5 FACTORS
to build a production grade
Yocto distro
Embedded / IoT Devices

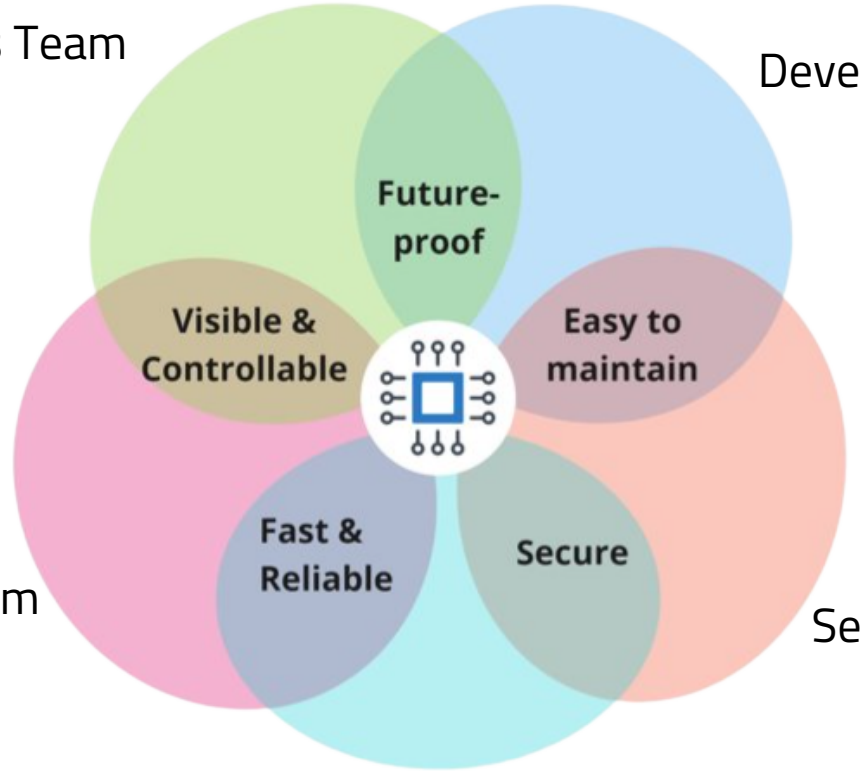
Witekio

AN AVNET COMPANY

A checklist with 5 success factors to deploy Yocto for production-grade embedded devices. The aim is to provide a 360° view.

Busines Team

Development Team



DevOps Team

Security Team

Your Customers
& Everyone

Witekio

AN AVNET COMPANY

Devices should be...

Easy to Maintain

Simple environment

- [ ] **Provide dev. environment**
  - Git, python, repo, ...
  - Container and companion scripts
  - Prefer prebuilt container image
    - Stable environment and package versions
- [ ] **Document the main dev. workflow**
  - Essential to newcomers and to share knowledge within the team
  - Crunch time = no time for doc but help wanted!
- [ ] **Enable build history**
  - Easier troubleshooting in case of regression
    - INHERIT += "buildhistory"
      BUILDHISTORY_COMMIT = "1"
- [ ] **Shrink build times**
  - Share downloads and sstate (NFS, SSHFS, ...)
    - site.conf:
      DOWNLOAD_DIR =
      SSTATE_DIR =
      SSTATE_MIRRORS =
  - Ask for a beefier machine 😇
- [ ] **Central package server (Nice to have)**
  - Prebuild and share packages
    - bitbake world
    - Run a package server
    - Use PR service to version packages

Devices should be...

Easy to Maintain

Reproducible builds

- [ ] **Automate Yocto builds**
  - Nightlies: relatively easy, cache downloads and state
  - Automate from cloud, build on premise
    - GitLab runners, Azure DevOps self-hosted agents
  - Pull request validation: highly valuable
    - Trigger build from another repo, override package or layer
    - SRCREV_pn-$PACKAGE_NAME =

- [ ] **Version your OS and pin everything**
  - Version code, layers, and configuration
    - repo, git submodules, ...
  - Pin meta-layers tags or commit
  - Tag your OS
  - Update “os-release”
    - Version, type of build, ...

- [ ] **Archive release build environment (Nice to have)**
  - Yocto: Reproducible Builds affiliated ✔
  - Your setup 1 year from now: likely KO ☹
    - Main culprit: external tools
  - Archive source, tools, environment
    - Downloads
    - VM or prebuilt image
  - Annotate your release with its build environment
  - Keep the build manifest: layers & packages version hash

Devices should be...

Secure

OS Security features

- [ ] **Minimize attack surface**
  - Define prod and prod-secure images
  - No dev/debug tools, or unnecessary packages
    - Image from scratch, core-image-minimal
    - Disable recommended packages
  - No root login, proper users, firewall
  - Disable serial, JTAG,USB (or exceptions)
    - meta-security

- [ ] **Provide a secure secret store**
  - TPM, Secure Element, TrustZone-based
  - Device keys, credentials, secrets

- [ ] **Use Secure boot**
  - BootROM, SSB, U-Boot, Kernel/initramfs/dtb
  - Rootfs if possible / readonly
    - dm-verity
  - Use key hashes, 1-2 backup secure boot keys, and support revocation
    - ... And test it!

- [ ] **Encrypt disks, prevent writes (Nice to have)**
  - Encrypt partitions, LUKS, dm-crypt, and secret store
  - Read-only filesystem (SquashFS), or mounted read only

Devices should be...

Secure

Other features

- [ ] **Apps: Least privilege principle**
  - MAC: SELinux, AppArmor
  - seccomp, cgroups, chroot, runc
  - Containers, AWS Greengrass, Azure IoT Edge
- [ ] **Monitor and address vulnerabilities**
  - INHERIT += “cve-check”
  - meta-timesys provides a few tools
  - Complex and time consuming
  - No package = no exploit
- [ ] **Automate on-device identity provisioning**
  - Unique x.509 certificates
    - Signed from an intermediate certificate
    - Or pre-provisioned in a secure element
- [ ] **Consider standards and regulations (Nice to have)**
  - Likely to become more important
  - Europe: ETSI EN 303 645
  - US: NIST 8259A
- [ ] **Avoid or automate device provisioning in the cloud**
  - Azure DPS, AWS
  - Option A. Authorize an intermediate cert
    - Used to sign devices, done once per intermediate cert
  - Option B. Pre-provisioned secure elements
    - Done during chip manufacturing, from a secure software factory (TO136)
  - Option C. Automate and secure
- [ ] **Support rolling device certs, and updating root CAs**
  - Software update or otherwise
  - Root CAs too!

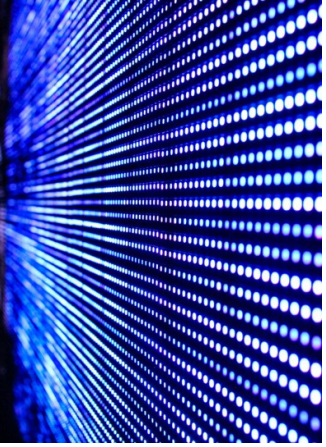
Devices should be…

Fast and reliable

Boot and runtime

- [ ] **Fast (enough) Boot time**
  - Start from a minimal image
  - Compile for size, link statically, strip binaries, use to musl or uCLibc
  - Postpone drivers and services
  - Btrfs, squashfs
  - … or just dump a small logo/animation from the U-Boot!
- [ ] **Fast (enough) & Responsive UX**
  - Compile for speed ▪ Leverage cores, CPU instructions, priority
  - 2D, 3D, and video hardware acceleration
  - Crypto hardware acceleration
  - Accelerated libs: GUI, AI inference, …
- [ ] **Yocto and test automation**
  - Based on AutoBuilder2, helper, and Buildbot
    - Not necessarily a good fit for you
- [ ] **Automate on-device tests**
  - Ptest, ptest-runner
    - DISTRO_FEATURES:append = " ptest"
      EXTRA_IMAGE_FEATURES += "ptest-pkgs“
  - LTP (Linux Test Project)
    - IMAGE_INSTALL:append = " ltp“
  - And device-specific tests
  - Automate with Labgrid, Pluma, Fuego, Lava, Buildbot, KernelCI
  - Integrate with GitLab runners

- [ ] **Automate tagging and deployment (Nice to have)**
  - Manual release “trigger”, automated release
  - More consistent, less errors
  - Can tag, archive build environment, …
  - Push to your OTA update backend and/or package host



Devices should be...

Visible (Observable)

Open -source licenses

- [ ] **Ensure OS licenses compliance**
  - Save manifest of all license, and ship in the binary
    - COPY_LIC_MANIFEST = "1"
    - COPY_LIC_DIRS = "1"
    - LICENSE_CREATE_PACKAGE = "1"
  - Archive source
    - INHERIT += "archiver"
      ARCHIVER_MODE[src] = "original" # OR
      ARCHIVER_MODE[src] = "patched"
      ARCHIVER_MODE[diff] = "1"
  - Some references:
    - Yocto's manual regarding compliance
    - OpenChain ISO 5230, Open Compliance Program
    - FOSSology license tracker
    - Various commercial tool

Devices should be...

Controllable

Over-the-air updates

- [ ] **Support OS updates**
  - OSTree, RAUC, Mender, swupdate
    - And their respective meta layers
    - Avoid non-atomic update like apt
  - Kernel, packages, ...
  - Sign your updates, look for delta updates

- [ ] **Support Application updates (Nice to have)**
  - Different frequency, more flexibility & less bandwidth
  - RAUC, Mender, OSTree, or managed deployment
    - Azure IoT Edge, AWS Greengrass, ...

- [ ] **Provide OTA update online dashboards**
  - Hawkbit, Mender, ThingsBoard, Full Metal Update

- [ ] **Provide a fallback mechanism**
  - What happens the OS, or update mechanism is KO?
  - Recovery initramfs, A/B, golden/base image
  - Or a combination of those
  - Test and re-test

Devices should be...

Controllable

Remote operations

- [ ] **Support remote device configuration**
  - Device twins… again!
  - Cloud “desired” configuration, stored in same JSON
    - Desired network config, logging mode, CPU throttling (hot device), …
  - Received whenever connected, and persistent

- [ ] **Support arbitrary operations (Nice to have)**
  - Provide a quick flexible way to support any operation
    - Specific repair job, test or experiment new features
  - Running jobs and applications from container
    - Azure IoT Edge, AWS IoT jobs, AWS IoT Greengrass

Manual control and debugging

- [ ] **Provide remote manual access**
  - Reverse SSH, OpenVPN/IPsec/wireguard, ngrok
  - As well as local access: GUI, USB drive script
  - Enabled on demand, for a limited duration
  - Update firewall rules, authorized devices, ...
  - or reboot in maintenance mode
  - After a secure call from your cloud platform and/or physical interaction

- [ ] **Support remote troubleshooting & debugging (Nice to have)**
  - Is it acceptable to ship the device back?
  - LTTng (meant for prod), ftrace,
  - Generate and save debug symbols
    - IMAGE_GEN_DEBUGFS = "1"
  - Install or run 'gdb-server' on-demand

Devices should be...

Reusable & Future proof

Yocto and reusing OS-level work

- [ ] **Limit "hacky" customizations**
  - Limit bbappends
  - Avoid patching more than a few files in one recipe
  - Tradeoff between upstream improvements, and maintenance

- [ ] **Prefer reusable meta and config**
  - Separate layers to allow reuse
    - Hardware, software, platform-specific features
  - WIC: custom wks and partitions
    - More tools to interact, customize and introspect
  - Device tree includes and overloads
  - LTS when it makes sense

- [ ] **Prepare by upgrading Yocto**
  - Once you know which is the ideal Yocto version
  - Otherwise very hard to reuse meta/recipes/classes across different Yocto versions

- [ ] **Contribute layers, recipes, classes (Nice to have)**
  - Generic layers, recipes, classes
    - Chances are others need it, and will help maintain them
    - Submit it to https://layers.openembedded.org/
  - Yocto features & issues
    - create-pull-request, send-pull-request
    - bugzilla.yoctoproject.org

Devices should be…

Reusable & Future proof

Application-level

- [ ] **Abstract device & OS specificities**
  - App software architecture
  - Rely on standard file location and mechanics
  - Makes development easier

- [ ] **Ensure application(s) modularity**
  - Core logic, connectivity, UI, GUI, storage
  - At a component level minimum:
    - Source components, plugins, or services
  - Future: headless, gateway, split in 2 devices

- [ ] **Self-contained application (Nice to have)**
  - Containers, snapd/flatpak, self-contained binaries (Golang, Rust, …)
  - Much easier to deploy/reuse, but larger
  - System or app update, managed services
    - Azure IoT Edge, AWS Greengrass, …

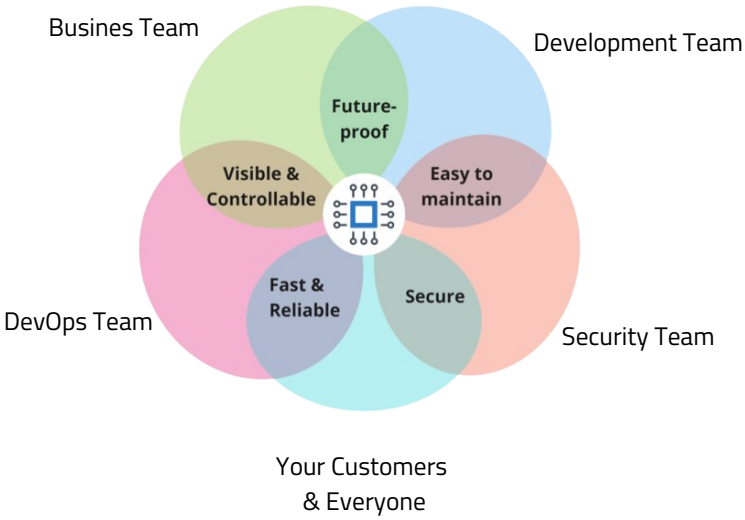
- [ ] **Use standard tools and protocols**
  - Most of the time, something already exists
  - Prefer what the industry uses (most of the time)
    - Conferences, blogs, Yocto mailing list, Gartner, …
    - yocto@lists.yoctoproject.org
  - When applicable, prefer standards for interop.
    - Matter, BLE profiles, …

Summary

Use the checklist before release. The sooner the better!

The 6th Success factor: the human one, YOU

- Contribute to its success, see something others didn't see
- Internally: review, ask, and suggest features
- Learning opportunity



Contact Us for more info

Witekio
AN AVNET COMPANY