"Automated Test System - build blocks"

Łukasz Majewski

Samsung R&D Institute Poland

Embedded Linux Conference Europe
Dublin, 05-10-2015
Outline

1. Introduction
2. System description
3. Building blocks
4. Summary
5. Discussion
Introduction
A few words about me

- Embedded systems programmer
- Using u-boot since 2008
- ”Blame me” for:
  - Device Firmware Upgrade (DFU)
  - Trats/Trats2 devices
Sharing knowledge and experience after building simple and cost effective setup for automated u-boot and kernel tests with using open source SW and off-the-shelf HW.
Testing our u-boot

- 2011: Build test infrastructure (u-boot-CI)
Testing our u-boot

- 2011: Build test infrastructure (u-boot-CI)
- 2014: USB DFU/USB Mass Storage test scripts in mainline
Testing our u-boot

- 2011: Build test infrastructure (u-boot-CI)
- 2014: USB DFU/USB Mass Storage test scripts in mainline
- 2014: Boot and flashing tests - HardWare Test (HWT) v1
Testing our u-boot

- 2011: Build test infrastructure (u-boot-CI)
- 2014: USB DFU/USB Mass Storage test scripts in mainline
- 2014: Boot and flashing tests - HardWare Test (HWT) v1
- 2015: HWT v2
Automated Test System
Automated Test System - overview

- Presentation (Wiki)
- Scheduling (Jenkins)
- Tests (dash)
- Power control (FT232RL)

Board 1, Board 2, Board 3, Board N
Automated Test System - block diagram
System description
Architecture

- Build bootloader and kernel
- Download rootfs
- Flash target board

- Execute script on target board
- Use procfs, sysfs, debugfs, etc.
- Test result:
  0 → OK
  Other → FAIL

Links to proper tests/setup repositories. Test/setup instances represented as directories.
Key design decisions

- Why: New vs. Reuse
Key design decisions

- Why: New vs. Reuse
- SW: Python vs. Shell
Key design decisions

- Why: New vs. Reuse
- SW: Python vs. Shell
- Console: Python (pexpect) vs. expect (CLI program)
Key design decisions

- Why: New vs. Reuse
- SW: Python vs. Shell
- Console: Python (pexpect) vs. expect (CLI program)
- Flashing: Ethernet (TFTP) vs. USB (DFU)
Key design decisions

- Why: New vs. Reuse
- SW: Python vs. Shell
- Console: Python (pexpect) vs. expect (CLI program)
- Flashing: Ethernet (TFTP) vs. USB (DFU)
- Scheduling: Jenkins
Key design decisions

- Why: New vs. Reuse
- SW: Python vs. Shell
- Console: Python (pexpect) vs. expect (CLI program)
- Flashing: Ethernet (TFTP) vs. USB (DFU)
- Scheduling: Jenkins
- Presentation: Wiki
Building blocks
Power control

- **FTDI FT232RL**
  - Note: [https://github.com/lmajewski/HWT_io_ctl.git](https://github.com/lmajewski/HWT_io_ctl.git)

- **Relays**
  - Note: GPIO-controlled relays module for Arduino

- **USB HUB**
  - Note: External and stable power supply
Host PC

- Ubuntu Long Term Support (LTS)
- USB udev configuration
- Runs:
  - Dash tests
  - Jenkins client
  - Repository mirroring
  - MediaWiki
Host PC - Test U-boot booting

- Test triggered
- Read board config
- Build u-boot [SHA1]
- Flash u-boot via DFU
- Set default envs
- Reboot
- Run "version" to get SHA1

FAIL

SHA1 equal

OK
Host PC - Test kernel

1. Test triggered
2. Read board config
3. Flash kernel and rootfs
4. Power up board
5. Log to board via serial/ssh
6. Setup target (copy test)
7. Execute test on target

Decision:
- $\neq 0$: FAIL
- $\neq 0$: OK
Host PC - Jenkins [optional for simple setup]

