Isar: Embedded Debian Development 2025

Baurzhan Ismagulov

Isar Community Meetup
Dec 8, 2025
Nuremberg, Germany

About

- Baurzhan Ismagulov
- ilbers GmbH
- Embedded software development
  - Boot loader, kernel, drivers, applications
- Isar introduction, migration, and trainings
- Isar maintenance
  - Patch review, testing and CI
  - Feature development
  - User support on mailing list and github

Timeline

2004 Custom build system in bash

2011 First bitbake implementation

2016 Open-source project

2025 Isar 0.11

Isar 0.11

- Updated distro support: Trixie, Noble, Sid
- Installer
- Filesystem types: squashfs, hybrid iso9660
- Machines: x86 PC, BeaglePlay
- mmdebstrap
  - Bootstrapping from multiple sources
  - Could be used without sudo (not in Isar yet)
- Reproducibility and stability improvements
- bitbake, OE-core libs, wic, kas, kernel version updates
- Cross-compile by default

After 0.11 and Ongoing

- bubblewrap
  - A step towards dropping sudo
- dracut – talk by Clara Kowalsky
- SBOM
  - debsbom: On the list – talk by Christoph Steiger, Felix Mössbauer
  - scancode
- Kernel module signing
- Testsuite improvements
  - make world
  - trixie testcases
  - env testsuite: nop, bitbake, build prebuilt-deb
- Security advisory: CVE-2025-65100 affects snapshot users
  https://github.com/ilbers/isar/security/advisories/GHSA-3r9w-6cp6-7hm4

From the Field

Wants:

- Mostly ARM, some x86
- Debian, newly Ubuntu
- OTA updates: swupdate, RAUC
- Secure boot; some EFI on ARM
- Encryption, some integrity checking
- Platform support
- Same tools for different platforms
- Multiple images
- Testsuite

Industries:

- Embedded hardware
- Industrial PC
- Automotive tools
- Railway
- Telecommunications
- Motors & control systems
- Industrial equipment
- Electrical equipment

Inputs from Downstreams

- Machine support
- Adding applications
- Downstream testing
- kas or bitbake?

1. Machine Support

- Want: Several platforms, same tools for different platforms
  - Debian: Traditionally strong x86 support
  - Possible approach: Provide only Debian for both x86 and ARM?
  - Platform support
    - Isar: No goal to be a BSP
    - Porting BSPs to Isar: Manageable effort
- Vendors need to provide Yocto
- Debian: Overall package attractive for end product builders
  - Don’t build from scratch
  - Many packages tested together
  - Security advisories

Debian Offerings

- Civil Infrastructure Platform
  https://gitlab.com/cip-project/cip-core/isar-cip-core
- Industrial platform
  https://eci.intel.com/docs/2.5/building.html
- Debian SDK
  https://www.nxp.com/design/design-center/software/embedded-software/linux-software-and-development-tools/nxp-debian-linux-sdk-distribution-for-i-mx-and-layerscape:NXPDEBIAN
- Industrial computer
  https://github.com/siemens/meta-iot2050
- Xenomai real-time framework
  https://gitlab.com/Xenomai/xenomai-images

2. Adding Applications

- Add sources
- Create an empty recipe file
- What next?

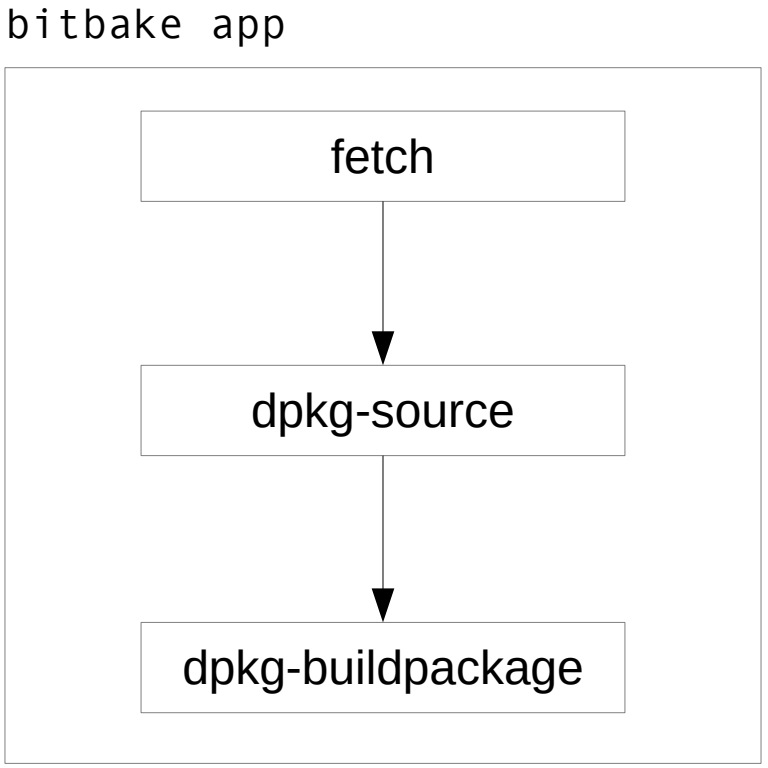
meta-product

```
recipes-app
├─app
│ └─files
│   ├─Makefile
│   └─app.c
└─app.bb
```

