Developement of Horizontal IoT Platform using DeviceHive Framework

Thakor Jay Chandrasinh
VLSI & Embedded System Design
GTU PG School
Ahmedabad, India
thakor.jay@gmail.com

Mr. Chaitannya Mahatme
Zeroes & Ones Technologies.
Pune,India
info@znotech.com

Abstract—This paper presents the Horizontal IoT(Internet of Things) Platform for IoT application development. This paper mainly describes the limitations of Vertical Approach for different domain applications and also presents generalized solution for limitations as Horizontal IoT Platform. In the context of IoT/M2M domain, devices perform the automated tasks data sensing, generation, logging and reporting to service. For this, a generic solution unifying the infrastructure and platforms that access and control different Vertical Domains leads to the *Horizontal Platform*. An Open Source DeviceHive Framework is used for developing this generic solution for building different IoT application.

Keywords—*Horizontal IoT Platform, M2M, DeviceHive Framework, Gateway.*

*****

I. INTRODUCTION

IoT is revolution in Internet Technology. IoT is the emerging technology and is very important one in next few years. As a global prediction by 2020, 50 billion objects are connected over the internet and communicating smartly over it[1]. IoT in its culmination, where we lived in data is defined as:" *The IoT creates an intelligent, invisible network fabric that can be sensed, controlled and programmed. IoT-enabled products employ embedded technology that allows them to communicate, directly or indirectly, with each other or the Internet*.[2]" Now using this connected devices called as Wireless Sensor Networks many applications are developed in various domains, like Smart Home, Smart Health, Smart Shopping and Smart Transportation to improve life. But the main problem of the application is the interoperability of various wireless protocol over which the sensor devices communicate. Such applications are built using the Vertical approach, so they can't be easily shifted in any other domain.

Such problem of interoperability is main concern of the recent development of IoT application. Which requires standardized "Protocol Layer" using which any device or protocol can communicate smoothly. Such a Protocol Layer is called "Service Capability Layer" between Transport Layer and Application Layer which is the main objective of this paper.

Here, as discussed the "Service Capability Layer"(SCL) is the key enabler for the Horizontal IoT Platform. This will help to develop generic solution for the IoT application across domains. For devices to communicate smartly with each other ,SCL requires certain features are: i) technologies that enable “things” to acquire contextual information,(Data Sensing and Acquisition) ii) technologies that enable “things” to process contextual information,(Data Analytics) and iii) technologies to improve security and privacy(Data Security).

II. IOT DOMAINS:FROM VERTICAL TOWARDS HORIZONTAL

Recently IoT applications are used in different domains as per the specific environment and needs. For this application up till now the available models are i.e., FP7 projects IoT-I [3] and IoT-A [4]. But these models specifically used for the vertical approach only. There is no availability of any standardized model for Horizontal approach. Since IoT is being percolated in every vertical there is to unify the infrastructure and architecture for a generic solution. A paradigm shift in M2M/IoT space from Vertical to Horizontal domains is being speculated for this purpose. There is very thin line between M2M and IoT as in every device will have specific IP address which makes them to communicate directly with server where as in M2M localized devices communicate with and aggregator device called as Gateway which in turn communicates with server over IP protocol. In recent years several standardization activities towards a horizontal service layer approach have been started by standardization organizations (SDO) world-wide. Here the activities at TIA in the TIA-50[5] group (M2M Smart Device Communication) in the USA, CCSA TC10 in China and the activities in the ETSI TC M2M[6] group in Europe should be explicitly mentioned.

ONEM2M[7] is other example of the M2M communication Framework. An open world-wide M2M service layer standard, based on the future ONEM2M standard, will open up the possibility for a broad range of companies and players to enter the market with different sets of possible business models.

A. M2M Standardization

As vertical proprietary M2M solutions provides some value to their vertical sectors, most challenges are identified in scalability and interoperability[8].Thus standardization of M2M communication is done by surveying different vertical domains and collect common ontology from it moving to a proper horizontal solution. As shown in figure 1 the industries are moving from Vertical towards Horizontal[9].

As shown in Figure 1 Specific algorithm is used for a business application in vertical domain whereas a standard protocol layer i.e., "Service Capability Layer", shown as Common Application Infrastructure is being used to build the application across verticals, This "Common Application Infrastructure" is transforming underlying framework from Vertical to Horizontal platform. There are many different business applications as well as many devices are connected to local network or gateways using different protocols. Mostly Restful APIs are used for development of this kind of Protocol Layer. As shown in Figure 1 the layer is not bounded between Application and Transport Layer. But it's some functionality is laid in between gateways and devices. So the Services given by the common layer is distributed in all the components of proposed architecture.

