity and extensibility refer to users can extend and customize the IOTGW to their needs. Modularity means that internal components of the IOTGW can interact only through small interfaces, thus allowing the evolution and exchange of individual parts of the system.

3.2 Architecture of Hardware and Software

According to above elements and principles, hardware architecture of IOTGW system is shown in Fig. 2.

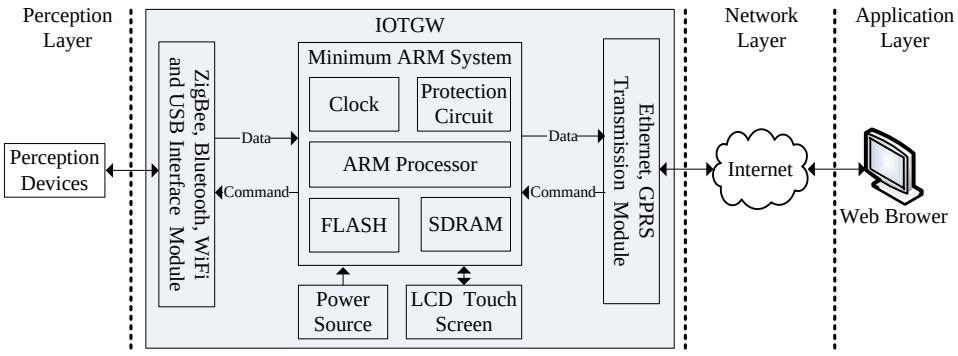

Fig. 2. Hardware architecture of IOTGW system

As shown in Fig. 2, the hardware architecture is composed of five major modules: minimum ARM system, interface module, transmission module, LCD touch screen and power source, each responsible for a well defined set of tasks. Minimum ARM

system is the kernel of the IOTGW, which provides the utilities of processing, control and storage. Embedded Web server with TCP/IP protocol suite is built in the ARM processor, and some web application programs, take protocol conversion, message parsing and device management for example, are written in ARM. The interface module makes the IOTGW accessible to different perception devices. With the help of software, these devices can be controlled and managed by appointing unique IP address, and their heterogeneity is shielded. The Ethernet and GPRS transmission module realizes the data transmission based on Internet by the means of wired and wireless respectively. The heterogeneity between these two protocols and perception devices is shielded too, so data and command can be exchanged and understood between perception layer and network layer. The LCD touch screen is used for output and input of data, command and status, which are useful during equipment debugging and testing. After the IOTGW is in proper working order, the LCD touch screen can be removed. The power source supplies direct current (DC) for all IOTGW components, such as 3.3V, ±5 V, and the source type contains electric supply and battery.

In order to implement the IOTGW's functionalities, the software is designed as Fig. 3.

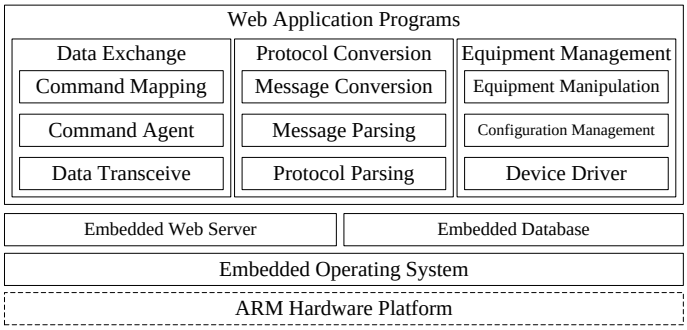

Fig. 3. Software architecture of IOTGW system

As shown in Fig. 3, Embedded Operating System (EOS) is installed on the hardware platform firstly, and then three functional blocks including data exchange, protocol conversion and equipment management are programming under the EOS.

4 Implementation of Secure IOTGW Based On AppWeb with Secure Sockets Layer

4.1 Selection of System Platform and Tools

Selection of ARM chip. The AM3352 microprocessor, based on the ARM Cortex-A8, is the product of Texas Instruments. The device supports Linux and Android operating systems and has excellent features including enhanced microprocessor unit, additional peripheral interfaces and real-time clock [11]. Consequently, we choose AM3352 as the hardware platform, corresponding, Linux, which is leading, free and

open source EOS, is adopted for hardware resources management and program running control.

Selection of embedded Web server. The common open source embedded Web servers which support Linux EOS are AppWeb, Goahead, Boa, Thttpd, Lighttpd, Shttpd and Mini_httpd. The comparisons of their major performances are shown in table 1.

