IoTivity

From devices to cloud

Free and Open Source Developers' European Meeting
#FOSDEM, Brussels, Belgium <2017-02-04>

Philippe Coval + Ziran Sun
Samsung Open Source Group / SRUK
philippe.coval@osg.samsung.com
ziran.sun@samsung.com
https://fosdem.org/2017/schedule/event/iot_iotivity/
Bonjour tout le monde !

- We're software engineers from Samsung OSG
- Ask Philippe Coval for IoTivity, Tizen, Yocto, Automotive
  - About OS/hardware support, build & usages (English, French)
- Ask Ziran Sun for IoTivity, Web
  - About internal, cloud (English, Chinese)
  - https://fosdem.org/2016/schedule/speaker/ziran_sun/
Agenda

- A Vehicle to Infrastructure IoT demonstration
- What is OCF/IoTivity?
- Prototyping using NodeJS
  - Sensor monitoring
  - Notification to cloud
- More cloud facilities
- Q&A or/and extras
“Any sufficiently advanced technology is indistinguishable from magic.”
~ Arthur C. Clarke
How to track defective street lights?

1. Measure if outside's **lighting** is too dark
   - Embedded sensor in car (demo: I²C sensor)
2. Get position from satellites (GPS, Galileo)
   - From: car, mobile or any (demo: simulated)
3. Send notice to Internet (Cloud)
   - Using mobile data
4. Forward information to city services (pull or push)
   - 5: Agent is assigned
      - 6: to fix defective light
   - 7: he can also check “open data” base from his mobile
   - ...
From devices to cloud AutoLinux demo
https://vimeo.com/202478132#iotivity-artik-20170204rzr

Vehicle To Infrastructure
Proof of concept
(From devices to cloud)
https://wiki.iotivity.org/automotive

Using
Iotivity, NodeJs, ARTIK Cloud, Auto Grade Linux
CC BY SA 3.0: https://blogs.s-osg.org/author/pcoval/
“Simplicity is the ultimate sophistication.”
~Leonardo da Vinci
Open Connectivity Foundation

• “Providing the software Linking the Internet of Things”
  - Creating a specification, based on open standards:
    • Resource based, RESTful architecture (Stateless. client/server...)
    • IETF, CoAP protocol (Web on UDP), CBOR (JSON in binary)...
  - Sponsoring an open source reference implementation (IoTivity)
• Join 190+ members to
  - Discuss specification, propose RFC
  - Test products in Plugfests & certify them
  - Propose new data models (OneIoTA.org)
Flow: Create, Read, Update, Delete, Notify

IoTivity Server

- Initialization as server
- Registration of resource
- Handling new clients
  - + CoAP
  - UDP Multicast
- Handling new requests
  - POST/PUT
  - GET
- Notify updated resource

Local IP Network

IoTivity Client(s)

- Initialization as client
- Discovery of resource
- Set/Get/ing properties values
- Observe resource change & Handling properties

Samsung Open Source Group

https://fosdem.org/2017/schedule/event/iot_iotivity/
IoTivity Framework for connecting devices

- Hardware: CPU, MCU, Desktop, SBC, Tizen devices
- OS: Many including Linux, Tizen, Yocto or baremetal...
- **C API**: Data transmission (flash footprint ~128KiB-)
  - Resource Model / Serialization (CBOR)
  - Connectivity Abstraction: CoAP, Local IP Network, BT, BLE...
  - Discovery (UDP, Multicast), Security (DTLS/TLS)
- **C++ API**
  - C++11 OOP, Provisioning Service...
- + High level services (Mostly C++)
  - Data/Device Management, Hosting, Encapsulation...
“Talk is cheap.  
Show me the code.”

~ Linus Torvalds
Welcome to JavaScript developers!

- IoT is not reserved for embedded (few) developers (many)
- NodeJS a run time environment of choice for prototyping
  - Huge community = Consistent repository of many modules
    - to be installed using node package manager
  - Packaged for many OSes: GNU/Linux, Tizen, Yocto
- **IoTivity-node**: npm install iotivity-node
  - binds IoTivity CSDK (Core Library) to Javascript
  - Of course is interoperable with native servers or clients
- Let's get started, with a yocto distro with node, npm, iotivity-node
BH1750 Digital Light Sensor

- Illuminance: [1 – 65535] lx
  - Datasheet: bh1750fvi-e.pdf
- Uses I²C bus interface
  - 5P: GND, ADD (to GND), SDA, SCL, VCC
  - Check presence:
    - /dev/i2c-1 on Raspberry Pi2
    - i2cdetect -y 1 : will tell the address to use
- NodeJS package(s) available:
  - https://www.npmjs.com/search?q=bh1750
  - npm install bh1750

```javascript
// https://www.npmjs.com/package/bh1750
var BH1750 = require('bh1750');
var device = '/dev/i2c-1';
var address = 0x23;
var options = {
  address: address, device: device,
  command: 0x10, // 1 lx resolution
  length: 2
};
var sensor = new BH1750( { options } )
sensor.readLight(function(value){
  console.log(value);
  // emit('update', value);
});
```
OCF: Resources Data Models: onelIoTa

- **Resource** is identified by an URI
  - Composed of properties
    - Declared by a ResourceType
  - Operations: CRUD+N
    - Create, Read, Update, Delete+ Notify
- Use existing known resource models
  - From onelIoTa.org repository
    - Ie: sensors, geolocation...
- Or create new ones (new names)
  - Share for interoperability

