Armbian documentation

Linux for ARM development boards

Armbian documentation team

© 2021 by Armbian

Table of contents

1. Welcome to the Armbian Documentation! 4
1.1 What is Armbian? 4
1.2 What is supported? 5
2. Get Involved! 5
3. User Guide 6
3.1 Armbian Quick Start Guide 6
3.2 Quick Facts 13
3.3 Hardware troubleshooting guide 15
3.4 Recovery 18
3.5 Frequently asked questions 20
3.6 Advanced Features 23
3.7 How to customize keyboard, time zone? 27
3.8 Armbian configuration utility 32
3.9 Device Tree overlays 35
3.10 Support Definitions, Criteria and Relationships 39
4. Hardware Notes 41
4.1 Generic howto for Allwinner devices 41
4.2 Allwinner A10 & A20 boards 46
4.3 Allwinner H3 boards 48
4.4 Allwinner H5 and A64 boards 50
4.5 Allwinner H6 51
4.6 Cubox and Hummingboard boards 52
4.7 Marvell Armada 55
4.8 Rockchip RK3399 boards 57
4.9 Rockchip RK3328 / RK 3288 57
5. Developer Guide 58
5.1 Building Armbian 58
5.2 Quick Start with Vagrant 61
5.3 Building with Docker 63
5.4 Build options 65
5.5 User Configuration 69
5.6 FEL/NFS boot explanation 71
5.7 Extensions 73
5.8 Extension Hooks 75

© 2021 by Armbian

6. Contributor Process 79
6.1 Collaborate on the project 79
6.2 Merge Policy 80
6.3 Automatic rebase of Pull requests 82
6.4 Merge request pipelines 82
6.5 Armbian Documentation 85
7. Release management 87
7.1 Release model 87
7.2 Changelog 92
7.3 Jira 143
7.4 Board Maintainers 146
8. Community 152
8.1 IRC Channel / Matrix / Discord 152
8.2 Maintainers 155

© 2021 by Armbian

armbian

Linux for ARM development boards

1. Welcome to the Armbian Documentation!

If you are **new to Armbian**, the *Getting Started* section provides a tutorial for everything you need to get Armbian running and answers many **F**requently **A**sked **Q**uestions. It then continues on to more advanced topics.

If you **need help** and have read through *Getting Started* check out *Troubleshooting*.

If you still cannot find what you need here visit the *Armbian forum* where your input can help to improve this documentation.

1.1 What is Armbian?

Armbian is a base operating system platform for single board computers (SBCs) that other projects can trust to build upon.

- Lightweight Debian or Ubuntu based Linux distribution specialized for ARM development boards
- Each system is compiled, assembled and optimized by *Armbian Build Tools*
- It has powerful build and software development tools to make *custom builds*
- A vibrant community

1.1.1 What is the difference between Armbian and Debian/Ubuntu?

- Debian or Ubuntu officially do not support most of those boards/boxes. Armbian does.
- Armbian userspace has many small but vital performance or security adjustments
- Armbian fancy some kernel development and a lot of its maintaining. Debian relies on upstream sources for ARM hardware which can be **years** behind and/or lack of many functions
- Armbian userspace is lean, clean but 100% Debian/Ubuntu compatible
- Many stock Debian bugs are fixed on the way, “better than original :)”
- The Armbian build system is a central part of this whole ecosystem. You can DIY. Debian is much harder.
- Dedicated support forums per boards/boxes
- Plug’n’Play vs. complicated install scenarios on stock Debian
- unified development scenarios and user experience vs. mess of different setup instructions scattered all around

1.1.2 Common features

- Armbian Linux is available as Debian and Ubuntu based images, compiled from scratch
- Images are reduced to actual data size and automatically expand across the SDcard at first boot
- Root password is `1234`. You are forced to change this password and (optional) create a normal user at first login
- Ethernet adapter with DHCP and SSH server ready on default port (22)
- Wireless adapter with DHCP ready (if present) but disabled. You can use `armbian-config` to connect to your router or create an access point
- NAND, SATA, eMMC and USB install script is included (`nand-sata-install`)
- Upgrades are done via standard `apt upgrade` method

© 2021 by Armbian

- Login script shows: board name with large text, distribution base, kernel version, system load, uptime, memory usage, IP address, CPU and drive temperature, ambient temperature from Temper if exits, SD card usage, battery conditions and number of updates to install

1.1.3 Performance tweaks

- `/var/log` is mounted as compressed device (zram, lzo), log2ram service saves logs to disk daily and on shutdown
- Half of memory is allocated/extended for/with compressed swap
- `/tmp` is mounted as `tmpfs` (optionally compressed)
- Browser profile memory caching
- Optimized IO scheduler (check `/etc/init.d/armhwinfo`)
- Journal data writeback enabled. (`/etc/fstab`)
- `commit=600` to flush data to the disk every 10 minutes (`/etc/fstab`)
- Optimized CPU frequency scaling with `interactive` governor (`/etc/init.d/cpufrequtils`)
- 480-1010Mhz @Allwinner A10/A20
- 480-1368Mhz @Allwinner H2+/H3
- 392-996Mhz @Freescale imx
- 600-2000Mhz @Exynos & S905
- eth0 interrupts are using dedicated core (Allwinner based boards)

1.2 What is supported?

Armbian will publish and distribute “stable” CLI images for supported boards through its mirror network. *Supported* is not a guarantee. *Supported* has a named maintainer and implies a particular SBC is at a high level of software maturity. Due to the complexity and lack-of-openness in the ecosystem it is unlikely that all accelerated and specialized functionalities (like 3D, VE, I²C…) will be available.

For more information is see the Board Support Guide

1.2.1 Supported boards

Check download page for recently supported list.

2. Get Involved!

- Contribute
- Community
- Contact

Our IRC channel is #armbian on Libera.Chat. More details here

© 2021 by Armbian

3. User Guide

3.1 Armbian Quick Start Guide

Mentioned links:

- https://gitlab.com/bztsrc/usbimager/
- https://forum.armbian.com/topic/4767-powering-through-micro-usb/
- https://docs.armbian.com/
- https://forum.armbian.com/profile/9032-werner/
- https://forum.armbian.com/topic/12803-armbian-irc-chat/

3.1.1 Prerequisites for new users

Please, make sure you have:

- a proper power supply according to the board manufacturer requirements (basic usage example: 5V/2A with DC Jack barrel or **thick** USB cable)
- a reliable SD card (see below “How to prepare a SD card?”)

3.1.2 What to download?

The download for each image consists of three separate files:

- a **xz-compressed image file**,
- a **sha file** for download verification
- and an **asc file** for image authentication.

For each board we usually provide:

- one CLI server image with Debian Buster userspace
- one CLI server image with Ubuntu Focal userspace
- one desktop image with Ubuntu Focal userspace **or** Debian Buster userspace

Other unsupported builds may also be available (like Debian Stretch/Bullseye or Ubuntu Disco/Eoan/Hirsute).

Some boards have different options due to their hardware specialities - router or IoT boards.

Legacy or current?

Only *current* kernel branch is considered fully supported and can bring up video acceleration for example. NAND support is there but is still experimental.

The level of kernel support does depend on the board family. If in your specific case something does not work well, you are always free to try an image with *legacy* kernel included.

What are testing images?

- made from stable branches
- not very well tested
- for end users

© 2021 by Armbian

What are experimental/bleeding edge images?

- made from unstable branches
- untested
- for experienced users only

Do not use testing or edge images in a productive environment. We do appreciate your constructive feedback to developers.

How to check download authenticity?

All our images are digitally signed and therefore it is possible to check their authenticity. You need to issue these commands (Linux/macOS, you might need to install dependencies first, eg. `apt-get install gnupg` on Debian/Ubuntu or `brew install gnupg` on macOS. on windows install the current simple gnupg Gnupg:

```
# download public key from the database
gpg --keyserver hkp://keyserver.ubuntu.com --recv-key DF00FAF1C577104B50BF1D0093D6889F9F0E78D5

# perform verification
gpg --verify Armbian_5.18_Armada_Debian_jessie_3.10.94.img.xz.asc

# proper response
gpg: Signature made sob 09 jan 2016 15:01:03 CET using RSA key ID 9F0E78D5
gpg: Good signature from "Igor Pecovnik (Ljubljana, Slovenia) <igor.++++++++++++@gmail.com>"

# wrong reponse. Not genuine Armbian image!
gpg: Signature made Sun 03 Jan 2016 11:46:25 AM CET using RSA key ID 9F0E78D5
gpg: BAD signature from "Igor Pecovnik (Ljubljana, Slovenia) <igor.++++++++++++@gmail.com>"
```

It is safe to ignore the message `WARNING: This key is not certified with a trusted signature!`.

How to check download integrity?

Since it might happen that your download got somehow corrupted we integrate a checksum/hash for the image. You can compare the image's SHA-256 hash with the one contained in the `sha256sum.sha` file.

On Windows, you can download and use the QuickHash GUI and follow the instructions in the gui.

while on Linux/macOS, in the directory in which you have downloaded the files ,you would do this

```
shasum -a 256 -c Armbian_*.img.sha
#good response
Armbian_5.35_Clearfogpro_Debian_stretch_next_4.13.16.img: OK
```

3.1.3 How to prepare a SD card?

Important note: Make sure you use a **good, reliable and fast** SD card. If you encounter boot or stability troubles in over 95 percent of the time it is either insufficient power supply or related to SD card (bad card, bad card reader, something went wrong when burning the image, card too slow to boot – 'Class 10' highly recommended!). Armbian can simply not run on unreliable hardware so checking your SD card with either F3 or H2testw is mandatory if you run in problems. Since counterfeit SD cards are still an issue checking with F3/H2testw directly after purchase is **highly recommended**.

Write the xz compressed image with USBImager or balenaEtcher on all platforms since unlike other tools, either can validate burning results **saving you from corrupted SD card contents**.

© 2021 by Armbian

Also important: Most SD cards are only optimised for sequential reads/writes as it is common with digital cameras. This is what the *speed class* is about. The SD Association defined *Application Performance Class* as a standard for random IO performance.

Application Performance Class	Pictograph	Miniumum Random Read	Minimum Random Write	Minimum Sustained (Seq. Write)
Class 1 (A1)		1500 4k IOPS	500 4k IOPS	10MBytes/sec
Class 2 (A2)		4000 4k IOPS	2000 4k IOPS	10MBytes/sec

At the time of this writing A1 and A2 cards are only widely available from SanDisk. Armbian recommends A1 rated SD-Cards **only** now (A2 rated cards need yet lacking driver support and therefore show lower overall and especially random IO performance). For example:

In case you chose an SD card that was already in use before please consider resetting it back to ‘factory default’ performance with SD Formatter before burning Armbian to it (explanation in the forum). Detailed information regarding ‘factory default’ SD card performance.

3.1.4 How to boot?

Insert SD card into a slot and power the board. (First) boot (with DHCP) takes up to two minutes with a class 10 SD card and cheapest board.

© 2021 by Armbian

3.1.5 How to login?

First boot will log you automatically on HDMI or serial console while for SSH login you need to login as **root** and use password **1234**. You will be prompted to change this password. You will then be asked to create a normal user account that is sudo enabled (beware of default QWERTY keyboard settings at this stage). Please use this tool, to find your board IP address.

```
odroidxu4 login: root (automatic login)

  ___      _          _     _  __  ___   _ _  _
 / _ \  __| |_ __ ___ (_) __| | \ \/ / | | | || |
| | | |/ _` | '__/ _ \| |/ _` |  \  /| | | | || |_
| |_| | (_| | | | (_) | | (_| |  /  \| |_| |__   _|
 \___/ \__,_|_|  \___/|_|\__,_| /_/\_\\___/   |_|

Welcome to Armbian 21.11 Jammy with Linux 5.4.160-odroidxu4

No end-user support: built from trunk

System load:   13%              Up time:       0 min
Memory usage:  7% of 1.94G      IP:            10.0.10.112
CPU temp:      48°C             Usage of /:    6% of 29G

[ 0 security updates available, 45 updates total: apt upgrade ]
Last check: 2021-11-21 10:32

[ General system configuration (beta): armbian-config ]

Last login: Sun Nov 21 10:32:42 UTC 2021 on tty1

Waiting for system to finish booting ...

New to Armbian? Documentation: https://docs.armbian.com Support: https://forum.armbian.com

New root password: ********
Repeat password: ********

Choose default system command shell:

1) bash
2) zsh

Shell: ZSH

Creating a new user account. Press <Ctrl-C> to abort

Please provide a username (eg. your forename): igorp
Create password: ********
Repeat password: ********

Please provide your real name: Igor

Dear Igor, your account igorp has been created and is sudo enabled.
Please use this account for your daily work from now on.

Detected timezone: Europe/Ljubljana

Set user language based on your location? [Y/n]

Generating locales: sl_SI.UTF-8

You selected ZSH as your default shell.
If you want to use it right away, please logout and login!

root@odroidxu4:~#
```

3.1.6 How to update?

```
apt update
apt upgrade
```

Update process can take hours in case of using cheap SD card and/or under heavy load.

If the kernel was upgraded during this process you will be prompted to reboot at next login.

