TIZEN: RT
A lightweight RTOS platform for low-end IoT devices

#FOSDEM, IoT Track
Brussels, Belgium <2018-02-04>

Philippe Coval
Samsung Open Source Group / SRUK
philippe.coval@osg.samsung.com
Who is Philippe Coval?

- Software engineer for Samsung Research
  - Open Source Group, EU team (@UK + DE + FR + CZ...)
- Interest: IoT, demos, usages, OS/hardware support, community
  - Projects: IoTivity, Tizen, Yocto, Automotive, etc
- Ask me online for help:
Agenda

- Technology overview:
  - **Tizen**, the OS of everything?
  - Software platforms
  - for **Low end** devices
    - NuttX, Tinyara, TizenRT
    - & relationships
  - Features & differentiation

- Crash course:
  - build sources,
  - log in, shell & apps
  - develop apps:
    - Native or Javascript?

- Demo & tips
- Resources and QA
What do you know of **TIZEN**?

- **Tizen** is an **Operating System** based on FLOSS
  - Powered by Linux Kernel
  - Open to platform and application developers:
    - Native (EFL), Web (HTML5), .Net...
- Shipped into **consumer electronics** products
  - TVs, Wearables, Home kitchen appliances...
  - With connectivity features (OCF, SmartThings, S-Connect)
- Supports: Architecture: ARM/Intel 32/64 bits (+ SBC)
  - 2013: Tizen-2.0: RD210: 1.2 GHz
  - 2014: Tizen-2.2 Gear2: 1GHz, 512MB, +4GB
  - 2016: Tizen-3 Wearable GearS3: 1GHz*2, 768MB
“I'm not crazy. My reality is just different from yours.”

~ Lewis Carroll
Tizen the OS of Everything?

- What about **low-end** and low-cost IoT devices?
  - Without display or rich UI/UX (just small LCD or buttons)
  - Battery powered, low consumption matters!
  - running ubiquitous **micro controllers** (MCU)
    - Cheap: Cost (<10$) and usage (Low Consumption <mW)
    - But can be very constrained in RAM+ROM
      - RFC7228: Class 0 from <10KiB+100KiB
- TizenRT is targeting middle configurations (RAM+Flash):
  - Cortex M3 (30KB+512KB), M4 (256KB+16MB), R4 (2MB+16MB)
Different software stack for low end devices

- For developers, there is gap between:
  - **Linux** kernel is flexible to some extend,
    - Typically: RAM=8MB + ROM=2MB (down for XIP: RAM=1MB ROM=4MB)
    - uCLinux is lowering requirements (No MMU, Reported, STM32F4’s M4: 256KB+2MB)
  - Baremetal: Optimal but not flexible: costly, slow
- For dedicated Oses: Genericity vs Speciality (~ trade of).
- Consider features, requirements, **learning curve** (tools), **licensing**
  - BSD: **NuttX**, TinyOS, MIT: FreeRTOS* Contiki,
  - Apache: Mbed, Zephyr, GPL: RIOT, ChibiOS, Inferno..
  - Unfree/Closed source: NucleusOS, eCosPro, ThreadX, VxWorks, QNX, uC/OS, RTX...
TizenRT's origin

- **TinyAra** 2015 project to collect, store, and deliver IoT sensor data using
  - IoTivity: IoT framework for seamless connectivity (+LWIP port for IPv6, +LWM2M)
  - AraStorage: Data management (SQL, b+ tree index)
    - sources released to public on tizen.org (SOSCON2016)
- Based on **NuttX RealTime** kernel (deterministic and priorities)
  - Initial release in 2007, community led by main author Gregory Nutt
  - Stable & Mature (2.5M LoC)
  - BSD Licensed:
    - used in many other OS projects or products, Industry adoption
      - PX4/PixHawk (Drones), Thingsee (IoT box)
- **TizenRT** is the whole stack: TinyAra kernel + middlewares (TDC2017)
NuttX is easy for Linux devs

- **UNIX/Linux** inspired OS (+ shell & apps)
- Comply with **standards** (POSIX/ANSI):
  - uses custom C library & C++ (uClibc++ or LLVM)
  - uses GNU tools, GCC, gmake, Kconfig, GDB, openocd...
- Filesystem (RO,RW), VFS (/dev/, /proc), MTD
  - handled by drivers (read, write, ioctl opts)
- Network: BSD sockets (uIP: TCP, UDP, IPv6, 6lowpan) NTP, FTP, HTTP etc
- Concurrency: Multi tasks & pthread support
  - +mutexes, message queues, signals, TLS, SMP, IPC, FIFO shed, preempt...
- **Modular** and configurable and scalable (Kconfig),
  - Low requirements: footprint is <16KB. Supports Many BSP (8 to 32 bits, 27 arch, 200 boards)
Tizen:RT Architecture and progress overview

