INTERNATIONAL EDUCATIONAL APPLIED RESEARCH JOURNAL (IEARJ)
Volume 04, Issue 03, March 2020
E-ISSN: 2456-6713

A TIME SYNCHRONIZATION TECHNIQUE FOR MQTT BASED HOME AUTOMATION SYSTEMS

Karthik L[1], Giridharan N[2]

[1]*Student, K.S.Rangasamy College of Technology, Tiruchengode, Tamilnadu, India.*

[2]*Assistant Professor, K.S.Rangasamy College of Technology, Tiruchengode, Tamilnadu, India.*

Abstract: *The advent of internet-of-things (IoT) - based home automation systems, time synchronization techniques for low power sensor modules are in high demand. This technology offers new and exciting opportunities to extend the connectivity of devices within the house for the aim of home automation. Home automation refers to regulate of home appliances using information technology. With the help of rapid expansion of the Internet, there is the potential to control and automate the home appliances. It is achieved by interfacing the internet with embedded systems. This paper deals with the idea of implementing the NodeMCU based interactive home automation system through internet of things in order to synchronize the time. In this project, I will also use the MQTT (Messaging Queuing Telemetry Transport) protocol which is considered a safe and secured protocol. It is an ISO Standard (ISO/IEC PRF 20922) publish-subscribe based messaging protocol. It works on top of TCP-IP protocol. The MQTT option field and a shim header are used to include time-stamps in the home automation system. The proposed scheme can thus be applied to both IP-based and non-IP-based home automation systems.*

Keywords: *MQTT, NodeMCU, IoT.*

1. INTRODUCTION:

As the user demands for communication between the home and the outside world increase, the requirement for (IoT) for home automation have also increased. As a result, IOT technologies have accelerated the development of communication protocol as well as sensor for home automation system. Currently home automation system platforms, which collect data from the sensors and appliances using commercial-of-the-shelf (COTS) wireless technologies, are developed alongside the IoT technologies. For Example: fire alarms and intrusion warning in home automation system need their data to be transmitted as quickly and reliable as possible. Hence, time synchronization between nodes is so crucial that it can severely affect the performance of the home automation system.

1.1 MQTT (MQ Telemetry Transport):

MQ (MQ Telemetry Transport) may be a lightweight messaging protocol that gives resource-constrained network clients with an easy thanks to distribute telemetry information.

The protocol, which uses a publish/subscribe communication pattern, is employed for machine-to-machine (M2M) communication and plays a crucial role within the internet of things (IoT).

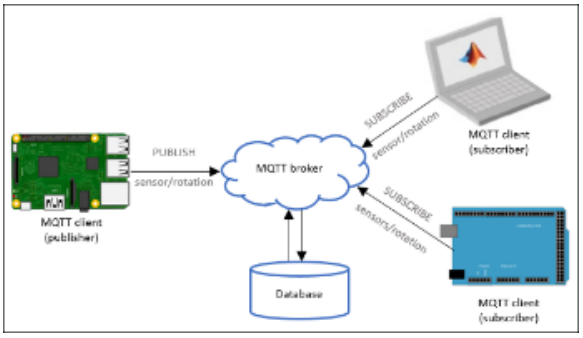

Figure 1.1.1 MQTT model

1.1.1 How MQTT works:

An MQTT session is split into four stages: connection, authentication, communication and termination. A client starts by creating a TCP/IP connection to the broker by using either a typical port or a custom port defined by the broker's operators. When creating the connection, it's important to acknowledge that the server might continue an old session if it's given a reused client identity. The standard ports are 1883 for non-encrypted communication and 8883 for encrypted communication using SSL/TLS. During the SSL/TLS handshake, the client validates the server certificate to authenticate the server. The client can also provide a client certificate to the broker during the handshake, which the broker can use to authenticate the client. While not specifically a part of the MQTT specification, it's become customary for brokers to support client authentication with

SSL/TLS client-side certificates. Because the MQTT protocol aims to be a protocol for resource constrained and IoT devices, SSL/TLS won't always be an option and, in some cases, won't be desired. In such cases, authentication is presented as a clear-text username and password that's sent by the client to the server as a part of the CONNECT/CONNACK packet sequence. Some brokers, especially open brokers published on the web, will accept anonymous clients. In such cases, the username and password are simply left blank.

MQTT is named a light-weight protocol because all its messages have a little code footprint. Each message consists of a fixed header – 2bytes -- an optional variable header, a message payload that is limited to 256 MB of information and a quality of service (QoS) level.

