JustIoT Internet of Things based on the Firebase Real-time Database

Wu-Jeng Li, Chiaming Yen, You-Sheng Lin, Shu-Chu Tung, and ShihMiao Huang

Abstract— **This paper designs an Internet of Things system (called JustIoT) which is mainly divided into four parts; back-end Google Firebase real-time database, front-end SPA (Single Page Application) web monitoring program (including mobile monitoring App), controller software-hardware, and intelligence server that support MQTT connection and condition control.**

JustIoT receives data from all kinds of controllers, allowing users to set the control rules and remote monitoring and control. JustIOT distinguishes users from managers, vendors, customers, registrants, and visitors. Users can build applications in the above system, to serve customers, and to run a business.

In the JustIoT, management web page based on Angular front-end technology is connected to the Firebase real-time database. The event of data modification of Firebase database can trigger Angular's two-way data binding to achieve Three-way data binding effect to implement server-less architecture easily. The data in Firebase database is read and written by the front-end devices (web apps, mobile apps, and controllers) directly.

The intelligence server is an MQTT server that supports the connections of relatively weak embedded controllers such as the Arduino controller. The intelligent server can be considered as an intermediary between the Firebase real-time database and weak controllers, which performs the transfer of data and remote commands.

The intelligent server is also the intelligent computing center of the JustIoT. It performs condition control.

Keywords: Internet of Things, Firebase, Embedded System, Supervisory Control.

I. INTRODUCTION

In 1999, Kevin Ashton first proposed the term Internet of things. In the same year Bill Joy proposed six devices and devices in the network, focusing on industrial production, unmanned plant applications. By now, the application of the Internet of Things is ubiquitous, according to international research firm Gartner predicts that in 2020 the global networking device will reach 26 billion units, about to create more than 300 billion US dollars in revenue, and the global economy to bring 1.9 trillion dollars attached Value, such a considerable business opportunities to drive the domestic industry chain as a whole, the downstream manufacturers, one after another out of the relevant positions.

Our team has been conducting research on the Internet of Things[1][2], using various open source embedded controllers (Arduino, Raspberry PI, Puppy Linux, Android [3][4]). The system was applied in the factory building sewage monitoring, home security[5][6], computer room environmental monitoring and other aspects[7].

This research is supported by Ministry of Science and Technology, Taiwan (MOST 106-2221-E-150-031).

W-J Li, C-M Yen, Y-S Lin and S-M Huang are with the Department of Mechanical Design Engineering, National Formosa University, Taiwan

S-C Tung is with the Department of Environmental Engineering, Kun Shan University, Taiwan.

In recent years, there are vast progress in many respects, including embedded controller, database structure, data cloud collection, data visualization, front-end technology, HTML standard, back-end data processing, big data, data mining, machine learning, etc.

In this paper, new technology and new architecture are used to build a new Internet of Things. The new system is expected to be more secure, more reliable, easier to develop and maintain.

II. STRUCTURE OF JUSTIOT

A. DESIGN GOAT

JustIoT is designed to achieve the following objectives:

1. Cloud computing architecture: the system is flexible to load demand.
2. New Web front-end technology (Angular SPA): the evolution of Web technology has huge impact on the development of IoT.
3. Real-time monitoring: mobile phone, tablet, desktop computer, laptop could be used for real-time monitoring.
4. System security: data in the form of encoding (SSL) transmission.
5. Integrated (commercially available) open source software and hardware architecture.
6. Universal platform: easy to use for general controller and Internet of things applications.
7. Business model: The system can be used for commercial operation.
8. Control rules: provide condition control[6] services.
9. Machine learning: can be easily used for big data machine learning and other artificial intelligence development structure.

