http://doi.org/10.35784/iapgos.923

SOFTWARE DEVELOPMENT FOR SMART HOME PROCESS CONTROL

Vitalii Kopeliuk[1], Vira Voronytska[1], Volodymyr Havryliuk[2]
[1]Rivne State University of Humanities, Department of Applied Math and Computer Science, Rivne, Ukraine, [2]International University of Economics and Humanities Academician Stepan Demianchuk, Department of Information Systems and Computing Methods, Rivne, Ukraine

Abstract*. Here we make an overview of the main stages of software development for server management system of "smart" building with a central controller via a mobile device. We have developed our own version of the concept.*

Keywords: Arduino, Internet of things, microcontroller, smart home, client-server architecture, sensors

OPRACOWANIE OPROGRAMOWANIA DO STEROWANIA PROCESAMI W BUDYNKU INTELIGENTNYM

Streszczenie. *Rozpatrzono etap podstawowy opracowania oprogramowania dla systemu sterowania inteligentnym budynkiem z centralnym kontrolerem, przez urządzenie mobilne. Zaproponowano własną wersję rozwiązania.*

Słowa kluczowe: Arduino, Internet rzeczy, mikrokontroler, inteligentny budynek, architektura klient-serwer, czujniki

Introduction

Smart Home (Home automation, smart home) is one of the most promising areas of information and communication technologies. Such type of systems connects all of the electrical devices of the house in one functional system which can be operated by a user with display-controller or with certain algorithms. Optimizing energy consumption today is one of the key objectives of the Smart Home systems.

Number of system types in this area is increasing: there are wireless technologies for the integration of devices into a single network, new kinds of sensors. At the same time, increased demand for Smart Home product makes the following problems extremely relevant:

- insufficient standardization and compatibility of different protocols;
- system reliability;
- safety and security systems from unauthorized access;
- costs and complexity of deployment.

This paper examines the main existing approaches to building Smart Home systems, analyzes network protocols and technologies used to build local networks, describes their advantages and disadvantages. In addition it investigates ways to improve existing solutions in this area. It justifies the choice of protocols, technologies and approaches for building Smart Home systems that are capable of solving main disadvantages.

The aim of this work is to build a device management system for the smart home, that will consist of the following elements:

- monitoring and controlling device (Smart Monitor),
- server that collects data from all devices in the local network and makes decisions regarding changes in the configuration of the network and switching devices on and/or off depending on the current state of the system.

1. Internet of things and cloud computing

At present time the vast majority of smart home systems do not have the function of remote control via the Internet. Meanwhile, mobile devices with constant network access have now become commonplace, they are practically everywhere. In 1999, the founder of the Research Center of Auto-ID Center at MIT Kevin Ashton proposed the term Internet of Things (Internet of Things). The basic idea is that the way a new generation of things will not only be "intelligent", but also will be connected in a network – Internet of Things [6]. The concept implies that devices such as smartphones, tablets, TVs, and various sensors and controlled devices with wireless modules, such as Wi-Fi and Bluetooth, can interact with each other as well as with users by using these modules. Due to the massive proliferation of mobile devices, the remote control of such systems as Internet of things, has become possible. Remote control has obvious advantages. The main one is of course security. When all of the inhabitants are not in the house, it is possible to remotely monitor the state of the house using cameras together with sensors and controllers.

