

# Automated Testing Laboratory for Embedded Linux Distributions

---

Paweł Wieczorek

October 11, 2016

Samsung R&D Institute Poland

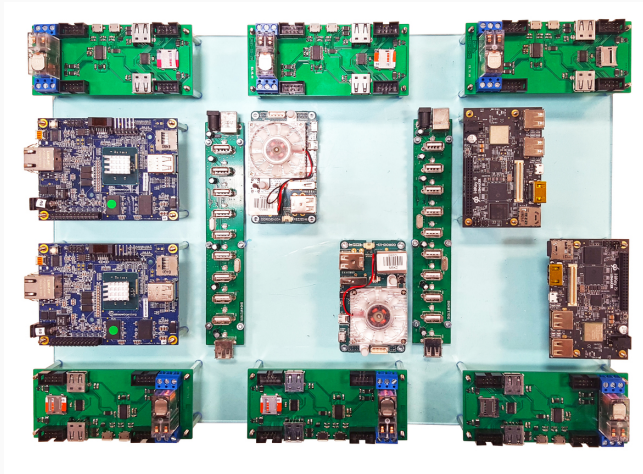
# Agenda

1. Introduction
2. Motivation
3. Automation opportunities with our solutions
4. Future plans
5. Conclusion

# Introduction

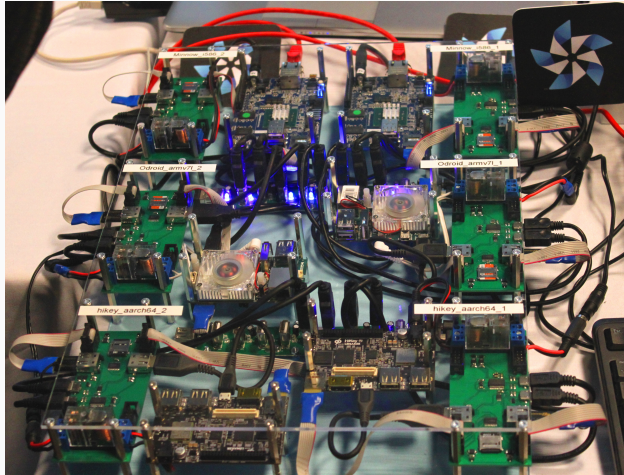
