Yocto Multiverse

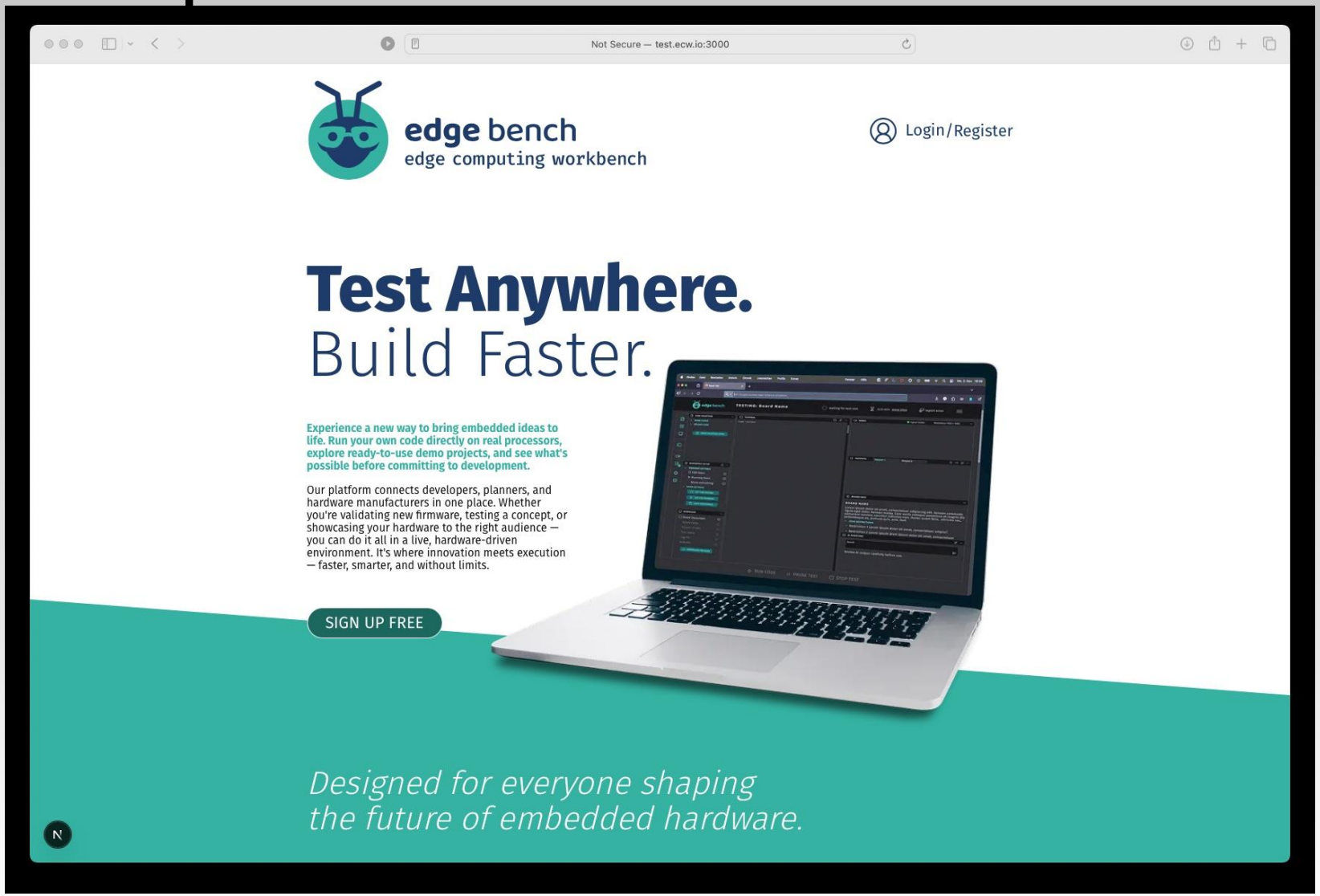
Powered by OpenEmbedded

Building a Distro That Survives Parallel Hardware Realities

Background

This universe is only one of an infinite number

Edge Compute Workbench



Edge Computing Workbench

- Provide remote access to real hardware
  - Console
  - Video
  - Upload and test code on real boards
  - Selection of hardware
    - SBCs
    - MCUs
    - Attached peripherals

Potential Boards

- Beaglebone Black
- PocketBeagle2
- Odroid N2+
- BeagleV-Fire
- STM32MP157F-DK2
- AmpereOne

- Basically – anything I have lying around in my office

Requirements

- Single image for all Linux systems
- Scalable
- Maintainable
- Flexible
- Not insecure

- May as well just list the features of Open Embedded!