Table 1. Performances comparisons of common embedded Web servers

Server	Benefits and features	Memory usage	Security and reliability
AppWeb	Over 4500 requests per second (r/s); Supports HTTP/1.0, HTTP/1.1, CGI/1.1, SSL RFC 2246, HTTP RFC 2617; Easy to configure and manage; Apache compatible configuration; Called as mini-Apache.	110 KB	Secure Socket Layer (SSL/TLS); Digest and basic authentication; Directory and URL location based authorization; Sandbox resource limits.
Goahead	Over 50 r/s; Supports HTTP/1.0, HTTP/1.1; Minimal security features.	60 KB	SSL/TLS; Basic, digest and form authentication; Security sandbox with DOS protection.
Boa	Over 726 r/s; Single-tasking HTTP server; Supports HTTP/1.0; No server side includes (SSI), access control lists (ACL) and chroot option.	95 KB	No SSL support.
Thttpd	Over 800 r/s; Supports HTTP/1.0, HTTP/1.1, IPv6; URL-traffic-based throttling; Suited to service high volume requests for static data.	150 KB	Basic authentication; Syslog is vulnerable to a buffer-overrun attack.
Lighttpd	Over 1000 r/s; Single-process design with only several threads; Doesn't support sending large files.	200 KB	SSL/TLS; Digest and basic authentication.
Shttpd	No configuration files and external dependencies; No SSI and ACL.	160 KB	SSL/TLS; Digest (MD5) authorization
Mini_httpd	Supports HTTP/1.0, HTTP/1.1, IPv6, CGI; Custom error pages; Suitable for low or medium traffic sites.	90 KB	SSL/TLS; Basic authentication.

In view of the requirements of IOTGW in the IGMS, such as high security, large requests, multi-business types and easy maintenance, the AppWeb is selected as the embedded Web server because of its extended security and higher performance [12].

Selection of embedded database. SQLite, which is contained in a small (about 350 KB) C programming library, is a software library that implements a self-contained, serverless, zero-configuration, cross-platform, transactional SQL database engine. It has some outstanding features, such as 1) Transactions are atomic, consistent, isolated, and durable (ACID) even after system crashes and power failures; 2) Faster than popular client/server database engines for most common operations; 3) Implements most of SQL92; 4) Several processes or threads may access the same database concurrently. So SQLite is employed accordingly.

Selection of development language. Common Gateway Interface (CGI) applications can be written in a number of different languages, such as C, C++, Java, PHP, Python and PERL. C is a structured, procedural programming language that has

been widely used both for operating systems and applications and that has been standardized as part of the Portable Operating System Interface (POSIX). C is employed to write the CGI scripts for the following reasons: 1) CGI programs written in C are stable and secure for the program is compiled and unmodifiable; 2) Linux is written in C with built-in C compiler, and it's a waste of resource to install other languages' compiler; 3) The scripts generated by C are relatively small and have faster execution speed; 4) The logic interactions of the hardware must use the C language;.

In a word, the system platform and tools for IOTGW implementation are AM3352 + Linux + AppWeb + SQLite + C language.

The development of IOTGW system includes the realization of hardware and software, and the specific contents are as shown in Fig. 2 and Fig. 3 respectively. Due to space constraints, the following focuses on the core part of the secure IOTGW development—compilation and transplantation of AppWeb with Secure Sockets Layer (SSL).

4.2 Compilation and Transplantation of AppWeb with SSL

In the IGMS, the communication between the server and the client contains the device status, user data, control commands and other sensitive information, so it's necessary to configure the secure transmission system in the AppWeb. The Secure Sockets Layer is a commonly-used protocol for managing the security of a message transmission on the Internet. SSL uses the public-and-private key encryption system from RSA, which also includes the use of a digital certificate, to establish the security of connection between the client and server and to prevent eavesdropping and tampering. Therefore, the SSL protocol is added in AppWeb to establish an encrypted data connection, and the main steps are as follows.

Building the cross compile environment. The host computer is a general-purpose PC running the Linux operating system, which communicates with the target, AM3352, via a serial port and network connection. The prepared cross compilation tool chain (cross-3.2.tar.bz2) can be downloaded from "ftp://ftp.arm.linux.org.uk/". In order to make the arm-linux cross compiler ready to use, the toolchain is decompressed and environment variable PATH is modified by following codes.

```
tar xvzf cross-3.2.tar.bz2
export PATH=/usr/local/arm/3.2/bin:$PATH
```

After the cross compilation tool chain is installed, the Configure file is modified by following commands.

```
export CC="/usr/local/arm/bin/arm-linux-gcc"
export AR="/usr/local/arm/bin/arm-linux-ar"
export LD="/usr/local/arm/bin/arm-linux-ld"
export RANLIB="/usr/local/arm/bin/arm-linux-ranlib"
export STRIP="/usr/local/arm/arm-linux-strip"
export CC_FOR_BUILD="gcc"
```

The IP addresses of host computer and target board are in the same segment, for example, the former is 192.168.1.6 and the latter is 192.168.1.15. The TFTP service is enabled to download the kernel image and file system mirroring to the target board. Using the NFS, the file directory of host computer is mounted to the target board via

"sudo mount 192.168.1.15:/tftpboot/mnt/nfs", consequently, host files can be directly read and written on the target board and application programs can be debugged expediently.