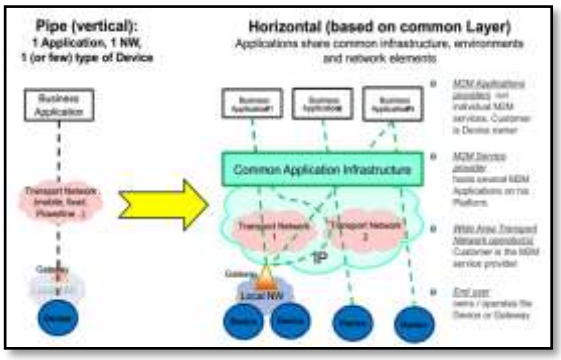

Figure 1 Moving From Vertical Towards Horizontal (Source:ONEM2M)

B. ETSI TC M2M Architecture

International standard developing organizations, such as the European Telecommunication Standard Institute (ETSI) have set their effort in the definition of a standard for a high level service layer platform for M2M. For the elaboration of such a standard, ETSI collected requirements from several important vertical markets of IoT/M2M. ETSI TC M2M was created in January 2009 in order to standardize specifications for developing and maintaining an end-to-end architecture for M2M systems. ETSI TC M2M has standardized various common features such as device management, discovery, security, location, etc. These common features can be used in various verticals. Such as Connected Vehicle, Smart Metering and e-Health.

As shown in Figure 2 the ETSI TC M2M Architecture has three different domains namely: i) Application ii) Network and iii) Device. The Network Domain comprises a collection of service capabilities in addition to core and access networks. The Device Domain contains sensors, actuators and other devices typically found in a wireless sensor network. ETSI also defines an M2M Gateway which also holds services that cannot be directly used by low level tiny devices like microcontroller. The Application Domain includes M2M Applications and Client Applications.

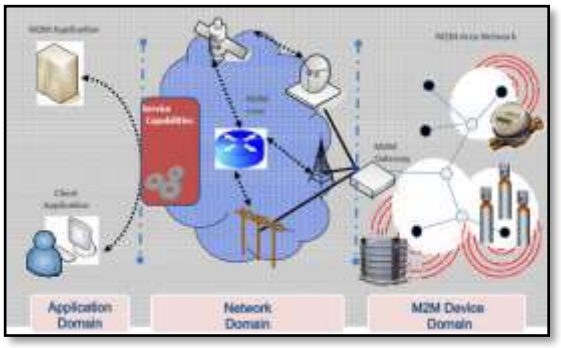

Figure 2 ETSI TC M2M Architecture (Source: ETSI)

III. PROPOSED SYSTEM ARCHITECTURE

The proposed system is divided into Hardware and Software Models. Figure 3 shows hardware components used in proposed system architecture. Here the Gateway is the most important component of the architecture. Raspberry Pi B+ module is being used as a gateway. And it works as the middleware between the devices/nodes and Server It is similar to ETSI M2M model. But here we are using Device Hive Framework for building the Service Capability Layer in the proposed architecture. Most of the services of common layer are provided in Gateway only and others are implemented on the device as well as on server side.

The comparative study is conducted among the available OneM2M standards and device hive as SCL module.

Table 1 Comparison between ONEM2M and DeviceHive Standards

Functional Entity	ETSI M2M	DeviceHive
M2M Service Capability hosted in the network domain	Yes, *Network Service Capability Layer (NSCL)*	Yes Device Hive API
M2M Service Capability hosted on an intermediary node	Yes, *Gateway Service Capability Layer (GSCL)*	Yes Gateway
M2M Service Capability hosted on an M2M Device	Yes, *Device Service Capability Layer (DSCL)*	Yes Device Base Class Device Host Class
Applications in the network domain	Yes, *Network Applications (NA)*	Yes Application Layer Client App.
Applications in the M2M Device	Yes, *Device Application (DA)*	Yes
M2M Device	Yes, *Device with/without Service Capabilities (D/D')*	Yes Device with OS Device w/o OS
AAA Server	Yes, *M2M Authentication Server (MAS), M2M Service Bootstrap function (MSBF)*	Two type of Authentication. i)Device ii)User(HTTP)

A. Hardware Model

The model mainly consist the following components as show in Figure 3.