B. Parts of JustIoT

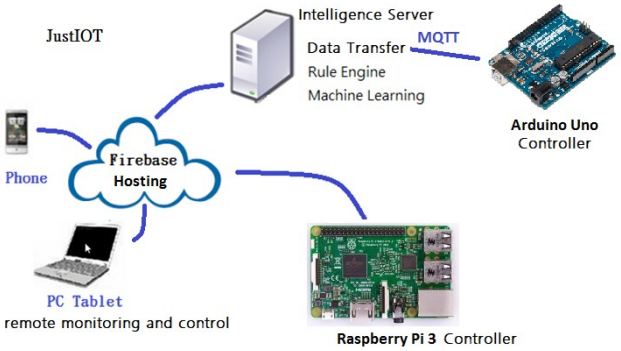

Figure 1. Structure of JustIoT

978-1-5386-3183-6/18/$31.00© 2018 IEEE

In order to achieve the above objectives, the structure of the JustIoT is designed as Figure 1. the focus is as follows:

1.Firebase cloud system:

Firebase is a combination of Google's many services in the cloud, including instant messaging, user authentication, real-time database, storage, hosting, and so on. This study mainly uses its user authentication and real-time database functions, supplemented by instant messaging, to complete the event notification and SMS notification function. Firebase cloud system provides SSL encryption data transmission.

2. Data collection:

One of the main functions of the Internet of Things is data collection. Based on Firebase Cloud, controllers were directly connected to its database to read and write data. Firebase provides access to its real-time database in iOS, Android, JavaScript SDK, REST API, Admin SDK. iOS and Android SDK are for mobile App development. JavaScript SDK is for web applications. And REST API is a general network service, it can be used for a general programming language. Admin SDK is used in the Node environment to implement JavaScript applications. All of these connections are encrypted.

For controllers with weak resources, like Arduino controller, their computation powers are not capable of real-time encryption operation. Intermediary layout between Firebase and controllers is used as shown in Figure 1.

3. Data visualization and management:

Another major function of the Internet of Things is data visualization and management. The data visualization and management of JustIoT is mainly through the browser and mobile phones.

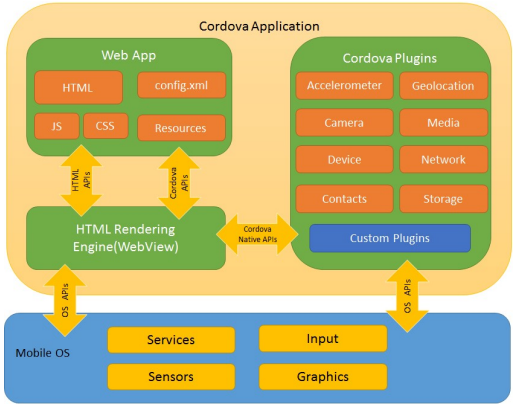

Figure 2. Cordova Application

There are many Mobile phone systems. It would be a huge burden to develop apps for all of them. For the sake of simplicity, Ionic SDK is used to develop mobile phone apps. The Ionic SDK is developed for hybrid apps, powered by Cordova as the underlying architecture[8], using web technology (HTML, CSS, JavaScript), developed on Angular front-end architecture, and can be used on different mobile platforms, including iOS, Android, and Windows Phone. With Ionic SDK, one can use one codebase to suit all mobile platforms.

Meanwhile, the Angular front-end Single Page Application (SPA) can be used in the web browser and mobile phone.

4. Intelligence Server:

The Intelligence server in JustIoT as shown in Figure 1 have three functions;

a. Data transfer:

The Intelligence server is a bridge between Firebase Cloud and weak controllers. It receives data from controllers and forwards the data to Firebase Cloud. At the same time, it receives remote commands from Firebase Cloud and forwards the commands to controllers for execution. The communication between the Intelligence server and weak controllers is on MQTT protocol. Therefore, the Intelligence server is an MQTT server.

b. conditional control Rule Engine:

The JustIoT is not just data collection and visualization. It provides condition control[6] for the system to enhance the system intelligence. In condition control, conditions are set. When some condition is met, certain actions are taken. There are three types of conditions; time condition, input/output condition, and location condition. And there are three kinds of actions; SMS notification, email notification, and output drive. Conditions, actions, and their connections are set by users in the Angular SPA.