Potential Problems

- Multiple board types
- Totally different architectures
  - ARM
  - ARM64
  - RISC-V
  - x86_64
- Different bootloaders
  - u-boot
  - grub

Design

We don’t get to choose our time

Design Choices

- Scarthgap
  - Want LTS stability to start with
    - May want to offer the Bleeding Edge later
- Use Standard Machine layers
  - Showcase the boards not the OS
- Bespoke Distro
  - Poky is a reference distribution!
- Bespoke Images
  - Want core-image-minimal +extras
    - Performance Tools
    - Demo code

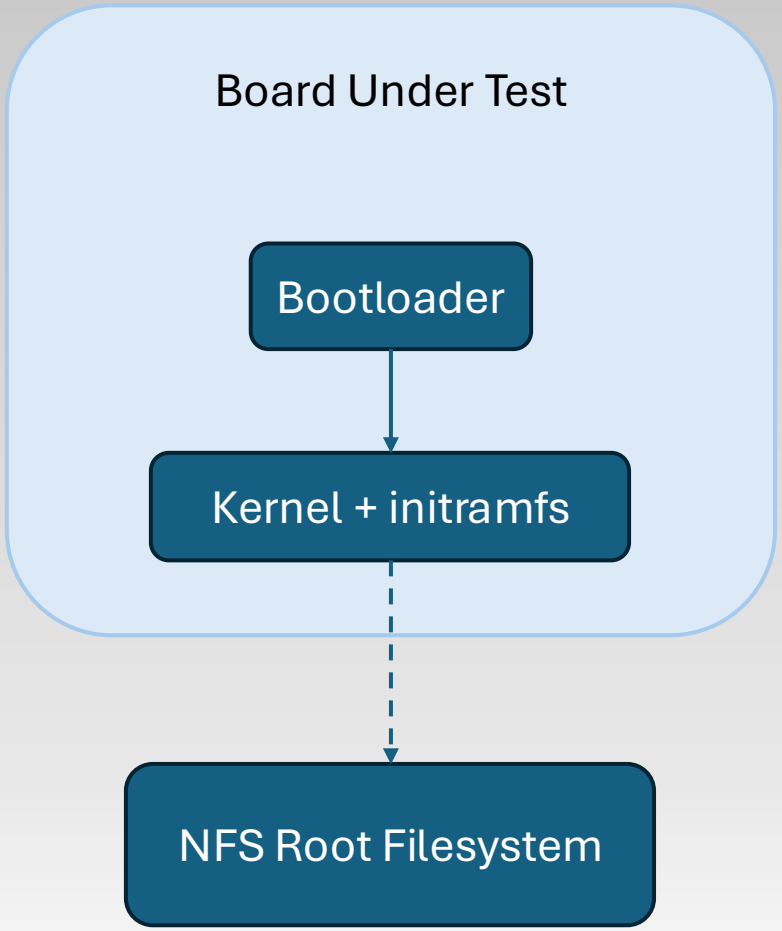
Design – NFS Rootfs

- Do not want to keep anything local on the board
  - Would have to re-flash between each use to clean it
- Faster to change the complete rootfs between users
- Simplifies uploading files
  - No need for SSH on the board images
- Allows simple persistence of user files between sessions

Design - initramfs

- Use initramfs to find and mount the rootfs
  - Cross-platform
  - Bootloader independent
  - Simple(?) to setup

Design Overview



Design - Setup

- Using KAS for consistency
  - May move to bitbake-setup
    - Make use of fragments

- Main + Board Specific
  - Use the kas combinations to select the board

```
~/kas/run-kas shell ./ecw.yaml:./kas/includes/beaglev-fire.yaml
```