Equally important is to increase user comfort when using smart home system. Often smart home control systems use scripts to manage light and heat automatically. However, some users prefer not to use these options. And the presence of the remote control option, for example, may himself at the approach to the house he needs to include devices (turn on lights, appliances, and include pre-heating or air conditioning). Implementation of remote access is possible through the use of cloud computing, where users are provided with universal access network computing resources, services and applications. There are several models of cloud computing. The most suitable model for a given problem definition is SaaS (software as a service). This model is based on providing customers access to software over the Internet. The main advantage of the SaaS model for end users is the absence of a need to install and update software, and caring about performance equipment, which operates the application. When using cloud computing systems smart home there are two options. First one is when the controller (server) for managing smart home devices is located not in the house, but on a cloud. In this case smart home system can be accessed from any point where internet access in available. As for the second option, controller can be located in the building, and the software is installed on a cloud. Also, in the second case, only additional modulus, that provide access to the internet are required, which reduces the requirements to the system. Also in the case of implementing remote control to the already existing system of smart home, there is no need to replace any equipment, it is enough to provide access for the controller to the cloud server. Direct remote control of the smart home systems can be done either via a web browser or through a special mobile application. One more important thing. Many modern devices that are used in smart homes, have their own specific protocols of the data transmission and can interact with internet services only via their own specific API's. So often it is impossible to expand the system of a smart home by adding extra devices, such as smart refrigerator, for example, since they are requiring completely different data transmission system. However, cloud-based system gives a general interface for controlling all of the devices via cloud. In this case all the devices interact with each other via cloud. The application of cloud technologies in the smart home will make them much more flexible, and will reduce maintenance costs and system expansion with any smart refrigerator, or other devices working on other data transmission protocols. However, with cloud service that will provide a common interface management of all systems and different devices will interact with each other through the cloud, it is possible to use devices from different manufacturers with different data transmission protocols. The application of cloud technologies in smart home will make them much more flexible, and will

reduce maintenance costs and system expansion. which any smart refrigerator, or add the devices working on other data transmission protocols. However, with cloud service that will provide a common interface management of all systems and different devices will interact with each other through the cloud, it is possible to use devices from different manufacturers with different data transmission protocols. The application of cloud technologies in smart home will make them much more flexible, and will reduce maintenance costs and system expansion. Is the ability to use devices from different manufacturers with different data transmission protocols.

2. Microservices

Microservice style of architecture [5] is an approach to developing a holistic applications as a set of small services, each of which runs on its own process and connect with others through lightweight mechanisms such as RPC or HTTP. Services are built according to a specific task and can be independent to be deployed by automated systems. There is some minimum centralized management of such services.

Traditional server systems are usually built as monoliths – logically separate executable programs. And this approach is natural: the whole logic of request processing works in a single process that allows you to use existing tools programming languages to divide an application into classes, functions, and namespaces. The monolith can be scaled horizontally by running multiple instances outside the load balancer.

Monolithic software is quite successful, but it has its disadvantages. Change cycles monoliths are bound together, that is, changing a small part of the system requires reassembly (compilation) and deployment. Over time, it becomes difficult maintain a good modular structure while keeping changes relevant specific module, only inside it. It is necessary to scale everything monolith instead of individual parts that require more resources.

Microservices can also be deployed and scaled independently. They also provide clear boundaries between modules, even allowing implement separate subsystems in different programming languages. That's it delimiting helps to manage the complexity of the codebase, as each the module will have a public API that will contain only the required functionality, and everything else will be encapsulated and not relevant for the development of other services, which depend on it.

The main disadvantage of microservice architecture is the increase in complexity error handling. Unlike a monolith, every call to a service can fail. Therefore, it is necessary to develop mechanisms for monitoring the condition services, check the various metrics of their functioning as well automate the restoration of the microservice if it fails.

The positive thing is that there are failures in microservice architecture largely isolated. If meeting the request requires a call many services and some of them are unavailable, perhaps less complete response (graceful degradation).

Difficulties can also be caused by providing consistency the deployment of dependent microservices, and the need for management transactions that interact with multiple subsystems.

Internal communication between servers occurs in binary TCP format for performance reasons, but from the point code view, each service provides a public API that encapsulates the creation messages to the corresponding service [4].

Apart from the division of responsibilities, the main advantage is the opportunity horizontal scaling of each service separately. For example, if the flow of operational data is greatly accelerated without increasing the flow administrative data, you can scale the corresponding DBMS (Database Management System) separately.