3.1.7 How to update u-boot?

First you need to update packages described in a previous “How to update” step. Then run armbian-config utility, go to system settings and proceed to:

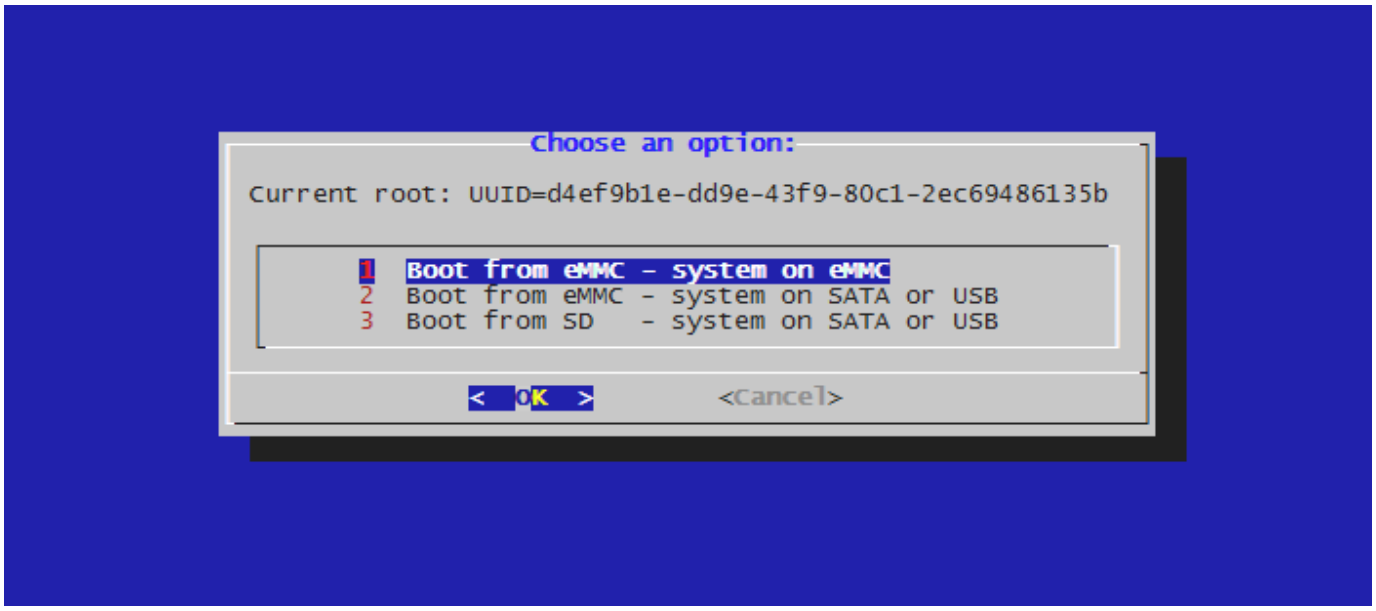
“Install” “Install to/update boot loader” -> Install/Update the bootloader on SD/eMMC

© 2021 by Armbian

3.1.8 How to adjust hardware features?

Use the Armbian configuration utility `armbian-config`

3.1.9 How to install to eMMC, NAND, SATA & USB?



Required condition:

NAND:

- kernel 3.4.x and NAND storage
- pre-installed system on NAND (stock Android or other Linux)

eMMC/SATA/USB:

- any kernel
- onboard eMMC storage
- attached SATA or USB storage

Start the install script:

```
nand-sata-install
```

and follow the guide. You can create up to three scenarios:

- boot from SD, system on SATA / USB
- boot from eMMC / NAND, system on eMMC/NAND
- boot from eMMC / NAND, system on SATA / USB

and you can choose the following file system options:

- ext2,3,4
- btrfs

On Allwinner devices after switching to boot from NAND or eMMC clearing the boot loader signature on the SD card is recommended: `dd if=/dev/zero of=/dev/mmcblkN bs=1024 seek=8 count=1` (replace `/dev/mmcblkN` with the correct device node – in case you run this directly after `nand-sata-install` without a reboot in between then it's `/dev/mmcblk0`). When booting from eMMC to get SD cards auto-detected on Allwinner legacy images please consider changing `mmc0`'s `sdc_detmode` from 3 to 1 in the board's fex file (see here for details).

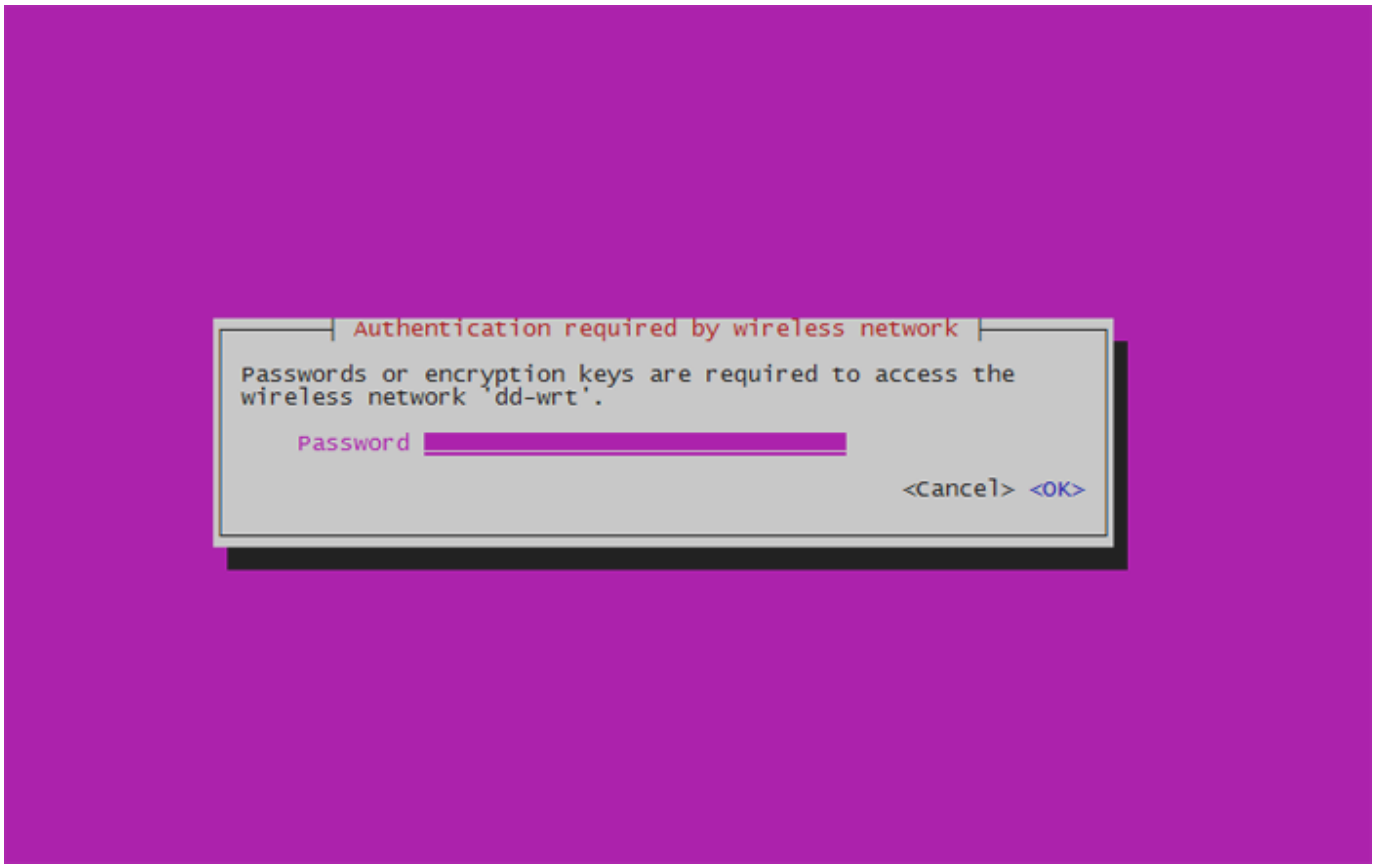
© 2021 by Armbian

3.1.10 How to connect to wireless?

Required condition: a board with onboard or supported 3rd party wireless adapter on USB

If you know what is your wireless SSID:

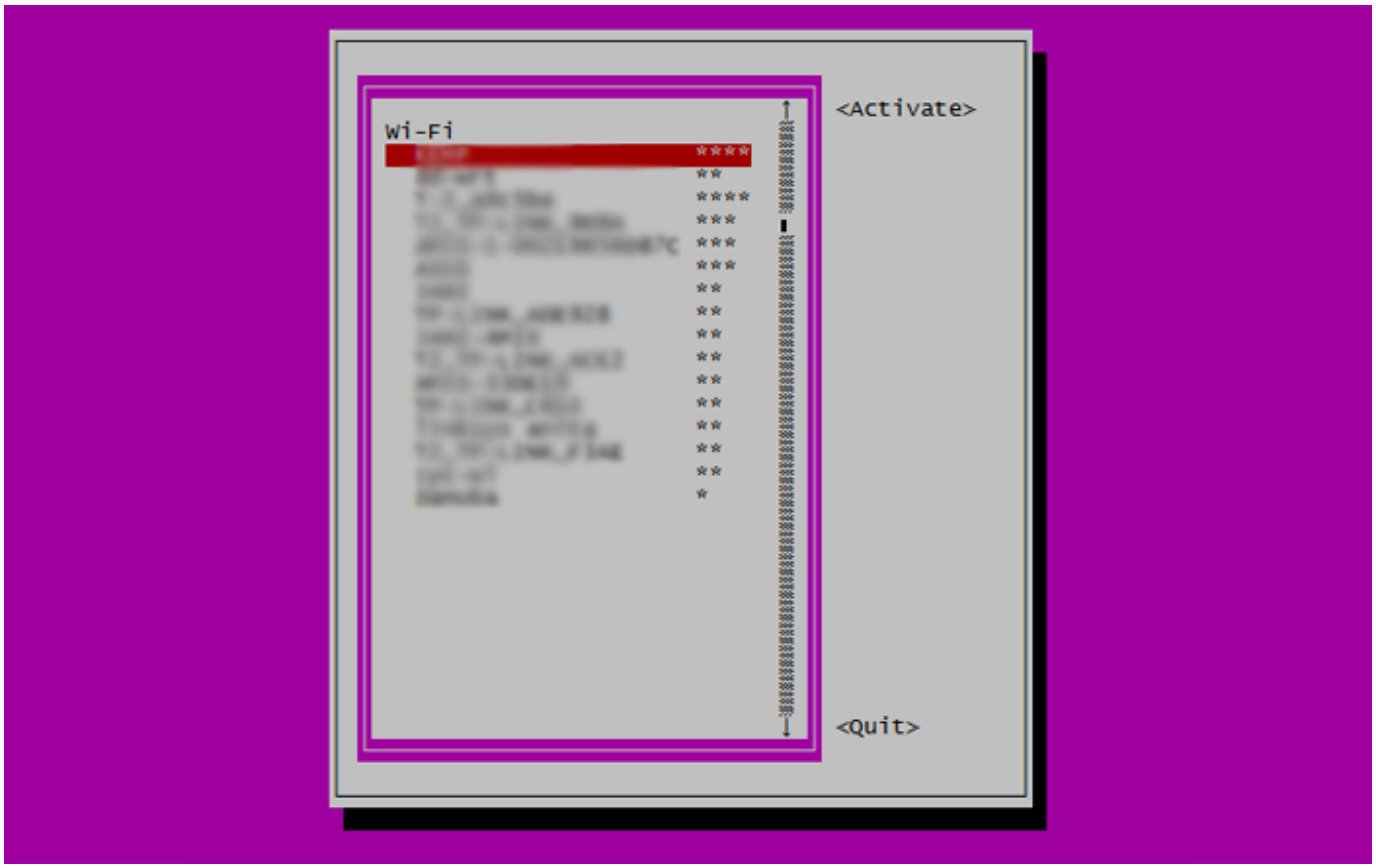
```
1  nmtui-connect SSID
```



If you do not know, you can browse and then connect

```
1  nmtui-connect
```


© 2021 by Armbian



3.1.11 How to set fixed IP?

By default your main network adapter's IP is assigned by your router DHCP server and all network interfaces are managed by **NetworkManager**:

```
user@boardname:~$ nmcli con show
NAME    UUID    TYPE    DEVICE
Wired connection 1  xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx    802-3-ethernet  eth0
```

The connection can now be edited with the following:

```
nmcli con mod "Wired connection 1"
  ipv4.addresses "HOST_IP_ADDRESS"
  ipv4.gateway "IP_GATEWAY"
  ipv4.dns "DNS_SERVER(S)"
  ipv4.dns-search "DOMAIN_NAME"
  ipv4.method "manual"
```

The same changes can also be done with NetworkManagers text user interface:

```
sudo nmtui
```


© 2021 by Armbian

3.2 Quick Facts

3.2.1 What is Armbian Linux?

Armbian Linux provides optimized Debian and Ubuntu Linux images for ARM-based SBCs. There is an incredible ecosystem of small computing platforms that are powerful alternatives to the Raspberry Pi. Armbian's mission is to provide a uniform system offering that is trustworthy to run on any of the dozens of OS-neglected ARM single board computers.

3.2.2 Challenges

Armbian is the opposite of Raspbian

Raspbian has dozens of contributors to focus on a single SBC platform. Armbian has a dozen contributors to focus on 100+ SBCs spread over 30 platforms.

Balancing Development and Support

Given the point above, resources are thin. Armbian developers have to focus on the core mission of maintaining the Armbian Build Platform. We heavily rely on other members of the community to support each other. Although Armbian does provide a lot of user friendly features, the reality is that Armbian is for more advanced users. If you are really struggling with your SBC, you may want to consider first getting more comfortable with Raspbian Linux on the Raspberry Pi.

