ROS on OpenEmbedded
Simpler Robotics Development

LG Electronics, USA

ROSCon 2019 - Macau
Agenda

● Introduction and Motivation: Mass-market Robots
● Development with ROS + OpenEmbedded
● webOS Robotics
● Demo
Introduction and Motivation: Mass-market Robots

Brian Shin
Robots for Mass Market: Expectations

- Full Experience on small form factor
- Multiple Variants & Reproducibility
- Optimized for Low Cost
- Advanced User Experiences
- Up-to-date with software and security

Possible path: reduce hardware cost + simplify software development and maintenance
Embedded Product: Introduction

- **Embedded Hardware** (Single Board Computers)
- Customized for dedicated function
- Low memory footprint
- Low power consumption
- Real-time performance
Embedded Software: Challenges

- Multiple Hardware variants
- Cross Development
- Significant Customizations
- Lot of Optimizations

https://elinux.org/images/0/0a/Embedded_Linux_Build_Systems.pdf
Robots for Mass Market: Possible Solution

- Embedded hardware (low cost)
- OpenEmbedded (simple base Embedded Linux)
- ROS on OpenEmbedded (meta-ros)
- webOS (rich product features on OpenEmbedded)
Development with ROS + OpenEmbedded

Herb Kuta
OpenEmbedded (OE): Introduction

- Build system for the Linux Foundation’s Yocto Project
- Cross-compilation for many CPU architectures
- Wide variety of chipset and board support
- Provides a customizable “OpenEmbedded Linux”
- Input: recipes and configurations (“metadata”)
- Output: package archive plus bootable images
- Completely reproducible builds
- Organized into independent layers => expandable

“create custom Linux-based systems regardless of the hardware architecture”

Recipe (.bb)

- DEPENDS # Build-time dependencies
- RDEPENDS # Run-time dependencies
- SRC_URI # Location of source
- do_fetch() # Fetch the upstream source
- do_patch() # Patch it with your changes
- do_configure() # Run CMake, autoconf, etc.
- do_compile() # Compile
- do_install() # Select what’s installed
- do_package() # Package what’s installed

**Super Flore**: An extended platform release manager for ROS

---

**Distribution File**

- `<ROS-DISTRO>/distribution.yaml`

**ROS Component Repos**

- `package.xml`

---

**Distribution Cache**

- `rosdistro_build_cache`

- `<ROS_DISTRO>-cache.yaml`

---

**Generated Recipes**

- `generated-recipes-<ROS_DISTRO>/*//*.bb`

---

**Generated Recipes**

- `superflore`

---

**rosdep Mapping Files**

- `rosdep/*.yaml`
meta-ros: How to use for development

### Build ROS Image

# Clone meta-ros
$ git clone git@github.com:ros/meta-ros

# Setup OE layers
$ meta-ros/scripts/mcf -f conf/ros2-dashing.mcf

# Setup build environment
$ source openembedded-core/oe-init-build-env

# Configure the build
$ vi conf/local.conf

# Build an image
$ MACHINE=raspberrypi4 bitbake <image>

### Add/Modify ROS Package

# Clone your ROS package source
$ git clone git@github.com:<ros-package>

# Generate (or regenerate) recipe
$ ros-generate-recipes <ros-package>

# Add your package to the image
$ vi conf/local.conf

# Build an image with your package
$ MACHINE=raspberrypi4 bitbake <image>

https://github.com/ros/meta-ros

OpenEmbedded Build Instructions
## Case Study: Simplified Deployment (TurtleBot3 Waffle Pi)

### Current Scheme (with remote PC)

1. Install Ubuntu, ROS and TB3 application packages on the remote PC
2. Install unspecialized Ubuntu and ROS for generic Arm architecture on RPi
3. Download source code for TB3 packages and build them on RPi
4. SSH into TB3 and setup device instance configuration (hostname, WiFi SSID, etc.)
5. Configure remote PC and RPi to communicate with each other
6. Launch roscore and TB3 ROS applications on remote PC
7. Launch TB3 ROS nodes on RPi. TB3 is operational

### OpenEmbedded Based (no remote PC)

1. Build image with specialized OpenEmbedded and ROS optimized for RPi on a build PC
2. Load device instance configuration (hostname, WiFi SSID, etc.) and TB3 ROS nodes to start upon boot on USB flash drive
3. Flash image on SD card, insert it and USB flash drive into RPi, and boot. TB3 is operational

---

webOS Robotics

Lokesh Kumar Goel
LG webOS: Path to complete product

webOS is adopted on many LG devices.