- [http://www.oneiota.org/revisions/1863](http://www.oneiota.org/revisions/1863)
- oic.r.sensor.illuminance.json
- /* ... */ "definitions": {
  "oic.r.sensor.illuminance": {
    "properties": {
      "illuminance": {
        "type": "number",
        "readOnly": true,
        "description": "Sensed luminous flux in lux."
      }
    }
  }
} /* ... */
IoTivity-node Server notifies

- Initialize server and register resource:
  ```javascript
  iotivity = require("iotivity-node/lowlevel");
  iotivity.OCInit(null, 0, OC_SERVER);
  iotivity.OCCreateResource(
    handleReceptacle, 
    resourceType, 
    OC_RSRVD_INTERFACE_DEFAULT, 
    "/IlluminanceResUri", // URL 
    handleEntity, 
    OC_DISCOVERABLE | OC_OBSERVABLE);
  ```

- resourceType define Payload's data and format:
  - // ie: "oic.r.sensor.illuminance"
  - `{ "illuminance": 42 }`

- `handleEntity` Is a callback on client(s) requests
  - Register observers
  - Respond to requests (GET, POST, PUT)

- `notify(value)` to `observers` using:
  - `iotivity.OCNotifyListOfObservers`

- Integrate ambient sensor by trapping events:
  - `source.on("update", notify)`

- Processing loop:
  ```javascript
  setInterval(function( 
    {iotivity.OCProcess();}, 1000);
  ```
var client = require("iotivity-node").client;

client.on("resourcefound", function(resource) {
    if ("/IlluminanceResUri" === resource.resourcePath){
        resource.on("update", function(resource) {
            console.log(JSON.stringify(resource.properties));
            // OR update UI, forward elsewhere?
        });
    }
});

client.findResources().catch( function(error) { process.exit(1); } );
Forward data to a cloud backend

- Login your artik.io dashboard
  - Select or define **data models**
    - https://developer.artik.cloud/dashboard/devicetypes
  - Declare devices: (Copy IDs)
    - https://my.artik.cloud/devices
  - Monitor:
    - https://my.artik.cloud/data
- Send data: (REST, WS, CoAP, MQTT)
  - From iotivity's resource **“update”** event
  - Using **http REST**

```
require("node-rest-client").Client;
client.post(url, message, callback);
```

- https://api.artik.cloud/v1.1/messages

```
message = {
headers: {
'Content-Type': 'application/json',
Authorization: 'bearer BADC0DE(...)DEADBEEF42'
},

data: {
  sdid: 'deadbeef(...)badc0de13',
ts: 1485178599672,
type: 'message',
data: { illuminance: 42 }
}
}
```
A Vehicle to Infrastructure notification service

function handle(illuminance) {
  if (gThreshold > illuminance) {
    var data = {
      illuminance: illuminance,
      latitude: gGeo.latitude, longitude: gGeo.longitude
    };
    sender.send(data); // { ARTIK's client.post(url...); }
  }
}

client.on("resourcefound", function(resource) {
  if ("/IlluminanceResURI" === resource.resourcePath) {
    resource.on("update", handle);
  } else if ("/GeolocationResURI" === resource.resourcePath) {
    resource.on("update", function(resource) {
      gGeo = resource.properties;
    });
  }
});
There is no limits to knowledge
IoTivity Clouds

- Cloud Interface
- Authentication
  - OAuth2
- Message Queue
  - Publish
  - Subscribe
- Directory (RD)
IoTivity Services

- A common set of functionalities to application development.
  - Resource Container
  - Notification
  - Resource Encapsulation
  - Scene Manager
  - Easy setup
Summary

- OCF establishes a **standard** for interconnecting things
- **Open Source** project IoTivity implements it in C and C++
- NodeJS is a nice tool to prototype a scenario
  - IoTivity node to use CSDK core implementation of OCF
  - + npm modules to support, hardware, cloud API
- ARTIK Cloud is providing a backend
- IoTivity native cloud extends connectivity to global
- IoTivity Service make app development easier
References

• Entry points:
  - https://wiki.iotivity.org/examples: git clone iotivity-example
  - https://wiki.iotivity.org/docker: cloud images from Ondrej Tomcik
  - http://wiki.iotivity.org/automotive

• Going further:
  - https://openconnectivity.org/resources/iotivity
  - https://openconnectivity.org/resources/oneiota-data-model-tool

• Keep in touch online:
  - https://wiki.iotivity.org/community
  - https://wiki.tizen.org/wiki/Meeting
Q&A or/and Extras?
Use GeoLocation resource in Tizen apps
https://vimeo.com/164000646#tizen-genivi-20160424rzr
CES2017: Smart Home & Automotive demos
https://youtu.be/3d0uZE6lHvo
IoTivity native cloud

- **Cloud Interface**
- **Account Server**
  - to support multi-user (secured connection)
  - OAuth2 over CoAP
- **Message Queue Server**
  - broker to support PUB/SUB
- **Resource Directory Server**
- **CoAP over TCP**
  - encoder/decoder with TLS
- **CoAP HTTP Proxy**
  - for message mapping/parsing
Merci / 谢谢
Thanks / 고맙습니다

Samsung OSG, SRUK, SEF, SSI, Open Connectivity Foundation and members, LinuxFoundation, FLOSS Communities: Tizen, Yocto, EFL, AGL, GENIVI, eLinux, Resources: xkcd.com, FlatIcons
(CC BY 3.0: Freepik, Scott de Jonge, Gregor Cresnar)
Tools: Libreoffice, openshot,

FOSDEM attendees & YOU!

Contact: