IoT Embedding with Sensors Data in Clouds

Neven Nikolov[1] and Svetlin Antonov[2]
[1, 2]Faculty of Computer Systems and Technologies at Technical University of Sofia
8 Kl. Ohridski Blvd
Sofia 1000, Bulgaria
{n.nikolov@tu-sofia.bg, tel}{svantonov@yahoo.com}

ABSTRACT: *In this work we studied the formation cloud structures for the communication by using IoT embedded systems. We have used the sensors data and applied it for cloud structure visualization. We used the steps and the nodes that are useful for designing the IoT cloud.*

Keywords: Embedded Systems, IoT, Protocol, Cloud, Node, Red

Received: 1 May 2021, Revised 15 July 2021, Accepted 24 July 2021

DOI: 10.6025/jic/2021/12/4/135-141

Copyright: with Authors

1. Introduction

The Internet of Things IoT are embedded systems that connect to Cloud (Server) by exchanging data between the cloud and other embedded systems [1-8]. Their role is to read sensors and manage different objects, depending on their purpose [9]. IoT are widely used in smart homes, medicine, electricity, industry 4.0 and other spheres. They need to communicate with Cloud Structures, where they publish data from sensors and parameters of managed objects, and these data must be processed and decisions made. The role of the Cloud is to collect data, make a decision based on pledged algorithms, and provide parameters to the embedded systems associated with it. For example, to manage the heating of smart houses, it is necessary to take into account the room temperature, the outside temperature and on the basis of other set parameters to activate the heating in the given room in the house [10]. The main idea is the automation of the whole process as well as the reduction of energy consumption, facilitating the user to monitor and manage his smart home. For this purpose, it is necessary to design the Cloud as intended.

This article describes building a private Cloud for IoT embedded systems, the technology behind Node Red IBM and the embedded NodeMcu Esp8266 system.

2. Experimental IoT Cloud

They have different varieties of the IoT Cloud. Depending on the purpose and type of use, it may be a private and public cloud. The public is convenient if we do not have confidential data and the system is scalable. For example, the number of connected devices increases continuously, resulting in increased server load and more and more computational capability. This is not a

data protection, as well as some platform-dependent features. For the construction of a private cloud, a Dell Power edge R510 server was used, with the Centos 7 operating system installed. For server technology, Node Red, a flow-based development tool developed by IBM, is used to connect together hardware devices, APIs and Online services that are part of the Internet of Things. Node red is an editor working in the browser and is useful for creating JavaScript functions. The application can be stored or shared as well as reused. To run Node Red, you need to install the Node.js - runtime server technology. The project described in Node Red is stored using the JSON type format, both for communication to the server being used and TLS encrypted connection.

For the built-in system that connects to the Cloud realized with Node Red, the NodeMcu Esp8266 embedded system, to which the DHT 22 temperature and humidity sensor is connected, is used. The Cloud Architecture is shown in Fig. 1 a of the embedded system of Figure 2 .

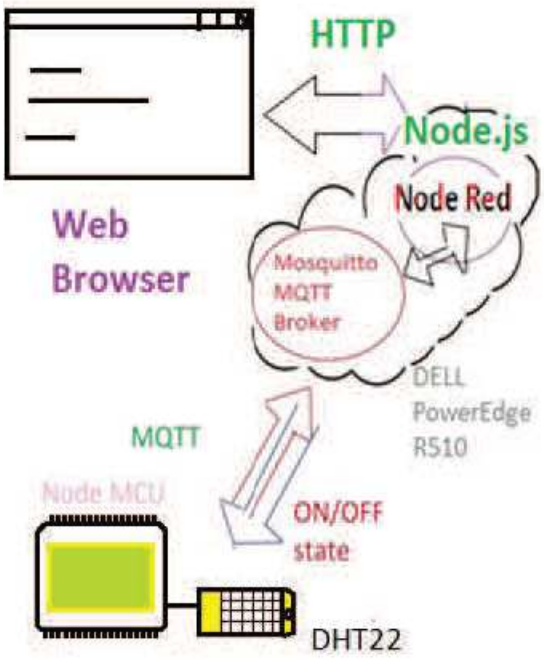

Figure 1. Cloud structure architecture built with Node Red

3. Software Realization and Architecture

With regard to Cloud Building, the following technologies were used - Node Red, Node.js, Mosquitto MQTT broker installed on the Centos 7 operating system on a Dell Power edge R510 server. A web based Node Red environment is used to build the backend. It is invoked by running the server by the command in the terminal:

> *node-red*

after which the system reports - Server now running at http://127.0.0.1:1880. By entering the IP address and port of the Node Red process into the Web browser, the graphical development environment is loaded Figure 3.