The three different quality of service levels determine how the content is managed by the MQTT protocol. Although higher levels of QoS are more reliable, I even have more latency and bandwidth requirements, so subscribing clients can specify the very best QoS level i might wish to receive.

The simplest QoS level is unacknowledged service. This QoS level uses a PUBLISH packet sequence; the publisher sends a message to the broker just one occasion and therefore the broker passes the message to subscribers just one occasion. There is no mechanism in situ to form sure the message has been received correctly, and therefore the broker doesn't save the message. This QoS level can also be mentioned as at the most once, QoS0, or fire and forget.

1.2 System Implementation:

A sample implementation of the proposed time synchronization system is illustrated. As depicted, a home automation system is designed for monitoring and controlling devices.

A general home gateway collects information from household devices being provided to the user and relays the user's control commands toward household devices in an IP network environment.

1.2.1 NodeMCU ESP8266:

Figure 1.2.1.1 NodeMCU ESp8266

NodeMCU is an open source firmware for which open source prototyping board designs are available. The name "NodeMCU" combines "node" and "MCU" (micro-controller unit). The term "NodeMCU" strictly speaking refers to the firmware rather than the associated development kits. Both the firmware and prototyping board designs are open source.

1.2.2 Home Automation Devices:

To determine the feasibility and effectiveness of the proposed system, three devices were developed: a fire detector, bulb controller and integrated environmental sensor.

Bulb Controller: A prototype bulb controller glows whenever there is a change in the LDR value. User can remotely monitor and control the state of the lamp.

Environmental Sensor: A prototype sensor incorporated four types of sensor: temperature- humidity, luminance, CO2, and NH3 sensors was developed and integrated with an RS485 communication interface.

1.3 Background and Assumptions:

In this section I describe standard methods for network time synchronization, sources for errors, basics for IoT technologies, and our assumptions for devices that are the target for time synchronization. I am also describe practical IoT deployment scenarios that could utilize our architecture to achieve accurate time synchronization.

1.3.1 Time Synchronization:

Synchronizing the clocks of networked devices is an important problem. Standard solutions include Network Time Protocol (NTP) and its variants including Simple Network Time Protocol (SNTP), Precision Time Protocol (PTP) , Datacenter Time Protocol (DTP) and Mobile NTP (MNTP), which have been developed for a variety of devices (e.g., routers and wired hosts use NTP, mobile phones use SNTP/MNTP, datacenters use DTP, etc.). To effectively counter and proper the differing rate at which clocks advance (also referred to as clock drift), these protocols estimate one-way delays (OWDs) between the devices during which I deployed and a hierarchy of clock references known as the stratum servers. Hierarchy starts with Stratum 0 servers, which are highly-precise clock sources (e.g., GPS or atomic clock), and goes right down to less-precise Stratum 15 servers.

Wired devices estimate clock offsets (defined as the difference in time between the client's clock and the remote reference) based on OWD measurements, which are determined using the timestamps exchanged with stratum servers in a process called polling. These timestamps are

vetted using another process called filtering, which uses heuristics to remove inaccurate offset estimates. The filtering heuristics are optimized using indicators such as round-trip delay, jitter and oscillator frequency as part of the clock discipline algorithm. Both the clock filtering and synchronization algorithms are implemented and run as part of the ntpd daemon in wired devices. Similarly, wireless devices like mobile phones use SNTP to accumulate clock estimates by polling the stratum servers, but without NTP's sophisticated clock synchronization algorithm and filtering heuristics. They also note that only the first octet of SNTP packets is set in the polling process employed by wireless devices, resulting in the exchange of less-accurate clock estimates. Furthermore, wireless devices are known to employ vendor-specific SNTP implementations with varying polling frequencies and retry rates in case of polling errors/failures.

1.3.2 Clock Synchronization Errors:

Synchronizing a client's local clock with a reference time source consists of calculating two interdependent components: (i) clock offset; and (ii) clock skew or rate-error, which is that the difference in rate or frequency between the client's clock and therefore the remote reference. While the offset synchronization (i.e., calculating offset) is sufficient for general coordination of events among distributed clients, rate synchronization (i.e., calculating clock skew) is necessary to achieve tight synchronization. Surprisingly, while skew is a predominant source of synchronization error, it is occasionally overlooked in distributed clock synchronization algorithms. As they discuss in s3, inexpensive clock hardware is comparatively unstable compared to traditional PC clock hardware and significant clock drift between synchronization requests exacerbates error.

