IoT use case for Yocto Project

SUMMER SCHOOL
Outline

Yocto and IoT
IoTivity
ResinOS
Test
Questions
As of 2015: 25 billion connected devices

By 2020: 50 billion connected devices

Building the right embedded Linux distro for the connected devices can be slow and expensive

Yocto Project provides:

- A great network of hardware partners
- Several standard feature layers providing a wide set of communication standards and protocols
- A manageable build system
IoT implementations

**Eccellenza Touch**: coffee machine developed using the Yocto project - [http://eccellenzatouchvki.com/](http://eccellenzatouchvki.com/)

**LG Smart TV powered by WebOS**: streaming media and interconnectivity based on Yocto Project - [http://www lg.com/uk/smarttv/index.html](http://www lg.com/uk/smarttv/index.html)

**Daikin Industries Rooftop Units**: connects Rebel rooftop units to cloud and aggregates, filter and shares data in a secure fashion using the Yocto Project - [http://www daikin.com/products/ac/lineup/rooftops/index.html](http://www daikin.com/products/ac/lineup/rooftops/index.html)

What is IoTivity

IoT:
- Interconnect devices with the digital world
- Deployment of Low Power Embedded devices

IoTivity
- High level APIs for IoT Application developers
- Exposes lots of resources available on network connected devices
- Discover and manipulate resources over network
- Utilize emerging IoT technologies
- Part of Open Connectivity Foundation

Open Connectivity Foundation:
- Provide software linking IoT
- Write specifications, establish a protocol
- Sponsors reference implementations (IoTivity)
- Certify products for its members
IoTivity Stack on an edge device

https://www.iotivity.org/documentation/features
Resources are identified by an URI
- Composed of properties: declared by ResourceType
- Operations: CRUD+N (Create, Read, Update, Delete+Notify)

Uses existing resource models or creates new ones
- [https://oneiota.org/documents?filter%5Bmedia_type%5D=application%2Framl%2Byaml](https://oneiota.org/documents?filter%5Bmedia_type%5D=application%2Framl%2Byaml)
- Sensors, geolocation etc.
- Share for interoperability

```json
text
/* ... */ "definitions": {
  "oic.r.sensor.illuminance": {
    "properties": {
      "illuminance": {
        "type": "number",
        "readOnly": true,
        "description": "Sensed luminous flux in lux."
      }
    }
  }
} /* ... */
```
Flow: Create, Read, Update, Delete, Notify

IoTivity flow
IoTivity client interaction

Client sets resource value

Server handles it and responds back

```
OCInit(..., OC_SERVER);
OCCreateResource(..., onOCEntity);
  { OCPProcess(); }
OCInit(..., OC_CLIENT);
OCDoResource(..., OC_REST_DISCOVER, ...)
onDiscover(... OCClientResponse ...)
```

```
onOCEntity(entityHandlerRequest) {
  switch entityHandlerRequest->method {
    case 'POST': // Create value
    case 'PUT':  // Update new resource
       // handling the change
    case 'GET':  // READ current value
      ... OCDoResponse(&response);
  }
}
```

```
OCDoResource(...OC_REST_PUT ...)
onPut(... OCClientResponse ...)
```

- Client sets resource's value
- Server is handling it
  - and responding
Emerging Open IoT Protocols

- 6LoWPAN: IPv6 over Low Power Wireless Personal Area Networks
- Bluetooth Smart
- IPSP: Internet Protocol Support Profile makes possible for Bluetooth Smart to support 6LoWPAN
- RPL: Routing over Low Power and Lossy Networks
- New RFCs being published followed by prototype Linux implementations
- Growing influence of Linux in IoT
IoT challenges

Heterogeneous nature of targets, CPUs etc.
- IoTivity needs to be ported and maintained separately for each variation.
- Not easily scalable.

IoT rapidly evolving with new protocols
- Modular approach needed for quick plug-in of new IoT protocol implementations

Embedded development now became mainstream with IoT
- Cohesive and uniform software development infrastructure is needed across multiple IoT targets

All these challenges are addressed by the Yocto Project...
Packages IoTivity and its dependencies in a target agnostic way:
Meta-oic

git://git.yoctoproject.org/meta-oic

https://wiki.iotivity.org/yocto

Samples
- Resource clients and servers
- Third-party protocol plug-ins

IoTivity
- APIs
- Service Model and Plug-in Manager
- Resource Model
- Base Framework

Dependencies
- Kernel Configuration
- Protocol implementations
- Middleware Updates
Enable IoTivity features

- Bbappend to extend existing kernel features (configuration fragments)
- Add a GATT interface to BlueZ
- Integrate protocols such as RPL, Xbee
- Security related features
- Provide SDK with IoTivity support for application development

```bash
#Enable features for IoTivity
CONFIG_BT_6LOWPAN=y
CONFIG_IEEE802154=m
CONFIG_IEEE802154_6LOWPAN=m
CONFIG_6LOWPAN_IPHC=m
CONFIG_MAC802154=m
```
What is ResinOS?

Host OS tailored for containers designed with reliability in mind and minimal footprint.