Smart TV
Digital Signage
Smart Watch
Smart Refrigerator
Robot
Open Source Edition

https://www.webosose.org/
OpenEmbedded + ROS + webOS: Advantages

- Optimized for embedded hardware (power, CPU, memory)
- Capabilities for advanced user experiences (apps/services)
- Rapid web-based application development
- Software update pipeline (from cloud to device)
- White-labeled OS (rebrandable & redistributable)
webOS Robotics: Common Base Platform

Common Base Platform

webOS Robotics

UI/UX
- Compositor
- Web-Runtime

UI Frameworks (Qt/web/native)

Services (node/JS/native)
- Software Update
- Power
- Media
- ROS-Bridge (web/native)
- Connectivity (BT/WiFi/wired/etc.)

meta-ros (ROS1 & ROS2)
- ROS Core
- Perception
- Control
- Planning
- Localization

OpenEmbedded

Base Layers
- Core
- Kernel
- Toolchains

Scripts

Tools

Board Support Package (BSP)
- Drivers
- Hardware Integration Layers
# webOS Robotics: Roadmap

<table>
<thead>
<tr>
<th>Variants</th>
<th>Service Robot (Service)</th>
<th>Airport Robot (Concierge)</th>
<th>Child Robot (Companion)</th>
</tr>
</thead>
<tbody>
<tr>
<td>webOS: Specific</td>
<td>Mobile App</td>
<td>Airport App</td>
<td>Play App (Youtube Kids, Story Telling, Game)</td>
</tr>
<tr>
<td>ROS: Specific</td>
<td>Pattern Navigation (different modes)</td>
<td>Follow-me</td>
<td>Emotion Engine (Face/Emotion detection)</td>
</tr>
<tr>
<td>ROS: Common</td>
<td>SLAM</td>
<td>Obstacle Detection</td>
<td>Parking Mode (return to charge station)</td>
</tr>
<tr>
<td></td>
<td>Navigation</td>
<td>Person Detection</td>
<td>Simulated (ROS + webOS)</td>
</tr>
<tr>
<td></td>
<td>Build Profiles (with/ without UX)</td>
<td>Flow Programming</td>
<td>Cloud (OTA, Recovery, Diagnostics, Control)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software Update (OTA)</td>
<td>Multi-display</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Remote Control</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Offline Mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference Hardwares</td>
<td>ROBOTIS TurtleBot3</td>
<td>NVIDIA TX(n)</td>
<td>Qualcomm RB3</td>
</tr>
</tbody>
</table>
webOS Robot Platform: Demo

- ROSCON 2018: BeanBird Bot
  [https://youtu.be/lCGa7LkDNp0](https://youtu.be/lCGa7LkDNp0)

- ROSCON 2019: Big Bean Bot
  [https://youtu.be/nZ3QQ2HL5Vg](https://youtu.be/nZ3QQ2HL5Vg)
QUESTIONS ABOUT webOS

ABOUT OpenEmbedded

QUESTIONS

ANSWERS
About us

Lokesh Kumar Goel

- >15 years in Embedded Systems
- Director of Engineering
lokeshkumar.goel@lge.com
https://www.linkedin.com/in/lokesh-kumar-goel-3206042b/

Charles “Herb” Kuta

- >10 years in Embedded Systems
- Project Lead & meta-ros maintainer
herb.kuta@lge.com
https://www.linkedin.com/in/kuta42/

Brian Shin

- Robotics and embedded systems
- Product Manager & ROS 2 TSC Rep
brian.shin@lge.com
https://www.linkedin.com/in/byunghyunshin/