---

# Automated Testing Laboratory

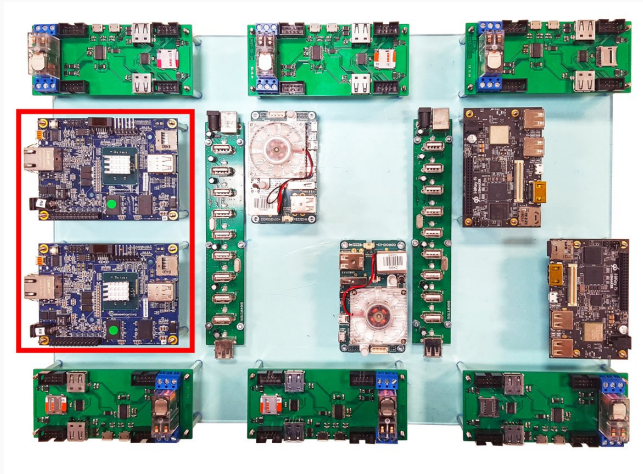




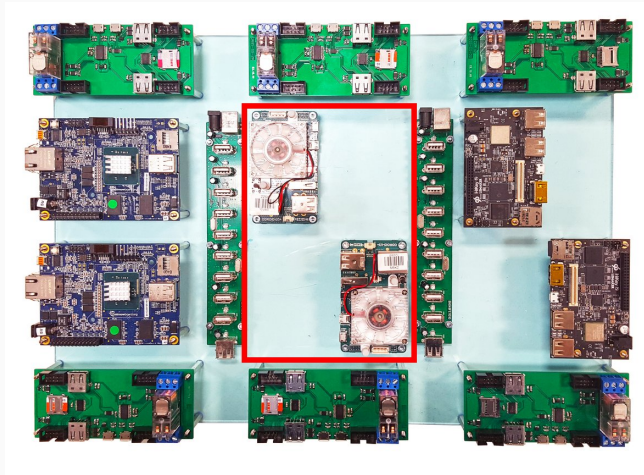
# Actual Automated Testing Laboratory



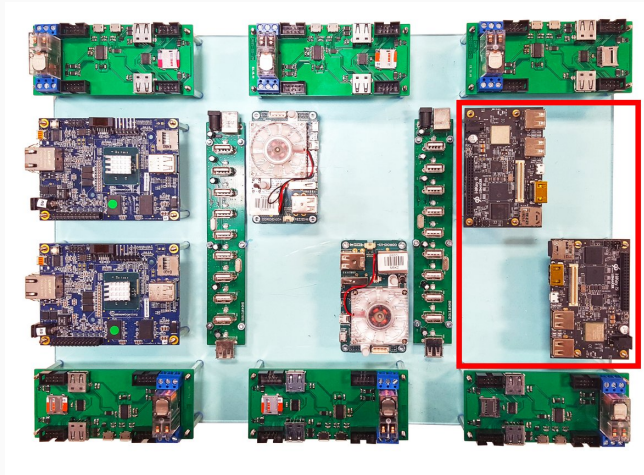
# Automated Testing Laboratory – MinnowBoard Turbot



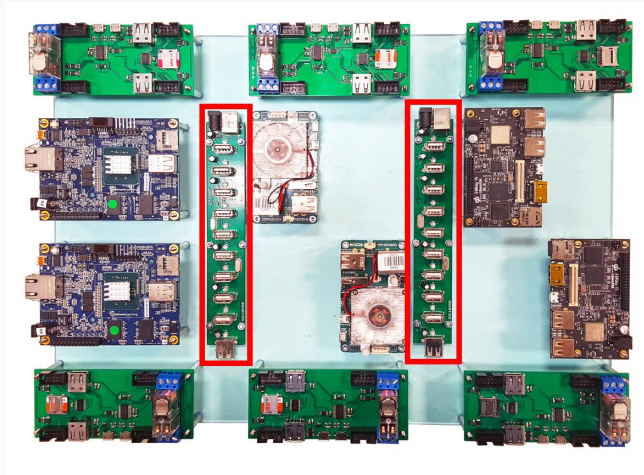
# Automated Testing Laboratory – Odroid U3+