More SBCs continuously coming to market

SBC and TV Box manufacturers love to design and ship new products. Unfortunately they do not like to spend time on software and instead rely on community projects such as Armbian to fill in the gaps.

3.2.3 Benefits

Simple

BASH or ZSH shell, standard Debian/Ubuntu utilities. Common and specific features can be with minimalistic menu-driven utility. Login is possible via serial, HDMI/VGA or SSH.

Light

No bloatware or spyware. Special utilities are completely optional. Suitable for newcomers and professionals.

Optimized

A distributed image is compacted to real data size and starts at around of 1G. Size is optimized for SD card usage. Bigger is better. Installing applications later severely reduces the life of your SD card. They were not designed for this type of usage.

Fast

Boards are optimized on kernel and userspace level. DVFS optimization, memory log caching, browser profile memory caching, swap usage tuning, garbage commit delay. Our system runs almost read-only and is one of the the fastest Linux for many development boards in just about every case.

Secure

Security level is on a stock Debian/Ubuntu level and can be hardened with the configuration utility. It provides a good starting point for industrial or home usage. The system is regularly inspected by professionals within the community. Each official stable build is thoroughly tested. Images are a direct base for all 3rd party builders.

© 2021 by Armbian

Supported

Providing long term updates, security fixes, documentation, user support.

Smart

Deep understanding how boards work, how operating system work and how hardware should be designed to run better. Involved in board design. Experience in Linux since early 90’. Specialized in ARM development boards since 2013.

Open

Open source build script and kernel development, maintenance and distribution for more than 30 different ARM and ARM64 Linux kernels. Powerful build and software development tools. Can run in fully parallel mode. Can run under Docker.

© 2021 by Armbian

3.3 Hardware troubleshooting guide

If you are experiencing at least one of these problems:

- board does not boot
- board freezes, crashes or reboots randomly or when connecting USB devices
- plugged in USB devices are not detected (not listed in `lsusb` output)
- error changing the root password at first boot (Authentication token manipulation error)
- error installing or updating packages due to read-only file system

and you are using a stable Armbian image, then most likely you have one of two common problems - **powering issue** or **SD card issue**.

Note that

- *“I know that my power supply is good”, “it worked yesterday”, “it works with a different device”,* etc. are **not objective reasons** to skip powering related diagnostics
- *“I know that my SD card is good”, “it worked yesterday”, “it works with a different device”,* etc. are **not objective reasons** to skip storage related diagnostics
- undervoltage can cause symptoms related to SD card problems such as filesystem corruptions and data loss, so powering has to be checked first

3.3.1 Powering notes

- Most boards, even ones fitted with PMIC (power management integrated circuit) do not have any measures to react to undervoltage that could prevent instability
- It does not matter what voltage your power supply outputs, it matters what voltage will reach the onboard voltage regulators
- Peak power consumption of popular boards can vary from 0.9A at 5V (H3 based Orange Pi PC) to 1.7A at 5V (RK3288 based Tinkerboard), both without any attached peripherials like USB devices
- Due to the Ohm’s law voltage drop due to cable and connector resistance will be proportional to the electric current, so most of the time problems will be experienced at current spikes caused by CPU load or peripherials like spinning up HDDs

Power supply

- Cheap phone chargers may not provide the current listed on their label, especially for long time periods
- Some cheap phone chargers don’t have proper feedback based stabilization, so output voltage may change depending on load
- Power supplies will degrade over time (especially when working 24/7)
- Some problems like degraded output filtering capacitors cannot be diagnosed even with a multimeter because of the non-linear voltage form

Cable

- The longer and thinner the cable - the higher its resistance - the greater the voltage drop will be under load
- Even thick looking cable can have thin wires inside, so do not trust the outside cable diameter

Connector

- MicroUSB connector is rated for the maximum current of 1.8A, but even this number cannot be guaranteed. Trying to pass larger current (even momentarily) may result in a voltage dropping below USB specifications
- Most of the boards can also be powered through GPIO pins. This can be used to bypass the microUSB connector and thus to improve stability

© 2021 by Armbian

3.3.2 SD card notes

- A SD card is a complex storage device with an embedded controller that processes read, erase and write operations, wear leveling, error and corruption detection, but it does not provide any diagnostic protocols like S.M.A.R.T.
- SD cards will degrade over time and may fail in the end in different ways - become completely or partially read-only or cause a silent data corruption

SD card brand

- Based on current prices and performance tests done by Armbian users Samsung Evo, Samsung Evo Plus and Sandisk Ultra cards are recommended
- Other good alternatives may be added to this page in the future

SD card size and speed class

- SD card speed class and size does not influence the reliability directly, but larger size means larger amount of lifetime data written, even if you are using 10-20% of the cards space

SD card testing

- There are many fake SD card around. eBay and Amazon Marketplace are notorious for selling fakes, but sometimes even reputable retailers get fooled.
- Most commonly low capacity cards will be reprogrammed to appear as bigger, but any files written beyond the true capacity will be lost or corrupted.
- We recommend to always test the capacity of each new SD cards using f3.

Writing images to the SD card

- If you wrote an image to the card it does not mean that it was written successfully without any errors
- so always verify images after write using some tools like *balenaEtcher* which is currently the only popular and cross-platform tool that does mandatory verify on write (more lightweight alternatives may be added to this page in the future)
- “Check for bad blocks” function available in some tools is mostly useless when dealing with SD cards
- Note that *balenaEtcher* verifies only 1-2GB that are occupied by the initial unresized image, it does not verify the whole SD card

3.3.3 Network

MAC Address Conflicts

If you experience strange network problems, especially if you are running several of these SOC-boards with the same operating system, then the problems may be sourced by not having a real hardware MAC address. The operating systems try to generate a hardware MAC address from the CPUid, but what if that SOC has no CPUid either?

Then you have to do it manually. Depending on system and network installation, there are several possibilities:

- the preferred way: change `/boot/armbianEnv.txt` and add a line:

```
ethaddr=XX:XX:XX:XX:XX:XX
```


© 2021 by Armbian

but that file is interpreted by u-boot, which happens early in boot process, but not every u-boot is able to read that file.

- next possibility to set mac-address is changing network configuration. On systems with **ifupdown** you can do that by changing `/etc/network/interfaces`. Add these lines:

```
auto eth0
iface eth0 inet dhcp
    hwaddress ether XX:XX:XX:XX:XX:XX
```

- if the above does not work, then your network is probably controlled by **Network-manager** . In directory `/etc/Networkmanager/system-connections` is a file `Wired connection 1.nmconnection`. Change entry *cloned-mac-address* of group **[ethernet]** :

```
[connection]
id=Wired connection 1
type=ethernet

[ethernet]
cloned-mac-address=XX:XX:XX:XX:XX:XX
```

3.3.4 Video

No Screen on 4k Resolution

Some combination of boards/kernel versions does not support 4k resolution. This may cause black screen on connecting the board to 4k resolution devices. A workaround to solve this without changing the kernel is forcing the video mode to 1080p. Add this directive to the `/boot/armbianEnv.txt` and reboot your system:

```
extraargs=video=HDMI-A-1:1920x1080@60
```

- Despite of this config, some apps may try to use 4k resolution, example: *Retroarch*. In this case you have to change app configuration to use the 1080p resolution.
- To edit files without video you could connect to the board using ssh. Other option is mount the sd card in another device.

3.3.5 Configuration

- Some boards require the setup of the correct device tree file or they will not boot. Check the board specific documentation for details.

3.3.6 How to report bugs properly

- Follow instructions on https://armbian.com/bugs

© 2021 by Armbian

3.4 Recovery

Important: If you came here since you cannot get Armbian running on your board please keep in mind that in 95 percent of all cases it is either a faulty/fraud/counterfeit SD card or an insufficient power supply that is causing these sorts of *does not work* issues!

The following are presented in (more or less) increasing levels of despair. But keep heart! :) And proceed in order.

3.4.1 U-Boot Shell Access

If you broke the system you can try to get in this way. You have to get to u-boot command prompt, using either a serial adapter or monitor and usb keyboard.

- Note: USB support in u-boot is currently not enabled on all H3 boards.

After switching power on or rebooting, when u-boot loads up, press some key on the keyboard (or send some key presses via terminal) to abort default boot sequence and get to the command prompt:

```
U-Boot SPL 2015.07-dirty (Oct 01 2015 - 15:05:21)
...
Hit any key to stop autoboot:  0
sunxi#
```

Enter the following commands, replacing root device path if necessary.

- Note: these are for booting with mainline kernel; check `boot.cmd` for your device for commands related to legacy kernel.

For serial:

```
sunxi# setenv bootargs init=/bin/bash root=/dev/mmcblk0p1 rootwait console=ttyS0,115200
```

For monitor:

```
sunxi# setenv bootargs init=/bin/bash root=/dev/mmcblk0p1 rootwait console=tty1
```

Then:

```
sunxi# ext4load mmc 0 0x49000000 /boot/dtb/${fdtfile}
sunxi# ext4load mmc 0 0x46000000 /boot/zImage
sunxi# env set fdt_high ffffffff
sunxi# bootz 0x46000000 - 0x49000000
```

System should eventually boot to bash shell:

```
root@host:/#
```

Now you can try to fix your broken system.

3.4.2 Replacing /boot

When something goes terribly wrong and you are not able to boot the system (and cannot gain access via u-boot as outlined above), this is the way to proceed. You will need some Debian based Linux machine where you can mount the failed SD card. With this procedure you will reinstall the kernel and hardware settings. In most cases this should be enough to unbrick the board.

It is recommended to issue a filesystem check before mounting. Replace `X` and `Y` below with your device and partition(s), respectively (if not a flash based device, it may even be `/dev/sdXY`, etc).

```
/ # fsck /dev/mmcblkXpY -f
```


© 2021 by Armbian

Mount the SD card.

```
/ # cd /mnt
/mnt # mkdir sdcard
/mnt # mount /dev/mmcblkXpY /mnt/sdcard
```

Make another temporary directory somewhere else (in our example `~/tmp/recovery`) and download the Linux root, kernel, firmware and dtb packages for your board and currently used OS.

- Note: This example is only for **Nanopi Neo 2** with **Ubuntu Focal**, **current** kernel (mainline) and **Armbian 20.08.13** firmware. Alter package names according to your device name, SOC-family, kernel and firmware version!

```
/mnt $ cd
~ $ mkdir -p tmp/recovery
~ $ cd tmp/recovery

(Root file system):
~/tmp/recovery $ wget https://apt.armbian.com/pool/main/l/linux-focal-root-current-nanopineo2/linux-focal-root-current-nanopineo2_20.08.13_arm64.deb

(Kernel):
~/tmp/recovery $ wget https://apt.armbian.com/pool/main/l/linux-5.8.16-sunxi64/linux-image-current-sunxi64_20.08.13_arm64.deb

(Firmware):
~/tmp/recovery $ wget https://apt.armbian.com/pool/main/a/armbian-firmware/armbian-firmware_20.08.13_all.deb

(Device Tree Binary (DTB)):
~/tmp/recovery $ wget https://apt.armbian.com/pool/main/l/linux-5.8.16-sunxi64/linux-dtb-current-sunxi64_20.08.13_arm64.deb
```

Extract all the Debian packages (`.deb` files) to the mounted sd card.

```
~/tmp/recovery # for f in *.deb; do dpkg -x $f /mnt/sdcard; done
```

Navigate to `/mnt/sdcard/boot` and create symlinks:

```
~/tmp/recovery # cd /mnt/sdcard/boot
/mnt/sdcard/boot # ln -s vmlinuz-5.8.16-sunxi64 zImage
/mnt/sdcard/boot # ln -s uInitrd-5.8.16-sunxi64 uInitrd
/mnt/sdcard/boot # ln -s dtb-5.8.16 dtb
```

If you upgrade from some very old build, you might need to update your boot script.

- Note: The following example is for Allwinner boards.
- Note: You will need a `u-boot-tools` package on your host system.

```
/mnt/sdcard/boot # wget https://raw.githubusercontent.com/armbian/build/master/config/bootscripts/boot-sunxi.cmd
/mnt/sdcard/boot # mv boot-sunxi.cmd boot.cmd
/mnt/sdcard/boot # mkimage -C none -A arm -T script -d boot.cmd boot.scr
```

Unmount SD card.

```
/mnt/sdcard/boot # cd /
/ # umount /mnt/sdcard
```

Move it to the board and power on. Check serial output for errors if problems persist.

3.4.3 Flashing boot loader

Sometimes we need to flash boot loader from some other Linux. Attach an SD card reader with your SD card and proceed this way:

```
/mnt $ cd
~ $ mkdir -p tmp/recovery
~ $ cd tmp/recovery
~ $ wget https://imola.armbian.com/apt/pool/main/l/linux-u-boot-nanopineo2-current/linux-u-boot-current-nanopineo2_20.08.13_arm64.deb
~ $ dpkg-deb -x linux-u-boot-current-nanopineo2_20.08.13_arm64.deb pack
~ $ bash pack/usr/lib/u-boot/platform_install.sh pack/usr/lib/linux-u-boot-nanopineo2-current /dev/XXX # replace XXX with the actual device /dev/sdb
```

Move it to the board and power on. Check serial output for errors if problems persist.

© 2021 by Armbian

3.5 Frequently asked questions

This information is mainly for new/inexperienced users but could be useful for others too.

3.5.1 Is Armbian an operating system?

Not per se. Armbian is a build framework that allows users to create ready-to-use images with working kernels in variable userspace configurations for various single board computers (SBCs).

We do provide various prebuilt images for some boards, but mostly for users convenience.

3.5.2 Why I cannot simply shove a random image into my board to work like on my PC?

x86 architecture always has a traditional BIOS or UEFI. This provides a standard framework for operating systems to interact with the hardware. Most SBCs do not. ARM is improving the situation with *ARM Server Ready* and *ARM System Ready certificates*, but most SBC vendors are not yet incentivized to meet these standards.

Without such standards, many vendors quickly fork low-level bootloaders such as u-boot and make the bare minimum modifications needed.

Great reference here

3.5.3 Why is Armbian constantly asking for money? Free software should be free.

Making free licence software also requires best people, expensive infrastructure, tooling. It has as much or more costs as proprietary while generating no income from the licence.

tl;dr: We are asking for help that developers and project maintainers do not lose their generosity and humanity which are the driving force that generates a value. For all of us! A great deal of our work represent a big pressure on our very limited private resources. We ask you to share that burden with us.

Development time

We are covering a large swath of diverse, custom designed ARM hardware in ways, extents, and under conditions nobody else does. Keeping this service up, keeping these low end hardware functional is laborious. When releases are approaching and a lot of testing and fixing is going on, this gets up, stress intensifies. This means we have to invest let's say at least 3.000 - 4.000 EUR of our time on top of fixed costs into this service every day just to keep it up. Without developing any serious features users wish to have. Fulfilling many of these wishes would easily cost tens of thousands in development time, which we don't have and which we can't get back due to it being free software. Nobody needs to buy licence for using it, and yet only a few people decide to respect the time and attention they are receiving from developers on forums.

Infrastructure and operations

We have to maintain our infrastructure where biggest costs are - once again - people's time, followed by electricity, then hardware itself. Often we get free hardware and very rare break even with electricity costs and with people that would maintain this for us. A new sponsored box usually brings us more costs then benefits – since benefit is anyway public.

Support time

The software is given, released free. However support, development, and documentation costs time, effort, hardware and technical ability, which incurs costs.

Each question that is directed towards our team is generating opportunity costs and taking away from development time. Some we are happy to cover, but not all. Especially when it goes for repetitive questions and demands.

© 2021 by Armbian

Questions associated with missing features represent another hit and miss for us. Complicated and critical upstream functions are missing (like video acceleration within a web browser, supporting a board that had very poor initial support and no community backing, etc.). This functionality is unique to hardware and implementing is extremely labor intensive and requires unique expertise. Our team is 10-15 volunteers that maintain this project during their own time. We cannot cover the job of Google Chromium team, Collabora, ARM, Rockchip and other vendors which have not provided sufficient support for their products.

All our work is done in public and we provide all sources which we are changing in the process. All our work is patent free and released under a free licence so anyone can re-use it further. The scale of SBCs Armbian supports is hard to beat, and consequently our work is repackaged and reused by other projects and vendors. Unfortunately the burden of support is often directed to us, while they focus on revenue.

3.5.4 Why does hardware feature XY work in old kernel but not in more recent one?

Vendors develop hardware specific support on fixed (usually old LTS) kernel and U-Boot fork and only do minimal adjustments to make board features work. Besides the fact that those adjustments are almost never pushed back to mainline they usually do not update their sources (if available at all) and pre-made kernels/boot loaders as well.

Armbian moves things forward and follows mainline kernel as much as possible, to provide both its features as well as security updates. The downside is that some features do not work since nobody ported specific drivers to recent mainline, and they can also break. Armbian can only afford to do brief testing of images and check if basic functions (boot-up, network, USB, etc.) work due to lack of both human and financial resources.

3.5.5 What does WIP/EOS/CSC mean?

- WIP: Work In Progress: Basic functions can be tested but not ready for production yet.
- CSC: Community Supported Configuration: Community contributed support. No official support from Armbian development team.
- EOS: End of life: Support ended.

3.5.6 I have no technical knowledge. How can I help?

We need many different profiles of people to run this project and just about any help is appreciated, not just help on development. Since otherwise developers have to fix web pages, developers have to run projects, developers have to seek for money, developers have to maintain servers, developers have to maintain forum, developers have to moderate forums, developers have to maintain infrastructure, developers have to maintain relations with partners, developers have to waste time on repeated support question, developers have to deal with “customers”, …

3.5.7 Why are old-stable distributions like Ubuntu Bionic or Debian Stretch not supported?

The Armbian project has very limited human and financial ressources so it can focus only on a few up-to-date operating system releases.

3.5.8 I have a TV Box/tablet from [insert random vendor]. Can I use Armbian on it?

No.

However some community members are commited to tinkering with these devices. They discuss their findings in a dedicated space in our forums. Take note that there is no support from the Armbian development team whatsoever.

3.5.9 Why does Armbian not support TV boxes? The market is huge!

There are some manufacturers who produce better quality then the others. In general they provide more or less accurate schematics and they have some engineers that are available for general public and you can ask them things here and there. Most of them try to keep up with the highest standards of hardware development. With proper documentation and minimal support,

© 2021 by Armbian

costs of software development are significantly lower. This is especially important, because we waste our precious private time to secure proper hardware functioning through the time.

However, in vast majority of cases, TV boxes are lacking any docuentation. There are frequent changes of components without notice whatsoever, boot mechanisms are closed source and almost all Armbian builds that exist in the wild are community hacks. Market is huge but since public does not have interest in covering of support - which in this case is even bigger - involvement in providing support is simply insane and stupid. It only eats our personal time and finances.

3.5.10 There is a new board on the market. Will Armbian officially support it?

Maybe. It depends on things like available documentation from both the vendor as well as SoC manufacturer, production samples to play with, available BSP and last but certainly not least human ressources. A Maintainer within the Armbian development team to say yes. Also if vendors decide to support Armbian there is certainly a higher chance to get it fully supported.

3.5.11 How can I compile my own kernel?

Normally on Debian or Ubuntu you would do something like `sudo apt-get build-dep linux linux-image-$(uname -r)`.

However Armbian's way of building kernel images is slightly different than the standard distribution method. The best way is to follow the procedures in the Developer Guide.

3.5.12 How do I upgrade from Armbian Buster to Bullseye?

Armbian does not offer a standardized way nor do we encourage users to upgrade their userspace like Bionic to Focal, Stretch to Buster, or Buster to Bullseye. We would love to do that but the reason why we cannot is simply the lack of ressources in time and devices to test such upgrades in random scenarios.

You can try to upgrade your userspace by following official ways from Debian/Ubuntu but make sure to freeze your firmware packages via `armbian-config` beforehand. Also you will not receive any help from Armbian if something goes wrong or have other issues with an upgraded system.

3.5.13 Why I cannot choose a specific kernel version (5.11.5 for example)?

Each kernel Armbian offers has a custom patchset on top which would be impossible to maintain compatibility to each and every kernel version out there. Therefore the choice is usually limited to up to three branches: legacy, current and edge. Depending on board/family the versions behind these branches may differ. You can lookup them in the source code.

3.5.14 Can I upgrade my userspace, like Buster to Bullseye or Focal to Jammy?

Userspace upgrades are neither tested nor supported. The reason behind is simply the lacking of (human) ressources to test different scenarios on different boards. A dist-upgrade may work but as stated if something goes wrong Armbian cannot provide assistance. You are on your own to fix issues.

© 2021 by Armbian

3.6 Advanced Features

3.6.1 How to switch kernels?

Check *this* for more info.

3.6.2 How to troubleshoot?

3.6.3 How to unbrick the system?

Both of above headings have been moved to a new page and expanded upon: Recovery

3.6.4 How to build a wireless driver?

Install and recreate kernel headers scripts (optional)

```
armbian-config -> install kernel headers
exit

cd /usr/src/linux-headers-$(uname -r)
make scripts
```

Go back to root directory and fetch sources (working example, use ARCH=arm64 on 64bit system)

```
cd
git clone https://github.com/pvaret/rtl8192cu-fixes.git
cd rtl8192cu-fixes
make ARCH=arm
```

Load driver for test

```
insmod 8192cu.ko
```

Check dmesg and the last entry will be:

```
usbcore: registered new interface driver rtl8192cu
```

Plug the USB wireless adaptor and issue a command:

```
iwconfig wlan0
```

You should see this:

```
wlan0   unassociated  Nickname:"<WIFI@REALTEK>"
        Mode:Auto  Frequency=2.412 GHz  Access Point: Not-Associated
        Sensitivity:0/0
        Retry:off   RTS thr:off   Fragment thr:off
        Encryption key:off
        Power Management:off
        Link Quality=0/100  Signal level=0 dBm  Noise level=0 dBm
        Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
        Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Check which wireless stations / routers are in range

```
iwlist wlan0 scan | grep ESSID
```

3.6.5 How to run Docker?

Docker works reliably with the distribution-provided builds of docker. It's as simple as `apt-get install docker.io`. If you prefer to use the latest docker builds provided directly by Docker. Please follow the guide below.

© 2021 by Armbian

Preinstallation requirements:

- Armbian 20.08.17 or newer with Kernel 3.10 or higher
- Debian Buster (might work elsewhere with some modifications)
- root access

This method is based on Docker Debian installation documentation. Execute this as root:

```
apt-get remove docker docker-engine docker.io containerd runc
apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add -
add-apt-repository "deb [arch=arm64] https://download.docker.com/linux/debian $(lsb_release -cs) stable"
apt update
apt-get install docker-ce docker-ce-cli containerd.io
```

Test if Docker works correctly:

```
docker run hello-world
```

If you get that kind of output, then Docker install went fine:

```
Hello from Docker!
This message shows that your installation appears to be working correctly.
```

If you would like to use Docker as a non-root user, you should now consider adding your user to the `docker` group with something like:

```
usermod -aG docker your-user
```

You will have to log out, and log in once more in order to be able to run Docker without being root.

Let's try a last test to see if a Docker container can be seen outside of your Armbian machine:

```
docker run -d -p 80:80 hypriot/rpi-busybox-httpd
```

... and point the browser of any device in the same network to `http://<IP OF YOUR DEVICE>/`

More info in this forum topic

3.6.6 How to set wireless access point?

There are two different HostAP daemons. One is **default** and the other one is for some **Realtek** wifi cards. Both have their own basic configurations and both are patched to gain maximum performances.

Sources: https://github.com/igorpecovnik/hostapd

Default binary and configuration location:

```
/usr/sbin/hostapd
/etc/hostapd.conf
```

Realtek binary and configuration location:

```
/usr/sbin/hostapd-rt
/etc/hostapd.conf-rt
```


© 2021 by Armbian

Since its hard to define when to use which you always try both combinations in case of troubles. To start AP automatically:

1. Edit /etc/init.d/hostapd and add/alter location of your conf file **DAEMON_CONF=/etc/hostapd.conf** and binary **DAEMON_SBIN=/usr/sbin/hostapd**
2. Copy **/etc/network/interfaces.hostapd** to **/etc/network/interfaces**
3. Reboot
4. Predefined network name: “BOARD NAME” password: 12345678
5. To change parameters, edit /etc/hostapd.conf BTW: You can get WPAPSK the long blob from wpa_passphrase YOURNAME YOURPASS

3.6.7 How to connect IR remote?

Required conditions:

- IR hardware
- loaded driver

Get your remote configuration (lircd.conf) or learn.

- Note: As of 2020-11-25, the above (Kodi / learn) link is broken. However I am not sure what to replace it with. If you know (or find out) please submit a PR. - TRS-80

You are going to need the list of all possible commands which you can map to your IR remote keys:

```
irrecord --list-namespace
```

To start with learning process you need to delete old config:

```
rm /etc/lircd.conf
```

Than start the process with:

```
irrecord --driver=default --device=/dev/lirc0 /etc/lircd.conf
```

And finally start your service when done with learning:

```
service lirc start
```

Test your remote:

```
irw /dev/lircd
```

3.6.8 Outdated

How to freeze your filesystem? (outdated)

In certain situations it is desirable to have a virtual read-only root filesystem. This prevents any changes from occurring on the root filesystem that may alter system behavior and it allows a simple reboot to restore a system to its clean state.

You need an ODROID XU4 or Allwinner A10, A20 or H3 board with legacy kernel where we added support for overlayfs. Works only on Ubuntu Xenial. Login as root and execute:

```
apt-get install overlayroot
echo 'overlayroot="tmpfs"' >> /etc/overlayroot.conf
reboot
```


© 2021 by Armbian

After your system boots up it will always remain as is. If you want to make any permanent changes, you need to run:

```
overlayroot-chroot
```

Changes inside this will be preserved.

© 2021 by Armbian

3.7 How to customize keyboard, time zone?

- **Attention: The preferred method to change most of this stuff is by using the interactive *armbian-config* tool which is shipped with all Armbian images.**

3.7.1 Keyboard:

```
dpkg-reconfigure keyboard-configuration
```

In some cases, e.g. when your keyboard standard is not available in previous command, you may also need to set your keymap config:

```
# Check your actual keymap config
localectl status | grep -i keymap

# Set the desired keymap config. Example below to 'br-abnt2'
localectl set-keymap br-abnt2
```

3.7.2 System language:

```
# Debian --> https://wiki.debian.org/ChangeLanguage
dpkg-reconfigure locales
# Ubuntu --> https://help.ubuntu.com/community/Locale
update-locale LANG=[options] && dpkg-reconfigure locales
```

3.7.3 Console font, codepage:

```
dpkg-reconfigure console-setup
```

3.7.4 Time zone:

```
dpkg-reconfigure tzdata
```

3.7.5 Sound output:

```
# Check the available sound output options:
pacmd list-sinks | less
# The default will be marked with "*"
# Press "q" to close


# Define the new default sound output
pacmd set-default-sink <NAME-OF-DESIRED-OPTION>
```

The name of HDMI sound output may change accordingly to the device. If you don’t wanna deal with different names you can:

```
pacmd set-default-sink $(pactl list short sinks | grep -i 'hdmi' | awk '{print $2}')
```

The command to define the default sound output is not persistent, to make it persistent add it to the file `~/.bashrc`

© 2021 by Armbian

3.7.6 Screen resolution on other boards:

```
nano /boot/boot.cmd

# example:
# change example from
# disp.screen0_output_mode=1920x1080p60
# to
# disp.screen0_output_mode=1280x720p60

mkimage -C none -A arm -T script -d /boot/boot.cmd /boot/boot.scr
```

3.7.7 Screen resolution within Xorg

- Thanks to user @maxlinux2000 in this forum post.

Find matching HDMI output:

```
xrandr --listmonitors
```

Calculate VESA CVT mode lines (example for 1440x900)

```
cvt 1440 900
```

Sample output:

```
1440x900 59.89 Hz (CVT 1.30MA) hsync: 55.93 kHz; pclk: 106.50 MHz
Modeline "1440x900_60.00"  106.50  1440 1528 1672 1904  900 903 909 934 -hsync +vsync )
```

Create new mode (example):

```
xrandr --newmode "1440x900_60.00" 106.50 1440 1528 1672 1904 900 903 909 934 -hsync +vsync
```

Add resolution (example):

```
xrandr --addmode HDMI-1 1440x900_60.00
```

Set current resolution (example):

```
xrandr --output HDMI-1 --mode 1440x900_60.00
```

If it works as expected add it to Xorg by editing `/etc/X11/xorg.conf.d/40-monitor.conf` and add (example):

```
Section "Monitor"
Identifier "HDMI-1"
Modeline "1440x900_60.00" 106.50 1440 1528 1672 1904 900 903 909 934 -hsync +vsync
Option "PreferredMode" "1440x900"
EndSection
```

Restart Xorg or reboot

3.7.8 How to alter CPU frequency?

Some boards allow to adjust CPU speed

```
nano /etc/default/cpufrequtils
```

Alter **min_speed** or **max_speed** variable.

```
systemctl restart cpufrequtils
```


© 2021 by Armbian

3.7.9 Swap for experts

By default Armbian implements ZRAM (writing nothing to ‘disk’ but compressing memory pages in RAM) but in case you often run into out of memory errors and your device has some capable storage (e.g. a securely attached NVMe or SATA SSD) you might want to use ZSWAP instead.

Check whether your kernel has zswap enabled (`dmesg | grep zswap` should output something) and if so create a swapfile or swap partition the traditional way, edit/uncomment `/etc/default/armbian-zram-config` so that it reads `SWAP=false`, reboot and you’re done.

Zswap performs a lot better than the combination of ZRAM and ‘swap on disk’ in parallel.

3.7.10 How to downgrade a package via apt?

This is useful when you need to fall back to previous kernel version.

```
apt install linux-image-sun8i=5.13
```

This example is for H3 legacy kernel. Check this page for others.

3.7.11 How to toggle boot output?

Edit and change boot parameters in `/boot/boot.cmd` (not recommended) or variables in `/boot/armbianEnv.txt`:

```
- console=both
+ console=serial
```

To disable console entirely (not recommended) set console to `none`.

Recompile boot.cmd to boot.scr if it was changed:

```
mkimage -C none -A arm -T script -d /boot/boot.cmd /boot/boot.scr
```

Reboot.

Serial console on imx6 boards are ttymxc0 (Hummingboard, Cubox-i) or ttymxc1 (Udoo).

3.7.12 How to toggle verbose boot?

Using Armbian 5.05 to 5.20 you would need to touch/rm `/boot/.force-verbose` to increase boot verbosity. With more recent Armbian builds you would have to alter the `verbosity=` line in `/boot/armbianEnv.txt` (defaults to 1 which means less verbose, maximum value is 7).

3.7.13 How to provide boot logs for inspection?

When your SBC behaves strange first step is to check power supply and integrity of boot media (`armbianmonitor -c "$HOME"`). Then look into your kernel logs. We made a tool that grabs info and pastes it to an online pasteboard service. Please increase boot verbosity as shown above (`verbosity=7`), reboot and then run

```
sudo armbianmonitor -u
```

Copy and past URL of your log to the forum, mail, …

© 2021 by Armbian

3.7.14 How to change network configuration?

To get Wi-Fi working simply use `nmtui`, a simple console based UI for network-manager (an example how to set up an AP with network-manager can be found here). To deal with different Ethernet/Wi-Fi combinations there are six predefined configurations available, you can find them in those files:

```
/etc/network/interfaces.bonding
/etc/network/interfaces.default
/etc/network/interfaces.hostapd
/etc/network/interfaces.network-manager
/etc/network/interfaces.r1
/etc/network/interfaces.r1switch
```

By default **/etc/network/interfaces** is a copy of **/etc/network/interfaces.default**

1. BONDING: your network adapters are bonded in fail safe / “notebook” way.
2. DEFAULT: your network adapters are connected classical way.
3. HOSTAPD: your network adapters are bridged together and bridge is connected to the network. This allows you to have your AP connected directly to your router.
4. All interfaces are handled by network-manager (`nmtui` / `nmcli` or using the GUI)
5. Router configuration for Lamobo R1 / Banana R1.
6. Switch configuration for Lamobo R1 / Banana R1.

You can switch configuration with copying.

```
cd /etc/network
cp interfaces.x interfaces
```

(x = default,hostapd,bonding,r1)

Then check / alter your interfaces:

```
nano /etc/network/interfaces
```

3.7.15 Choosing an apt mirror

Armbian has its own apt repository and mirrors for armbian-specific packages. The default of `http://apt.armbian.com` is a round-robin to all mirrors. If you are having trouble updating or slow speeds you may want to choose a specific mirror.

Do the following:

Assure `jq` is installed

`apt install -y jq`

Get a list of available mirrors from our `https://apt.armbian.com/mirrors` endpoint.

```
curl -s http://apt.armbian.com/mirrors|jq
```


© 2021 by Armbian

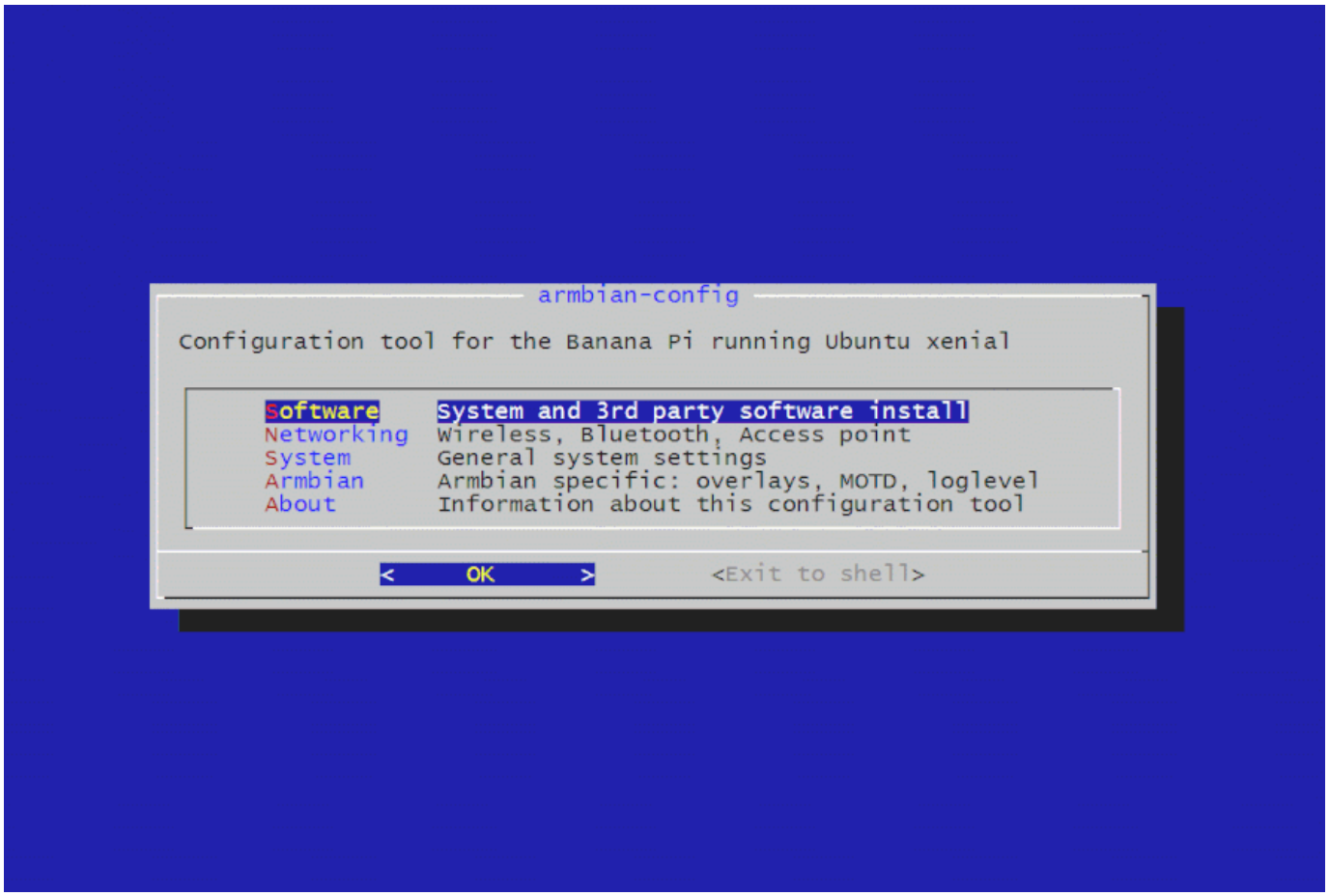
You will see a result set similar to this, listing mirrors by region:


```
{
  "AS": [
    "https://mirrors.tuna.tsinghua.edu.cn/armbian/",
    "https://minio.k-space.ee/armbian/apt/"
  ],
  "NA": [
    "https://armbian.tnahosting.net/apt/",
    "https://us.mirrors.fossho.st/armbian/apt/"
  ],
  "EU": [
    "https://minio.k-space.ee/armbian/apt/",
    "https://armbian.systemonachip.net/apt/",
    "https://mirrors.netix.net/armbian/apt/",
    "https://mirrors.dotsrc.org/armbian-apt/",
    "https://armbian.hosthatch.com/apt/"
  ],
  "default": [
    "https://armbian.tnahosting.net/apt/",
    "https://us.mirrors.fossho.st/armbian/apt/",
    "https://minio.k-space.ee/armbian/apt/",
    "https://armbian.systemonachip.net/apt/",
    "https://mirrors.netix.net/armbian/apt/",
    "https://mirrors.dotsrc.org/armbian-apt/",
    "https://armbian.hosthatch.com/apt/"
  ]
}
```


Edit `/etc/apt/sources.list.d/armbian.list` and replace the url `http://apt.armbian.com` with your preferred mirror.

© 2021 by Armbian

3.8 Armbian configuration utility



Is a base utility for configuring your board, divided into four main sections:

- **S**ystem - system and security settings,
- **N**etwork - wired, wireless, Bluetooth, access point,
- **P**ersonal - timezone, language, hostname,
- **S**oftware - system and 3rd party software install.

The tool needs root privileges to work and can be launched by entering `sudo armbian-config` at the terminal prompt or by clicking to the armbian-config menu item on desktop images.

3.8.1 System

- **I**nstall - installs to SATA, eMMC, NAND or USB. It gives you an option to install the system to more resilient and faster internal or external media. You can also change filesystem type to ext2,3,4 or BTRFS (if supported),
- **F**reeze - freeze or unfreeze kernel and board support packages, to avoid upgrading,
- **N**ightly - switch between nightly automated beta and stable builds,
- **B**ootenv - edit boot environment and alter kernel boot parameters,
- **H**ardware - toggle board low level functions: UART, I2C, SPI, ...
- **S**witch - switch to/between alternative kernels: legacy, current, edge
- **S**SH - reconfigure SSH daemon. Permit root login, toggle ssh key and mobile phone authentication,

© 2021 by Armbian

- **F**irmware - execute apt update and upgrade to update your system,
- **Z**shell - toogle stock BASH and ZSH with Oh My ZSH and tmux
- **E**nable - toggle desktop on and off (on desktop images)
- **L**ightdm - change login managers from none to lightdm (on desktop images)
- **R**DP - toggle remote desktop from Windows (on desktop images)
- **O**verlayroot - toggle overlayroot (Ubuntu images)
- **M**inimal - install minimal Armbian XFCE powered desktop,
- **D**efault - install Armbian XFCE powered desktop with web browser and extras.

3.8.2 Network

- **I**P - choose to select dynamic or edit static IP address,
- **H**otspot - create or manage wireless access point. If your wireless adapter is recognized by a kernel, then armbian-config utility auto selects best mode on the selected device. It can detect 802.11n, 802.11a and 802.11ac. It also knows how to handle some special Realtek adapters,
- **I**PV6 - toggle IPV6 for apt and system,
- **I**perf3 - toogle network troughput tests daemon,
- **L**TE - 3G/4G LTE modem management
- **W**iFi - manage wireless networking. Connect with Wifi network. You can create multiple wireless connections at the same time. They are managed by Network Manager,
- **B**T install - pair Bluetooth devices without PIN code,
- **A**dvanced - edit network config manually,
- **F**orget - disconnets and clear all wireless connections.

3.8.3 Personal settings

- **T**imezone - change timezone,
- **L**ocales - reconfigure language and character set,
- **K**eyboard - change console keyboaard settings,
- **H**ostname - change hostname,
- **M**irror - change to backup APT repository mirror in case of troubles,
- **W**elcome - toggle welcome screen items.

© 2021 by Armbian

3.8.4 Software

Software installation menu provides automated install of the following packages.

- **s**ofty
- TV headend *(IPTV server)*
- Syncthing *(personal cloud)*
- SoftEther VPN server *(VPN server)*
- Plex *(Plex media server)*
- Radarr *(Movie downloading server)*
- Sonarr *(TV shows downloading server)*
- Transmission *(torrent server)*
- ISPConfig *(WEB & MAIL server)*
- NCP *(Nextcloud personal cloud)*
- Openmediavault NAS *(NAS server)*
- PI hole *(ad blocker)*
- UrBackup *(client/server backup system)*
- Docker *(Docker CE engine)*
- Mayan EDMS *(Document management system within Docker)*
- MiniDLNA *(media sharing)*
- **M**onitor = simple CLI monitoring
- **D**iagnostics = create a summary of logs and upload them to paste.bin
- **T**oggle kernel headers, RDP service, Thunderbird and Libreoffice (desktop builds)

3.8.5 Sources

https://github.com/armbian/config

© 2021 by Armbian

3.9 Device Tree overlays

Most in-circuit and GPIO based interfaces (SPI, I2C, I2S, UART, ...) don't have a mechanism for detecting and identifying devices connected to the bus, so Linux kernel has to be told explicitly about the device and its configuration details.

While Device Tree is a way of describing hardware configuration to the kernel, Device Tree overlays are a way for modifying the DT in order to provide the kernel and kernel drivers with details about external devices or to activate interfaces disabled by default.

Note: from the Linux kernel maintainer perspective all unused in-circuit type interfaces that use GPIO pins should be disabled by default and all pins on pin headers or soldering pads will be configured as standard GPIOs.

Note: from the Linux kernel maintainer perspective all dedicated interfaces like USB, Ethernet or analog audio that are wired to soldering pads or a pin headers instead of specialized sockets (like USB socket, Ethernet socket or 3.5mm jack) will be left disabled by default.

3.9.1 Armbian specific notes

- DT overlays are a Work-in-Progress (WIP) feature, present **only in certain images**.
- Please note that different SoCs will have different sets of available overlays.

3.9.2 Quick start

1. Check the `README.<soc-id>-overlays` in `/boot/dtb/overlay/` (32-bit SoCs) or `/boot/dtb/allwinner/overlay/` (64-bit SoCs) for a list of provided overlays, their required and optional parameters
2. Add names of overlays you want to activate to `overlays=` line in `/boot/armbianEnv.txt`, separated with spaces
3. Add required parameters with their values to `/boot/armbianEnv.txt`, one per line
4. Add optional parameters with their values to `/boot/armbianEnv.txt` if you want to change the default value, one per line
5. If you didn't find the required overlay or want to change one of provided overlays, refer to "Using custom overlays" section
6. Reboot

3.9.3 Using custom overlays

1. Check here for some example overlays
2. Copy or create your overlay file (with `.dts` extension) on the device
3. Change I2C or SPI bus number, GPIO and pinctrl pins, `compatible` string to match your SoC if necessary
4. Compile and activate the overlay by running `armbian-add-overlay <overlay_file.dts>` as root, i.e. `sudo armbian-add-overlay sht15.dts`
5. Reboot

3.9.4 armbianEnv.txt entries reference

- `overlay_prefix` - prefix for the DT and overlay file names, set at OS image creation time
- `overlays` - list of overlays to activate from kernel directory
- `user_overlays` - list of overlays to activate from `/boot/overlay-user/` directory
- `param_*` - overlay parameters

© 2021 by Armbian

3.9.5 Kernel provided vs user provided overlays

Overlays can be loaded from 2 locations:

- `/boot/dtb/overlay/` (`/boot/dtb/allwinner/overlay/` for 64-bit SoCs) - kernel provided overlays
- `/boot/overlay-user/` - user provided overlays

Main differences between these locations:

- Kernel provided overlays are updated with the kernel packages, any changes to this directory (including new files) will be lost on kernel upgrade
- Kernel provided directory may contain overlays for different SoCs, so overlay file name pattern will be `<prefix>-<name>`, for example `sun8i-h3-i2c0.dtbo`, where `sun8i-h3` is the prefix and `i2c0` is the name
- Kernel provided overlays are activated by the overlay name (i.e. `i2c0`), and the prefix is set at OS image creation time
- User provided overlays directory is empty by default and is meant for storing and using user created overlays that are not present in the kernel packages or modified stock overlays
- User provided overlays are activated by the file name (excluding the extension), i.e. for file `adafruit13m.dtbo` overlay name would be `adafruit13m`

3.9.6 Activation

DT overlays are activated by editing u-boot environment file `/boot/armbianEnv.txt`

- Kernel provided overlays are activated by adding a name to the `overlays` variable
- User provided overlays are activated by adding a name to the `user_overlays` variable
- No more than one `overlays` line and one `user_overlays` line can be present in the environment file
- Multiple names should be separated by space
- If activated overlays have parameters marked as “Required”, those parameters have to be set to proper values
- Reboot is required for any changes to take effect

3.9.7 Overlay parameters

Some overlays have additional parameters that can be set.

Parameters marked as “Required” have to be set if overlay with these parameters is activated, other parameters are not mandatory if default value is suitable.

Parameters are set by adding their name and value to `/boot/armbianEnv.txt`, each parameter should be added on a new line.

Please refer to `README.<SoC_prefix>-overlays` files in `/boot/dtb/overlay/` (`/boot/dtb/allwinner/overlay/` for 64-bit SoCs) directory for supported parameters, i.e. `README.sun8i-h3-overlays` for H3 based boards.

Parameters of type `pin` require special format:

- Value consists of a letter `P`, a letter that signifies the pin bank and a number of the pin in the bank
- Letters should be upper case
- Numbers should not contain leading zeros
- Examples: good - `PA9`, `PG12`; bad - `pa2`, `PG08`

3.9.8 Overlay bus selection

SoCs may contain multiple bus controllers of the same type, i.e. Allwinner H3 contains 2 SPI controllers and Allwinner A20 contains 4 SPI controllers.

Please refer to your board documentation and schematic to determine what pins are wired to the pin headers and thus what bus number should be used in each case.

© 2021 by Armbian

3.9.9 Overlay pinmux conflicts

Some controllers may share the SoC pins in some configurations. For example on Allwinner H3 UART 3 and SPI 1 use the same pins - `PA13`, `PA14`, `PA15`, `PA16`. In this case activating both UART 3 and SPI 1 would result in a conflict, and on practice only one interface (either SPI or UART) will be accessible on these pins.

Please check the SoC specific README, board schematic, SoC datasheet or other documentation if you are unsure about possible conflicts if activating multiple overlays for the controllers that use shared (muxed) pins.

3.9.10 Overlay device endpoint conflicts

Overlays for devices that use addresses or similar mechanisms (i.e. SPI chip selects) can't be activated simultaneously if addresses (chip selects) are identical.

For example A20 SPI controller 1 has only one hardware chip select, so `spi-spidev` and `spi-jedec-nor` overlays cannot be activated both if they would use the same bus number and chip select.

3.9.11 Overlay compatibility

Device Tree overlays for different platforms and SoCs are not directly compatible. This, for example, means that overlays for H3 may need some changes to work on A20, and that Raspberry Pi overlays will need adjustments in order to be used on Allwinner based boards.

Rework may include changing labels, references (phandles) and pinconf bindings.

3.9.12 Notes regarding SPI and I2S overlays

Activating a device on SPI or I2S bus may require more than one overlay. In case a bus overlay like `spi0` or `i2s0` exist for the target SoC they need to be activated in addition to a slave device overlay (provided or custom/user-made). Please note that these overlays (`spi0`, `i2s0`) do not enable any slave devices (like spidev or I2S codec).
In some cases it might be necessary to change `param_spidev_spi_bus` to `1`.

3.9.13 Debugging

As overlays and overlay parameters are applied by the u-boot, it is impossible to get any debugging information (such as error messages) from the OS.

Serial console on UART 0 is **required** to debug DT overlay related problems.

3.9.14 Example `/boot/armbianEnv.txt` contents:

```
verbosity=1
console=serial
overlay_prefix=sun8i-h3
rootdev=UUID=bd0ded76-1188-4b52-a20a-64f326c1f193
rootfstype=ext4
overlays=w1-gpio uart1 i2c0 spi-spidev
param_w1_pin=PA20
param_w1_pin_int_pullup=1
param_uart1_rtscts=1
param_spidev_spi_bus=0
```


© 2021 by Armbian

3.9.15 Example of serial console log when using several overlays:

```
## Executing script at 43100000
U-boot loaded from SD
Boot script loaded from mmc
265 bytes read in 182 ms (1000 Bytes/s)
5074230 bytes read in 532 ms (9.1 MiB/s)
5702664 bytes read in 579 ms (9.4 MiB/s)
Found mainline kernel configuration
32724 bytes read in 269 ms (118.2 KiB/s)
882 bytes read in 277 ms (2.9 KiB/s)
Applying kernel provided DT overlay sun8i-h3-w1-gpio.dtbo
506 bytes read in 326 ms (1000 Bytes/s)
Applying kernel provided DT overlay sun8i-h3-uart1.dtbo
374 bytes read in 377 ms (0 Bytes/s)
Applying kernel provided DT overlay sun8i-h3-i2c0.dtbo
788 bytes read in 347 ms (2 KiB/s)
Applying kernel provided DT overlay sun8i-h3-spi-spidev.dtbo
4327 bytes read in 268 ms (15.6 KiB/s)
Applying kernel provided DT fixup script (sun8i-h3-fixup.scr)
## Executing script at 44000000
tmp_bank=A
tmp_pin=20
## Loading init Ramdisk from Legacy Image at 43300000 ...
   Image Name:   uInitrd
   Image Type:   ARM Linux RAMDisk Image (gzip compressed)
   Data Size:    5074166 Bytes = 4.8 MiB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 43000000
   Booting using the fdt blob at 0x43000000
   reserving fdt memory region: addr=43000000 size=9000
   Loading Ramdisk to 49b29000, end 49fffcf6 ... OK
   Loading Device Tree to 49b1d000, end 49b28fff ... OK

Starting kernel ...
```


© 2021 by Armbian

3.10 Support Definitions, Criteria and Relationships

3.10.1 Overview

As mentioned in our announcement, made adjustments to ease the burden on the team, make the meaning “supported” status of an SBC clear to the community and vendors.

Support Status from Lowest to Highest are the following:

1. EOL
2. CSC
3. WIP
4. Standard
5. Gold
6. Platinum

3.10.2 EOL - End of Life

EOL devices are not under active development by any entity or community. Can be removed from Armbian code base at any time. Left as a courtesy in case a community member wants to attempt to resurrect development.

No benefits provided

3.10.3 Community Support

Community Support SBCs are exclusively supported by the community.

No Images are provided for Community Supported SBCs

- no stable
- no periodic / nightly
- users must use Armbian Build Framework to produce an image

Benefits for Community Support

- pull requests are accepted
- Armbian team will interact on Github with contributors considered to be acting in good faith
- periodic / Nightly Debian packages are built and published into Armbian’s community apt repository

Requirements for Community Support

- patch or component does not break Armbian Build Framework
- patch or component does not break build of supported or other CSCs
- generally considered to “work most of the time”
- generally considered to recieve periodic maintenace from community

3.10.4 WIP - Work in progress

WIP status is for when a maintainer has commited to an SBC, but is not ready to ship stable images.

Benefits of Community Supported SBCs apply to WIP.

© 2021 by Armbian

Additional Benefits provided for a WIP SBC

- periodic / nightly CLI images are published by Armbian
- Armbian will work with SBC maintainer to assure compatiblity within the Armbian Build System
- eligible for promotion to Standard Support

Criteria for WIP status

- must satisfy standard support criteria
- must show active development

3.10.5 Armbian Standard Support

Benefits provided for a Supported SBC

- Armbian will publish and distribute “stable” CLI images through its mirror network
- Armbian will publish and distribute “periodic / nightly” CLI and desktop images
- Armbian will work with SBC maintainer to assure compatiblity within the Armbian Build System
- Armbian will provide the team’s unique expertise to assist maintainer with general challenges
- best-effort compensation will be provided to maintainer from the “Armbian Community Fund”

Criteria for Supported

For an SBC to be considered supported:

- a named individual as “*maintainer*” along with GitHub ID must be clearly identifed in the Board Configuration File
- a named individual must commit to providing “*best effort*” support for their SBC on the Armbian forums
- maintainer must participate in the Release Process
- maintainer must sign-off that device has been tested, is stable, and ready for release during release process
- maintainer must have physical access to the SBC they are supporting

Additional Caveats:

- If the burden placed on the Armbian team is too high funding maybe needed to assure:
- R&D bills paid. Failsafe in case Armbian members or a team was hired
- support bills covered by vendor or end users with crowdfunding campaign
- supported is **not** applied to a “board family” or group of related SBCs. It is per SBC
- a maintainer can support multiple devices but must satisfy all requirements above per SBC
- any individual can be a maintainer for a standard support SBC
- missed released will result in immediate forfeit of “Armbian Supported” status and demotion to CSC status unless Armbian team grants exemption

3.10.6 Gold and Platinum Support

Gold and Platinum Support are intended to be business relationships with the Armbian project or team and are out of scope of this document. Please connect with Armbian team for more information.

© 2021 by Armbian

4. Hardware Notes

4.1 Generic howto for Allwinner devices

4.1.1 Legacy or current kernel ?

Many Armbian images come in two flavours : *Legacy* (using an older kernel version) and *current* (up-to-date LTS kernel). Depending on kernel version, the procedure to enable/disable features is not the same.

- *Legacy* kernel (5.4.x): DT (Device Tree) overlays
- *Current* kernel (5.10.x) : DT (Device Tree) overlays

Note: Support for older kernel versiones (like 3.4.x or 3.10.x) has been dropped.

What flavour am I using ?

Best way to know is by checking your kernel version :