Design - Distribution Configuration

- Keep it clean and simple
- Make use of ‘soft assignments’ (?= and ?==) to allow for overrides elsewhere
- Implement personal prejudices
  - Systemd
  - ipk

Implementation

If I tell you what happens, it won’t happen

Main Configuration 1/3

```
header:
  version: 19

repos:
  poky:
    url: https://git.yoctoproject.org/poky
    branch: scarthgap
    path: layers/third-party/poky
    layers:
      meta:
      meta-poky:
```

Main Configuration 2/3

```
meta-openembedded:
  url: https://github.com/openembedded/meta-openembedded.git
  branch: scarthgap
  path: layers/third-party/meta-openembedded
  layers:
    meta-oe:
    meta-initramfs:
    meta-python:
    meta-networking:

meta-ecw:
  path: layers/project/meta-ecw
  layers:
    meta-ecw-core:
```

Main Configuration – 3/3

```
local_conf_header:
  General: |
    PACKAGE_CLASSES = "package_ipk"
    SSTATE_DIR = "${HOME}/yocto/share/sstate-cache/${MACHINE}"
    DL_DIR = "${HOME}/yocto/share/downloads"
```

Distro – Key Configs: Prejudices

```
MICROFORGE_DEFAULT_DISTRO_FEATURES = "usrmerge systemd systemd-resolved networkd"

PACKAGE_CLASSES ?= "package_ipk"
IMAGE_FSTYPES += "tar.bz2 ext4 "

MICROFORGE_INIT_MANAGER = "systemd"
INIT_MANAGER = "${MICROFORGE_INIT_MANAGER}"
```

Distro – Key Configs: Initramfs

```
# Set up the intramfs
INITRAMFS_IMAGE ?= "ecw-image-minimal-initramfs"
INITRAMFS_FSTYPES ?= "cpio.gz"
INITRAMFS_IMAGE_NAME ?= "${INITRAMFS_IMAGE}-${MACHINE}"
INITRAMFS_IMAGE_BUNDLE ?= " 1"
```

- Sets the name of the initramfs recipe
- Defines the filesystem type that is built
- Sets the name of the final image
- Defines to include the image in the kernel or not

Allow for Machine Specific Options

- Keep extra config options ordered by machine name for easier organisation

```
meta-ecw-core/
  conf/
    distro/
      microforge.conf
      microforge/
        <machine name>/
              machine-distro-extras.inc
```

```
include conf/distro/${DISTRO}/${MACHINE}/machine-distro-extras.inc
```

- Use **`include`** not **`require`** as the file may not exist

Note: BBPATH vs FILESPATH

- From the manual:
  - Through the use of the BBPATH variable, BitBake locates class files (.bbclass), configuration files, and files that are included with include and require statements

- This is why you need to need to be explicit when using require and include

Initramfs Framework

- https://docs.yoctoproject.org/dev-manual/building.html#building-an-initial-ram-filesystem-initramfs-image

- Provides helpers for many configurations
  - LVM
  - Debug
  - **NFS Root**

- Easily extensible

Initramfs – Key changes from Base

- Selection of INITRAMFS_SCRIPTS
  - Remove
    - initramfs-module-setup-live
    - initramfs-module-install
  - Add
    - initramfs-module-ecw-nfsroot

- Add `riscv64` to COMPATIBLE_HOST

Initramfs Recipe – Changes

```
DESCRIPTION = "Small image capable of booting a device. \
Based on layers/third-party/poky/meta/recipes-core/images/core-image-minimal-initramfs.bb"

INITRAMFS_SCRIPTS ?= "\
                initramfs-framework-base \
                initramfs-module-udev \
                initramfs-module-install-efi \
                initramfs-module-ecw-nfsroot \
"

PACKAGE_INSTALL = "${INITRAMFS_SCRIPTS} ${VIRTUAL-RUNTIME_base-utils} udev base-passwd
${ROOTFS_BOOTSTRAP_INSTALL}"
...
# Use the same restriction as initramfs-module-install
COMPATIBLE_HOST = '(x86_64.*|i.86.*|arm.*|aarch64.*|loongarch64.*|riscv64.*)-(linux.*|freebsd.*)'
```

nfsroot script

- Installed to `/init.d/80-ecw-nfsroot.sh` in the initramfs
- Checks bootargs for `root=/dev/nfs` and existence of `server_ip=`
- The main changes from the base (`poky/meta/recipes-core/initrdscripts/initramfs-framework/nfsrootfs`)
  - Uses the MAC Address of the board for the NFS root path
  - Specify NFS v4 in mount options to simplify the firewall rules
- Can extend later to add additional checks, etc