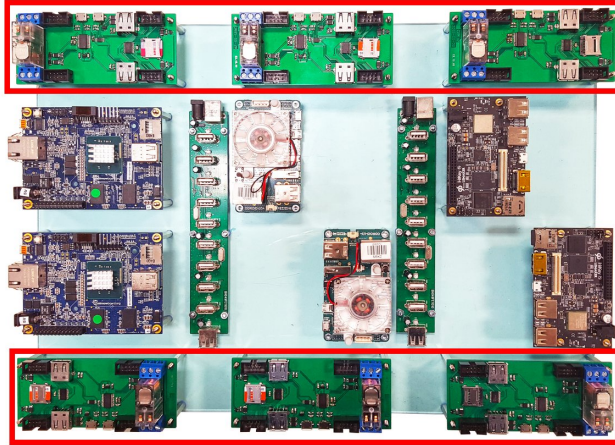
# Automated Testing Laboratory – HiKey

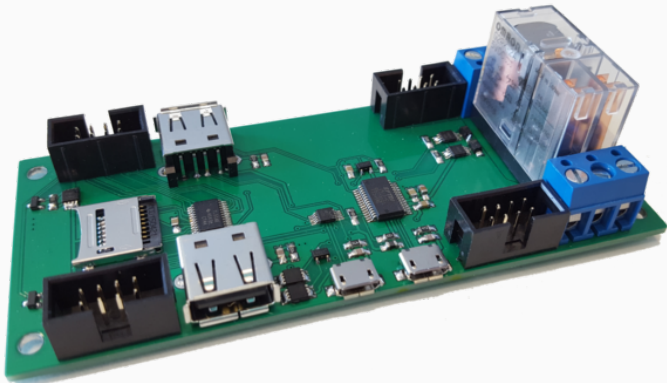


# Automated Testing Laboratory – Supporting hardware



# Automated Testing Laboratory – SD MUX



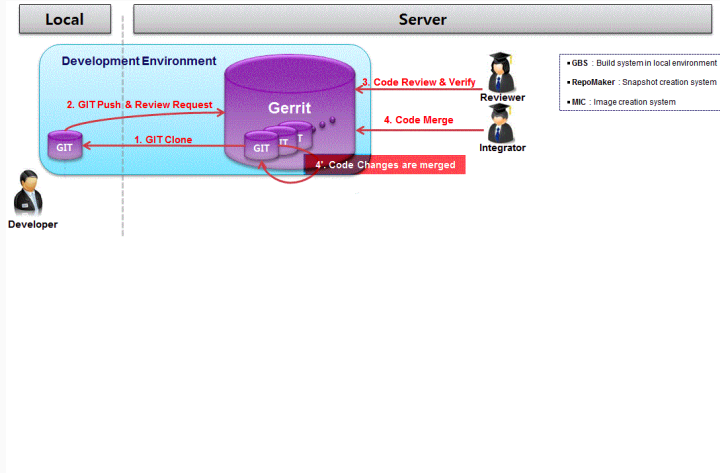


# Motivation

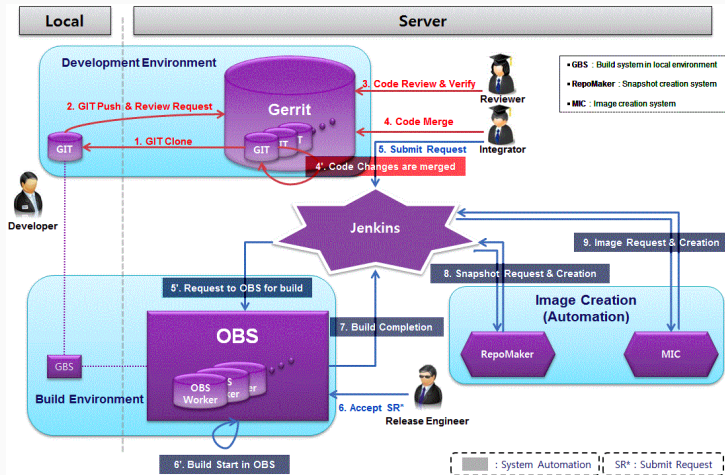
---



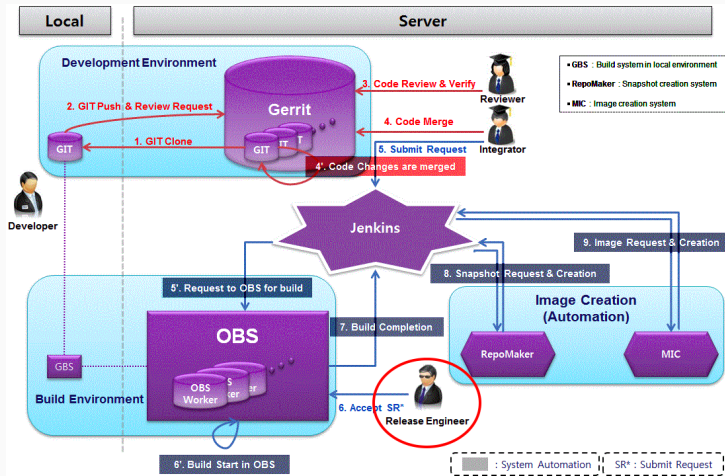
# Change life cycle



# Change acceptance



# Release engineering





**Open Build Service**



**Jenkins**

## Release Engineer role

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

- Complete image testing on multiple devices takes much time:

$$t_{total} = t_{download} + n_{targets} \times (t_{flash} + t_{test})$$

- 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:

$$t_{total} = t_{download} + n_{targets} \times (t_{flash} + t_{test})$$

- 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

- **Software**
  - **Infrastructure**
    - Internal
    - External
  - **Hardware**
- Polling OBS for 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