The embedded system NodeMcu Esp8266 should connect to MQTT Mosquitto Broker first and after Node Red will receive message. For this purpose, in the Node Red functional module search box, mqtt input nodes are selected for temperature and humidity and they are set to connect to MQTT Mosquitto broker. For this purpose, the following settings are shown in Figure 4, localhost: 1883 which is the address and gateway of the MQTT broker server and the name of Topic - temperature. On Figure 5 are shown Edit switch node configuration for LED control of embedded system and setting the MQTT input nodes configuration for humidity in Figure 6.

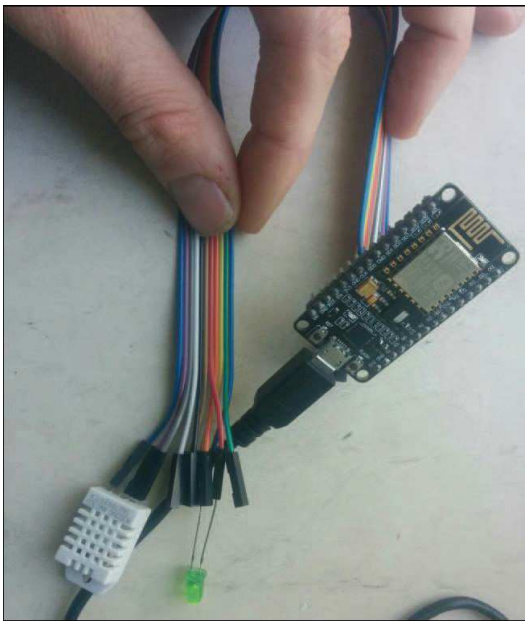

Figure 2. Experimental staging of NodeMcu Eps8266 embedded system

The parameters with humidity settings are the same, only the difference is the Topic - humidity name is set. To the temperature and humidity input nodes are connected a chart node for temperature readings and a gauge node for humidity. To the embedded system as shown in Figure 1 is an LED connected, which will be managed via a Web browser. For this purpose, a switch node in Node Red is used, which is a button with two defined states - On/Off. The MQTT output node is linked to the button and its configuration is a connection to the MQTT broker server and Topic name - switchLed.

Regarding the software implementation of the embedded system Esp8266, an MQTT library is used, and the device send to the MQTT Broker server status of sensor. The embedded system sends the values read from the DHT22 sensor for tempera-