Core Image

- The core image is common to all boards
  - Mainly performance monitoring tools

- Adds the dependency of the initramfs image

```
IMAGE_DEPENDS = "ecw-image-minimal-initramfs"
```

Allow for Machine Specific Options again

- Keep extra recipes ordered by machine name for easier organisation

```
include conf/distro/${DISTRO}/${MACHINE}/machine-distro-extras.inc
```

- Again, use **`include`** not **`require`** as the file may not exist

- Allows for sample applications and tools specific to that board type

Configure

The only thing you can do is be the best version of yourself

Machine YAML files

- Contain
  - Additional repositories needed to support the machine
  - machine definition
  - Any local configuration options specific to the machine

Example - Generic ARM64

```
header:
  version: 11

repos:
  poky:
    layers:
      meta-yocto-bsp:

machine: genericarm64

local_conf_header:
  InitramFS: |
    INITRAMFS_IMAGE_BUNDLE = "1"
```

Extend the poky definition to include the meta-yocto-bsp layer

Include the initramfs in the Kernel

Exmaple - Odroid N2+

```
header:
  version: 11

repos:
 meta-meson:
  url: https://github.com/superna9999/meta-meson
  branch: scarthgap
  path: layers/third-party/meta-meson

machine: hardkernel-odroidn2plus

local_conf_header:
 ODroid: |
  KERNEL_IMAGETYPE = "fitImage"
  KERNEL_CLASSES:append = " kernel-fitimage"
```

Need to include an additional layer to support the board

Use a fitImage to hold the initramfs

Deploy

The bill comes due. Always

NFS Root Mount Point

- Each board needs to have a unique mount point
  - Use the MAC Address from the system
  - Determined by the init script

- Specified for each board in the
  - `bootargs` u-boot variable
    - bootargs ‘root=/dev/nfs server_ip=…
  - (boot)/extlinux/extlinux.conf
    - append root=/dev/nfs server_ip=…

Image Creation

- Building the image results in two sets of files
  - The core image files

    ecw-core-<MACHINE>*
  - If using FIT Images

    fitImage-<Kernel-version>*

  - The initramfs files

    ecw-image-minimal-initramfs-<MACHINE>*
  - If using FIT Images
    - fitImage-ecw-image-minimal-initramfs-<MACHINE>
- Make sure you are using the right ones in the right places

Initramfs

- For the initramfs bundled into the kernel
  - Use the zImage-initramfs-<MACHINE> **NOT** the zImage

- For a fitImage
  - Use the fitImage file with the initramfs in the name
  - Note: fitImage-its-* is the image *configuration* file
    - If you copy that to the board, it will not boot
      - **Sometimes you need a break and more coffee!**

Root filesystem

You can now take `ecw_core-<MACHINE>.tar.bz2` from the `deploy/images/<MACHINE>` direrectory and extract it to your NFS server

I use a bind mount to mount a rootfs for each user/board combination

Remember to export the rootfs directory no_root_squash

Conclusions

You can’t control everything. Sometimes you just have to let go.