Download and decompression of source. The AppWeb source package and MatrixSSL open source are downloaded from "http://appwebserver.org/" and "http://www.matrixssl.org/" respectively, and then the following commands are executed to decompress them.

```
tar -xvzf appweb-4.3.0-1-src.tgz-C/usr/local/src
tar -xvzf matrixssl-3-4-2-open.tgz-C/usr/local
```

Cross compilation of AppWeb with MatrixSSL. The CPU types of host computer and target board are i686 and ARM Cortex-A8, so the Configure commands are as follows.

```
./configure --host=arm-Cortex-A8-linux --build=i686-pc-linux
--port=80 --type=RELEASE --enable-log --disable-shared-libc
--disable-access-log --disable-shared --enable-static
-- enable-multi-thread --disable-test --disable-samples
--prefix=/usr/appweb --webDir=/var/www
--with-cgi=builtin --with-copy=builtin --with-auth=builtin
--with-esp=builtin --with-upload=builtin --with-matrixssl=builtin
--with-ssl=builtin --with-matrixssl-dir=/usr/local/matrixssl/src
--with-matrixssl-libpath=/usr/local/matrixssl/src
--with-matrixssl-iflags="-I/usr/local/matrixssl/src"
--with-matrixssl-libs=libmatrixsslstatic
make
```

The cross compilation result is shown in Fig. 4.

```
cp conf/standard.conf install.conf
chmod +w install.conf
BLD_PREFIX="/etc/appweb" BLD_WEB_PREFIX="/var/www/appweb-default" \
               BLD_DOC_PREFIX="/usr/share/doc/appweb" BLD_LIB_PREFIX="/usr/lib/appweb" \
               BLD_SERVER=`hostname` BLD_LOG_PREFIX="/var/log/appweb" \
               BLD_HTTP_PORT=80 BLD_SSL_PORT=4443 \
               patchAppwebConf install.conf
# Return from make -C ./appweb

make -C ./samples compile
# Return from make -C ./samples

make -C ./package compile

make -C ./package/LINUX compile
# Return from make -C ./package/LINUX
# Return from make -C ./package
tequlia@tequlia-laptop:~/tequlia_paper/appweb+matrixssl/appweb-src-2.4.1$ ls
appweb  buildConfig.h     configure  esp          http         make.dep  obj            releases
bin     buildConfig.make  doc        fileList.txt INSTALL.TXT  Makefile  package        samples
build   buildConfig.sh    ejs        FILES.TXT    lib          mpr       README_SRC.TXT tools
```

Fig. 4. Cross compilation result of AppWeb with MatrixSSL

Then, the executable and configuration files of AppWeb and dynamic and static linking libraries of MatrixSSL will be generated.

Transplantation of AppWeb with MatrixSSL. The files in /etc/appweb, such as appweb.conf, conf/, groups.db, users.db, mime.types, are moved to working directory firstly, then the appweb.conf is modified as follows.

Open the Web service. Finally, the Web service is enabled via "./appweb －f appweb.conf &", and the AppWeb server can be visited in PC browser by "httpS://192.168.1.15:4443", as shown in Fig. 5.

```
ServerRoot ".”
ErrorLog “logs/error.log”
DocumentRoot "/var/www"
LoadModulePath "/usr/appweb/modules"
Listen 80
<if SSL_MODULE>
Listen 4443
</if>
<if SSL_MODULE>
LoadModule ssl libsslModule
<if MATRIXSSL_MODULE>
LoadModule matrixSsl libmatrixSslModule
</if>
</if>
LoadModule cgi libcgiModule
AddHandler cgiHandler .cgi .cgi-nph .bat .cmd .pl .py
ScriptAlias /cgi-bin/ "$DOCUMENT_ROOT/cgi-bin"
SSLCertificateFile "$SERVER_ROOT/server.crt"
SSLCertificateKeyFile "$SERVER_ROOT/server.key.pem"
```