# Automation tasks examples

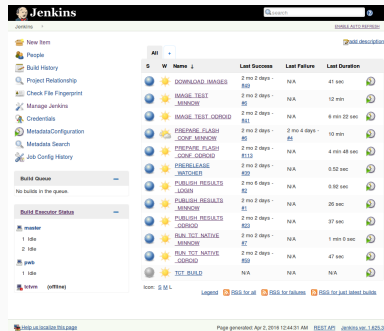
- |                  |   |   |
|------------------|---|---|
| • Software       | • Polling OBS for new images              |  |
| • Infrastructure | • Getting new images from OBS             |  |
| • Internal       | • Controlling hosts and targets           |  |
| • External       | • Publishing test results                 |  |
| • Hardware       | • 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



The screenshot shows the Jenkins web interface. On the left is a sidebar with navigation links like 'New Item', 'People', 'Build History', etc. The main area displays a table of build jobs. The table has columns for status, name, last success, last failure, and last duration. The jobs listed include 'DOWNLOAD\_IMAGES', 'IMAGE\_TEST\_MINION', 'IMAGE\_TEST\_ORROID', 'PREPARE\_FLASH\_IMAGE\_MINION', 'PREPARE\_FLASH\_IMAGE\_ORROID', 'PRERELEASE\_WATCHER', 'PUBLISH\_RESULTS\_LOGIN', 'PUBLISH\_RESULTS\_MINION', 'PUBLISH\_RESULTS\_ORROID', 'RUN\_TCT\_NATIVE\_MINION', 'RUN\_TCT\_NATIVE\_ORROID', and 'TCT\_BUILD'. Each job has a corresponding icon (sun for success, lightning bolt for failure) and a link to its description.

| S | W | Name                                       | Last Success       | Last Failure     | Last Duration |
|---|---|--|--------------------|------------------|---------------|
|   |   | <a href="#">DOWNLOAD_IMAGES</a>            | 2 mo 2 days - #62  | N/A              | 41 sec        |
|   |   | <a href="#">IMAGE_TEST_MINION</a>          | 2 mo 2 days - #5   | N/A              | 12 min        |
|   |   | <a href="#">IMAGE_TEST_ORROID</a>          | 2 mo 2 days - #41  | N/A              | 6 min 22 sec  |
|   |   | <a href="#">PREPARE_FLASH_IMAGE_MINION</a> | 2 mo 2 days - #5   | 2 mo 4 days - #4 | 10 min        |
|   |   | <a href="#">PREPARE_FLASH_IMAGE_ORROID</a> | 2 mo 2 days - #113 | N/A              | 4 min 48 sec  |
|   |   | <a href="#">PRERELEASE_WATCHER</a>         | 2 mo 2 days - #39  | N/A              | 0.52 sec      |
|   |   | <a href="#">PUBLISH_RESULTS_LOGIN</a>      | 2 mo 6 days - #2   | N/A              | 0.92 sec      |
|   |   | <a href="#">PUBLISH_RESULTS_MINION</a>     | 2 mo 2 days - #1   | N/A              | 26 sec        |
|   |   | <a href="#">PUBLISH_RESULTS_ORROID</a>     | 2 mo 2 days - #23  | N/A              | 37 sec        |
|   |   | <a href="#">RUN_TCT_NATIVE_MINION</a>      | 2 mo 2 days - #7   | N/A              | 1 min 0 sec   |
|   |   | <a href="#">RUN_TCT_NATIVE_ORROID</a>      | 2 mo 2 days - #59  | N/A              | 47 sec        |
|   |   | <a href="#">TCT_BUILD</a>                  | N/A                | N/A              | N/A           |

Icon: S M L Legend: SSG for all SSG for failures SSG for just failed builds

Page generated: Apr 2, 2018 12:44:31 AM Jenkins ver. 1.629.3

Jenkins

## 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

Default choice

**SDB**

(Smart Development 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

2016-01-30 03:48:25 - tizen-common\_20160108.1 [\[edit\]](#)