Current status:
V1.1
+ ~650 patches
JavaScript runtime:
“IoT.js is to NodeJs what JerryScript is to v8”

(For more, Welcome to FOSDEM’s IoT room at 15h05 for Ziran's Sun presentation)
Tizen:RT plan for reliability

- **Memory protection unit** (MPU > MMU)
  - User/Kernel separation
    - Per thread mem protection
- **Micro kernel** architecture
  - Only for scheduling tasks, memory, IPC
  - + user space services (Net, drivers...)
- **Fault Tolerance**
  - Self healing
    - restart services (and dependencies)
  - Fallback option: Live update (DM)
Tizen:RT for IoT: Connectivity & Security

**Connectivity:**
- Standard protocols: LWIP (IPv6), mDSN, DHCP, BSD/Web/Sockets, MQTT (Eclipse's mosquitto)
- OCF's IoTivity: CoAP Discovery, Messaging (CA, RA, Cloud), Security (PM, DTLS, SRM)
- WiFi, WPASupplicant, APIs for onboarding (ARTIK app using QR codes)...
- Cloud: ARTIK=>SmartThings cloud, AWS... + S.Connect App

**Device Management Framework:** Monitor connectivity & power, report
- OMA-based Lightweight M2M
  - Over the air (FOTA...)
  - Eclipse Wakaama (formerly liblwm2m)

**More security features:** Crypto (AES 128/256, RSA, ECC....)
- Secure: Boot, Flash Storage, Channel (DTLS using mbedtls), Certs
Hardware support

**ARTIK**: 05x (053, 053s, 055, 055S)
- Ie: SoM ARTIK 055S + Interposer board:
  - 32-bit ARM® Cortex® R4 @ 320MHz
    - R for **Real Time** and safety critical
    - S for **Secure**: SE+TEE, Secure Sub System (sssfw.bin)
      - Crypto (Certs, Key, ECDSA, RNG, PUF),
      - Boot: ROM>bl1.bin> U-Boot (bl2.bin)> TizenRT OS (signed with artik05x_codesigner)
  - 1280KB RAM, 8MB Flash, WiFi (firmware wlanfw.bin)
  - IO: GPIO*29 (3.3VDC), ADC*4 (1.8V 12bits@6Mhz), PWM*5
    - UART*4 (2-pin), SPI*2, I2C*2, I2S, RTC, JTAG (lock)
- Others?: QEmu, SIDK_S5JT200
  - docs/HowToAddnewBoard.md
  - Or port NuttX BSPs (STM32, ESP32? ...)

https://fosdem.org/2018/schedule/event/tizen_rt/
“Simplicity is the ultimate sophistication.”
~Leonardo da Vinci
First boot of “TizenRT ARTIK SDK” on 055s

**sudo screen /dev/ttyUSB1 115200**

**TASH>>Hello, World!!**

**TASH>> help**

```
TASH command list

    cat    cd    date    df
    dhcpd  exit  free    getenv
    heapinfo hello  help    ifconfig
    ifdown  ifup  iperf  kill
    killall  logm  ls
    mkrd  mksmartfs mount
    onboard  ping  ps
    reboot  rm  rmdir
    setenv  sh  sleep
    tls_client  tls_server  umount
    uptime  wifi  unsetenv
```

CPU: Exynos200 @ 320 MHz
Model: ARTIK-053 based on Exynos200
DRAM: 722 KiB
WARNING: Caches not enabled
BL1 released at 2017-3-13 1
SSS released at 2016-12-30
WLAN released at ???.-??-.??
Flash: 8 MiB
**Warning - bad CRC, use with caution**

In: serial@80180000
Out: serial@80180000
Err: serial@80180000
Hit any key to stop autoboot...
gpio: pin gpg16 (gpio 46)
## Starting application at
s5j_sflash_init: FLASH Quantum
uart_register: Registering serial
uart_register: Registering serial...
(System Information:
  Version: 1.0
  Commit Hash: 1371111
  ...)
Explore system using TASH Shell and apps