- Responsible for scheduling tests
- Provides more sophisticated functionality to the test setup:
  - Gerrit push trigger
  - Repository/branch change trigger
- For simple test system it can be omitted - cron managed test triggering with results uploaded to wiki
## Host PC - Wiki

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Status</th>
<th>Repository</th>
<th>Branch</th>
<th>Image</th>
<th>Log</th>
</tr>
</thead>
<tbody>
<tr>
<td>KSFW_HWT_HQ_SCENARIO_1</td>
<td>OK</td>
<td>linux-arm64-exynos</td>
<td>exynos-next-4.1</td>
<td>tizen-mobile_20150914.1_mobile-wayland-arm64-n4.tar.gz [1]</td>
<td>[2]</td>
</tr>
<tr>
<td>KSFW_HWT_HQ_SCENARIO_2</td>
<td>OK</td>
<td>linux-arm64-exynos</td>
<td>exynos-next-4.1</td>
<td>tizen-mobile_20150914.1_mobile-wayland-arm64-n4.tar.gz [3]</td>
<td>[4]</td>
</tr>
<tr>
<td>KSFW_HWT_HQ_SCENARIO_3</td>
<td>OK</td>
<td>linux-arm64-exynos</td>
<td>exynos-next-4.1</td>
<td>tizen-mobile_20150914.1_mobile-wayland-arm64-n4.tar.gz [5]</td>
<td>[6]</td>
</tr>
<tr>
<td>KSFW_HWT_HQ_SCENARIO_4</td>
<td>OK</td>
<td>linux-arm64-exynos</td>
<td>exynos-next-4.1</td>
<td>tizen-mobile_20150914.1_mobile-wayland-arm64-n4.tar.gz [7]</td>
<td>[8]</td>
</tr>
<tr>
<td>KSFW_HWT_HQ_SCENARIO_5</td>
<td>OK</td>
<td>linux-arm64-exynos</td>
<td>exynos-next-4.1</td>
<td>tizen-mobile_20150914.1_mobile-wayland-arm64-n4.tar.gz [9]</td>
<td>[10]</td>
</tr>
<tr>
<td>KSFW_HWT_HQ_SCENARIO_6</td>
<td>OK</td>
<td>linux-arm64-exynos</td>
<td>exynos-next-4.1</td>
<td>tizen-mobile_20150914.1_mobile-wayland-arm64-n4.tar.gz [11]</td>
<td>[12]</td>
</tr>
<tr>
<td>KSFW_HWT_HQ_SCENARIO_8</td>
<td>OK</td>
<td>linux-arm64-exynos</td>
<td>exynos-next-4.1</td>
<td>tizen-mobile_20150914.1_mobile-wayland-arm64-n4.tar.gz [13]</td>
<td>[14]</td>
</tr>
<tr>
<td>KSFW_HWT_HQ_SCENARIO_9</td>
<td>OK</td>
<td>linux-arm64-exynos</td>
<td>exynos-next-4.1</td>
<td>tizen-mobile_20150914.1_mobile-wayland-arm64-n4.tar.gz [15]</td>
<td>[16]</td>
</tr>
<tr>
<td>KSFW_HWT_HQ_SCENARIO_10</td>
<td>OK</td>
<td>linux-arm64-exynos</td>
<td>exynos-next-4.1</td>
<td>tizen-mobile_20150914.1_mobile-wayland-arm64-n4.tar.gz [17]</td>
<td>[18]</td>
</tr>
<tr>
<td>KSFW_HWT_HQ_SCENARIO_11</td>
<td>OK</td>
<td>linux-arm64-exynos</td>
<td>exynos-next-4.1</td>
<td>tizen-mobile_20150914.1_mobile-wayland-arm64-n4.tar.gz [19]</td>
<td>[20]</td>
</tr>
<tr>
<td>KSFW_HWT_HQ_SCENARIO_12</td>
<td>FAILED</td>
<td>linux-arm64-exynos</td>
<td>exynos-next-4.1</td>
<td>tizen-mobile_20150914.1_mobile-wayland-arm64-n4.tar.gz [21]</td>
<td>[22]</td>
</tr>
<tr>
<td>KSFW_HWT_HQ_SCENARIO_13</td>
<td>OK</td>
<td>linux-arm64-exynos</td>
<td>exynos-next-4.1</td>
<td>tizen-mobile_20150914.1_mobile-wayland-arm64-n4.tar.gz [23]</td>
<td>[24]</td>
</tr>
<tr>
<td>KSFW_HWT_HQ_SCENARIO_14</td>
<td>OK</td>
<td>linux-arm64-exynos</td>
<td>exynos-next-4.1</td>
<td>tizen-mobile_20150914.1_mobile-wayland-arm64-n4.tar.gz [25]</td>
<td>[26]</td>
</tr>
</tbody>
</table>
## Minimal HW costs