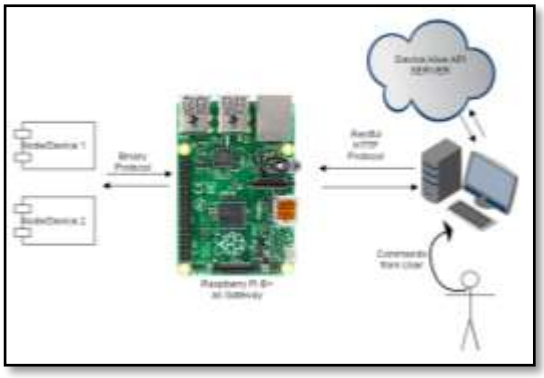

Figure 3 Proposed Architecture

System Architecture Components:-

(i) Different nodes (sensors, actuators, etc.)

(ii) Gateway(H/W:Raspberry pi b+, S/W:Device Hive Framework)

(iii) Server (Device Hive API)

(iv) Client Application (PC)

Here we are developing a generic solution so the device nodes can be of any vertical domains for instance Home Automation has different sensors like temperature, PIR,

 ISSN: 2321-8169
2574 - 2578

pressure, etc. In case of Smart Transport it could be any GPS module. A server for the proposed architecture is designed as mentioned in DeviceHive Framework[10] which has web based interface.

B. Software Model

Device Hive Framework is used as a software framework for building the IoT Horizontal Platform on a Gateway i.e., Raspberry Pi B+. Device Hive is a C++ based Framework skeleton used for building software to the Hardware Model. The most important part of software is Gateway Configuration and Device(Gateway, Sensors) Registration with server seamless connection between the Server and Devices. for data communication. For this the flow diagram is shown Figure 4.

We are using two types of protocols for communication among system components.

1) Binary Protocol:-

The DeviceHive Binary Protocol is used for transferring messages between various low-level devices and the gateway. In the DeviceHive architecture, the gateway acts as a proxy between devices and the DeviceHive cloud and converts binary messages to REST API service calls and vice versa. Devices connected to the gateway are considered to be very simple equipment which do not implement TCP/IP stack on their own and have no direct access to the Internet. To route messages from/to the cloud, these devices connect to the gateway using various wired as well as wireless interfaces such as RS-232,RS-485, SPI or ZigBee, and use the Binary Protocol to communicate about various events in the system.

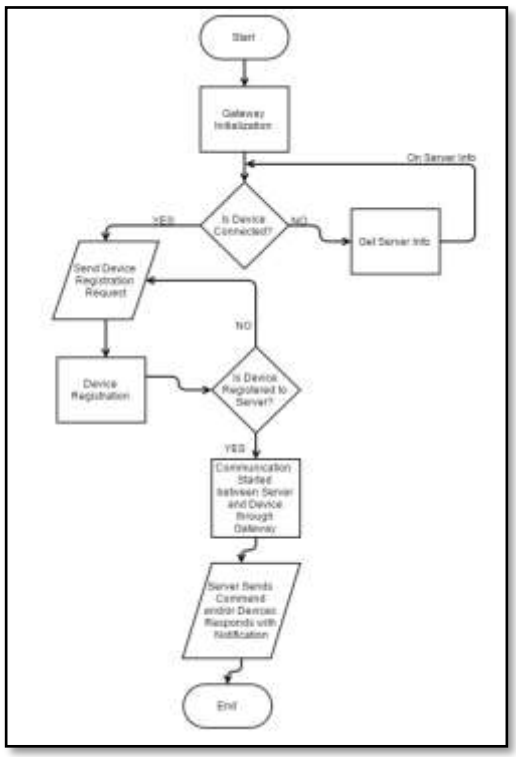

Figure 4 Flow Diagram

2) Restfull Protocol:-

The DeviceHive API (Restful API) is the central part of the framework which allows different components to interact with each other. The API provides access to information about registered components in the system, and allows them to exchange messages in real time. Three types of users are there which are using the DeviceHive API as shown in below Figure 5.

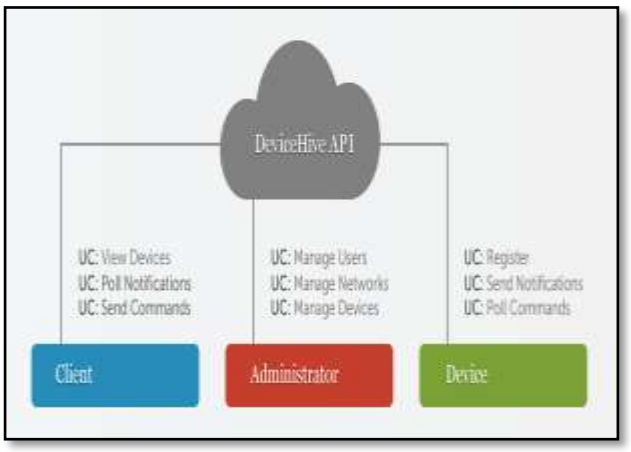

Figure 5 DeviceHive API use cases (Source:DeviceHive)