Modern security features availability

Environment defined in a Dockerfile for predictability

OS can be used as a container

https://github.com/resin-os/

https://resinos.io/docs/custombuild/
Reliability

Root partition is never written to while in use

Strive to do atomic operations everywhere

Compartmentalization of failures

- Devices can survive data partition corruption
- Most I/O activity happens in there
Why Yocto Project?

Minimal
Low footprint
Build system allows for easy patching
Board vendors usually supply Yocto BSP
Easy new device support
Yocto layer architecture

One repo per board
Submodules for dependent layers
Each board can move independently

resin-<board-name>

submodules
- board BSP
- meta-resin
- poky
- meta-oe
Meta-resin

Main resinOS layer
Automatic aufs (union filesystem, for Docker) patching
BSP independent kernel configuration
Can prepopulate docker images
Kernel headers for out-of-tree module development
Cleaner separation
OTA updates are much easier
Enables diff based updates
We can`t leave state behind
Configuration stored in state partition
◦ Network configuration
◦ Random seed
◦ Clock at shutdown
Some states are also stored in tmpfs
◦ DHCP leases
◦ Limited logs
Systemd

Leverage a lot of systemd features
- Adjusting OOM score for critical services
- Running services in separate mount namespaces
- Very easy dependency management
- NTP

Socket activation for SSH
- Saves RAM since ssh is running only when needed
Networking

DNS is hard
- dnsmasq
- Integration of Docker with host’s dnsmasq

NetworkManager
- Excellent D-Bus API

ModemManager
- Excellent D-Bus API
- Lots of documentation
Docker

AUFS driver
- Allows support for NAND based devices

Currently on docker 1.10.3
- Backported stability patches

Journald logging driver
- Avoids SD card wear

Seccomp enabled
Log management

All logs end up in journald
In RAM 8MB buffer by default
Configurable log persistence
Journald allows for structured logs
◦ Container logs are annotated with metadata
Easy to send logs to a central location to store and process
Other features

Two stage flashing
- Automatic copy to internal storage
- Feedback through LEDs

Host OS updates
- Resinhup: https://github.com/resin-os/resinhup/

Dual root partition method
- Docker/ostree both viable solutions

Automatic emulated testing
- Integrated with Jenkins

Automatic hardware testing
- Built a board that instruments boards: GPIO, provisioning, SD muxing, wifi testing etc.
Development mode

Development images have
- Open SSH server
- Docker socket exposed over TCP
- mDNS exposed metadata

Device is at <hostname>.local
Resin Device Toolbox

- Image configuration
- Wifi credentials
- Hostname
- Persistent logging
- Automatically detects removable storage
- Won’t wipe your drive!
- Validates after writing
- Docker development
- Finds device in local network
- Continuously syncs code into the container
- Rebuilds when necessary

```
$ rdt configure ~/Downloads/resinos-dev.img
  ? Network SSID super_wifi
  ? Network Key super_secure_password
  ? Do you want to set advanced settings? Yes
  ? Device Hostname resin
  ? Do you want to enable persistent logging? no
Done!
```

```
$ rdt flash ~/Downloads/resinos-dev.img
  ? Select drive /dev/disk3 (7.9 GB) - STORAGE DEVICE
  ? This will erase the selected drive. Are you sure? Yes
  Flashing [==================================] 100% eta 0s
  Validating [==================================] 100% eta 0s
```

```
$ rdt push --source .
  * Building..
  - Stopping and Removing any previous 'myapp' container
  - Removing any existing container images for 'myapp'
  - Building new 'myapp' image
```
1. What is Bitbake?
   a) A build system
   b) A set of rules to write recipes
   c) A make-like build tool

2. Metadata is represented by:
   a) Recipes and configuration files
   b) Configuration files, bb and bbclass files
   c) Only recipes, configuration files are Bitbake specific

3. A Yocto Project distribution usually consists of:
   a) An USB bootable OS image
   b) Bootloader, kernel and rootfs images
   c) Kernel and rootfs tar archive

4. SDK is used by Yocto Project for:
   a) Writing recipes
   b) Developing application and images
   c) Compiling source code
5. ADT means?
   a) Additional Development Toolset
   b) Additional Development Toolkit
   c) Application Development Toolkit

6. Meta-python is available in the following repository:
   a) Meta-openembedded
   b) Poky
   c) Meta-openstack

7. IoTivity is part of:
   a) Linaro
   b) Open Connectivity Foundation
   c) Linux Foundation

8. ResinOS is tailored for:
   a) Kubernetes containers
   b) LXC containers
   c) Docker containers
1. What is Bitbake?
   c) A make-like build tool

2. Metadata is represented by:
   a) Recipes and configuration files

3. A Yocto Project distribution usually consists of:
   b) Bootloader, kernel and rootfs images

4. SDK is used by Yocto Project for:
   b) Developing application and images

5. ADT means?
   c) Application Development Toolkit

6. Meta-python is available in the following repository:
   a) Meta-openembedded

7. IoTivity is part of:
   b) Open Connectivity Foundation

8. ResinOS is tailored for:
   c) Docker containers
Questions?