Application Building

- Fetch source repo
- Build Debian source package
- Build Debian binary packages

- Missing: Debianization



Application Debianization

- Debianize before building in Isar:
  - mkdir app-0.1
  - cd app-0.1
  - Create Debian package template: dh_make
  - Edit debian/* files to taste
- Test manually:
  - dpkg-source -b .
  - Source package:
    - hello_0.1.dsc
    - hello_0.1.tar.gz
  - dpkg-buildpackage -uc -us
  - Binary package:
    - hello_0.1_amd64.deb

app-0.1

```
app
├─debian
│ ├─changelog
│ ├─control
│ └─rules
├─Makefile
└─app.c
```

Adding Applications: Classic Way

- Layout
  - Sources in app repo
  - Debianization in app repo
  - Recipe in layer
  - Class: dpkg
- Example: libhello
- If unsure, use this

app.git

```
app
├─debian
│ ├─changelog
│ ├─control
│ └─rules
├─Makefile
└─app.c
```

meta-product

```
recipes-app
└─app
  └─app.bb
```

app.bb

```
SRC_URI = ”git://app.org/app.git”
SRCREV = ”0123456789abcdef”

inherit dpkg
```

Adding Applications: Debianize On the Fly

- Layout
  - Sources in layer
  - Debianization on the fly
  - Recipe in layer
  - Class: dpkg-raw
- Example: example-raw
- If unsure, book a training :)

app.bb

```
SRC_URI = “ \
  file://Makefile \
  file://app.c \
”
inherit dpkg-raw

do_install() {
  install -v Makefile ${S}
  install -v app.c ${S}
}

do_prepare_build() {
  echo app /usr/bin \
    >${S}/debian/${PN}.install
  deb_debianize
}
```

4. Downstream Testing

- Isar testsuite provides classes for reusing downstream
- Testsuite runs from command line or CI environment
- Summary provided in json, tap, xml formats
- Jenkins
- GitLab
- Azure
- LAVA – talk by Diogo Mattioli, Lukas Rabener

Testcases

- Image build and execution testcases

meta-product/test/build_test.py

```
#!/usr/bin/env python3
from cibase import CIBaseTest
class BuildImage(CIBaseTest):
  def test_build_image(self):
    targets = [ 'isar-image-base', ]
    self.init(build_dir='/work/build')
    self.perform_build_test(targets, compat_arch=False)
```

meta-product/test/exec_test.py

```
#!/usr/bin/env python3
from cibase import CIBaseTest
class ExecImage(CIBaseTest):
  def test_exec_image(self):
    self.init(build_dir='/work/build')
    self.vm_start('amd64', 'bookworm', skip_modulecheck=True)
```

Running Tests

- Testcases can be run e.g. from kas

```
$ ./kas-container shell --command /work/isar/scripts/ci_setup.sh kas-icm2025.yml
builder@a60c84d6b954:/work/build$ export PYTHONPATH=${PYTHONPATH}:${TESTSUITEDIR}
```

```
builder@a60c84d6b954:/work/build$ avocado run /work/test/build_test.py:BuildImage.test_build_image
JOB ID     : 1af33f962e21a8a87aa76a0281a4c2dd88209b6e
JOB LOG    : ~/avocado/job-results/job-2025-12-07T12.00-1af33f9/job.log
 (1/1) /work/test/build_test.py:BuildImage.test_build_image: STARTED
 (1/1) /work/test/build_test.py:BuildImage.test_build_image: PASS (1.15 s)
RESULTS    : PASS 1 | ERROR 0 | FAIL 0 | SKIP 0 | WARN 0 | INTERRUPT 0 | CANCEL 0
JOB TIME   : 1.73 s
```

```
builder@a60c84d6b954:/work/build$ avocado run /work/test/exec_test.py:ExecImage.test_exec_image
JOB ID     : 6cf1bf5182c884d0354208fcabbbd3688ae63ea0
JOB LOG    : ~/avocado/job-results/job-2025-12-06T21.31-6cf1bf5/job.log
 (1/1) /work/test/exec_test.py:ExecImage.test_exec_image: STARTED
 (1/1) /work/test/exec_test.py:ExecImage.test_exec_image: PASS (38.61 s)
RESULTS    : PASS 1 | ERROR 0 | FAIL 0 | SKIP 0 | WARN 0 | INTERRUPT 0 | CANCEL 0
JOB TIME   : 39.12 s
```

4. kas or bitbake?

- Logic implementation: kas or bitbake?
- bitbake:
  - Dependencies
  - Multiconfigs

kas-icm2025.yml

```
header:
  version: 14

build_system: isar

distro: debian-trixie
target: isar-image-base
machine: qemuamd64

repos:
  meta-icm2025:
    layers:
      .:

  isar:
    url: https://github.com/ilbers/isar
    commit: 2efd5d4ca3b4abf2386fe0089594029becdf2801
    layers:
      meta:
      meta-isar:
      meta-test:

local_conf_header:
  app: |
    IMAGE_INSTALL += "app"
  multiconfig: |
    BBMULTICONFIG += "qemuamd64-trixie"
  ssh-tests: |
    IMAGE_INSTALL += "isar-ci-ssh-setup"
```

Future Work

- Pre-fetch packages, bootstrap from local files
- Persistent apt frontend for sstate-cache
- Support Debian dependencies at bitbake level
- Drop sudo
- Documentation

Summary

- Vendors provide Yocto, Isar integration downstream
  - Debian offerings growing
- From C to Isar: Debianize
  - Prefer application repos
- Use Isar CI classes to quickly add testcases downstream
- Which logic in kas, which in bitbake? To be analyzed

Questions?