A Project Report on

Implementing a OneM2M based IoT platform

For the course

Internet of Things (EEE F411)

Submitted by

Dikshit Gautam bearing ID No.: 2018A8PS0816P

And

Sunil Saharan bearing ID No.: 2018B3A30957P

Submitted to

Dr. Vinay Chamola
Assistant Professor in
Department of Electrical and Electronics Engineering
BITS Pilani
Pilani Campus

Contents

1 Introduction

The task of IoT device communication and management has always been challenging due to heterogeneity of things in terms of communication, types of data generated, access control for clients, accessing and erasing information, etc. To solve these kinds of issues and interoperate with other Machine to Machine systems, the “oneM2M global initiative” was established.

Its aim is to develop standards and specifications to enable the machine to machine communications market to fully develop and take off. The oneM2M aims to develop standards to support machine to machine communication service providers, application developers, and device manufacturers to help develop a larger market and ecosystem for their products and so grow the oneM2M market.

Our objective is to deploy a server using oneM2M standard which can be used by application developers as a middleware to connect with lower level devices such as sensors, actuators and exchange data with those devices.

In this paper we’re presenting all the different tools, configurations and methods we have used to deploy that server and make the communication happen.

2 Literature

2.1 M2M Communication

M2M (Machine to Machine) communication enables the devices on a common network to exchange information. Generally M2M systems take the sensor data from one node and transmit it to the network so that other devices can access it. Control commands can also be given to a node device through M2M technology. It often uses Ethernet or cellular networks. One good example of M2M communication is an e-health system[i] in which patient's medical data like temperature, pulse rate, sugar level etc. is sent to the monitoring device in real time through M2M technology. Many control commands can also be given to different equipment.

We should take care of some important things before deciding which M2M technology to use. For example, an M2M system should be scalable and it should work efficiently if the number of devices connected is increased or decreased. It should also be less power consuming so that the devices connected can work for a long time as battery replacement is not easily possible in many cases. A good M2M system is also a secure system so that no unauthorized access to sensor data and control over devices is possible.

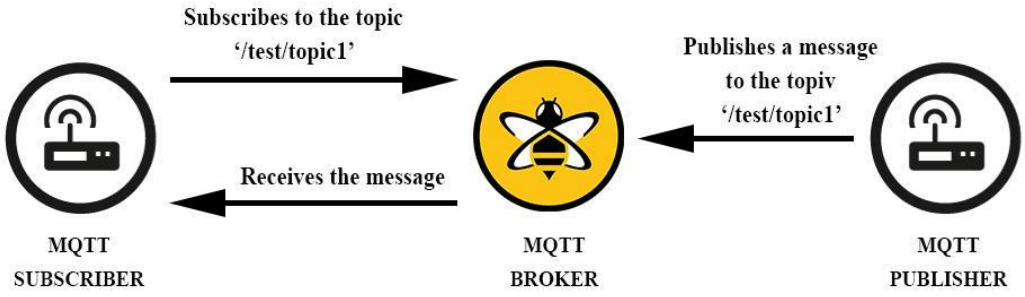
Many M2M standards have been developed over the years. Some of them include OMA LightweightM2M, oneM2M, HyperCat etc.

2.2 MQTT and OneM2M standards

MQTT (Message Queue Telemetry Transport) is a lightweight network protocol which is used by devices to communicate with each other. It was developed by Dr. Andy Stanford-Clark and Dr. Arlen Nipper in 1999.

This protocol is based on a publish-subscribe architecture in comparison to HTTP which uses a request-response architecture. It is a low power consuming protocol and that's why it is suitable for M2M communication in IoT systems. It is having an event-driven architecture and it pushes the messages to its clients whenever it receives any message.

