

# How to combine Debian and Yocto/Bitbake?



Manuel Traut <[manuel.traut@linutronix.de](mailto:manuel.traut@linutronix.de)>

ELCE 2017 - Prague

# What's next?

1. Why Yocto?
  2. About Debian
  3. Benefit of a combination
  4. Existing solutions
  5. Perfect combination
-

# 1) Why Yocto?

- ★ What is Yocto
- ★ Typical usage
- ★ Limitations

---

# What is Yocto?

- Tooling for building your own Linux distribution and SDK
- Defines a format that eases sharing of compile recipes and patches
- Powerful configuration management for different but similar images
- Based on Open-Embedded
- Example distribution “Poky” available

# Typical usage

- Use Poky example distribution
- Add meta-layers from chip and/or hardware vendor
- Add 3rd party layers, e.g. for QT5
- Add own layer with image customization and own applications

# Limitations

- Recipes from different layers might be incompatible
- Packages need to be built before they can be used
- Quality of recipes is hard to verify
- Security tracking/updates need to be done
- No LTS/updates available
- Reproducibility is not completely given (host dependencies)

## 2) About Debian

- ★ The universal OS
  - ★ Debian and embedded?
  - ★ Usage
  - ★ Limitations
  - ★ E.L.B.E.
-

# The universal OS

- Debian provides more than a pure OS, it comes with over 51 000 packages
- The infrastructure, documentation and build-tools are open-source
- Debian takes security very seriously
- Many security advisories are coordinated with other free software vendors and are published the same day a vulnerability is made public



## Debian and embedded?

**Packages are available for**

amd64 arm64 armel armhf i386 mips mips64el mipsel  
powerpc (not in stretch) ppc64el s390x

Also **cross-toolchains** for different architectures are available in Debian/stretch

## 4 Debian as CIP primary reference distribution

- What does the primary distribution means?
  - CIP will select CIP Core package from Debian packages
  - CIP would like to work with Debian community
  
- CIP members also interested in Yocto Project as a build tool
  - CIP might create meta-cip layer
    - Users can get SLTS benefit from CIP Core packages
    - Other OE-layers could be extend CIP Core (Will not SLTS by CIP)