On the Intelligence server, there is a Node application which communicates with Firebase Cloud with the Admin SDK. it reads the control rules in the Firebase real-time database, executes them, and send action commands to Firebase real-time database for distributing the commands to respective controllers.

c. Machine Learning: (Subsequent Study)

C. *Firebase Application development*

The use of the Firebase cloud system begins with enabling the user authentication and planning database. JustIoT uses email/password authentication.

Firebase real-time database is a NoSQL-type database. In order to reduce traffic, the structure of the database is designed as flat and does not conform to the normalization required by the general database[9].

Before using the Firebase real-time database, user must set up access rules according to the user's database plan. The settings are somewhat complicated, and the Bolt compiler [10] is used to assist the user by setting the access rules in a simple syntax. Here is a small part of JustIoT bolt file:

```
path /controllers/{uid} {
index() {['createdAt']}
}
path /controllers/{uid}/{controllerid} is Controller {
read() {isSignedIn()}
write(){isOwner(uid)||isAdmin()||isCompanyAdminForUser(uid)}
}
type Controller {
 id: String,
 title:  String,
 description:   String | Null,
```

```
  userid:   String,
  active:   Boolean,
  type: Number | Null,   //  1: Arduino
  geolocation: Number[] | Null,
  productid: String | Null,
  offline: Boolean | Null,
  samplingtime: Number | Null,
  createdAt: Number
}
```

JustIOT's database mainly contains users, companies, controllers, data points, control rules, and data records. JustIOT distinguishes users from managers, vendors, customers, registrants, and visitors. Users who want to upload data are required to register on the system, log in to the system, register controllers and data points. According to the user's planning, user integrate these controllers and data points to form an IoT application, such as home security system.

Once the controllers are connected to the JustIoT, the user can monitor and remotely control his system. A user with mature IoT application can be promoted to be a service provider to service customers. The customers cannot manage the controller settings and can only monitor and remotely control its system. Visitors can only visit open systems in JustIoT.

B. Angular SPA monitoring program

The traditional web service is a three-tier architecture, front-end web pages (HTML, CSS, JavaScript), an intermediate web server, and a back-end database server. Based on Firebase Cloud, JustIoT is a two-tier serverless architecture.

The monitoring and management program of JustIoT is an Angular single-page application. it is a static page and therefore is easy to be hosted in a general cloud system. The monitoring and management program of JustIot is hosted on Firebase.

The Angular SPA program of this research is based on BlurAdmin[11]. Its development uses the framework of the Bootstrap CSS Framework, Sass, Gulp build, AngularJS, and Chart.js. Besides, HTML Canvas Gauges Suite [12] is used for data visualization.

The main functions of the monitoring and management program are user registration, system-related information setup, condition control rule settings, data visualization, historical data viewing and remote control.

D. Arduino Controller

The Arduino controller cannot use SSL-encrypted https transmission. Therefore, it cannot use Firebase's REST API web service to access the Firebase database. In the JustIoT system, an intelligence server which is an MQTT server is utilized as a bridge between the Firebase Cloud and Arduino controller.

An Arduino controller connected to the JustIoT, is a MQTT client. Its program structure is as follows.

```
#include <SPI.h>
#include <Ethernet.h>
#include <PubSubClient.h>  // MQTT client library
const char* username= "xxx@xxx.edu.tw";  // account
const char* pw="xxxxxxxx";              // password
const char* controllerid="xxxxxxx";          // controller id
const char* dptitle="亮度 2 彩燈 1";  // data point setup
//0:digit input 1:digit output 2:analog input 3:analog output
IPAddress ip(140, 130, 17, 233);        //controller ip
IPAddress dnServer(8, 8, 8, 8);         //DNS server ip
IPAddress gateway(140, 130, 17, 254);   //gateway ip
IPAddress subnet(255, 255, 255, 0);     // network mask
IPAddress mqtt_server(xxx, xxx, xxx, xxx); //MQTT server ip
void callback(char* topic, byte* payload, unsigned int length) {
// callback function for data from MQTT server
if((char)payload[0] == 's' && (char)payload[1] == 'e' &&
(char)payload[2]=='t') // remote command ('set')
{... // command execution
}
}
EthernetClient ethClient;
PubSubClient client(mqtt_server, 1883, callback, ethClient);
void setup()
{
Serial.begin(115200);
pinMode(FLASHPIN, OUTPUT);
Ethernet.begin(mac, ip, dnServer, gateway, subnet);
}
void loop() {
if(!client.connected()) {reconnect();} // MQTT connection
client.loop();
long now = millis();
if (now - lastMsg > SAMPLEPERIOD) {
sendData();
}
}
```

In the above program, an MQTT library (PubSubClient) is used to connect to JustIoT MQTT server.