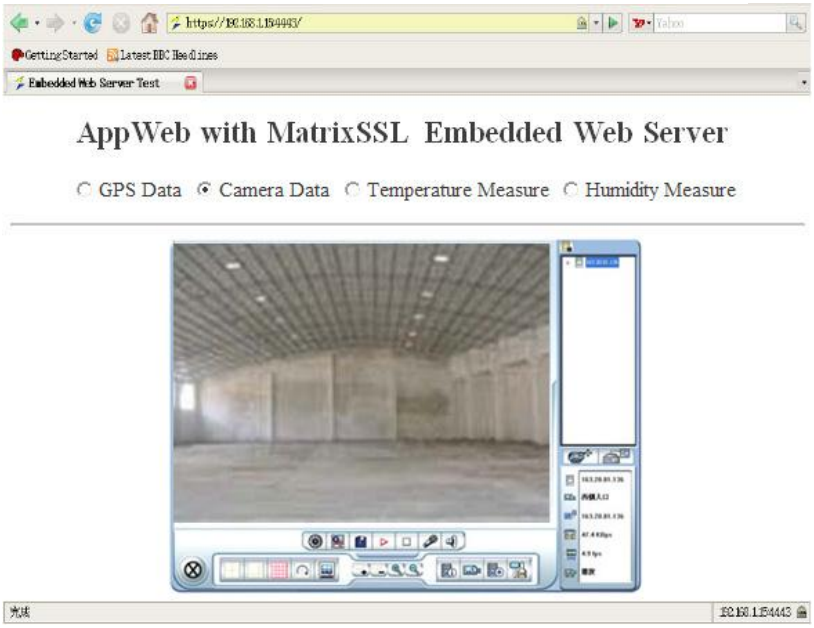

Fig. 5. Interface of video surveillance in the browser

5 Conclusions and Future Work

Internet of Things is a huge global information system composed of hundreds of millions of objects that can be identified, sensed and processed based on standardized and interoperable communication protocols. The IOT based granary management can intelligently process the perception devices' state, provide management and control for decision-making, and even make them cooperate with each other autonomously without manager's intervention. IOTGW plays an important role in IGMS, which facilitates the seamless integration of perception objects and Internet, and the

management and control with perception devices. This paper presents a secure IOTGW based on AppWeb and MatrixSSL, which has guarantee of information security in addition to achievement of data forwarding, protocol transformation and device control. Furthermore, the IOTGW framework is flexible, making it usable for many applications, such as smart home, industrial monitoring, smart grid, environment monitoring and so on. In future works, the IOTGW will be put into practice to build a county intelligent granary management system, then improvement and production will be considered.

Acknowledgment

This work has been funded by Advance Research Project of Xi'an Communications Institute under grant No.XATYB013. The authors would like to thank Prof. Yanpu Chen for his help and valuable comments.

References

1. Xiao Kun. Design of a Wireless Granary Temperature Monitoring System [D]. Hefei: Hefei Industrial University, 2008, 5-12(in Chinese).
2. Zhang Zhen. Research of Granary Monitoring System Based on Wireless Sensor Network [D]. Zhengzhou: Zhengzhou University, 2010, 6-16(in Chinese).
3. Xu Baisen, Zhang Dexian, Yang Weidong. Research on architecture of the Internet of Things for grain monitoring in storage [C]. Proceedings of International Conference on Communications in Computer and Information Science, 2012(312):431-438.
4. Luigi Atzori, Antonio Iera, Giacomo Morabito. The Internet of Things: a survey [J]. Science Direct Journal of Computer Networks, 2010, 54(15):2787-2805.
5. Xie Yanxin, Cheng Haimin, Pan Yexing. Application of network of things technologies on granary monitoring system [J]. Hubei Agricultural Science, 2012, 51(20):4645-4650.
6. Xu Peng, Yao Yindi. Design of a granary monitoring system based on Internet of Things [J]. Journal of Xi'an University of Post and Telecom, 2013, 18(3):122-124.
7. Chen Hao, Jia Xueqin, Li Heng. A brief introduction to IoT gateway [C]. IET International Conference on Communication Technology and Application, 2011, 610-613.
8. Zhu Qian, Wang Ruicong, Chen Qi et al. IOT gateway: bridging wireless sensor networks into Internet of Things [C]. Proceedings of 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, 2010, 347-352.
9. Till Riedel, Nicolaie Fantana, Adrian Genaid et al. Using Web service gateways and code generation for sustainable IoT system development [C]. Internet of Things 2010-Second International Conference for Academia and Industry, 2010, 1-8.
10. Sarika Chhatwani, K.B. Khanchandani. Embedded Web server [J]. International Journal of Engineering Science and Technology, 2011, 3(2):1233-1238.
11. Texas Instruments Incorporated. AM335x ARM Cortex-A8 Microprocessors [EB/OL]. http://www.ti.com. 2013-05-11.
12. Embedthis Software. Appweb for Dynamic Applications [EB/OL]. http://appwebserver.org. 2013-04-13.