## Summary [\[edit\]](#)

### Test environments & result summary

| Arch   | Device    | Pass rate | Succeeded | Failed | Blocked | N/A |
|--------|-----------|-----------|-----------|--------|---------|-----|
| armv7l | Odroid U3 | 100.0%    | 66        | 0      | 0       | 0   |

## Results for specific device [\[edit\]](#)

### Odroid U3 (armv7l)

| Category | Pass rate | Succeeded | Failed | Blocked | N/A |
|----------|-----------|-----------|--------|---------|-----|
| CTC      | 100.0%    | 2         | 0      | 0       | 0   |
| UTC      | 100.0%    | 53        | 0      | 0       | 0   |
| ITC      | 100.0%    | 11        | 0      | 0       | 0   |

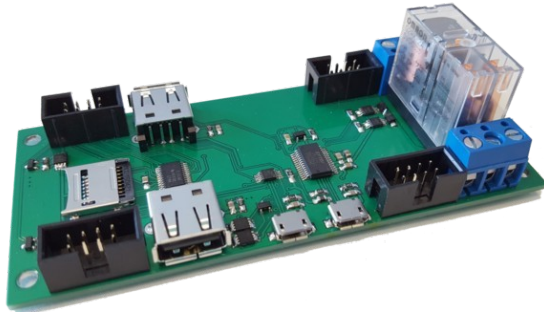
[Details \(2016-01-30 03:48:25\)](#)

**MediaWiki edited  
by Pywikibot**

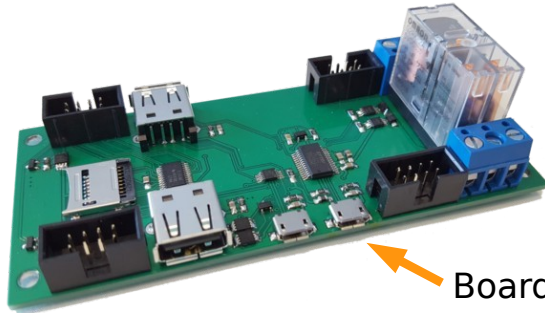
## Hardware – flashing target devices with new images