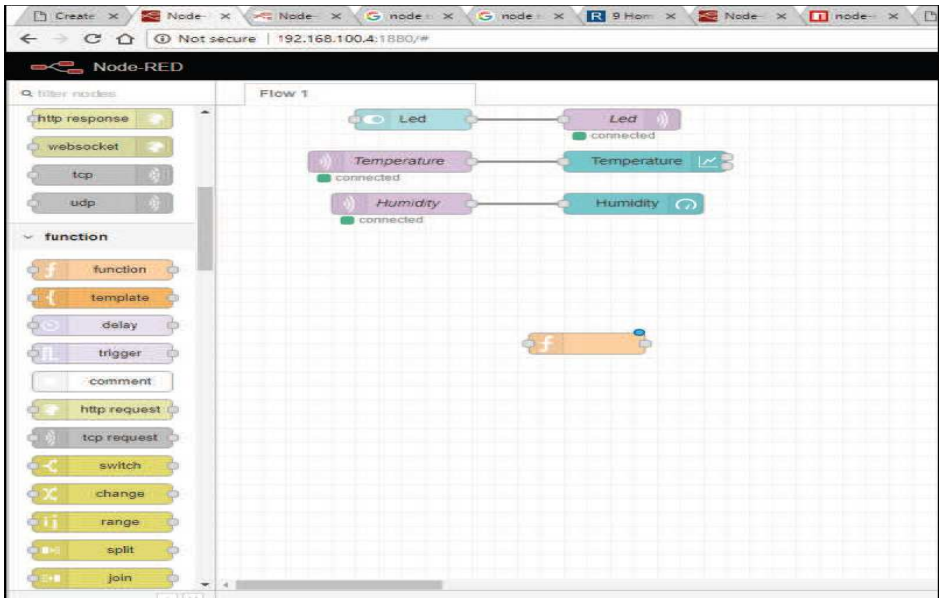

Figure 3. Development environment for Node Red IBM

ture and humidity periodically within 10 seconds to the broker. It monitors continuously for commands from MQTT broker, in the case of activating the LED through the user's web browser Figure 8. Esp8266 works as Publisher / Subscriber mode, because he receive command for turn of and off the LED diode and send data to Cloud.