1.4 IoT Ecosystem:

IoT Devices: IoT devices are available for an increasingly wide variety of applications. IoT devices differ in processing power, network connectivity, and packaging. Specifically, the processing capabilities range from microcontrollers running real-time operating systems to low-power (mobile) processors that are designed to function as gateways to connect to other IoT devices. In addition, IoT devices often use compact System-on-Chip (SoC) construction with a low-power radio chip to enable connectivity via cellular, WiFi, ZigBee, etc. The SoC form factor enables transformation of standard devices (e.g., thermostat) into "smart" devices and the creation of entirely new types of devices such as wearables and voice assistants.

IoT Cloud: Current popular IoT cloud offerings, such as Amazon AWS IoT and Microsoft Azure IoT , are designed to serve billions of IoT device endpoints. IoT devices to communicate with services/applications running in the IoT cloud via protocols including HTTP , WebSockets , AMQP and MQTT . In particular, the widely-deployed MQTT protocol uses a publisher-subscriber model. IoT devices supporting MQTT can function both as publishers and subscribers of data . To coordinate the communication between publishers and subscribers, the MQTT protocol requires a message broker, which runs in the cloud and has the ability to scale with the number of cloud-connected IoT devices.

IoT Time Synchronization: IoT devices typically use SNTP or MQTT to synchronize with reference clocks. For the latter, either the cloud platform pushes a timestamp over MQTT to the device similar to the classic time protocol, or the device obtains a timestamp via a standard REST API to synchronize with reference clocks. In either case, the IoT cloud server or message broker publishes the present time to the IoT device without accounting for forward and reverse OWDs of the packets carrying the timestamp and any response message. As such, any OWD contributes to synchronization inaccuracy at the device. On the other hand, due to resource constraints in IoT devices, sophisticated OWD sample filtering heuristics from the NTP protocol suite are too heavyweight and cannot be employed. Furthermore, as a consequence of resource constraints in these devices, other variants of NTP (e.g., PTP) are generally ill-suited for synchronizing devices' clocks within the IoT ecosystem.

2. EXISTING SYSTEM:

In general, household devices that support both a non-IP environment and an IP network are mixed within a home automation system. Accordingly, general protocols, such as TPSN, RBS, RTSP, and IEEE 1588, I have been used to synchronize time in the home automation system. However, it remains difficult to effectively apply these protocols in wireless and non-IP environments that feature a limited communication interface having low processing performance and low power consumption because these conventional protocols are based on IP network access.

Time synchronization is usually performed by a sequence of ordering sensor data obtained from sensor nodes, distinguishing duplicate events occurring from peripheral sensor nodes, and publishing a sensor node's timestamp and concurrency control sensor nodes in the home automation system alternate between sleep and active modes to be more efficient at power consumption accurate time scheduling can play an important role in saving energy costs by considering the time-varying price of electric power in a home energy management system (HEMS).

The network time protocol (NTP) was proposed in order to provide time synchronization for a larger and static network such as the internet. However, as the number of sensor nodes

increases, the number of messages that are transmitted also greatly increase, resulting in a large amount of power being consumed. The sensor nodes revise the time error and synchronize time, depending on the error factors, through a linear regression analysis of the duplicate messages received.

This method is dependent on the system environment and requires a time delay for the initial time synchronization.

3. PROPOSED SYSTEM

A time synchronization technique is proposed for a MQTT-based home automation system. Many physical interfaces have been proposed for configuring home automation systems, including serial communication, Ethernet, WiFi, etc., which can be classified as IP-based and non-IP-based interfaces. Home applications usually require accurate and concurrent timing information in order to transmit their data in a timely manner. A time synchronization technique was proposed here to support a non-IP environment. The proposed scheme comparatively reduces network overhead because it only uses MQTT protocol.It is the standard for time synchronization protocols. Another advantage is that it does not require an increase in performance, as experimental results indicate that the proposed scheme has a reasonable synchronization error when compared to NTP for existing distributed systems. This pre-processing step is used to implement the MQTT-based time synchronization technique in a non-IP environment. The proposed scheme consists of two steps: a time-sharing process and a network delay compensation process.

The time sharing process is a procedure for updating the time information of the MQTT servers (e.g., household device) by obtaining the time information of the MQTT clients.

