Article

OIC and OCF Spec Based Smart Switch Board with Raspberry PI 3

Matta Jayaram[1], V Sandhya[2]

Abstract

In this paper we describe about designing and development of OIC and OCF Spec based Smart Switch Board with Raspberry pi 3.Raspberry pi 3 board running Raspibian OS with IoTivity Frame work specifically, attention is given to understand the IoTivity Frame work by running an Smart Switch Board with Raspberry pi 3 as IoTivity Server Device where IoTivity ported Ubuntu PC with Eclipse Simulator will act as IoTivity Client Device. The intended out-come of this work is to understand the IoTivity Frame work and how to Discover the Devices, how to know the Status of Devices and Control the Devices.

Keywords: IoTivity Framework, Raspberry Pi 3, OCF, OIC, Coap, Ison, Cbor, Scons

Introduction

Each day more and more devices are adding to the ever-growing Internet of Things (IoT). Analysts and techies are agree that the IoT will grow to many billions of devices over the next decade.

The challenge for the IoT ecosystem is to ensure guidelines and rules to the IoT devices to connect securely and reliably to the Internet and to each other.

IoTivity Framework defines standards for connectivity requirements. It Ensures interoperability of billions of Internet of Things (IoT) devices. An open source software framework implementing OIC Standards. Ensures seamless device-to-device connectivity to address the emerging needs of the Internet of Things. The IoTivity project is sponsored by the Open Connectivity Foundation (OCF).

Related Work

Block Diagram abd Working Principle

Initially IoTivity Client and Server Devices are Connected to Local WiFi Router. Smart Switch Board with Raspberry pi 3 as IoTivity Server Device where as IoTivity ported Ubuntu PC with Eclipse Simulator will act as IoTivity Client Device.

[1]PG Student, [2]Asst.Prof, Dept of ECE, BVC Engineering College, JNTUK, A.P, India.

Correspondence: Mr. Jayaram Matta, BVC Engineering College, JNTUK, A.P, India.

E-mail Id: mattajayaram@gmail.com.

Orcid Id: http://orcid.org/0000-0002-6256-7040

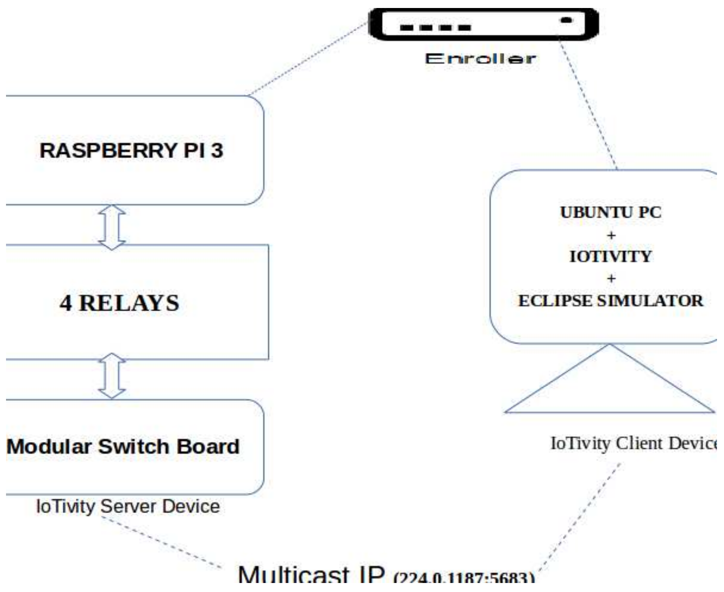
How to cite this article: Jayaram M, V Sandhya. OIC and OCF Spec Based Smart Switch Board with Raspberry PI 3. *J Adv Res Embed Sys* 2017; 4(1&2): 9-14.

ISSN: 2395-3802

Initially both the Devices are Connected to Common Multicast IP (224.0.1.187) with 5683 port number. 1.Initiate Multicast Device Discover Request from IoTivity Ported Ubuntu PC. It will Discovers all Iotivity devices which are connected to that comman Multicast IP. Example: Multicast GET coap://224.0.1.187:5683/oic/res 2.Then Initiate GET request for knowing it's present status Example:

1. Unicast GET coap://192.168.4.32:5508/a/sb
2. Unicast GET coap://192.168.4.32:5508/a/switch1

3. For Controlling the Device Initiate POST with Different Payload States Example: 1. Unicast POST coap://192.168.4.32:5508/a/sb payload [state=true / false] 2. Unicast POST coap://192.168.4.32:5508/a/switch1 payload [state=true / false]



IoTivity Framework

IoTivity is an open source project , it is hosted by the Linux Foundation and sponsored by the Open Connectivity Foundation (OCF) .OCF is a group of technology companies such as Samsung Electronics and Intel who together will develop standard specifications, promote a set of interoperability guidelines and provides a certification program to enable the Internet of Things. This project is independent from the OCF. Any individual or company can contribute to the project.