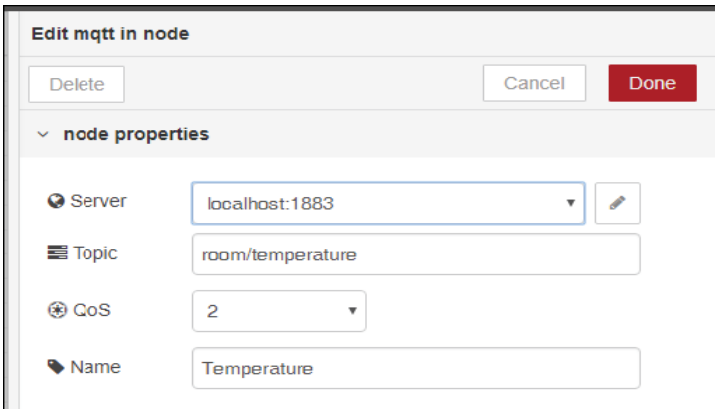

Figure 4. MQTT input node configuration for temperature from embedded system

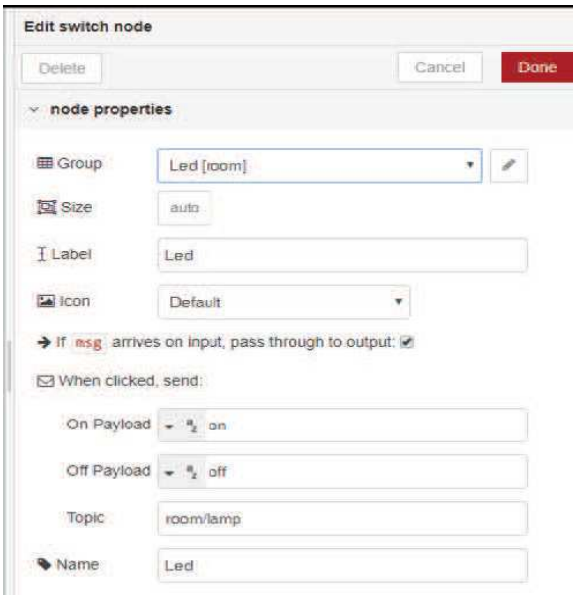

Figure 5. Edit switch node configuration for LED control of embedded system

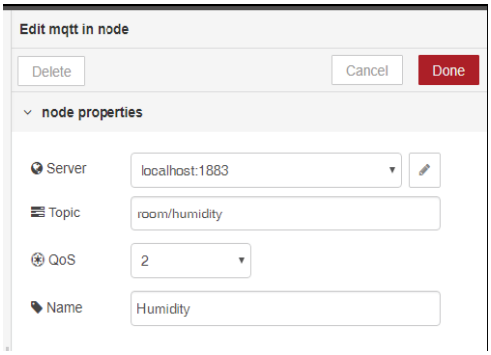

Figure 6. Setting the MQTT input nodes configuration for humidity

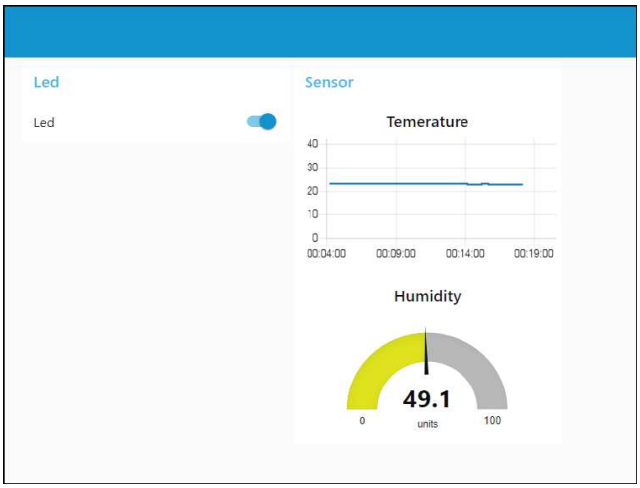

Figure 7. Temperature and humidity reading in Web interface

In Figure 9,10,11 are shown communication between Cloud and IoT embedded device and there are shown MQTT Packet anatomy for message topics. For this measurement are used program tool named WireShark, with whom we can measure the anatomy of packages. With program we can see packet parameters like Frame length, Data length and more parameters. On figures we can see the messages in ASCII format in MQTT protocol, because MQTT used port 1883 to Mosquito Broker and that port isn't encrypt. In table I are shown parameters Frame Length bytes, Data length bytes and Values of sensors.

MQTT Topic	Frame Length Bytes	Data Length Bytes	Value of sensor	Sensor
room/lamp	56	2	off	LED
room/ temperature	80	26	27.36	DHT22
room/humidity	77	23	40.69	DHT22

Table 1. Parameters of MQTT Messages between IoT Device and Cloud

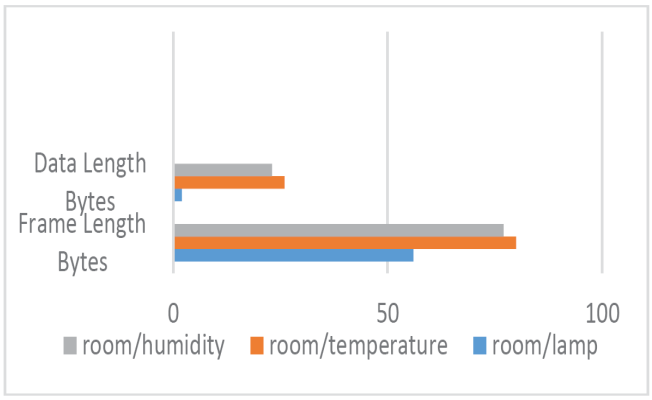

Figure 8. Size of bytes in MQTT messages

In Figure 8 are shown the parameters of MQTT messages between IoT device and Cloud. From table can calculate the header part of MQTT message with Eq 1:

Frame length – Data length = Header MQTT Packet (1)

and header values of MQTT protocol in messages between Cloud and IoT device are shown in Table 2.

MQTT Topic	**Frame Length**	**Data Length**	**Header Frame – Data =**
room/lamp	56	2	54
room/temperature	80	26	54
room/humidity	77	23	54

Table 2. Header Value of MQTT Protocol in Messages between cloud and IoT Device

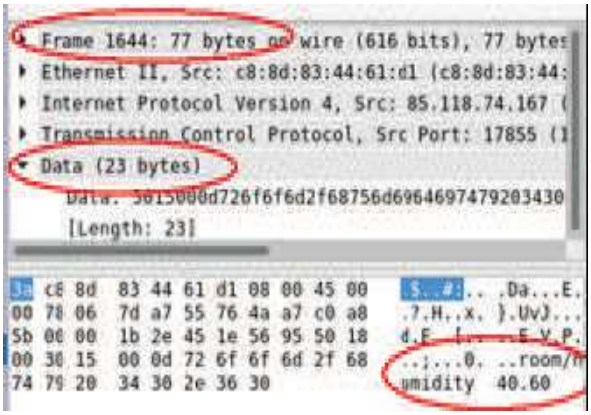

Figure 9. MQTT packet anatomy for message topic “room/humidity”