4. CONTROLLING HOME LIGHT USING WIFI NODEMCU AND RELAY MODULE:

As we all know that IoT revolution has started from previous couple of years, since then ESP series Wi-Fi modules become very fashionable among hobbyists and industries. It is a very small chip and can be embedded in small devices. Also, it is cost effective and anyone who knows the basics of Arduino programming can program it and use it in their projects to make it an IoT enabled device.

To take you one step ahead towards IoT development, i will be able to make our own web server to host a webpage and control any appliance remotely from anywhere within the world.

In simple terms, Web Server may be a place where we will store the online pages, process them and deliver them to the online clients. Now, What Is Web Client? It is nothing but a Web Browser installed in our mobiles and pcs. A protocol is used to establish and transfer information between Web Client and Server. This protocol is known as Hypertext Transfer Protocol (HTTP).

In this protocol, communication is initiated by making a request for a particular web page using HTTP GET request and the server responds with the content of that web page. If server doesn't respond it'll through a mistake message i.e. 404 Error. Webpages delivered by the server are mostly in HTML coding.

All the websites are hosted on some webserver, mostly Linux based OS is employed on webservers. Any computer can be converted into a webserver, provided that it is connected to the network. We have previously done many webserver projects with different microcontrollers. Raspberry pi have already got inbuilt Wi-Fi module so it doesn’t need the other hardware to show it into a webserver, whereas other microcontroller needs some network connecter (ESP module) for a webserver.

4.1 Station (STA) Mode:

When ESP module connects to an existing Wi-Fi network i.e. network created by your router then this is often a Station mode (STA). In this mode ESP gets IP from wireless router to which it's connected. Using this IP address, ESP will found out an internet server and send the online pages to all or any devices which are connected with the prevailing WiFi network.

4.2 Soft Access Point (AP) Mode:

In this mode, ESP creates its own Wi-Fi network and sets SSID and IP address thereto and acts like Wi-Fi router. Unlike WiFi router, it is not connected to a wired network. Using the generated IP address, it'll send sites to all or any devices which are connected with its own WiFi network. Also, the utmost number of devices that are connected to the ESP is restricted to 5 . This is same like converting it to a Wireless access point.

Figure 4.1 Controlling LED by webpage (OFF state)

Figure 4.2 Controlling LED by webpage (ON state)

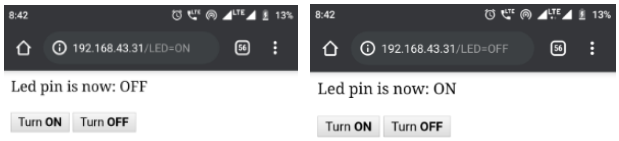

Figure 4.3 Webpage for control LED (ON/OFF state)

5. GETTING CURRENT TIME FROM NTP SERVER:

Every once during a while you'll encounter a thought where keeping time a major concern. For example, imagine a relay that has got to be activated at a particular time or a knowledge logger that has got to store values at precise intervals. The first thing that comes in your mind is to use an RTC (Real Time Clock) chip. But these chips aren't perfectly accurate so, you would like to try to to manual adjustments over and once again to stay them synchronized.

The solution here is to use Network Time Protocol (NTP). If your ESP8266 project has access to the web , you'll get date and time (with a precision within a couple of milliseconds of UTC) for free of charge.

5.1 What is an NTP?:

An NTP stands for Network Time Protocol. It's a typical Internet Protocol (IP) for synchronizing the pc clocks to some reference over a network. The protocol are often wont to synchronize all networked devices to Coordinated Greenwich Mean Time (UTC) within a couple of milliseconds (50 milliseconds over the general public Internet and under 5 milliseconds in a LAN environment). Coordinated Greenwich Mean Time (UTC) may be a world-wide time standard, closely associated with GMT (Greenwich Mean Time). UTC doesn't vary, it's an equivalent worldwide.

NTP sets the clocks of computers to UTC, any civil time zone offset or day light saving time offset is applied by the client. In this manner clients can synchronize to servers no matter location and zone differences.