Architectural goal of IoTivity is to create and Implement a new standard by which billions of wired and wireless devices will connect to the internet and to each other.

Key Focus of OIC and IoTivity

Defines the standards for connectivity requirements Ensures interoperability of billions of Internet of Things (IoT) devices.An open source software framework implementing OIC Standards. Ensures seamless device-to-device connectivity to address the emerging needs of the Internet of Things. OIC Standards addresses multiple vertical domains including Home Automation, Automotive, Enterprise, HealthCare, Industrial scenarios. Initial focus on Smart Home & Office solutions Adopt Open Standards like IETF when applicable & standardize on areas, not addressed Open Source Framework implementing OIC Standards.Licensed under Apache License Version 2.0.Available on TIZEN, Android, Arduino, Linux(Ubuntu) Platforms

Hard Ware Requirements

Raspberry pi 3

The Raspberry Pi 3 is the third generation Raspberry Pi.It is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation . It replaced the Raspberry Pi 2 Model B in February 2016. Compared to the Raspberry Pi 2 it has:

- A 1.2GHz Broadcom BCM2837 64-bit quad-core ARMv8 CPU
- BCM43438 802.11n Wireless LAN
- Bluetooth 4.1 and Bluetooth Low Energy (BLE)

Like the Pi 2, it also has:

1GB RAM, 4 USB ports, 40 GPIO pins, Full HDMI port, Ethernet port, Combined 3.5mm audio jack and composite video, Camera interface (CSI), Display interface (DSI), Micro SD card slot (now push-pull rather than push-push), VideoCore IV 3D graphics core.

The Raspberry Pi 3 has an identical form factor to the previous Pi 2 (and Pi 1 Model B+) and has complete compatibility with Raspberry Pi 1 and 2.

Raspberry pi3 Supports Noobs, Rasbian, Ubuntumate, Snappy Ubuntu Core, windows 10 IoT Core, OSMC, Libreelec, Pinet, RISC OS with Custom we build we can also flash Mac OS, Android OS.

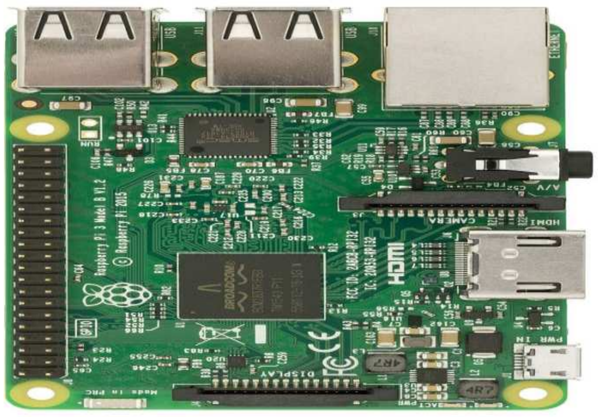


Figure.Raspberry pi 3

Relays

A Relay can be defined as a Electric switch.In Switches are used to close or open the circuit manually .Relay is also a switch that connects or disconnects two circuits. But instead of manual operation a relay is applied with an electrical signal, which in turn connects or disconnects another circuit.