C. Modules of Implementation

- Study the Device Hive Framework for the Implementation of the proposed Solution.
- Building of Boost Libraries[11] . Which mainly contains boost.asio , boost.smart_ptr , boost.bind
- Build the boost for Linux using the Cross Compile Toolchain build+ using crosstool-ng tool.
- Gateway Compilation and Configuration
- Connection of Gateway to Server as well as Devices Connected to it.
- Implement the common service provided by Service Layer Capabilities.

IV. COMPLITION OF COMMUNICATION LINK

The flow starts with the initialization of the gateway as shown in the figure 4. The gateway is Raspberry Pi B+ module. The device hive framework is used for the initialization and connection to the server as well as devices. First of all the gateway is activated. For this the gateway must be configured as per the requirement. For gateway configuration some command line arguments have to be passed such as --serial port --server server link --no ws. After the gateway is configured and made up & running, it polls the device if it connected to it or not. If the device is connected to it then it send request to the device to register itself. If the device didn't respond to request before timeout expired ,then the gateway is connected to server and gets the server information. On getting information of the server the gateway again send request to device for registration. In the registration request the following bit stream is passed from gateway.

ISSN: 2321-8169
2574 - 2578

Gateway converts it to JSON[12] value to binary format in a frame.

Signature		Version	Flags	Length	Intent	Data	Checksum
0xC5	0xC3	0x01	0x00	0x00	0x00	0x0000	0x76

As shown in the bit stream all fields have significance as mentioned in Table 2:-

Table 2 Binary Message Format

Field	**Significance**
Signature	A sequence of two bytes identifies the start of new message.
Version	Current Protocol Version. Should be 0x01.
Flags	Reserved for future use. Should be 0.
Length	Length of Data Block in bytes
Intent	Identifies the message intent.
Data	Data Block
Checksum	Represent a check byte.

During first registration request from gateway to device, gateway, in mean time, initiates the connection with server and gather information. Upon the first time device registration failure, gateway re-polls the device for registration. Registration Data is the most important data for the gateway to communicate the proper way with device. So it's very important that the data is send to the gateway in proper format only. After the Device is Registered Successfully on the Server. The device and Server communicate in both the way. The device notifies the server on the connection and also responds with the Updated State every time it got the commands from the Server Side.

V. RESULTS

As mentioned in the Proposed System the components are connected as shown in Figure 6.

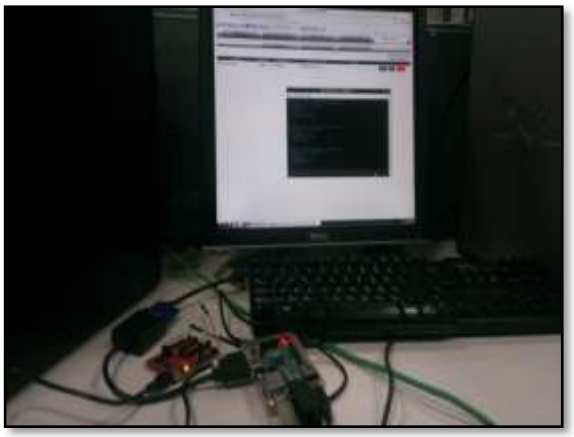

Figure 6 Component Setup

A. Gateway and Server Connection

The screen shot of the result below in Figure 7 is taken on the Raspberry Pi B+ module. It is showing the gateway is connected to server using restful protocol. And also showing the server information on gateway terminal.

```
pi@raspberrypi ~/Desktop $ ./simple_gw --serial /dev/ttyUSB0 --buadrate 115200
--server http://nn7596.pg.devicehive.com/api --no-ws
/Application: RESTful service is used
/devicehive/rest: getting server info
/Application: got stream device OPEN
/Application: sending frame #0 data: {"data":null}
/Application: process received frame #0 data: {"data":null}
/Application: invalid intent: 0, ignored
/devicehive/rest: got "server info" response: {
  apiVersion: "1.3.0",
  serverTimestamp: "2015-05-06T10:22:43.550000",
  webSocketServerUrl: "ws:\/\/nn7596.pg.devicehive.com:8010"
}
```