```
root@bananapipro:~# uname -a
Linux bananapipro 4.5.2-sunxi #11 SMP Thu Apr 28 21:53:25 CEST 2016 armv7l GNU/Linux
```

In this example the kernel version is 4.5.2 so you can use DT to tweak some settings. If you get a kernel version 3.X then you'll be certainly using FEX like on an Orange Pi Plus 2E :

```
root@orangepiplus2e:~# uname -a
Linux orangepiplus2e 5.4.45-sun8i #10 SMP PREEMPT Wed Jun 1 19:43:08 CEST 2016 armv7l GNU/Linux
```

4.1.2 Enable Hardware Features

Some boards require some manual configuration to turn on/off certain features. In some cases, the procedure is "less than obvious", so we document some basic examples here.

How to reconfigure video output?

This affect *current* kernel only.

U-Boot supports HDMI and LCD output on Allwinner sunxi SoCs, LCD output requires the `CONFIG_VIDEO_LCD_MODE` Kconfig value to be set.

© 2021 by Armbian

The sunxi U-Boot driver supports the following video-mode options:

- `monitor=[none|dvi|hdmi|lcd|vga|composite-*]` - Select the video output to use
- `none` : Disable video output.
- `dvi/hdmi` : Selects output over the hdmi connector with dvi resp. hdmi output format, if edid is used the format is automatically selected.
- `lcd` : Selects video output to a LCD screen.
- `vga` : Selects video output over the VGA connector.
- `composite-pal/composite-ntsc/composite-pal-m/composite-pal-nc` : Selects composite video output, note the specified resolution is ignored with composite video output.
- Defaults to `monitor=dvi` .
- `hpd=[0|1]` - Enable use of the HDMI HotPlug Detect feature 0: Disabled. Configure DVI/HDMI output even if no cable is detected 1: Enabled. Fallback to the LCD / VGA / none in that order (if available) Defaults to `hpd=1` .
- `hpd_delay=<int>` - How long to wait for the HDMI HPD signal in milliseconds When the monitor and the board power up at the same time, it may take some time for the monitor to assert the HPD signal. This configures how long to wait for the HPD signal before assuming no cable is connected. Defaults to `hpd_delay=500` .
- `edid=[0|1]` - Enable use of DDC + EDID to get monitor info 0: Disabled. 1: Enabled. If valid EDID info was read from the monitor the EDID info will overrides the xres, yres and refresh from the video-mode env. variable. Defaults to `edid=1` .
- `overscan_x/overscan_y=<int>` - Set x/y overscan value This configures a black border on the left and right resp. top and bottom to deal with overscanning displays. Defaults to `overscan_x=32` and `overscan_y=20` for composite monitors, 0 for other monitors.

For example to always use the HDMI connector, even if no cable is inserted, using edid info when available and otherwise initalizing it at 1024x768@60Hz, use: `setenv video-mode sunxi:1024x768-24@60,monitor=dvi,hpd=0,edid=0` .

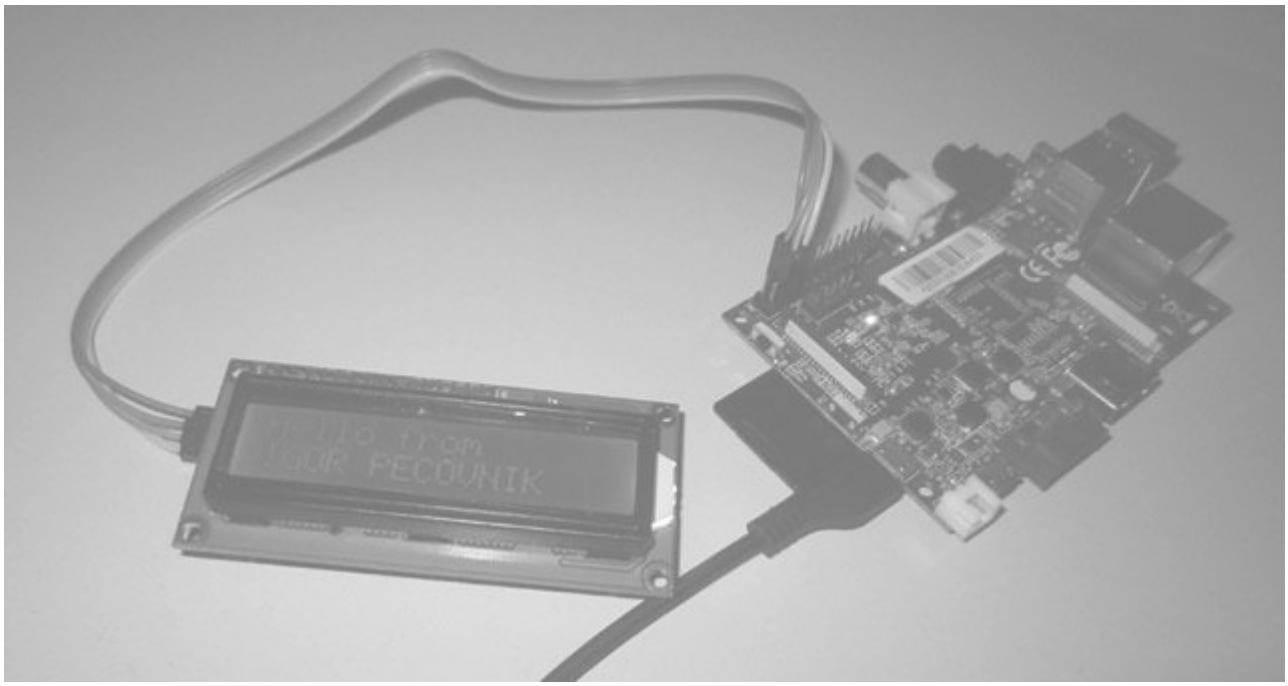
Parameters regarding video must be saved into U-Boot environment file since they must be read before reading boot script. You can do this by adding `saveenv` command at the end of boot script (boot.cmd). Remember to recompile boot.cmd to boot.scr and note that changes will come into action after second boot.

Connect your LCD display

I tried three different display connection types: I2C, (4bit) parallel and SPI. All of them are working perfectly with my image. I didn't took a picture of the third one. It's a standard Hitachi HD44780 based 20×4 LCD, wired and tested according to wiringBP example.

© 2021 by Armbian

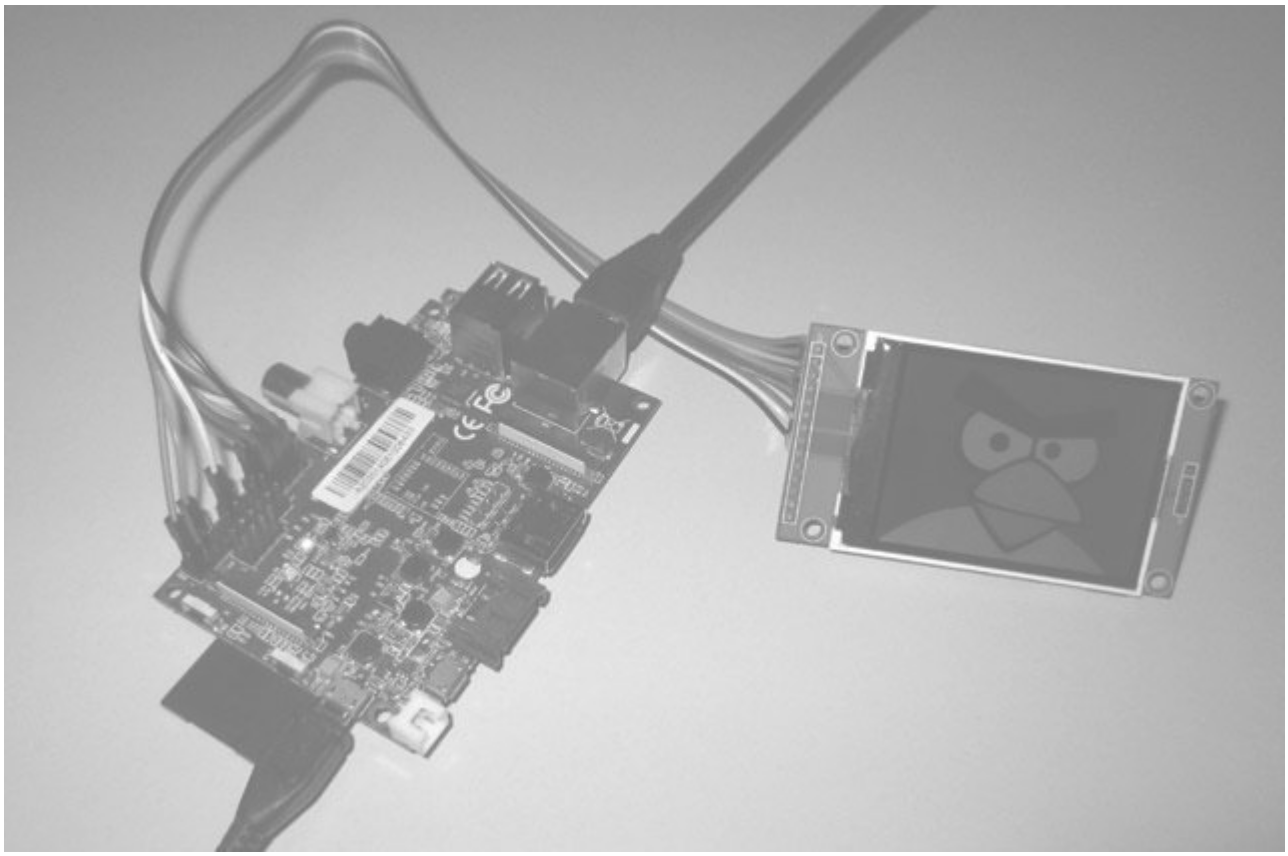
I2C



I am using this code for mainline kernel and with changed line: /dev/i2c-%u = /dev/i2c-2 for Legacy kernel.

© 2021 by Armbian

SPI



- I am using 2.4″ 240×320 SPI TFT LCD Serial Port Module+5/3.3V Pbc Adapter Micro SD ILI9341
- Wire according to this map.
- You have to use Armbian 1.5 or newer. Currently working only under Legacy kernel.
- Add this to your /etc/modules: `fbtft_device name=adafruit22a rotate=90 speed=48000000 fps=50 gpios=reset:25,led:19,dc:24`
- Reboot
- Test – display some picture on the screen: `fbi -d /dev/fb2 -T 1 -noverbose -a yourimage.jpg`

LVDS

Currently working only under Legacy kernel.

Image has pre-loaded settings for two LVDS display.

To enable 7 inch.

```
ln -sf /boot/bin/bananapilcd7.bin /boot/script.bin
```

To enable 5 inch.