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

## Hardware – SD MUX

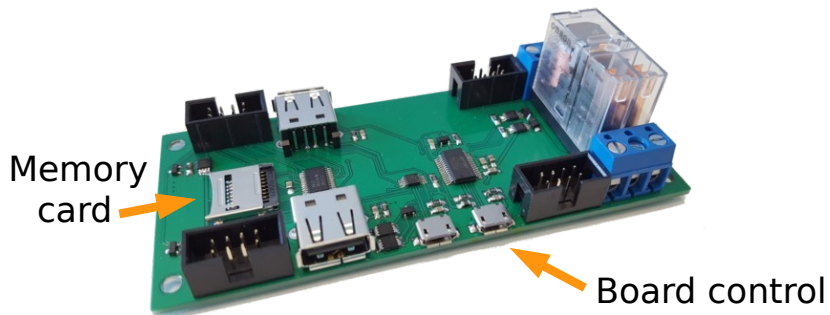


## Hardware – SD MUX



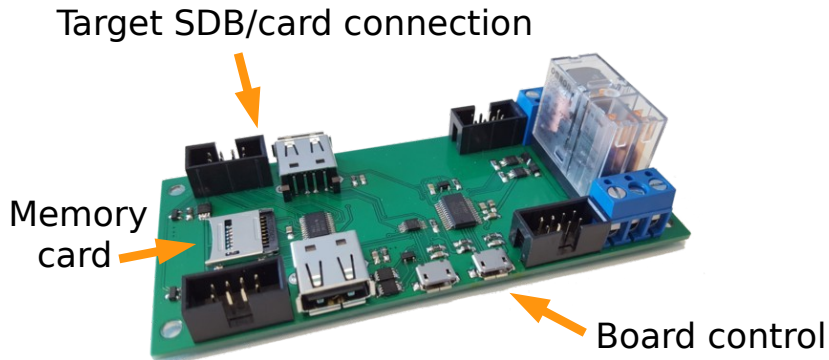
Board control

## Hardware – SD MUX

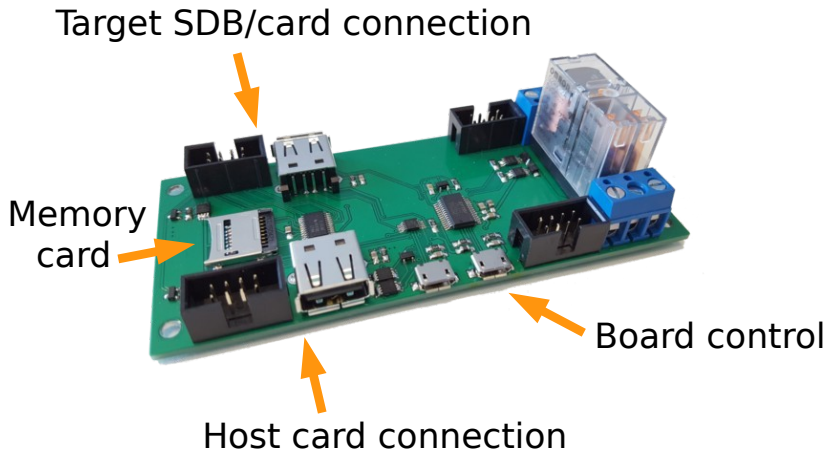




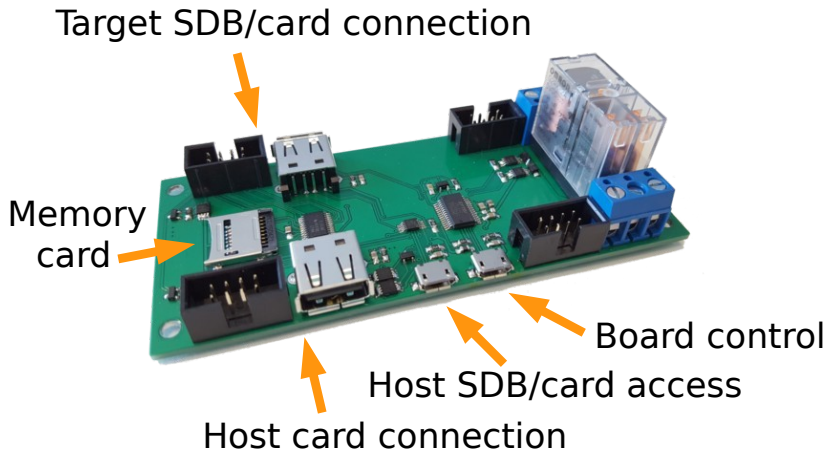
## Hardware – SD MUX



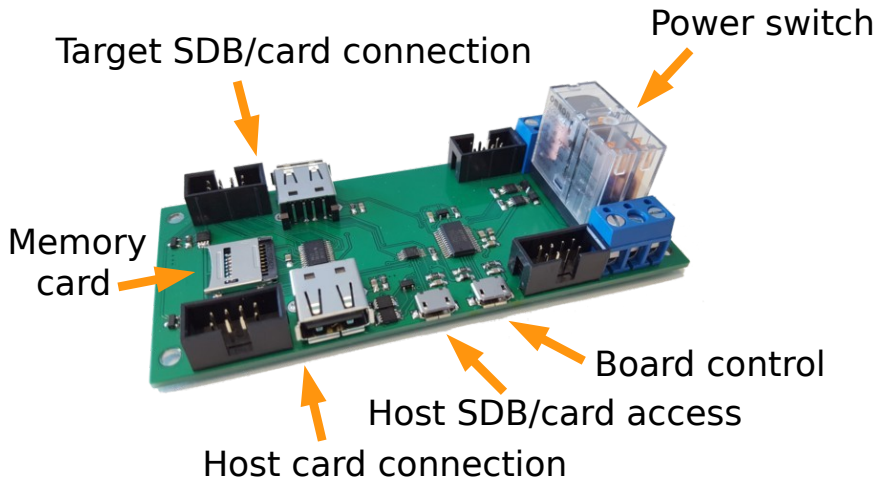
## Hardware – SD MUX



## Hardware – SD MUX

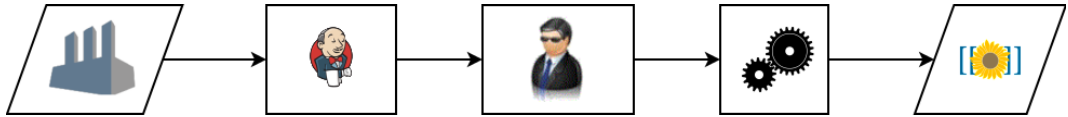


## Hardware – SD MUX



```
$ sdmuxctrl --help
Usage: sdmuxctrl command
  -l, --list
  -i, --info
  -o, --show-serial
  -r, --set-serial=STRING
  -t, --init
  -u, --status
(...)
```

## Former work flow



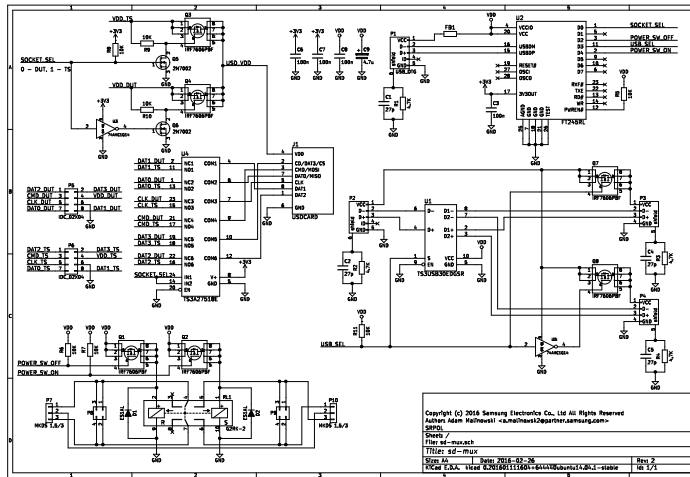
Requires release engineer's interaction