Figure 7 Server Information on Gateway

B. Gateway and Device Connection

The result below is the Device information sent to the gateway on the response of the request from the gateway. The device sends the device information in a particular registration data structure using binary protocols to gateway.The device also sends notification to server on connection as well as after every command is executed on the device is shown in the Figure 8.

```
/Application: process received frame #258 data: {"parameters":{"equipment":"TS_1
","local":32767,"temprature":35,"time":"Mon Jan  1 00:04:08 1900\n"}}
/Application: process received frame #256 data: {"parameters":{"equipment":"BTN"
,"state":true}}
/Application: process received frame #256 data: {"parameters":{"equipment":"LED_
R","state":false}}
/Application: process received frame #256 data: {"parameters":{"equipment":"LED_
G","state":false}}
/Application: sending 4 pending notifications
/devicehive/rest: inserting notification: {
  notification: "sensordata",
  parameters: {
    equipment: "TS_1",
    local: 32767,
    temprature: 35,
    time: "Mon Jan  1 00:04:08 1900\n"
  }
}
/devicehive/rest: inserting notification: {
  notification: "equipment",
  parameters: {
    equipment: "BTN",
    state: true
  }
}
```

Figure 8 Device Notification on Gateway

C. Device and Server Connection via Gateway

These results are taken on server side. Here when the device is connected to gateway it sends registration data, and the gateway sends it to server using which only device and sever communication is occurred. This is shown in Figure 9. The same way the commands from server is passing to the device and in response to it the commands the device is ending

ISSN: 2321-8169
2574 - 2578

notification to the server via gateway only is shown in Figure 10 and 11 respectively.

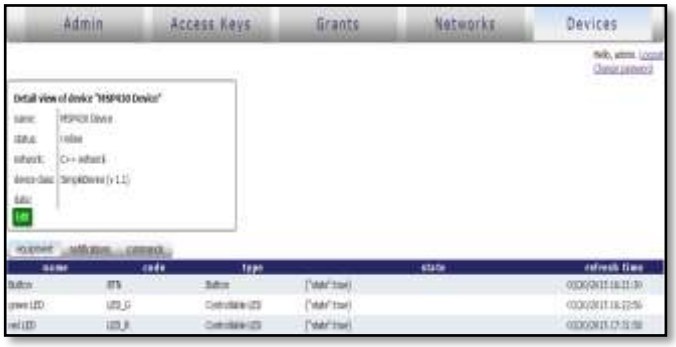

Figure 9 Device Registration on Server

Figure 10 Device Notification on Server

Figure 11 Commands on Server

VI. Conclusion

The main goal of this research work is to build a Horizontal Platform for IoT application development. The development of such platform along with Service Capability Layer is done using DeviceHive Framework.. In this the discovery and registration of the device on gateway as well as on Server is also added. On the device side MSP430 board is used for a demo device connection. From the server side we also send commands to board and its also executed on the board. And also send the notifications to server via gateway. The device coding is done in C language being a non OS node. Further work on the scalability and security is under progress.

References

[1] Ericsson White Paper, "More than 50 Billion Connected Devices,"
2011.

[2] Texas Instrument White Paper, "The Evolution of the Internet of Things, 2013.

[3] “IoT-i.” [Online]. Available: http://www.iot-i.eu/public K. Elissa, “Title of paper if known,” unpublished.

[4] “IoT-A.” [Online]. Available: http://www.iot-a.eu/public.

[5] “TIA website,” [Online], available: http://www.tiaonline.org/.

[6] TS102690, “ETSI TS 102690, Machine-to-Machine communications (M2M);. Functional architecture,” Aug. 2013.

[7] “ONEM2M website,” [Online], available: http://www.onem2m.org/.

[8] Martin Floeck, Apostolos Papageorgiou, Anett Schuelke, and JaeSeung Song, “Horizontal M2M Platforms Boost Vertical Industry: Effectiveness Study for Building Energy Management Systems”, page-15-20 in Internet of Things (WF-IoT), IEEE World Forum on,2014.

[9] Stefano Severi, Francesco Sottile, Giuseppe Abreu, Claudio Pastrone, Maurizio Spirito *and* Friedbert Berens, “M2M Technologies: Enablers for a Pervasive Internet of Things”, pages-1-5 in Network and Communication (EuCNC), European Conference on, 2014.

[10] “Device Hive website,”[Online],available:http://www.devicehive.com

[11] "BoostLibraries,"[Online],available:http://www.boost.org/.

[12] "JSON,"[Online],available:http://tools.ietf.org/html/rfc4627