3. Cloud server interaction with smart home devices

In order to interact with the cloud server, smart home devices need to select protocol for sending messages. Currently there are several dozens of data transfer protocols that allow communication IoT devices. In our implementation we preferred protocol MQTT (Fig. 1).

MQTT (Message Queue Telemetry Transport) - Simplified network protocol that runs on top of TCP/IP. It is used to exchange information between devices on the basis of publish-subscribe. The first version of the protocol was developed by Dr. Andy Stanford Clark (IBM) and Arlen Nipper (Arcom) in 1999 and published under license royalty-free. MQTT 3.1.1 specification was standardized OASIS consortium in 2014.

The main advantages of the protocol are:

- the pattern of interaction on the basis of publish-subscribe solutions most suitable for working with different kinds of sensors,
- easy to use. This software unit does not contain any unnecessary functionality and can be easily embedded into any complex system,
- it is easy to administer,
- provides work in constant communication loss or other problems on the line,
- there are no restrictions on the format of the data.

MQTT requires broker messages. The broker is responsible for distributing messages to all devices that are signed in this newsletter. MQTT defines methods (so-called "verb") to indicate the desired action that should be performed on the identified resource, which can be either existing data or data that is generated dynamically, depending on the server implementation. Often resource corresponds to a given file or it is a result of a file processing on a server.

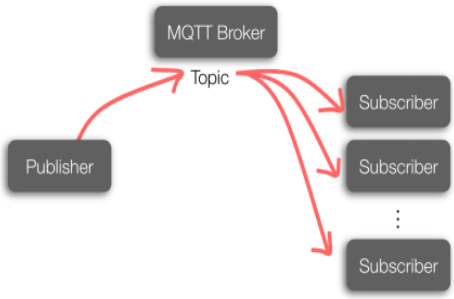

Fig. 1. MQTT broker scheme

Here are examples of the following methods:

Connect. Connect: Waiting for a connection to the server.

Disconnect. Disconnect: Waiting until the customer MQTT finish any work that must be done, and the session TCP/IP is broken.

Subscribe. Subscribe: Pending completion of Subscribe method.

UnSubscribe. Unsubscribe: server asks the client to unsubscribe from one or more topics.

Publish. Post: immediately returns to the application flow after customer requests pass MQTT.

Currently there are several open Brokers MQTT: Emqttd, ActiveMQ, Apollo, Mosquitto, RabbitMQ and others. They differ only in their feature set, and some add-ons over the standard version of the protocol MQTT.

4. The operating data reception subsystem

Because data arrives at high speed and can occur bursts of traffic, you need to have a buffer between the database and the front servers for provide load balancing. For this platform architecture involves using a message queue.

Message queues define an asynchronous communication protocol. this is means that the sender and recipient of the message should not interact with message queue at the same time. Messages that are queued stored until received by the receiver.

An important feature of the message queue is to ensure resilience and reliability, which are implemented using various data storage strategies. To increase the reliability of message delivery, it is possible to save them on drive before they are received by the recipient. Even if the receiver program or message queue will stop working due to failure, messages will be safe and will be available to recipients, only the system will be operational again.

Using Message Queuing can support much more messages. In general, applying queue architecture is a good strategy for organizing asynchronous processing of big data.

We formulate the requirements for the message queue:

- **High performance**. Need to provide high speed writing messages to the queue.
- **Horizontal scalability**. Increase through put system ability by increasing the number of servers.
- **Ability to save messages to disk**. Need to reduce the amount of data lost when the message queue server is down. This is very necessary when reading data from the queue in large packets with large intervals - in this case, in the absence of permanent storage You can lose a lot of data.
- **Replication**. For some increase in data storage guarantees and maintaining the availability of the message queue for recipients in case of shutdown part of the servers.
- **Ability to configure storage reliability messages**. That is, how many servers are replicated and how often saved to disk. Needed in order to adjust the system so that it is more reliable to store messages from devices that have a long period between data transfer operations. With a heavy load, it should be possible to loosen such guarantees.
- **The ability to divide messages into groups by topic.** In this way, can send recipients of messages that implement a specific for a certain type of device logic, to receive messages with a specific queue in which the relevant data is placed.