Image Sanity Check

- Compare the manifest files in the deploy directory
  - These were the same with the exception of differing kernel and kernel modules
- All boards should therefore function in a similar way

Discoveries

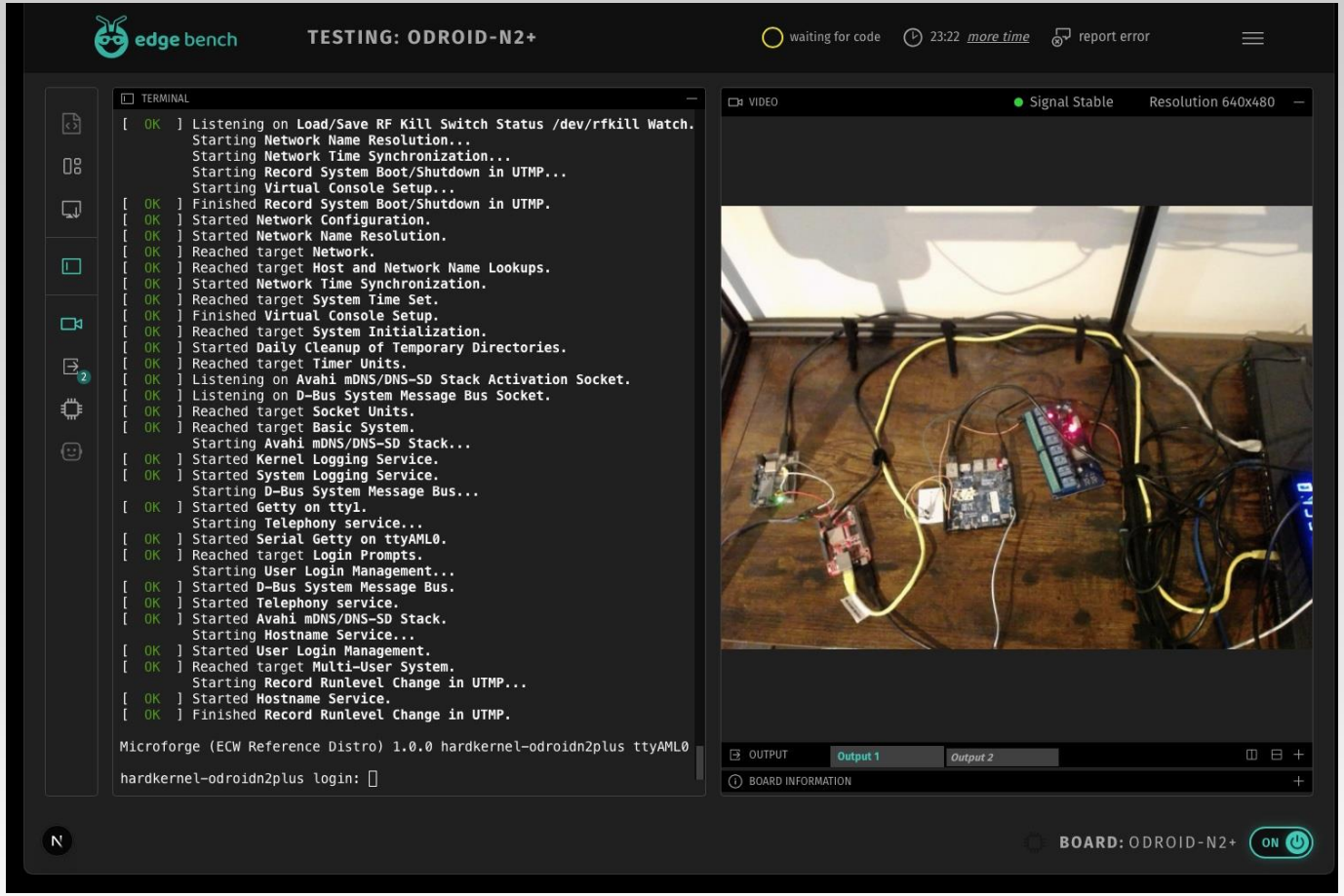
- Expectations of difficulty
  - Not being familiar with all platforms
  - Having too many war stories on board bring-up

- Reality
  - It was actually quite straight forward

Future Enhancements

- Use the MAC Address of the board are the mount point ID
  - Would need more admin on the server side, but then makes the image easier to maintain
  - Done since I wrote the slides!
- Try the mainline kernel meta-linux-mainline
  - Have more consistency across boards
- Increase the number of supported boards
  - Always want to know what boards people want to use
- Integrate Zephyr into the builds for MCU boards

Working Proof of Concept



Questions?

Every decision makes sense once you ask the right question