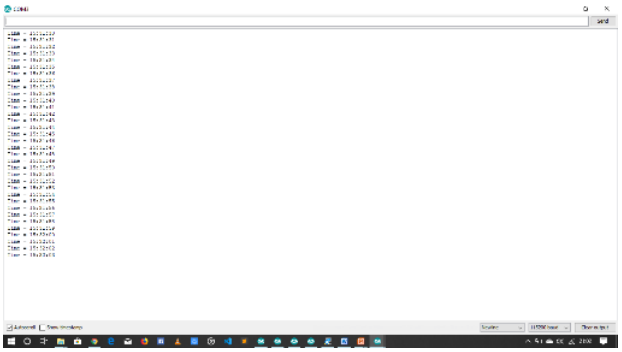

Figure 5.1 NTP server time but differ on current time

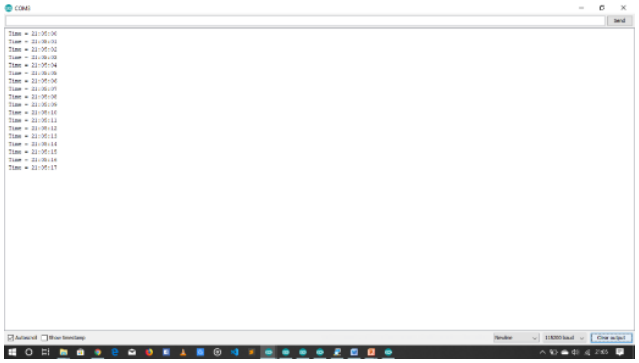

Figure 5.2 NTP server time same as current time

6. TIME SYNCHRONIZATION ON LED LIGHT BY USING NODEMCU TIME ANALYZER

The Hypertext Transfer Protocol (HTTP) is standard application layer protocol which functions as a request response protocol in between server and client. It is widely utilized in IoT (Internet of Things) embedded applications, where every sensor is connected to a server and that we have access to regulate them over the web. NodeMCU has Wi-Fi functionality available on board. With this Wi-Fi functionality NodeMCU can hook up with any wi-fi network as client or it can create a network to which other wi-fi enabled devices can connect.

NodeMCU wi-fi has Access Point (AP) mode through which it can create Wireless LAN to which any wi-fi enabled device can connect. We can set SSID and Password for AP mode which can be wont to authenticate other devices while connecting there. NodeMCU has Station (STA) mode using which it can hook up with existing wi-fi network and may act as HTTP server with IP address assigned by that network. NodeMCU gets IP from Wi-Fi router to which it's connected.

With this IP address, it can act as an HTTP server to which any wi-fi device can connect.

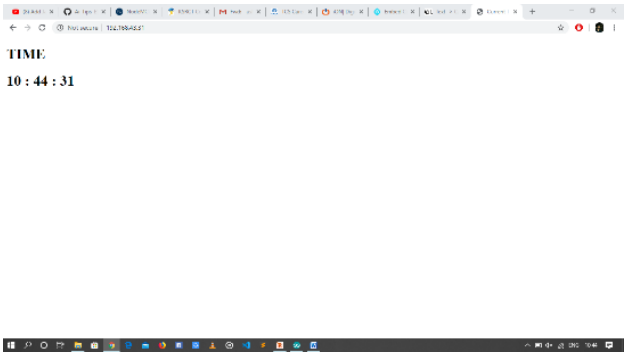

Figure 6.1 NTP current time in webpage

7. MQTT COMMUNICATION BETWEEN SERVER AND CLIENT:

MQTT is lightweight publish-subscribe based messaging protocol. It is quicker (faster) than other request-response based APIs like HTTP. It is developed on the base of TCP/IP protocol. It allows remote location devices to connect, subscribe, publish etc. to a specific topic on the server with the help of message broker. MQTT Broker/Message broker is a module in between the sender and the receiver. It is an element for message validation, transformation and routing. The broker is responsible for distributing messages to the interested clients (subscribed clients) of their interested topic.

8. CONCLUSION & FUTURE SCOPE:

The requirements imposed on the design of the system in this paper have shown that MQTT is an appropriate protocol for connecting low power devices in constrained environments. MQTT provides a simple, flexible architecture that allows for the easy synchronization of device states with minimum overhead, and a selection of open source libraries make its implementation comparatively simple. Analysis of our design has shown that MQTT meets system response time requirements for mildly unstable networks, but will begin to fail as greater packet losses occur. MQTT also successfully delivered all packets in constrained environments, which may be more important depending on the exact requirements of a system. MQTT was relatively efficient, utilizing less than 40% of the CPU during any operations, indicating that it would be appropriate for even lower power devices. These tradeoffs will need to be considered for anyone attempting to design a practical IoT system with synchronized device states.