TASH>>mount /mnt type smartfs
   /proc type procfs
   /sss type smartfs

TASH>>cat /proc/partitions

Name   Start  Size
b1l    0      0
ssro   0      0
bl2    0      0
ssfw   0      0
wlanfw 0      0
factory 0    0
ota     0    0
user    0    0
nvram  0    0
sssro  0    0

TASH>>cat /proc/version
Version: 1.0
Commit Hash: 13711f7a7
Build User: root@gateway
Build Time: 2017-08-17 01:22

TASH>>free
Data: 779296 153632 625664 610288

<table>
<thead>
<tr>
<th>PID</th>
<th>STATUS</th>
<th>SIZE</th>
<th>PEAK_STACK</th>
<th>PEAK_HEAP</th>
<th>TIME THREAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ACTIVE</td>
<td>1024</td>
<td>1024</td>
<td>81840</td>
<td>30755 Idle Task</td>
</tr>
<tr>
<td></td>
<td>(...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>ACTIVE</td>
<td>4076</td>
<td>844</td>
<td>3584</td>
<td>30755 task</td>
</tr>
<tr>
<td></td>
<td>(...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ACTIVE</td>
<td>2028</td>
<td>300</td>
<td>10256</td>
<td>30755 logm</td>
</tr>
</tbody>
</table>

TASH>>killall logm

TASH>>http
TASH>>gpio
TASH>>cloud
TASH>>sdk modules
TASH>>sensorbd

TASH>>wifi
TASH>>wifi startsta
TASH>>scan
TASH>>wifi join $SSID $PASS
TASH>>ifconfig w11 dhcp
TASH>>websocket connect wss://echo.websocket.org/
TASH>>mqtt_sub -d -h 198.41.30.241 -t $SYS/
“Talk is cheap. Show me the code.”
~ Linus Torvalds
Build from source

- **git clone**
  ```
  https://github.com/samsung/tizenrt
  ```
- **cd tizenrt & &**
  ```
  TIZENRT_BASEDIR="$PWD"
  ```
- Setup toolchain
  ```
  tar xvjf gcc-arm-none-eabi-4_9-2015q3-20150921-linux.tar.bz2 ; export PATH=$ {PWD}/...:$PATH
  ```
- Configure:
  ```
  - cd ${TIZENRT_BASEDIR}/os/tools
  - ./configure.sh <board>/<configuration_set>
  - make -C ${TIZENRT_BASEDIR}/os
    menuconfig
  ```
- Build
  ```
  - make -C ${TIZENRT_BASEDIR}/os
  ```
- Eventually sign image
  ```
  - (artik05x_codesigner -h)
  ```
- Flash using **openocd**
  ```
  - make -C ${TIZENRT_BASEDIR}/os download
  ```
Add sample native app

- Inspire from sample app: TizenRT/apps/examples/hello
  - hello_main.c:
  - Makefile
    - to build & register as a TASH command
  - Kconfig_ENTRY & Kconfig
    - to be enabled on build configuration
      - make menuconfig
  - Make.def
    - Build app in FW image (no DL, static libs)

- C API are mostly there
  - functions could missing: enable | use alternative | reimplement

- Exercise: create netcat app using <sys/socket.h> and <arpa/inet.h>
App sample JS app

- **Enable IoT.js**
  - in menuconfig (runtime):
  - To check: edit? simple javascript line:
    - console.log(JSON.stringify(process));
    - TASH>> iotjs /mnt/index.js
      - { "env":{ (...) },
        "platform":"tizenrt", (...),
        "iotjs":{ (...) } }

- **Exercise:**
  - Upload js from host to target
    - using netcat, websocket, http etc

- **Then load .js from ROM partition**
  - Enable ROMFS with menuconfig:
    - CONFIG_FS_ROMFS=y
    - *_FLASH_PART_* (400, romfs, rom)
    - *_AUTOMOUNT_ROMFS_*:
      - /dev/mtdblock11 on /rom
  - Install javascript to FS dir and rebuild FW:
    - ./TizenRT/tools/fs/contents/*.js
    - TASH>> iotjs /rom/index.js

- **Exercise:**
  - Make a luncher app to run iotjs at boot
“The secret to getting ahead is getting started.”