An MQTT connection is a publish-subscribe-based protocol; after logging in to an MQTT server, you can post a message to a topic or subscribe to a topic. When a post is posted to a topic, all users who subscribe to the topic will receive the message.

In order to plug into the JustIoT system, a controller must perform following actions;

1. Login to the JustIoT system.
2. System configuration. System configuration should be done in the Angular SPA management program. The system will assign a controller id to the controller. A system configuration string in a controller is optional and is used solely for quit setup.
3. The controller should periodically send measurement data to the JustIot system.

4. A controller with MQTT connection must subscribe a specific topic to a receive remote command from MQTT server.
5. The controller should execute the remote command.

III. Result and Validation

"Start for free, then pay as you go." With Firebase Cloud, we started a free project to construct the JustIoT system. After enabling the user authentication and planning database with appropriate access rules, we focused on developing the Angular SPA monitoring and management program.

For testing, a simple smart house IoT application with Arduino controller was designed. As described above an MQTT server was needed as a bridge between an Arduino controller and Firebase database.

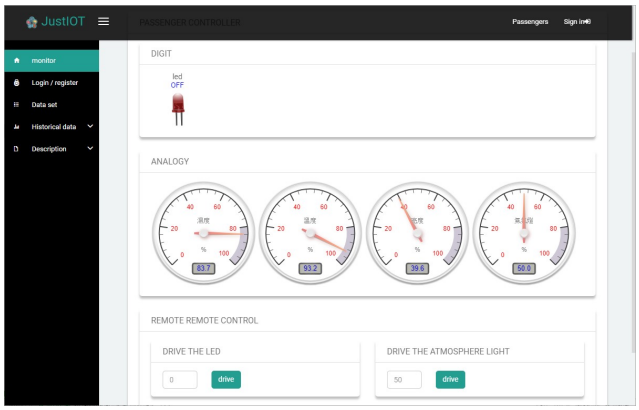

Figure 3. Smart house application (https://justdemo-df23d.firebaseapp.com/)

The MQTT server resident in the intelligence server was built with Mosca[13]. Mosca is a Node.js MQTT server/broker. We overwrote its 'authenticate' function to be consistent with the authentication function of the Firebase project.

The smart house system has Three inputs: humidity, temperature, and brightness and two outputs: living room light (digital output) and bedroom mood light (analog output). Once the Arduino controller connected to the MQTT server, we could monitor the smart house system and remotely drive its outputs with the JustIoT SPA monitoring and management program in a browser as shown in Figure 3.

With condition control of the JustIoT system, we made the smart house system smart. For example, we set the time condition for living room light. We set IO condition between brightness measurement and bedroom mood light [6].

When the JustIoT system was stable, the project was upgraded to a pay plan to provide reliable connections and rapid responses for many users. The usage fee depends on the quantities of database data stored, database data downloaded, website hosted, and website pages transferred. "Pay as you go." Therefore, the JustIoT system is flexible to load demand.

The portal page of the JustIot system is located in https://justdemo-df23d.firebaseapp.com/. Once the page is loaded, the user is a visitor. He can browse public (open) applications of the system. After registering/logging in as a registrant, a user can then register controllers and data points of his application. Then he can plug his system into the JustIoT system by starting his controllers. Since the system is hanged in the JustIoT, he can setup condition control rules, and remotely monitor and control his system. A JustIoT application of a registrant is a public (open) application which a visitor and other registrants can browser.