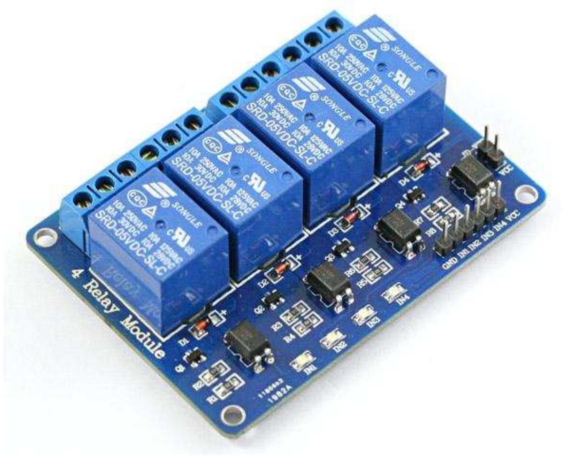


Fig.Four Relay Module

Four or Six Modular Switch Board

We need a Four or Six Modular Switch Board

Project Setup

Building IoTivity 1.2.0 for Raspberry Pi 3

- **Installing Raspbian on Raspberry Pi**

Download Raspbian Desktop version from belowLink https://www.raspberrypi.org/downloads/raspbian/ By using Win32diskimager Tool Write image to the SD card

- **Installing IoTivity 1.2.0 on Raspberry Pi 3**

a. a. Download Source code from https://www.iotivity.org/downloads/iotivity-1.2.0 and extract it.
b. Add SwitchBoard.c & .h files by making necessary changes in Scon Scripts This file contains the Smart Switch Board Source code for 4, 6 and 8 Modular Switch Boards.
c. From the Terminal go to the Location where the code base is extracted and Execute below command $ scons TARGET_ARCH=arm TARGET_OS=linux TARGET_TRANSPORT=ALL -j8

Building IoTivity 1.2.0 for Ubuntu 12.04/14.04 /16.04

Install necessary Tools like gitcore, scons, build essential.boost libraries, doxygen packages using apt. 1. Download the source code for IoTivity 1.2.0 from https://www.iotivity.org/downloads and Extract it to your preffered folder Location. 2 From the Terminal Navigate to the iotivity-1.2.0 directory.

```
$ cd /home/preffered folder Location/iotivity-1.2.0
```

3.Execute the scons build command from the iotivity-1.2.0 directory $ scons

Eclipse simulator setup

Follow the steps below for eclipse installation:

1. Download eclipse from http://www.eclipse.org/downloads
2. Start Eclipse and Select Help -> Install New Software.... In the dialog appears enter below mentioned URL in Work With text field and press enter key.

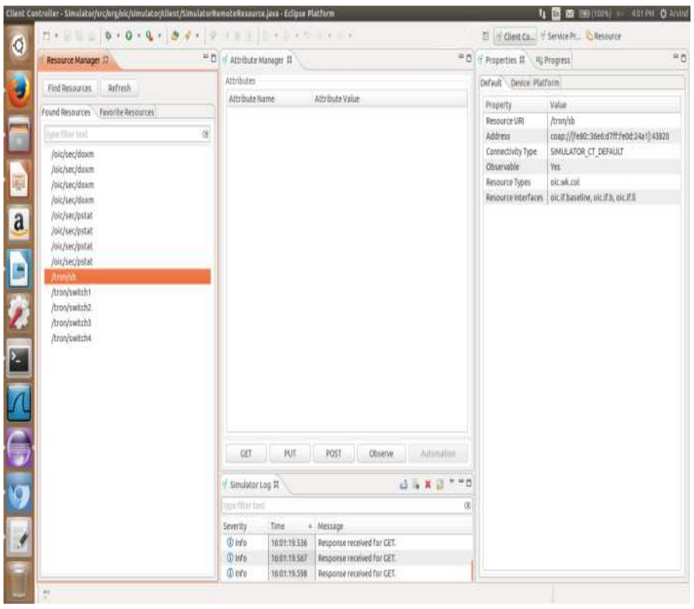
- https://downloads.iotivity.org/tools/simulator/latest
- Install below plugins and makesure that simulator properly will work.
- Service Provider Plugin: ~/<IoTivity home directory>/ service/simulator/java/eclipse-plugin/ServiceProvider Plugin
- Client Controller Plugin: ~/<IoTivity home directory>/service/simulator/java/eclipse-plugin/ClientController Plugin
- Simulator Java SDK: ~/<IoTivity home directory>/service/simulator/java/sdk

Result Analysis

1. Connect Raspberry pi3 to Wifi and Provide Power supply to it.
2. Program executable files will load automatically because I kept it in Raspberry pi3 startup.
3. Run the IoTivity Client Controller from Eclipse Simulator.

a. Click on Find Resources

It will Discovers Smart Switch Board with Four Switches as Resources.