The principle of publish-subscribe architecture is entities which are interested in certain information 'subscribe' to a specific topic and they are called as subscribers. While entities which want to send some information to interested entities, which are called publishers, 'publish' a message to that topic. All this is possible with the help of a software called MQTT broker which is responsible for forwarding the messages from the senders to their true receivers



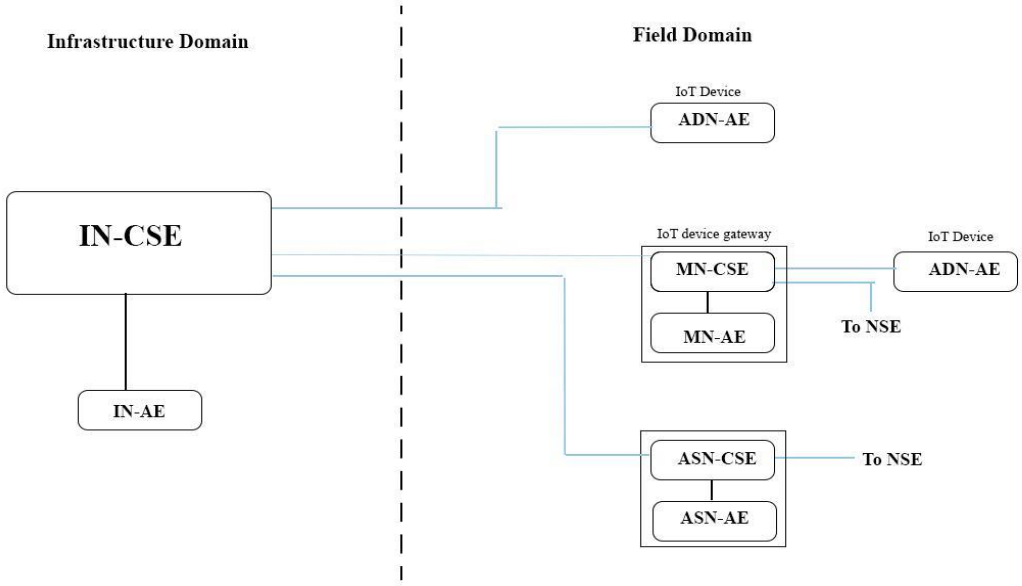
.

OneM2M [ii]is the standards for M2M and IoT. It is a worldwide partnership project which aims on providing a common service layer which is used to connect different M2M devices with application servers worldwide. In this way it enables interoperability for the complete IoT ecosystem.

OneM2M uses MQTT, REST and CoAP as its communication protocol. The request and response messages in oneM2M are serialized using JSON

OneM2M[iii] supports resource based architecture. The oneM2M framework is shaped by various entities called nodes. Its components are divided into two domains. One is field domain and the other is infrastructure domain. Field domain contains Application Service Nodes (ASNs), Middle

Nodes (MNs) and Application Dedicated Nodes (ADNs) and Infrastructure domain contains Infrastructure Nodes (INs). ASN, MN, AND node must contain either Common Service Entities (CSEs) or Application Entities (AEs). CSEs provide Common Service Functions such as data management, resource management, event management etc. While AEs provide Application Service Logics such as sensing and actuation. Network Service Entities are used to provide networking services. These entities reside in the network service layer.



Since it uses a resource based architecture so its every service is denoted by a resource. Every resource is having an address called Uniform Resource Identifier (URI). An *AE* resource is created under the *CSE* resource of the system. An AE must be registered with only one CSE but a *CSE* resource can have many AEs registered under it. An ADN-AE is the IoT device in the field. Sensors and actuators are connected to it directly or indirectly through the device gateway. ADN-AE communicates with the CSE using REST API, CoAP and MQTT . An *AE* resource has many *container* resources under it. Each *container* resource has a *cin* resource under it. The data is stored as an attribute of a *cin* resource.

An IN-AE can be an android application which can view and modify the server data and also can give control commands using some operations such as CREATE, RETRIEVE, UPDATE and DELETE operations. The communication between IN-CSE (IoT server) and IN-AE takes place over HTTP or HTTPS using some REST APIs[iv].

2.3 OCEAN (Open allianCE for iot stANdard)

OCEAN[v] is an open source-based global partnership for implementing oneM2M standards and is developed by KETI. There are many IoT platforms available for use[vi]. OCEAN is one of them. OCEAN has two platforms: one is Mobius[vii] server and the other is &Cube[viii].