<table>
<thead>
<tr>
<th>Device</th>
<th>Price [EUR]</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT232RL</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Relays</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>USB HUB</td>
<td>40</td>
<td>External power</td>
</tr>
<tr>
<td>Host PC</td>
<td>0</td>
<td>Old reused</td>
</tr>
<tr>
<td>Cables/connectors</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Power supply</td>
<td>0</td>
<td>Included with target board</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>65</strong></td>
<td></td>
</tr>
</tbody>
</table>
Summary
Solved problems

- Problem: Breaking the default boot
Solved problems

- **Problem:** Breaking the default boot
  - **Solution:** Use C/C++ with nanosleep()
Solved problems

- Problem: Breaking the default boot
  - Solution: Use C/C++ with nanosleep()

- Problem: Jenkins logs readability
  - Solution: Concatenate logs and link them to Wiki

- Problem: Target power control
  - Solution: HW rework to also control VUSB

- Problem: Lack of persistent USB configuration
  - Solution: Test system connectivity managed by udev rules, which use USB device serial numbers
Solved problems

- **Problem**: Breaking the default boot  
  - **Solution**: Use C/C++ with nanosleep()

- **Problem**: Jenkins logs readability  
  - **Solution**: Concatenate logs and link them to Wiki
Solved problems

- Problem: Breaking the default boot
  - Solution: Use C/C++ with nanosleep()

- Problem: Jenkins logs readability
  - Solution: Concatenate logs and link them to Wiki

- Problem: Target power control
Solved problems

- **Problem**: Breaking the default boot
  - **Solution**: Use C/C++ with nanosleep()

- **Problem**: Jenkins logs readability
  - **Solution**: Concatenate logs and link them to Wiki

- **Problem**: Target power control
  - **Solution**: HW rework to also control VUSB
Solved problems

- **Problem:** Breaking the default boot
  - **Solution:** Use C/C++ with nanosleep()

- **Problem:** Jenkins logs readability
  - **Solution:** Concatenate logs and link them to Wiki

- **Problem:** Target power control
  - **Solution:** HW rework to also control VUSB

- **Problem:** Lack of persistent USB configuration
Solved problems

- **Problem:** Breaking the default boot
  - **Solution:** Use C/C++ with nanosleep()

- **Problem:** Jenkins logs readability
  - **Solution:** Concatenate logs and link them to Wiki

- **Problem:** Target power control
  - **Solution:** HW rework to also control VUSB

- **Problem:** Lack of persistent USB configuration
  - **Solution:** Test system connectivity managed by udev rules, which use USB device serial numbers
Found bugs:

- u-boot: SD card timeout for writing data
- u-boot: Wrong error status passing from FAT to upper layers
- Linux: ”rootwait” problem in Trats/Trats2
YES, it was worth to build this setup.
Conclusion

- **YES**, it was worth to build this setup.

- **Benefits:**
  - Tizen.org: build, boot and functionality tests
  - u-boot-dfu: build, boot (u-boot and kernel), USB: Mass Storage, DFU, THOR
Discussion