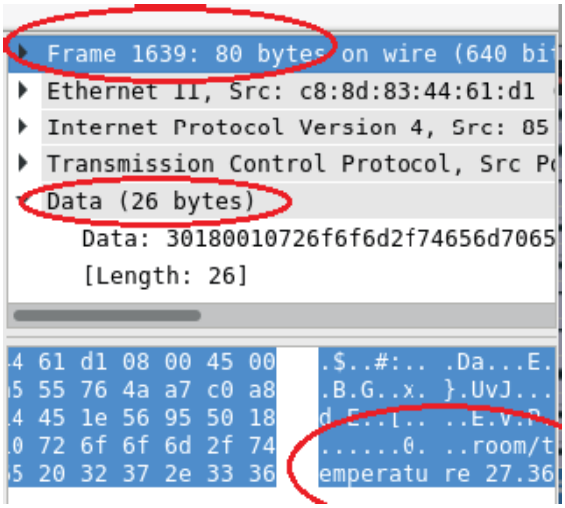

Figure 10. MQTT Packet anatomy for message topic “room/temperature”

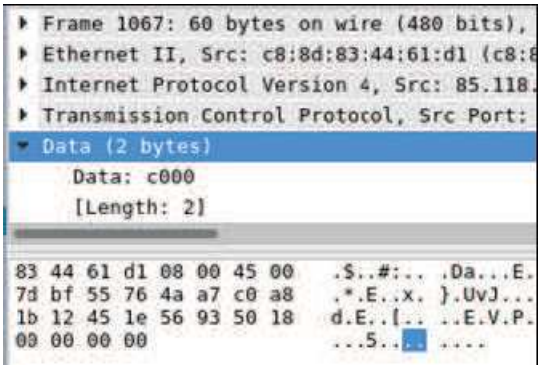

Figure 11. MQTT Packet anatomy for message topic “room/lamp”

5. Conclusion

In this article is described how is released Cloud system for controlling an LED diode and reading data from temperature and humidity sensor. The management and control are realized via Web interface. It is achieved a real-time measurement of parameters such as temperature and humidity.

A concept for building a IoT Cloud is shown using Node Red technology. In section IV are measured the protocol between Cloud and IoT embedded system. In Table 2 are shown packet payload Data and Frame Length. From all these studies, it see the MQTT protocol parameters for all messages. The system has big advantage over analog systems – Cloud or IoT devices can be anywhere and far away without losing any information.

References

[1] Weinman, J. (2015). The Strategic Value of the Cloud, *IEEE Cloud Computing*, volume 2, 66-70.

[2] Peng Xu, W., Yang, L. T. (2018). *Secure Data Collection, Storage and Access in Cloud-Assisted IoT*, *IEEE Cloud Computing*.

[3] Alshehri, A., Sandhu, R.(2017). Access Control Models for Virtual Object Communication in Cloud-Enabled IoT, *IEEE Computer society*, 16-25.

[4] Truong, H., Dustdar, S.(2015). *Principles for Engineering IoT Cloud Systems*, 2, 68-76.

[5] Nastic, S., Truong, H., Dustdar, S. (2017). A programming model for resource-constrained iot cloud edge devices, *Banff, IEEE international Conference on systems*, 2017

[6] Lingg, E., Leone, G., Spaulding, K., Far, R. B. (2014). Cloud based employee health and wellness integrated wellness application with a wearable device and the HCM data store, *Internet of Things* (WF-IoT), Seoul.

[7] Fujii, N., Koike, N. (2017). *A FPGA-based Remote Laboratory in the Hybrid Cloud*, Chester, Cyberworlds.

[8] Merlino, G., Bruneo, D., Longo, F., Puliafito, A., Distedano, S. (2015). A Novel Paradigm for Smart Cities through IoT Clouds, IEEE press.

[9] Lekic, M., Garadasevic, G. (2018). IoT sensor integration to Node-RED platform, East Sarajevo, Infoteh-Jahorina, 2018

[10] Obuchi, Y., Yamasaki, T., Aizawa, K., Toriumi, S., Hayashi, M. (2018). Measurement and evapuation of comfort levels of apartments using IoT sensors, Las Vegas, *Consumer Electronics (ICCE)*.