Automated Testing Laboratory for Embedded Linux Distributions

Paweł Wieczorek
April 6, 2016

Samsung R&D Institute Poland
1. Introduction

2. Motivation

3. Automation opportunities with our solutions

4. Future plans

5. Conclusion
Introduction
Actual Automated Testing Laboratory
Automated Testing Laboratory – MinnowBoard MAX
Automated Testing Laboratory – SD MUX
Motivation
Change acceptance
Release engineering
Primary tools

- Open Build Service
- Jenkins
1. **Release engineer** investigates build failures (if any)
2. **Release engineer** checks whether new images introduce any regressions
3. **Release engineer** approves inclusion of verified changes to the main repository
Release Engineer headache

- Complete image testing on multiple devices takes **much** time:
  \[
  t_{\text{total}} = t_{\text{download}} + n_{\text{targets}} \times (t_{\text{flash}} + t_{\text{test}})
  \]
- Monotonous – involves repeating **the same** set of actions
- Requires focus – processing **similar** results calls for an observant person
1. Can we test images less frequently?
2. Can we run fewer tests on new images?
3. Can we assume that successfully built packages work properly?
1. Resolve an issue as soon as it is discovered
2. Look for a solution, not just workaround
3. Don't release software that was never run on an actual device
• Complete image testing on multiple devices takes much time:

\[ t_{\text{total}} = t_{\text{download}} + n_{\text{targets}} \times (t_{\text{flash}} + t_{\text{test}}) \]

• Monotonous – involves repeating the same set of actions
• Requires focus – processing similar results calls for an observant person
Automation opportunities with our solutions
Automation tasks categories

- Software
- Infrastructure
  - Internal
  - External
- Hardware
Automation tasks examples

- **Software**
- **Infrastructure**
  - Internal
  - External
- **Hardware**

- Polling OBS for new images
- Getting new images from OBS
- Controlling hosts and targets
- Publishing test results
- Flashing target devices with new images
Automation tasks examples

- Software
- Infrastructure
  - Internal
  - External
- Hardware
- Polling OBS for new images
- Getting new images from OBS
Automation tasks examples

- Software
- Infrastructure
  - Internal
  - External
- Hardware
- Polling OBS for new images
- Getting new images from OBS
- Controlling hosts and targets
Automation tasks examples

- Software
- **Infrastructure**
  - Internal
  - External
- Hardware
- Polling OBS for new images
- Getting new images from OBS
- Controlling hosts and targets
- Publishing test results
- Flashing target devices with new images
Automation tasks examples

- Software
- Infrastructure
  - Internal
  - External
- Hardware
- Polling OBS for new images
- Getting new images from OBS
- Controlling hosts and targets
- Publishing test results
- Flashing target devices with new images
Software – polling OBS and getting new images

- OBS lacks event mechanism
- Human-readable naming conventions require parsing
- New image discovery is run on multiple levels

- Scheduling tasks
- Queueing tasks
Internal infrastructure – reliable communication with devices

OpenSSH
- Depends on other services
- Requires network connection

Serial console
- Lower rate of data transfer
- Less flexible than alternatives

Neither could be chosen

SDB
(Smart Debug Bridge)
- Testlab-handbook on its own is **not enough**
- All changes in configuration are tracked in Testlab-host
- Improved deployments
- No more **snowflakes!**
External infrastructure – results publishing

- Easily available
- With possibility for future reuse
- Preferably using existing services

- Sharing test environment information
- Publishing test results
- Providing data for future reuse
Hardware – flashing target devices with new images

- Current interface focused on user interaction
- Designed for single target device per host
- Architecture-specific procedure
Hardware – SD MUX

Board control
Hardware – SD MUX

Board control

Memory card
Board control
Target SDB/card connection
Memory
card
Board control
Board control
Host card connection
Target SDB/card connection
Memory card

Hardware – SD MUX
Hardware – SD MUX

- Target SDB/card connection
- Board control
- Host SDB/card access
- Host card connection
- Memory card
Power switch
Board control
Host SDB/card access
Host card connection
Target SDB/card connection
Memory card
Host SDB/card access
Board control
Host card connection
$ sdmuxctrl --help
Usage: sdmuxctrl command
   -l, --list
   -i, --info
   -o, --show-serial
   -r, --set-serial=STRING
   -t, --init
   -u, --status
(...)

Former work flow

Requires release engineer's interaction
SD MUX work flow

Fully automated process
SD MUX – open-source

https://git.tizen.org/cgit/tools/testlab(sd-mux.git
Future plans
What is next?

- Pre-test cases development
- More detailed monitoring of differences between tested images
- Improved fail management
- Improved resource management
- System distribution
Conclusion
1. No need for reinventing the wheel in modern automation
2. Enforced limitations can be overcome with software
3. Custom hardware can simplify tasks
4. Automation pays off in the long term
Questions?
Thank you!

Paweł Wieczorek
p.wieczorek2@samsung.com
Samsung R&D Institute Poland
Further read

- https://wiki.tizen.org/wiki/Laboratory
- https://wiki.tizen.org/wiki/SD_MUX
- https://git.tizen.org/cgit/tools/testlab
• https://wiki.tizen.org/w/images/9/95/Testlab.JPG
• http://openbuildservice.org/images/obs-logo.png
• https://wiki.jenkins-ci.org/download/attachments/2916393/logo.png
• https://wiki.tizen.org/w/images/5/57/Tizen_Build_Process.gif
• https://by-example.org/wp-content/uploads/2015/08/openssh-logo.png
• https://pixabay.com/en/gears-options-settings-silhouette-467261/
• https://commons.wikimedia.org/wiki/File:Notification-icon-MediaWiki-logo.svg