/oic/sec/doxm
/oic/sec/doxm
/oic/sec/doxm
/oic/sec/doxm
/oic/sec/pstat
/oic/sec/pstat
/oic/sec/pstat
/oic/sec/pstat
/tron/sb
/tron/switch1
/tron/switch2
/tron/switch3
/tron/switch4

Property	Value
Resource URI	/tron/sb
Address	coap://[fe80::36e6:d7ff:fe0d:24a1]:43820
Connectivity Type	SIMULATOR_CT_DEFAULT
Observable	Yes
Resource Types	oic.wk.col
Resource interfaces	oic.if.baseline, oic.if.b, oic.if.ll

Severity	Time	Message
Info	16:01:19.536	Response received for GET.
Info	16:01:19.567	Response received for GET.
Info	16:01:19.598	Response received for GET.

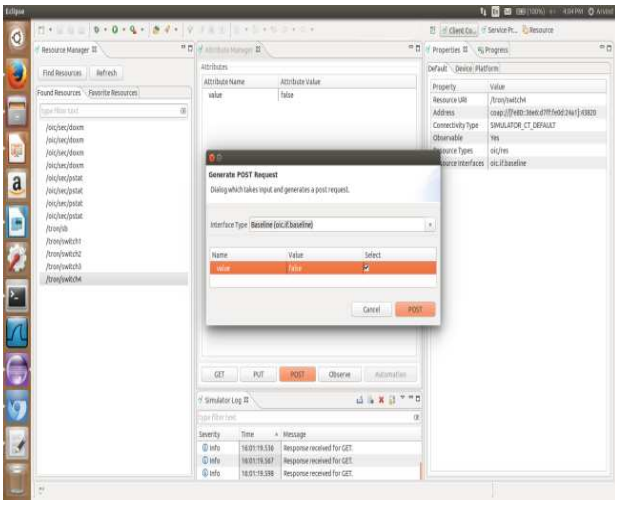
b. Click on GET

It will give Switches present Status ON / OFF

c. Click on POST

Change state of the Individual Switches by changing true / false

All OFF: state or value is false



/oic/sec/doxm
/oic/sec/doxm
/oic/sec/doxm
/oic/sec/doxm
/oic/sec/pstat
/oic/sec/pstat
/oic/sec/pstat
/oic/sec/pstat
/tron/sb
/tron/switch1
/tron/switch2
/tron/switch3
/tron/switch4

Attributes

Attribute Name	Attribute Value
value	false

Property	Value
Resource URI	/tron/switch4
Address	coap://[fe80::3ee6:d7ff:fe0d:24a1]:43820
Connectivity Type	SIMULATOR_CT_DEFAULT
Observable	Yes
Resource Types	oic/res
Resource Interfaces	oic.if.baseline

Generate POST Request

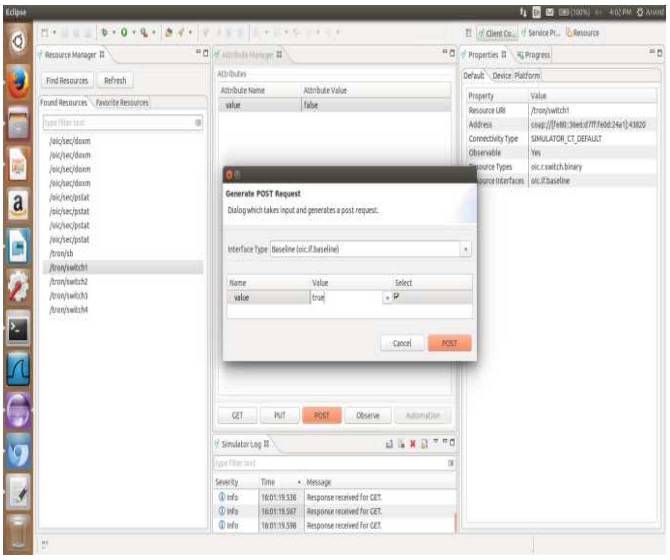
Dialog which takes input and generates a post request.

Interface Type: Baseline (oic.if.baseline)

Name	Value	Select
value	false	✓

Severity	Time	Message
Info	16:01:19.536	Response received for GET.
Info	16:01:19.567	Response received for GET.
Info	16:01:19.598	Response received for GET.