~Mark Twain
First world problems: Air Quality

- **World Health Org.:**
  - 92% of population
  - 11% cause of death

- **Survival plan:**
  - Collect and present *data*
    - To *systems* or *users*
  - Change behaviors

---

https://fosdem.org/2018/schedule/event/tizen_rt/
Monitoring air quality? let's prototype:

- Input Sensor(s):
  - Particle Pollution (PM), Dust, Gas
    - AQ-*, MQ-*, MG-811, DSM501A
  - MQ-2 Analog 5V sensor for CO, smoke, propane
- TizenRT supports **analog** inputs
  - + IoT.js module to handle ADC (/dev/adc0)
  - For ARTIK S55 1.8V in pin A0 (Add Voltage divider, + LED)
- Let's create a javascript class inspired from:
  - [W3C](https://www.w3.org) generic sensor and OCF airquality model
  - Configure, loop on read, emit ondata event
Emit radio alerts to Smart City

- Using LpWan (LoRa, SigFox...)
  - SubGHz low bandwidth radio (like 1 SMS per hour)
  - Try LoRa® RN2483 modem: SODAQ LoraBee
- TizenRT supports UART (RX/TX) a la UNIX
  - KConfigure /dev/ttyS1's baudrate to 56200
  - Use IoT.js UART module to handle port
    - (First I prototyped on node.js then ported serialport.js)
- Register device on favorite LoRaWAN network
  - BTW: Greetings to Rennes.fr's IoT communities.

```bash
> mac set devaddr BADC0DE1
< ok
> mac set nwkskey B4DC0DE2...
< ok
> mac set appskey BADC0D33...
< ok
> mac join abp
< ok
< accepted
> mac tx uncnf 1 142
< ok
< mac_tx_ok
```
“Any sufficiently advanced technology is indistinguishable from magic.”
~ Arthur C. Clarke
Live demo!
https://youtu.be/S7zpBpnpfIU#tizen-rt-lpwan-20180204rzr

AirQuality LPWAN monitoring
Proof of concept
Using: Tizen:RT + IoT.js
https://fosdem.org/2018/schedule/event/tizen_rt/

Running on
ARTIK 055s + LoRaWAN RN2483
CC BY SA 3.0: https://blogs.s-osg.org/author/pcoval/
DIY: mobile air quality monitor, and beyond?

Demo code (WIP):
https://github.com/rzr/TizenRT

https://fosdem.org/2018/schedule/event/tizen_rt/
Summary

- Tizen:1,2,3,4... is based on Linux kernel for high end devices
- Tizen ecosystem is now addressing more **constrained** IoT devices
  - Tizen:RT was originally based on NuttX kernel,
    - then evolved into micro kernel architecture
- Tizen:RT is **open** to application developers by providing
  - **Native**, C, C++ POSIX standard libraries
  - Middleware: Security, Connectivity, Device management, Database, IoTbus, Audio...
  - **Javascript** runtime, using IoT.js + JerryScript interpreter
- **Open Community** (github.com/Samsung) & Tools: SDK (CLI or IDE)
References

- **Entry points:**
  - https://wiki.tizen.org/Category:RT
  - https://github.com/Samsung/TizenRT/

- **Keep in touch online:**
  - https://wiki.tizen.org/wiki/Meeting
  - https://wiki.tizen.org/wiki/Events
  - https://wiki.tizen.org/wiki/FOSDEM

Bonus tip:
for demo convenience
I am drafting helper recipes (WIP)
to rebuild all from scratch using docker:

```bash
git clone https://github.com/rzr/TizenRT
       ./run.sh
make help menuconfig deploy run console```

More:
- https://source.tizen.org/documentation/tizen-rt
- https://www.artik.io/modules/artik-05x/
- https://developer.artik.io/documentation/artik-05x/getting-started/
- http://www.nuttx.org/
- http://youtube.com/channel/UC0QcilICUnj3kLSy3JBmluw
- https://www.slideshare.net/MyungJooHam/tdc-mjham-armkernelintizen3noani
- http://opensource.samsung.com/
Thank you
Merci, Danke Schoen, Gracias,
맞습니다, ありがとう, شكرا, Trugarez!


BTW greetings to:

Resources: flaticons CC
<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Gateway EUI</th>
<th>BASE64 data</th>
<th>RSSI</th>
<th>SNR</th>
<th>MIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-02-03 23:46:22.020844</td>
<td>AA555A0000090238</td>
<td>Rw==</td>
<td>-121</td>
<td>-13.5</td>
<td>OK</td>
</tr>
</tbody>
</table>