5. Software smart home central controller

The system of interaction between devices in the "smart" house built (Fig. 2) on the architecture of the central controller (server). Therefore all requests coming from client applications are added to the message queue and only then turn to the commands that will be distributed to the microcontroller (in the case of multiple rooms/facilities).

Also, for the additional reliability, all the data is synchronized with cloud server, which helps to easily track errors. Such architecture has many advantages:

- Prior processing of user requests; only those requests that have been processed on a server will be sent to the microcontrollers; this helps to enable multi-user mode as well as organize all of the commands in a queue without any errors.
- Removing the burden of the microcontroller since it has a limited set of memory, saving a large amount of requests on it might influence the quality of the performance. When server is used microcontroller receives only a final command to execute, which are of a type ON-OFF.
- Scalability. When microcontroller-application system is used, we are having a problem with switching between the tasks for different devices in different rooms of the house. When a central server is used, it will take care of this problem by statically describing all of the controllers. Final user only has to define desired settings.
- Enhanced security system. Given the low processing power of microcontroller, in order to organize more or less reliable encryption on it is impossible. Unlike microcontrollers server has sufficient resources to perform encryption/decryption information.

Despite the obvious advantages, systems of this kind have also a number of disadvantages:

- Need of an extra device to perform the functions of the server. That adds complexity to the system and its maintenance.
- Need of an additional software for the organization of the server.

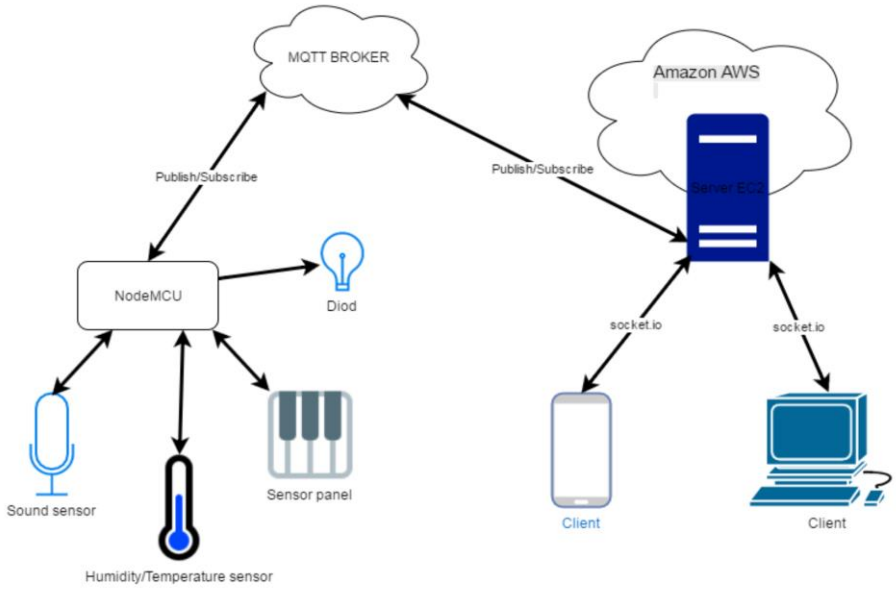

Fig. 2. An example of the finished system

6. Conclusion

We developed our own software for server management system of "smart" building via mobile device with a central controller, which is the main module of communication between the application and domestic electronic devices. We also developed basic functions such as storage configuration and information link between devices, encryption system, communication with cloud storage, energy forecasting, and more.