The Mobius[ix] server acts like IN-CSE in an IoT system. It is a Node.js based software. Mobius server is like a bridge between various devices and applications so that they can communicate with each other. It has parts such as MySQL database to store the data, MQTT broker and proxy to communicate with the gateway and REST server to communicate with the applications over HTTP/HTTPS. For changing and retrieving data, Mobius offers a suite of REST APIs. Using these APIs through the API development platform Postman or any other platform, one can easily retrieve, change or delete the data.

&Cube is a suite of many Field domain softwares which act as device platforms. Many &Cube softwares are available such as &Cube: Rosemary, &Cube: Thyme, &Cube: Lavender etc. In this project we have used &Cube Thyme as an IoT gateway which acts as an MN-AE. &Cube Thyme is Node.js based software. It communicates with the Mobius server through the MQTT broker and MQTT proxy by sending serialized data using MQTT protocol and oneM2M standards.

3 Methodology

In this project we are using HC-SR04 Ultrasonic sensor to test the server. This sensor will send the distance data to the IoT server and we can access the data by REST APIs through HTTP for further processing. We are also using a servo motor as an actuator to test the server.

We have deployed a Mobius server as a CSE on an Ubuntu machine. Eclipse mosquitto has been used as an MQTT broker[x]. We have deployed &Cube Thyme on raspberry pi 3 Model B+ as an AE. Various tools and resource tree structure used for the project are as follows:

Mobius Server

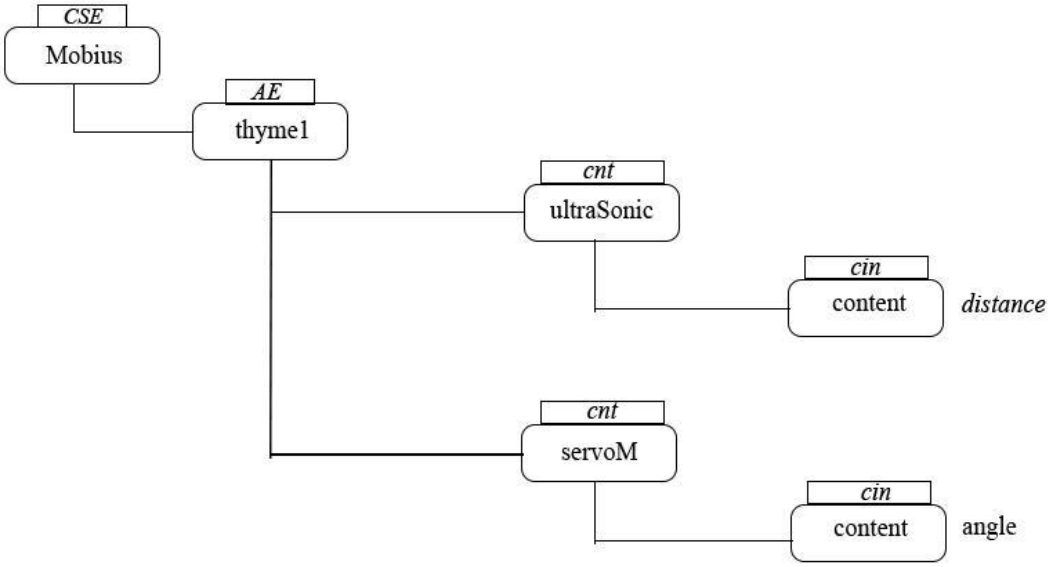
Mobius server version 2.4.36 has been used. MySQL version 5.7.29 is used for storing and handling data. Mosquitto MQTT broker version 1.6.9 has been used for MQTT communication between the CSE and AE. Its CSE name and CSE ID are 'Mobius' and '/Mobius2' respectively by default. OCEAN has provided a suite of REST APIs for accessing and modifying the Mobius server data such as creating or deleting its resources, viewing and deleting the data sent from sensors, giving control commands to the actuators. These APIs can easily be tested using the Postman platform.

&Cube Thyme

&Cube Thyme[xi] version 2.2.0 has been used and it has been deployed on raspberry pi. It is used as an IoT device gateway which connects IoT devices to the IoT server, i.e., Mobius server. Thyme communicates with Mobius server through mosquitto broker running on port 1883 of Mobius server. It sends the serialized data to Mobius for sending requests or responses.