All ON: state or value is true



/oic/sec/doxm
/oic/sec/doxm
/oic/sec/doxm
/oic/sec/doxm
/oic/sec/pstat
/oic/sec/pstat
/oic/sec/pstat
/oic/sec/pstat
/tron/sb
/tron/switch1
/tron/switch2
/tron/switch3
/tron/switch4

Attributes

Attribute Name	Attribute Value
value	false

Property	Value
Resource URI	/tron/switch1
Address	coap://[fe80::3ee6:d7ff:fe0d:24a1]:43820
Connectivity Type	SIMULATOR_CT_DEFAULT
Observable	Yes
Resource Types	oic.r.switch.binary
Resource Interfaces	oic.if.baseline

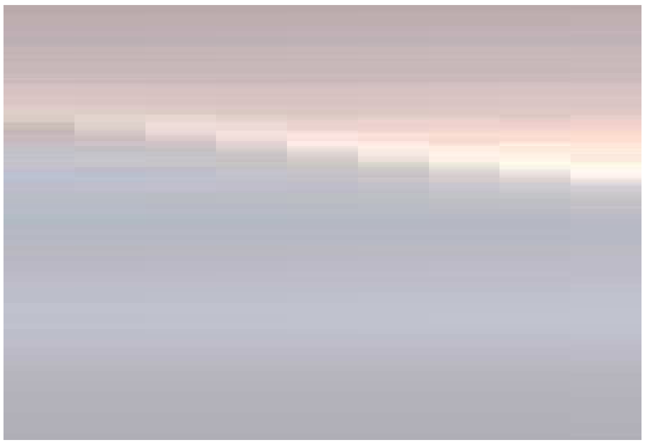
Generate POST Request

Dialog which takes input and generates a post request.

Interface Type: Baseline (oic.if.baseline)

Name	Value	Select
value	true	✓

Severity	Time	Message
Info	16:01:19.536	Response received for GET.
Info	16:01:19.567	Response received for GET.
Info	16:01:19.598	Response received for GET.



Conclusion

Nevertheless, this paper gives an insight into how the IoTivity Frame work will work, how it address the IoT challenges, Implementing standards and addressing the Interoperability by making open specifications and openness of the code it helpful for those who need to modify or existing implementation of their IoT based Designs and products.

Currently this IoTivity Frame work supports Various Constrained Os's like Contiki, RioT, Zephyr, Linux and Rich Side SDK's like Linux, windows, Android, Tizen.

The Available Development Boards Manufacturers or Vendors at least supporting the one of the above Constrained Os's or Rich SDK's.

For understanding this Framework I prepared an OIC and OCF spec based Four Modular Switch Board with Raspberry pi 3.

May be in the near future this IoTivity Frame work becomes New standard by which billions of wired and wireless devices will connect to each other and to the internet.

References

1. “Oic - open connectivity foundation brings massive scale to iot ecosystem.”http://openconnectivity.org/news/open-connectivity-foundation-brings-massive-scale-to-iot-ecosystem, feb 2016. Accessed 28/4 2016.
2. “Oic - oneiota data model tool.” http://open connectivity.org/resources/oneiota-data-model-tool. Accessed 28/4 2016.
3. “Iotivity programmers guide.”https://www.iotivity.org/documentation/linux/programmers-guideAccessed 29/4 2016.
4. “Oic core specification v1.0.0.” http://open connectivity.org/resources/specifications, 2015. Accessed 19/4 2016.
5. H. Virji, “The layered architecture of iotivity - samsung open source groupblog.”https://blogs.s-osg.org/layered-architecture-iotivity/, nov 2015. Accessed 2/5 2016.
6. “Iotivity features.” https://www.iotivity.org/documentation/featuresAccessed 29/4 2016.
7. “Registering a resource | iotivity.” https://www.iotivity.org/documentation/linux/programmers-guide/registering-resource.Accessed 29/4 2016.
8. “Finding a resource iotivity.” https://www.iotivity.org/documentation/linux/programmers-guide/finding-resource. Accessed 29/4 2016.
9. “Querying resource state [get] iotivity.” https://www.iotivity.org/documentation/linux/pro grammers-guide/querying-resource-state-get. Accessed 29/4 2016.
10. “Setting a resource state [put] iotivity.” https://www.iotivity.org/documentation/linux/programmers-guide/setting-resource-state-put. Accessed 29/4 2016.
11. “Observing resource state [observe] iotivity.” https://www.iotivity.org/documentation/linux/programmers-guide/observing-resource-state-observe. Accessed 29/4 2016.
12. www.raspberry.org.