

Silicon Blade

Yocto Multiverse

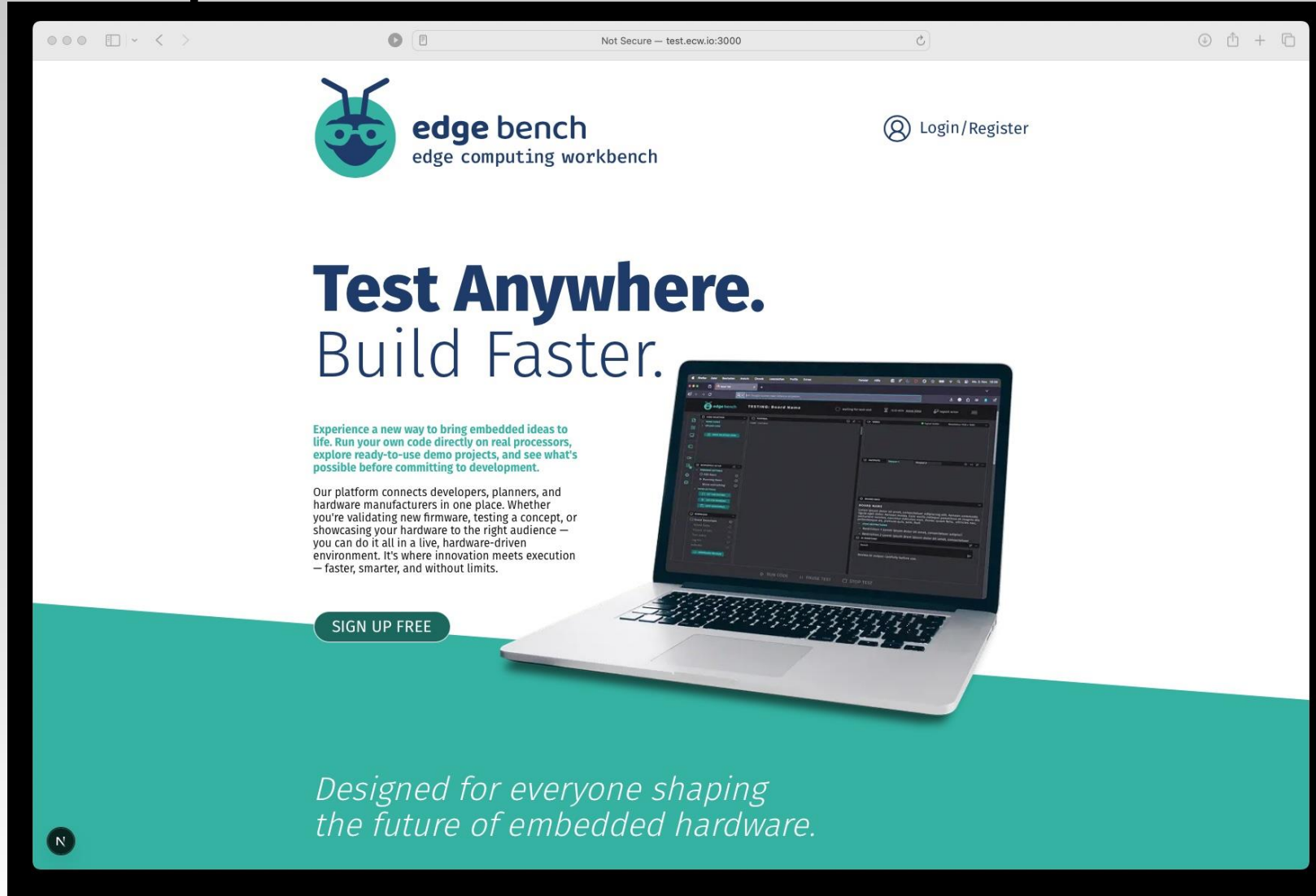
Powered by OpenEmbedded

Building a Distro That Survives Parallel Hardware Realities

Background

This universe is only one of an infinite number

Edge Compute Workbench



The image shows a browser window displaying the Edge Compute Workbench website. The browser's address bar shows "Not Secure — test.ecw.io:3000". The website features a teal and white color scheme. At the top left is the logo, a stylized green robot head with glasses, followed by the text "edge bench" and "edge computing workbench". At the top right is a "Login/Register" link with a user icon. The main heading reads "Test Anywhere. Build Faster." Below this is a paragraph: "Experience a new way to bring embedded ideas to life. Run your own code directly on real processors, explore ready-to-use demo projects, and see what's possible before committing to development." Another paragraph follows: "Our platform connects developers, planners, and hardware manufacturers in one place. Whether you're validating new firmware, testing a concept, or showcasing your hardware to the right audience — you can do it all in a live, hardware-driven environment. It's where innovation meets execution — faster, smarter, and without limits." A teal button with white text says "SIGN UP FREE". To the right of the text is a laptop displaying the workbench interface. At the bottom, a teal banner contains the text "Designed for everyone shaping the future of embedded hardware." and a small circular logo with the letter "Z" in the bottom left corner.

edge bench
edge computing workbench

Login/Register

Test Anywhere. Build Faster.