Since it is an AE so must be registered under some CSE. To do that it sends a register request called as reg_req to the Mobius server and gets its AE name and AE ID. In this project we have named this AE resource as 'thyme1'. Two *cnt* resources have also been created having names 'ultraSonic' and 'servoM'. The sensor data is stored as the value of the content attribute of the *cin* resource created under the *resource* ultraSonic and command to servo motor is given as content attribute of servoM resource.

The sensor or actuator can either be directly connected to the raspberry pi through its GPIO pins or through an end node device such as Arduino UNO or NodeMCU or any other devices connected to the raspberry pi using WiFi or Ethernet. We have used NodeMCU as an end node device which is connected to raspberry pi using WiFi. And Actuators and sensors are directly connected to the NodeMCU.



TAS (Things Adaptation software)

TAS is a small middleware program. The TAS program takes the sensor data, converts it into oneM2M defined format and then sends it to Thyme server running on port 3105 for further routing. It can also take actuation commands from the Thyme and convert them into the actuator accepted command format. In this project we have developed two TASs, one for controlling servo motor and the other for sending ultrasonic sensor data to Thyme.

oneM2MBrowser

OneM2MBrowser[xii] is a resource monitoring tool for Mobius developed by OCEAN. It is used to easily monitor various CSEs or AEs connected in the IoT system and also to get the real time data through *cin* resource. As this software is just for making monitoring easy and giving it a graphical user interface so using this software is optional and can be skipped.

Note that the ADN-AEs (IoT devices) can also be directly connected to IN-CSE (Mobius server) through REST APIs, MQTT and CoAP without using MN-AE (Thyme) but using Thyme makes the communication process very simple and easy.

4 Results

We were able to control the servo motor connected to the raspberry pi through WiFi using NodeMCU by sending POST requests to change the the content attribute of *cin* resource under *servoM* resource via Postman using OCEAN provided REST APIs. The servo motor takes commands from the gateway via a middle TAS program.

One TAS program was also written for providing ultrasonic sensor's data to the gateway. We were also able to retrieve and change data by sending HTTP requests by accessing the *cin* resource under the *ultraSonic* resource using the REST APIs. Hence the IoT communication system has been successfully tested and is working as desired.

These two examples of an actuator and a sensor can be used as an example to design other IoT systems which use some other sensors and actuators. Only the TAS programs have to be modified for those cases.

References

[i] Zhong Fan and S. Tan, "M2M communications for e-health: Standards, enabling technologies, and research challenges," 2012 6th International Symposium on Medical Information and Communication Technology (ISMICT), La Jolla, CA, 2012, pp. 1-4.

[ii] J. Swetina, G. Lu, P. Jacobs, F. Ennesser and J. Song, "Toward a standardized common M2M service layer platform: Introduction to oneM2M," in *IEEE Wireless Communications*, vol. 21, no. 3, pp. 20-26, June 2014.

[iii] C. Gezer and E. Taşkın, "An overview of oneM2M standard," 2016 24th Signal Processing and Communication Application Conference (SIU), Zonguldak, 2016, pp. 1705-1708.

[iv] https://github.com/IoTKETI/oneM2M-API-Testing

[v] - http://www.iotocean.org/main

[vi] Lucero, Sam. "IoT platforms: enabling the Internet of Things." *White paper* (2016).

[vii] http://developers.iotocean.org/archives/module/mobius

[viii] http://www.iotocean.org/download/?tab1=2

[ix] Ahn, Il-Yeup, et al. "Development of an oneM2M-compliant IoT Platform for Wearable Data Collection." *TIIS* 13.1 (2019): 1-15.

[x] Light, (2017), Mosquitto: server and client implementation of the MQTT protocol, Journal of Open Source Software, 2(13), 265, doi:10.21105/joss.00265

[xi] http://developers.iotocean.org/archives/module/ncube-thyme-nodejs

[xii] http://developers.iotocean.org/archives/module/onem2mbrower