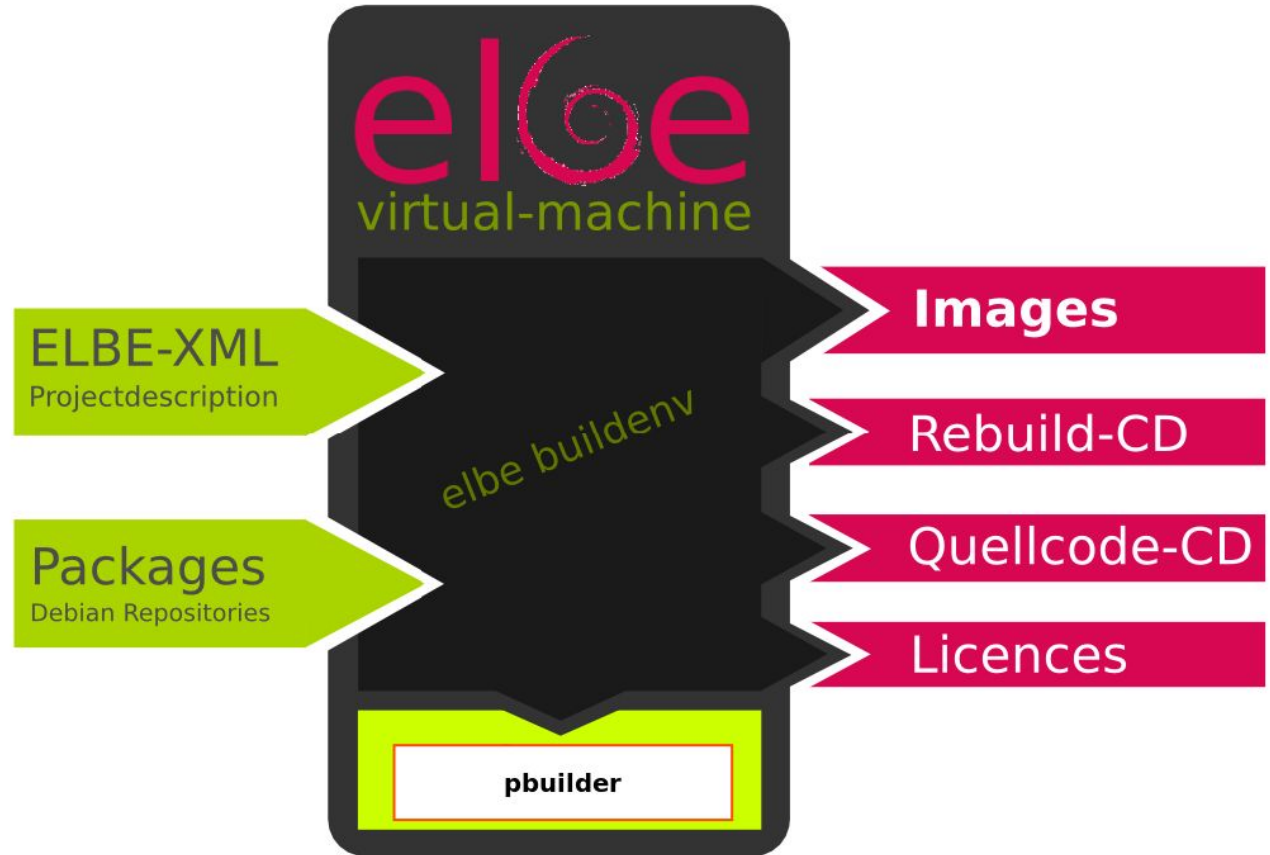
# Usage

- Debootstrap embedded RFS (e.g. for arm) into a directory
- Use pbuilder or a cross-compiler to build own applications and copy to RFS-dir
- Remove unneeded files (man-pages, i18n, ...) from RFS-dir
- Build FS (ext4, etc) or disk / UBI images using some tools and scripting
- Extract licence information and retrieve source-code of all used packages

## Limitations using Debian

- Only limited number of HW architectures supported
- No HW specific binary packages like special gstreamer plugins are available
- SDCard / UBI / etc. image generation
- SDK generation and licence information and source package extraction
- Reduce image footprint
- Own application integration

## E.L.B.E.



## Limitations using Debian & E.L.B.E.

- Only limited number of HW architectures supported
- No HW specific binary packages like special gstreamer plugins are available
- ~~SDCard / UBI / etc. image generation~~
- ~~SDK generation and licence information and source package extraction~~
- ~~Reduce image footprint~~
- ~~Own application integration~~

### 3) Benefit of a combination

★ Yocto + Debian = ?

---

# Yocto + Debian = ?

## Use from Yocto

- Task scheduling
- Configuration management
- (cross-) compile from source  
if necessary
- SDK generation

## Use from Debian

- Well maintained packages
- Security tracking
- Binary packages  
if available and useful
- Source packages  
if necessary



## 4) Existing solutions

- ★ meta-debian
  - ★ Isar
  - ★ nmeta-elbe
  - ★ Comparison table
-

# meta-debian

- ~600 .bb recipes, using sources from Debian/jessie
- Build rules optimized for embedded and retrieved from 'debian/rules'
- Long-term Linux kernel from CIP (Civil Infrastructure Platform)
- Supports SDK generation
- Very active ~2000 commits on github
- Not compatible with existing Debian binary packages

# lsar

- Uses Debian binary packages from stretch, jessie, wheezy or raspbian-jessie
- Optional: building Debian source packages in a chroot (with qemu-user)
- Needs 'sudo' with nopasswd for several tasks
- Default image size ~300MB
- ~100 commits on github

## nmeta-elbe

- Proof of concept E.L.B.E. frontend (nmeta-elbe 9 / elbe ~2000 commits on github)
- Uses Debian stretch binary pkgs (tested with armhf)
- Optional: build binary pkgs from source within elbe-pbuilder
- Source and binary pkgs built with pbuilder available in a signed Debian repository
- Bitbake generates elbe-xml and schedules elbe-pbuilder and elbe-image-build jobs
- Builds licence information
- SDK generation currently not implemented, but easy because available in E.L.B.E.