Experience a new way to bring embedded ideas to life. Run your own code directly on real processors, explore ready-to-use demo projects, and see what's possible before committing to development.

Our platform connects developers, planners, and hardware manufacturers in one place. Whether you're validating new firmware, testing a concept, or showcasing your hardware to the right audience — you can do it all in a live, hardware-driven environment. It's where innovation meets execution — faster, smarter, and without limits.

[SIGN UP FREE](#)

*Designed for everyone shaping
the future of embedded hardware.*

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

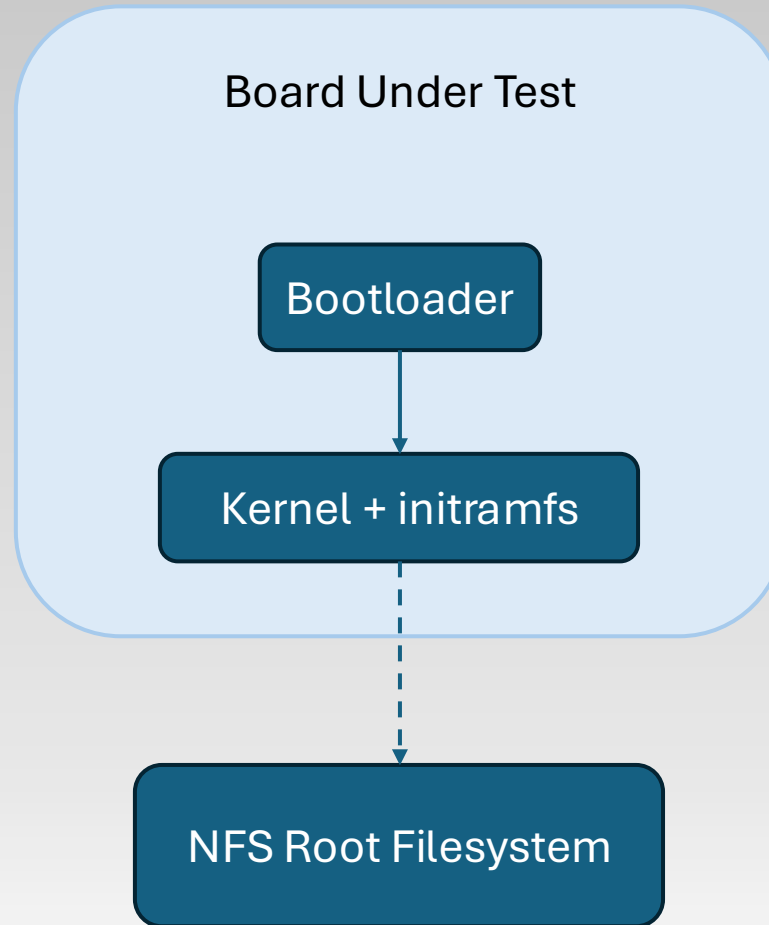
Design – NFS Rootfs

- 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:
```

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



```
machine: genericarm64
```

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

Include the initramfs in the Kernel



Exmample - Odroid N2+

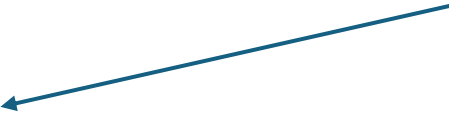
```
header:
  version: 11

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

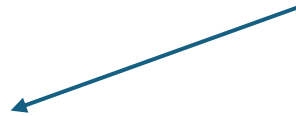
machine: hardkernel-odroidn2plus

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>` directory 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