## SD MUX work flow



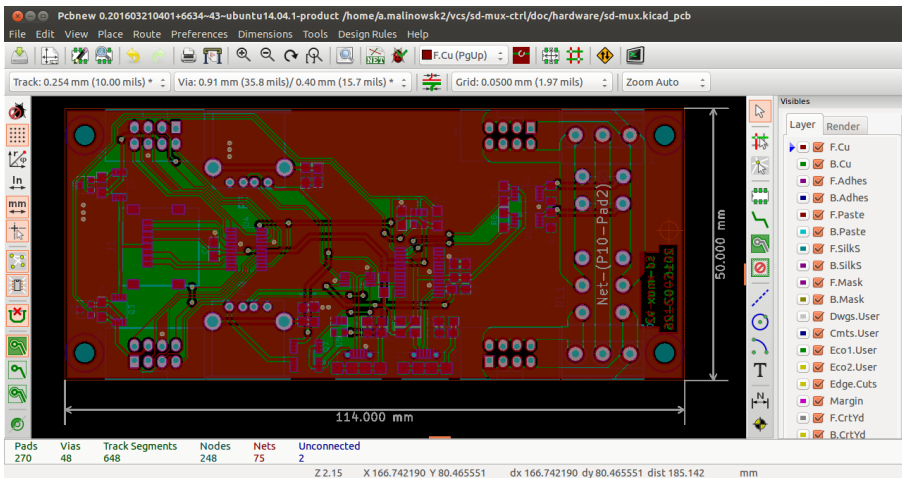
Fully automated process

# SD MUX – schematics





# SD MUX – open-source



<https://git.tizen.org/cgit/tools/testlab/sd-mux>

## **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. Custom hardware can simplify tasks
3. 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://wiki.tizen.org/wiki/SD_MUX)
- <https://git.tizen.org/cgit/tools/testlab>



# Pictures used

- <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://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/terminal-console-shell-cmd-dos-153150/>
- <https://pixabay.com/en/gears-options-settings-silhouette-467261/>
- <https://commons.wikimedia.org/wiki/File:Notification-icon-MediaWiki-logo.svg>