REFERENCES:

1. A. A. Khan and H. T. Mouftah, (2011) “Web services for indoor energy management in a smart grid environment,” in IEEE 22nd Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC), pp. 1036–1040.
2. A. Zaballos, A. Vallejo, and J. Selga, (2011) “Heterogeneous communication architecture for the smart grid,” IEEE Netw., vol. 25, no. 5, pp. 30–37.
3. B. Becker, A. Kellerer, and H. Schmeck, (2012) “User interaction interface for energy management in smart homes,” in IEEE PES Innov. Smart Grid Technol. (ISGT) Conf., Washington, DC, USA, pp. 1–8.
4. C. Lien, Y. Bai, H. Chen and C. Hung, (2009) “Home Appliance Energy Monitoring and Controlling Based on Power Line Communication”, IEEE ICCE.
5. C. Y. Leong, A. R. Ramli, and T. Perumal, (2003) “A Rule-Based Framework for Hetegeneous Subsystems Management in Smart Home Environment,” IEEE Trans. Consumer Electron., vol. 55, no. 3, pp. 1208-1213.
6. D. –M. Han and J. –H. Lim, (2010) “Smart Home Energy Management System using IEEE 802.15.4 and ZigBee” IEEE Trans. Consumer Electron., vol. 56, no. 3, pp. 1403-1410.
7. D. Tao, P. Wei, H. Jiang and L. Xiyu, (2007) “Adaptive Energy Saving Monitoring Technology for Air Condition in Telecommunication Equipment Room”, pp.155-160, IEEE INTELEC.
8. E. Spanò, L. Niccolini, S.D. Pascoli, and G. Iannaccone, (2015) “Last-Meter Smart Grid Embedded in an Internet-of-Things Platform,” IEEE Trans. Smart Grid, vol. 6, no.1, pp. 468-476.
9. [9] F. Benzi, N. Anglani, E. Bassi, and L. Frosini, (2011) “Electricity smart meters interfacing the households,” IEEE Trans. Ind. Electron., vol. 58, no. 10, pp. 4487–4494.
10. H. C. Jo, S. Kim, S. K. Joo, (2013) “Smart heating and air conditioning scheduling method incorporating customer convenience for home energy management system,” IEEE Trans. Consumer Electron., vol. 59, no. 2, pp. 316-322.
11. I. Choi, J. Lee, and S.-H. Hong, (2011) “Implementation and evaluation of the apparatus for intelligent energy management to apply to the smart grid at home,” in IEEE Instrum. Meas. Technol. Conf. (I2MTC), pp. 1–5.
12. J. Han, C. –S. Choi and I. Lee, (2011) “More Efficient Home Energy Management System Based on ZigBee Communication and Infrared Remote Controls,” IEEE Trans. Consumer Electron., vol. 57, no. 1, pp. 85-89.
13. M. Inoue, T. Higuma, Y. Ito, N. Kushiro and H. Kubota, (2003) “Network Architecture for Home Energy Management System,” IEEE Trans. Consumer Electron., vol. 49, no. 3, pp. 606-613.
14. Q. Hu and F. Li, (2013) “Hardware design of smart home energy management system with dynamic price response,” IEEE Trans. Smart Grid, vol. 4, no. 4, pp. 1878–1887.

15. Seung-Chul Son, Nak-Woo Kim, Byung-Tak Lee, C.H. Cho, J.W. Chong, (2016) ‘‘A Time Synchronization Techniquefor CoAP-based Home Automation Systems,” IEEE Trans. Consumer Electron, vol. 62, no. 1.
16. T. Neagoe, V. Cristea, and L. Banica, (2006) “NTP versus PTP in Computer Networks Clock Synchronization,” in Proc. IEEE International Symposium on Industrial Electronics, Montreal, Canada, vol. 1, pp. 317-362.
17. T. Sauter and M. Lobashov, (2011) “End-to-end communication architecture for smart grids,” IEEE Trans. Ind. Electron., vol. 58, no. 4, pp. 1218–1228.
18. X. Guan, Z. Xu, and Q. S. Jia, (2010) “Energy-efficient buildings facilitated by microgrid,” IEEE Trans. Smart Grid, vol. 1, no. 3, pp. 243-252.
19. Y. S. Son, T. Pulkkinen, K. D. Moon, and C. Kim, (2010) “Home Energy Management System based on Power Line Communication,” IEEE Trans. Consumer Electron., vol. 56, no. 3, pp. 1380-1386.