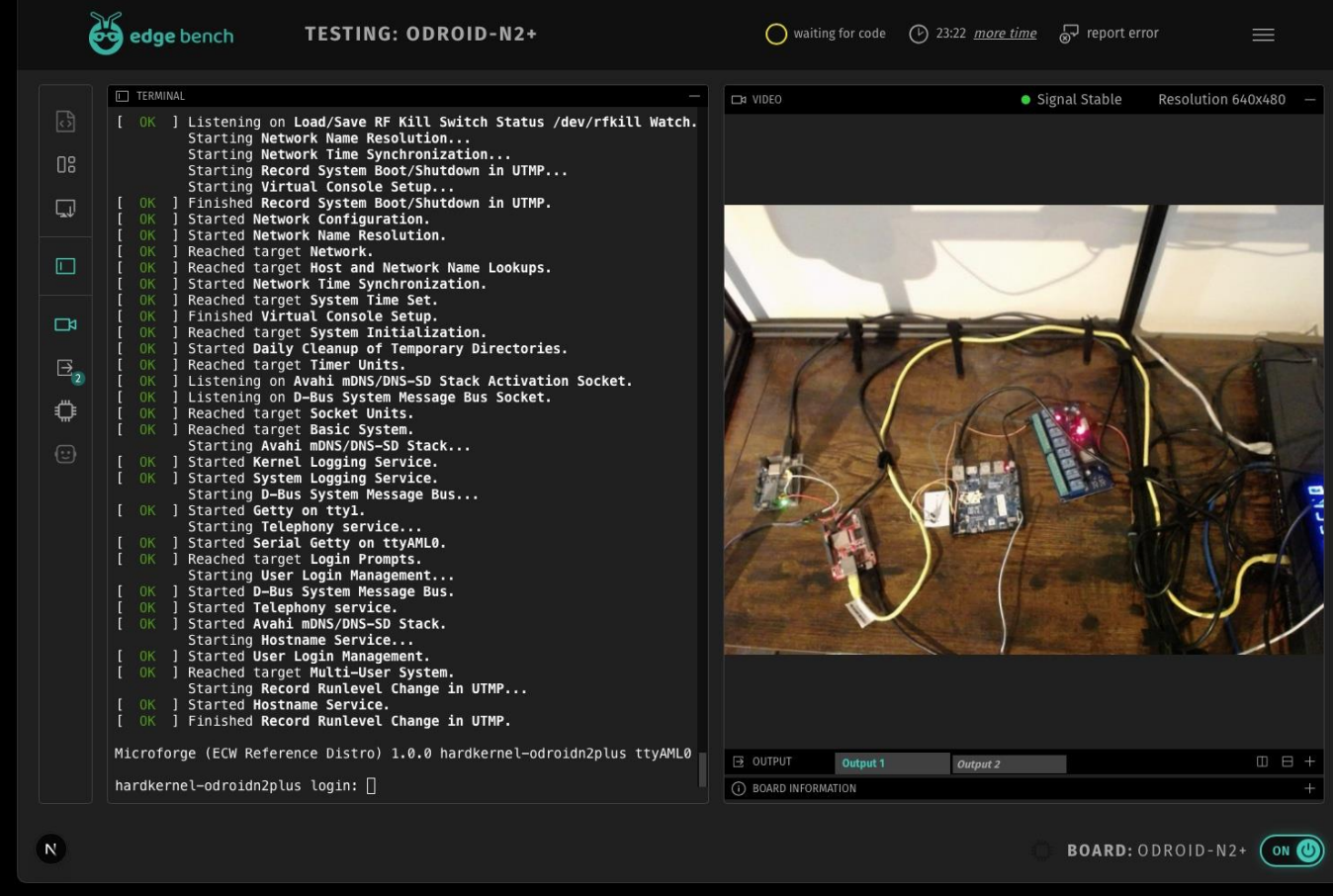
Discoveries

- 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 as 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



The screenshot displays the edge bench interface for testing an ODROID-N2+ board. The interface is divided into several sections:

- Header:** "edge bench" logo, "TESTING: ODROID-N2+", "waiting for code", "23:22 more time", and "report error".
- Terminal Window:** Shows the system boot process. The output includes:

```
[ OK ] Listening on Load/Save RF Kill Switch Status /dev/rfkill Watch.  
Starting Network Name Resolution...  
Starting Network Time Synchronization...  
Starting Record System Boot/Shutdown in UTMP...  
Starting Virtual Console Setup...  
[ OK ] Finished Record System Boot/Shutdown in UTMP.  
[ OK ] Started Network Configuration.  
[ OK ] Started Network Name Resolution.  
[ OK ] Reached target Network.  
[ OK ] Reached target Host and Network Name Lookups.  
[ OK ] Started Network Time Synchronization.  
[ OK ] Reached target System Time Set.  
[ OK ] Finished Virtual Console Setup.  
[ OK ] Reached target System Initialization.  
[ OK ] Started Daily Cleanup of Temporary Directories.  
[ OK ] Reached target Timer Units.  
[ OK ] Listening on Avahi mDNS/DNS-SD Stack Activation Socket.  
[ OK ] Listening on D-Bus System Message Bus Socket.  
[ OK ] Reached target Socket Units.  
[ OK ] Reached target Basic System.  
Starting Avahi mDNS/DNS-SD Stack...  
[ OK ] Started Kernel Logging Service.  
[ OK ] Started System Logging Service.  
Starting D-Bus System Message Bus...  
[ OK ] Started Getty on tty1.  
Starting Telephony service...  
[ OK ] Started Serial Getty on ttyAML0.  
[ OK ] Reached target Login Prompts.  
Starting User Login Management...  
[ OK ] Started D-Bus System Message Bus.  
[ OK ] Started Telephony service.  
[ OK ] Started Avahi mDNS/DNS-SD Stack.  
Starting Hostname Service...  
[ OK ] Started User Login Management.  
[ OK ] Reached target Multi-User System.  
Starting Record Runlevel Change in UTMP...  
[ OK ] Started Hostname Service.  
[ OK ] Finished Record Runlevel Change in UTMP.
```

The prompt at the bottom is "Microforge (ECW Reference Distro) 1.0.0 hardkernel-odroidn2plus ttyAML0" and "hardkernel-odroidn2plus login: |".
- Video Window:** Shows a live video feed of the hardware setup. The text above the video reads "Signal Stable" and "Resolution 640x480". The video shows a wooden desk with various components: a red power supply, a blue USB hub, a keyboard, and several cables connected to the board.
- Bottom Bar:** Includes "BOARD: ODROID-N2+" and a power button icon labeled "ON".

Questions?

Every decision makes sense once you ask the right question