# nmeta-debian architecture

## nmeta-elbe

base.bbclass

elbeproject.bbclass

source.xml.mako

image.bbclass

pbuilder.bbclass

./conf/distro/jessie.conf

./conf/distro/stretch.conf

./conf/machine/mymachine.conf

## nmeta-elbe-ext

libgpio\_git.bb

```
SRC_URI = "git://github.com/linutronix/libgpio"  
SRCREV = "${AUTOREV}"
```

```
S = "${WORKDIR}/git"
```

```
inherit pbuilder
```

hello\_git.bb

simple-image.bbappend

	meta-debian	lsar	nmeta-elbe
Yocto-style config management & app integration			
HW-specific SW like kernel / bootloader buildable			
Use Debian sources			
Default footprint / reducible?	10MB	300MB / with Yocto methods	300MB / not yet
Non-Debian archs buildable			
use an arch not supported by Debian			
Export used source code	download dir		easy to develop
Yocto-style SDK with cross-toolchain generation available			easy to develop
Generate licence information	csv		XML & plain-text
Reproducibility	pkg v. by git tags	no VM / shared chroot for all builds	VM/ pbuilder
Bitbake file per Debian package needed	+ a git repo	not for bin-pkg	not for bin-pkg
Use Debian binary packages			
# of available Debian packages	limited / ~600 dsc	all	all
Effort needed to adapt buildsystem to new Debian release	very high		
Generate signed Debian repos of self built packages	unsigned deb	unsign. deb (dsc/sign easy to add)	dsc + deb

# 5) Perfect combination

- ★ My personal wishlist
- ★ Conclusion
- ★ Your ideas

---

## My personal wishlist

- Collaborate with 'rebootstrap.sh'  
to bootstrap Debian with settings from Bitbakes machine config
- Use Debian multiarch for cross-compiling any (modified) src pkg  
for a self bootstrapped architecture
- Mix usage of cross-built Debian pkgs via Bitbake  
with official Debian binary pkgs (for official supported architectures)
- Having reproducible builds for all Debian packages



# Conclusion

- 3 implementations but only 2 use-cases
  - meta-debian is good for architectures that are NOT available in Debian
  - Isar and nmeta-elbe can only be used if the architecture is available in Debian
  - nmeta-elbe is a proof-of-concept but it's already very powerful thanks to the E.L.B.E backend
- Porting Debian bootstrapping to Bitbake might be interesting for Debian and Yocto

# Your ideas

To improve the usage of Debian in embedded Linux projects?

# References

**nmeta-elbe / ELBE.**

<http://elbe-rfs.org>

<http://github.com/linutronix/nmeta-elbe>

<http://github.com/linutronix/nmeta-elbe-extended>

[http://elinux.org/images/e/e5/Using\\_ELBE\\_to\\_Build\\_Debian\\_Based\\_Embedded\\_Systems.pdf](http://elinux.org/images/e/e5/Using_ELBE_to_Build_Debian_Based_Embedded_Systems.pdf)

**Debian**

<https://wiki.debian.org/HelmutGrohne/rebootstrap>

<https://wiki.debian.org/ReproducibleBuilds>

**Isar**

<http://github.com/ilbers/Isar>

<https://events.linuxfoundation.org/sites/events/files/slides/isar-elce-2016.pdf>

**meta-debian**

<http://github.com/meta-debian>

<https://elinux.org/images/2/2e/MiniDebianConfJapan-Yoshi.pdf>

## Contact

**Manuel Traut**

<manuel.traut@linutronix.de>

**Linutronix GmbH**

<https://linutronix.de>

Bahnhofstraße 3  
88690 Uhldingen  
Germany