The developed platform meets the set functional and non-functional requirements and contains the implementation of the following subsystems: acceptance, storage, processing of operational data; authentication and security; work with administrative data; monitoring. This allows it to be used as such dedicated user groups: devices, administrators, analysts, and security administrators.

Such distributed service oriented service has been proposed and implemented (microservice) platform architecture that delivers high performance and almost endless scalability. This allows you to increase power systems for servicing device networks of all sizes and for increasing the capabilities of the data analytics subsystem. Effective implementation system components to optimize hardware costs platform.

The platform also guarantees a significant level of resiliency, which reduces the probability of losing operational data and providing a high level of it availability. The investigated and implemented aspects of security provide sufficient level of data protection during transmission and protect the system from the various information attacks.

During the development, the platforms were properly and properly selected modern systems, protocols, data formats, libraries and databases are applied data. Third-party solutions have open source that simplifies system expansion, has economic and other benefits.

Various aspects of the chosen architectural features were described in detail solutions: from justifying high-level division into functional modules to features of effective implementation of I/O and the influence of the selected scheme authentication for scalable platform architecture.

Unlike existing platforms, the solution developed is good expandable in the sense that it can be easily adapted to analytics data coming from different devices and sensors, setting the chains data processing. Data protocols and formats were chosen not only in light of them efficiency, and also given the convenience and ease of extending the platform.

Other key features of the platform are monitoring and the functionality of securely transferring service information between devices and by administrators. These capabilities allow you to effectively manage your network devices, and flexibly configure alerting tools possible problems and promptly respond to them.

There are architectural and technical solutions proposed and applied in the work versatile enough to allow them to be built various big data processing systems and other platforms for the Internet things specializing in narrower areas.

References

[1] Aberer K.: Smart Earth: From Pervasive Observation to Trusted Information. International Conference on Mobile Data Management, Mannheim, 2007, 3–7
[2] Darianian M., Michael M.P.: Smart Home Mobile RFID-Based Internet-of-Things Systems and Services. International Conference on Advanced Computer Theory and Engineering, Phuket, 2008, 116–120.
[3] EPCglobal.EPC information services (EPCIS) version 1.0.1 specification. EP-Cglobal, Lawrenceville 2007.
[4] Garg V.K.: Elements of Distributed Computing. John Wiley & Sons, New York 2002.
[5] Marz N., Warren J.: Big Data: Principles and best practices of scalable realtime data systems. Manning, Shelter Island 2015.
[6] Sathi A.: Big Data Analytics: Disruptive Technologies for Changing the Game. Mc Press, Boise 2012.
[7] Tamer Özsu M., Valduriez P.: Principles of Distributed Database Systems. Springer, New York 2011.
[8] Tel G.: Introduction to Distributed Algorithms. University Press, Cambridge 2000.

Vitalii Kopeliuk
e-mail: vkopeluk@gmail.com

Master Student at the Department of applied math and computer science of Rivne State University of Humanities, Rivne, Ukraine.

Robotics, computer technology and computer technology, programming, programming of microcontrollers, artificial intelligence.

http://orcid.org/0000-0002-3538-7028

M.Sc. Vira Voronytska
e-mail: vera.voronitska@gmail.com

Senior Lecturer at the Department of applied math and computer science of Rivne State University of Humanities, Rivne, Ukraine.

Development of sites, development of multimedia courses, computer science and computer technologies, programming.

http://orcid.org/0000-0003-0014-1121

Ph.D. Volodymyr Havryliuk
e-mail: V.i.Havrilyuk@gmail.com

Associate professor at the Department of Information Systems and Computing Methods International University of Economics and Humanities Academician Stepan Demianchuk, Rivne, Ukraine. Engaged in applied and computational mathematics, mathematical modeling of technological processes, numerical modeling and analysis, differential equations in applied mathematics, physics and engineering, computer science and computer technologies, programming.

http://orcid.org/0000-0003-3377-6465

otrzymano/received: 21.12.2019 *przyjęto do druku/accepted: 26.06.2020*