The JustIoT system was used in university courses to promote IoT application development of students. The MQTT communication between Arduino controller and the intelligence server in the JustIoT is not secure. The usage of MQTT communication in the JustIoT system is solely for educational purposes.

Despite that, the JustIoT can be used to run a business. When a registrant matures his application with business potential. JustIoT system administrator can register him as company admin (vendor) to provide services to his customers. He can then set up IoT systems for his customers in the JustIoT system. A customer system is not public (open) to the visitors or regular registrants.

Another JustIoT system was put into business usage as shown in Figure 4[14]. The system was separated from the one for educational usage for security reason.

Figure 4. JustIoT business model

IV. Conclusion

This paper designs an Internet of Things system called JustIoT. It can be used for educational purposes and help students to design IoT applications.

Firebase Cloud had recently introduced the Cloud Functions feature which allows users to write programs in JavaScript language and put them on the Firebase cloud platform. The programs are triggered by relevant events. In JustIoT the implementation of intelligent server condition control can be turned into a cloud function on the Firebase Cloud system. However, Firebase cloud function service seems not yet stable at this moment.

With JustIoT to collect huge data, a machine learning framework will be integrated to enhance the system intelligence in the near future.

Acknowledgment

This research is supported by Ministry of Science and Technology, Taiwan (MOST 106-2221-E-150-031).

REFERENCES

[1] Wu-Jeng Li, and Shu-Chu Tung, A Web-based Multiple Stations Supervisory Control Framework, Advanced Science Letters, volume 8, 274-278 (2012).

[2] Wu-Jeng Li, Shu-Chu Tung, and ShihMiao Huang, Server-side Ladder Logic of a Web-based Supervisory Control System, The First International Conference on Engineering and Technology Innovation 2011 (ICETI2011), Kenting, Taiwan, November 11-15, 2011.

[3] Shu-Chu Tung, Wu-Jeng Li, Shih-Miao Huang, A Web-based Android Supervisory Control System, Applied Mechanics and Materials Vols. 284-287, pp 3211-3215, January 2013.

[4] Shu-Chu Tung, Wu-Jeng Li, Shih-Miao Huang, A Web-based Arduino Supervisory Control System, Applied Mechanics and Materials Vols. 284-287, pp 3216-3220, January 2013.

[5] W. J. Li, S. C. Tung, S. M. Huang, "Home Security Service Enhanced with Information Stations", Appl.ied Mechanics and Materials, Vols. 764-765, pp. 915-918, May 2015.

[6] Shu-Chu Tung, Wu-Jeng Li, Shih-Miao Huang, Home Security Service and Condition Control, Applied Mechanics and Materials, vol. 479-480: 661-664, December 2013.

[7] Shu-Chu Tung, Wu-Jeng Li, K. C. Lee, Jui-Chang Lin, Environment Supervisory Control of Computer Room Based on Android Device, Applied Mechanics and Materials, Vol. 418, pp124-127, September, 201379-480: 661-664, December 2013.

[8] Diagram on architecture of the Ionic, https://forum.ionicframework.com/t/is-there-a-graphic-diagram-available-that-shows-how-ionic-2-and-all-related-technologies-connect/88375.

[9] Firebase, 'Structure Your Database', https://firebase.google.com/docs/database/web/structure-data#best_practices_for_data_structure.

[10] Bolt Compiler, https://github.com/firebase/bolt.

[11] Angular Bootstrap Admin Panel Framework, https://github.com/akveo/blur-admin.

[12] HTML Canvas Gauges, https://github.com/Mikhus/canvas-gauges.

[13] Lelylan Blog, 'How to Build a High Availability MQTT Cluster for the Internet of Things', https://medium.com/@lelylan/how-to-build-an-high-availability-mqtt-cluster-for-the-internet-of-things-8011a06bd000.

[14] Wu-Jeng Li, Chiaming Yen, You-Sheng Lin, Shu-Chu Tung, and ShihMiao Huang, Cloud Supervisory Control System Based on JustIoT, submitted to Proceedings of the 2018 International Symposium on Semiconductor Manufacturing Intelligence (ISMI2018)