```
ln -sf /boot/bin/bananapilcd5.bin /boot/script.bin
```

If you need touch screen support, add this module to your /etc/modules

```
ft5x_ts
```

FRAMEBUFFER

Linux Framebuffer drivers for small TFT LCD display modules.

© 2021 by Armbian

4.1.3 FEX (outdated/unsupported, informational only)

Which file should I edit

Armbian embed a lot of BIN files, but a symlink point to the one in use :

```
root@orangepiplus2e:~# ls -la /boot/script.bin
lrwxrwxrwx 1 root root 22 Jun  1 20:30 /boot/script.bin -> bin/orangepiplus2e.bin
```

Updating a FEX

You may need to use `sudo` with all the following commands.

The whole process won't overwrite any of your files. If you're paranoid, you can make a proper backup of your BIN file :

```
cp /boot/script.bin /boot/bin/script.bin.backup
```

Then you can decompile your BIN into a FEX :

```
bin2fex /boot/script.bin /tmp/custom.fex
```

Finally you can edit your FEX file with your favorite text editor and compile it back to a BIN :

```
fex2bin /tmp/custom.fex /boot/bin/custom.bin
```

The last step is to change the symlink to use your custom BIN :

```
ln -sf /boot/bin/custom.bin /boot/script.bin
```

H3 based Orange Pi, legacy kernel

ENABLE SERIAL /DEV/TTYS3 ON PINS 8 AND 10 OF THE 40 PIN HEADER

Update the FEX configuration (which is compiled into a .bin) located at /boot/script.bin

Decompile .bin to .fex

```
cd /boot
bin2fex script.bin > custom.fex
rm script.bin # only removes symbolic link
```

Edit .fex file

```
[uart3]
uart_used = 1 ; Change from 0 to 1
uart_port = 3
uart_type = 2 ; In this case we have a 2 pin UART
uart_tx = port:PA13<3><1><default><default>
uart_rx = port:PA14<3><1><default><default>
```

Compile .fex to .bin

```
fex2bin custom.fex > script.bin
```

Reboot

Notice that /dev/ttyS3 appears. That is your new UART device.

© 2021 by Armbian

4.2 Allwinner A10 & A20 boards

4.2.1 Overview

Both *legacy* and *current* kernels are stable and production ready and mostly sharing the same features through backports.

- **Note: Kernel 3.4.x has been dropped due to lacking upstream support which has ended in 2017.**

Features

- Enabled audio devices: analog, 8 channel HDMI, spdif and I2S (if wired and enabled in HW configuration)
- Bluetooth ready (working with supported external keys)
- Enabled overlayfs (outdated)
- I2C ready and tested with small 16×2 LCD. Basic i2c tools included.
- SPI ready and tested with ILI9341 based 2.4″ TFT LCD display.
- Drivers for small TFT LCD display modules.
- Clustering / stacking
- Onboard LED attached to SD card activity (script.bin)
- Docker ready
- Enabled audio devices: analog, SPDIF (if available) & USB
- USB / UAS – more efficient disk access over USB (A20 and H3)
- CAN bus – Controller Area Network
- USB OTG connector – OTG or host mode
- Bluetooth ready (working with supported external keys)
- I2C ready and tested with small 16×2 LCD. Basic i2c tools included.
- Onboard LED attached to SD card activity (not enabled on all boards yet)

Bugs or limitation

- NAND install sometime fails. Workaround: install ~~Lubuntu to NAND~~ with ~~Phoenix tools~~ and run install again. (Links no longer available - 27/01/21 Werner)
- Shutdown results into reboot under certain conditions.
- SATA port multiplier support is disabled by default, can be enabled by adding kernel parameter `ahci_sunxi.enable_pmp=1`
- Screen output from kernel is set to HDMI by default. Boot loader can detect and switch, kernel not.

© 2021 by Armbian

4.2.2 Desktop

- HW accelerated video playback (if available)
- MALI Open GLES (if available)
- Pre-installed: Firefox, LibreOffice Writer, Thunderbird
- Lightweight XFCE desktop
- Autologin, when normal user is created – no login manager (/etc/default/nodm)

4.2.3 Notes

Setting non-standard monitor settings for A10, A20 and A31 based boards in u-boot

Following commands (example) needs to be executed in u-boot command prompt:

```
setenv video-mode sunxi:1024x768-24@60,monitor=dvi,hpd=0,edid=0,overscan_x=1,overscan_y=2
saveenv
```

Since environment is reset after flashing u-boot, you need to do this after every u-boot upgrade or put this to u-boot script

4.2.4 Resources

Armbian packages repository

© 2021 by Armbian

4.3 Allwinner H3 boards

4.3.1 Overview

The H3 SoC from Allwinner is meant for OTT boxes and therefore its reference design is *not* accompanied by a separate PMIC (power management IC) unlike *A series* Allwinner SoCs (like A10, A20, A64, ...). No PMIC means also that there is no battery charging/monitoring implemented so H3 is not that much suited for mobile devices. On the other hand some pretty cheap H3 boards were released that can be driven with rather low consumption and therefore combining H3 devices with a battery became a real use case with boards like Orange Pi One/Lite, NanoPi NEO and Neo AIR.

As usual SoC and device information can be found in Linux-sunxi wiki. Same applies to status of mainlining kernel efforts. Adding to the usual SoC feature set (I2C, SPI, PWM, UART, SDIO, GPIO and so on) H3 has one USB OTG port, 3 real USB host ports (not exposed on all devices), Fast- and Gigabit Ethernet capablities (board specific), a Mali400MP2 GPU and Allwinner's video encoding/decoding engine.

When CPU or GPU cores are fully utilized H3 tends to overheat over time like any other popular ARM SoC released within the last 2-3 years. With Armbian we provide sane dvfs (dynamic voltage frequency scaling) settings that help a lot with throttling. In case you plan to operate your H3 device constantly under high load please check Armbian forums first since boards behave differently (related to voltage regulation and PCB size and design – some use copper layers to spread the heat away from the SoC). Also consider applying a heatsink to the SoC (a fan should not be necessary unless you want to do number crunching on your board and then you obviously chose the wrong device).

You find some differentiation criteria regarding supported H3 devices as well as an overview/history of H3 software support in our forums or use Jean-Luc's nice comparison table (both slightly outdated since more H3 devices have been released in the meantime).

4.3.2 Kernel support

Almost all features of the H3 SoC are supported on Armbian's *current* branch. Please refer to the Linux sunxi support sheet.

4.3.3 Default settings

- CPU frequency settings are 480 MHz to 1.37 GHz on most boards (cpufreq governor is *interactive* therefore the boards only increase CPU speed and consumption when needed). Varity might occur due to different voltage regulators and heat dissipation behaviour. Theoretically lower frequencies are possible but disabled by default due to known stability issues.
- Armbian unlike older/other H3 OS images uses the green led as 'power on' indicator (blinking means 'ready to login' or 'shutting down'), the red led (blue on NanoPis) can be used for your own purpose.

4.3.4 Tips and tricks (general)

- An insufficient power supply **is the root cause of many weird symptoms/problems**. Never trust in ratings written on the PSU since they might be wrong, the PSU might be old/dying and cable/contact resistance adds to problems. In other words: Before you blame Armbian for strange behaviour please try at least one second power supply (this applies to both PSU and cable between PSU and board if this is separate – especially USB cables really suck due to high resistance leading to severe voltage drops).
- In case you experience instabilities check your SD card using `armbianmonitor -c $HOME` and think about installing RPi-Monitor for H3 to get an idea whether you suffer from overheating (`sudo armbianmonitor -r` will install everything needed).
- Especially for desktop images the speed of your SD card matters. If possible try to use our *nand-sata-install* script to move the rootfs away from SD card. The script also works with USB disks flawlessly (some background information).

4.3.5 Tips and tricks (H3 specific / lowering consumption) (outdated)

Recent research showed that H3 boards operated as wired IoT nodes need way less power compared to Raspberry Pis in the same situation (ethernet active). If you want to use your H3 device headless (server/IoT) and care about power consumption then

© 2021 by Armbian

there exist a couple of tweaks to get your board being more energy efficient when using in the meantime unsupported 3.x kernel (no tests done yet with up-to-date *legacy*/*current* kernel):

- Disabling HDMI/GPU saves ~200mW.
- Allowing to temporarely only negotiate a Fast Ethernet connection on GbE capable boards saves +350 mW.
- Adjusting DRAM clockspeed is surprisingly another way to control consumption (on NanoPi NEO for example changing DRAM clockspeed between 132 MHz and 672 MHz results in consumption differences of 470mW).
- Limiting maximum CPU clockspeed will help with lowering maximum consumption (think about scripts running amok or something going terribly wrong), the same applies to limiting the count of active CPU cores.
- Choosing a board with Fast instead of Gigabit Ethernet or disabling GbE on the latter using `ethtool` or `ifconfig` saves at least 150 mW (board specific).

As an example: We chose default Armbian settings for NanoPi NEO to ensure this board is not able to exceed 2W consumption when running with no peripherals connected. This resulted in CPU and DRAM clockspeed of just 480/408 MHz while *booting* (the first ~20 seconds). In normal operation we limit maximum CPU clockspeed to 912 MHz to stay below the 2W consumption barrier even in worst case scenarios.

In case you want to have a few more percent maximum CPU performance you would need to set maximum cpufreq to 1200 MHz instead of 'just' 912 MHz maximum CPU clock using our new h3consumption tool. Be warned: This will both heavily increase consumption and SoC temperature since exceeding 912 MHz CPU clockspeed means feeding the SoC with 1.3V instead of 1.1V core voltage (most smaller H3 devices use a voltage regulator only switching between two voltages to feed the SoC based on load).

Walking this route in the other direction is more interesting: In case you want to use an H3 device as IoT node you might want to limit both idle and maximum consumption. That should involve disabling stuff not needed (eg. HDMI/GPU since this saves 200mW) or limiting ressource consumption: Lowering maximum clockspeeds for both CPU and DRAM or even disabling CPU cores (which helps *not* with idle consumption since ARM cores enter low-power modes if not needed but can help lowering maximum consumption requirements).

Since all of this stuff is based on recent research and being still WiP please consider reading through relevant threads in Armbian forums **and** join development/research/discussions: SBC consumption/performance comparisons and Default settings for NanoPi NEO/Air.

© 2021 by Armbian

4.4 Allwinner H5 and A64 boards

4.4.1 Overview

See the generic Allwinner page

4.4.2 Warning

Using the board without cooling in conjunction with the stable release of Ambian using kernel 5.4.y there is a risk of the board getting **permanently damaged** due to overheating. If you decide to try the board without cooling, you can use `sudo armbianmonitor -r` to keep an eye on the temperatures.

© 2021 by Armbian

4.5 Allwinner H6

See also generic Allwinner page,

4.5.1 CPU frequency

~~The H6 CPU frequency has ben soft-capped at 1,48 GHz to avoid thermal throttling too fast. This limit can be lifted by editing~~ ~~`/etc/default/cpufrequtils`~~ ~~and set~~ ~~`MAX_SPEED`~~ ~~to~~ ~~`1810000`~~~~.~~
With the release of Armbian 20.05 “Kagu” new thermal zones have been added making this limitation obsolete and therefore has been removed. All H6 boards now clocking at the highest possible value OOB.

Warning Adding proper cooling is highly recommended.

4.5.2 PCIe (un-)supported

Some H6 SoC based boards (like Pine H64 Model a, discontinued) are shipped with a PCIe slot. This slot **cannot** work out of the box as it has to be considered as broken by design. Linux-Sunxi writes about this:

> Allwinner H6 has a quirky PCIe controller that doesn’t map the PCIe address space properly (only 64k accessible at one time) to CPU, and accessing the PCIe config space, I/O space or memory space will need to be wrapped. As Linux doesn’t wrap PCIe memory space access, it’s not possible to do a proper PCIe controller driver for H6. The BSP kernel modifies the driver to wrap the access, so it’s also not generic, and only devices with modified driver will work.

Icenowy is working on a wrapper to make PCIe work. Check forums.

4.5.3 Networking

Trouble getting ethernet connection? Check here

© 2021 by Armbian

4.6 Cubox and Hummingboard boards

4.6.1 Legacy

System images with legacy kernel

- Kernel 3.14.x with large hardware support, headers and some firmware included
- Docker ready – what is Docker?
- PCI-E operational (Hummingboard Pro, Gate & Edge)
- mSATA / m2 operational (Hummingboard Pro & Edge)
- Enabled audio devices: HDMI, spdif, analogue
- Bluetooth ready (working with Cubox-i/HB PRO on-board device or external key)
- I2C ready and tested with small 16×2 LCD. Basic i2c tools included.
- SPI ready and tested with ILI9341 based 2.4″ TFT LCD display.
- Drivers for small TFT LCD display modules.
- USB redirector – for sharing USB over TCP/IP (disabled by default /etc/init.d/rc.usbsrvd)

Bugs or limitation

- Gigabit ethernet transfer rate is around 50% of its theoretical max rate (internal chip bus limitation)

4.6.2 Mainline

System images with mainline kernel

- Mainline with large hardware support, headers and some firmware included
- Docker ready – what is Docker?
- PCI-E operational (Hummingboard Pro, Gate & Edge)
- mSATA / m2 operational (Hummingboard Pro & Edge)
- Enabled audio devices
- Bluetooth ready (working with supported external keys)

Set board type

By default the mainline Hummingboard / Cubox-i Armbian images are configured for the Cubox-I with a dual or quad core SOM.

To run the image on another platform or SOM the correct device tree file needs to be specified. These files are stored in `/boot/dtb`. To set the correct device tree blow a `fdt_file=` parameter needs to be added to `/boot/armbianEnv.txt`.

Set the `fdt_file=` parameter according to the list below:

IMX6 Solo / DualLite:

- [Hummingboard] `fdt_file=imx6dl-hummingboard.dtb`
- [Hummingboard v2] `fdt_file=imx6dl-hummingboard2.dtb`
- [Cubox-I] `fdt_file=imx6dl-cubox-i.dtb`

IMX6 Dual / Quad:

- [Hummingboard] `fdt_file=imx6q-hummingboard.dtb`
- [Hummingboard v2] `fdt_file=imx6q-hummingboard2.dtb`
- [Cubox-I] `fdt_file=imx6q-cubox-i.dtb` (Default)

© 2021 by Armbian

If a v1.5 SOM (with eMMC) is used the device tree names need to be changed, e.g.: `imx6q-cubox-i-som-v15.dtb` or `imx6q-cubox-i-emmc-som-v15.dtb`.

Bugs or limitation

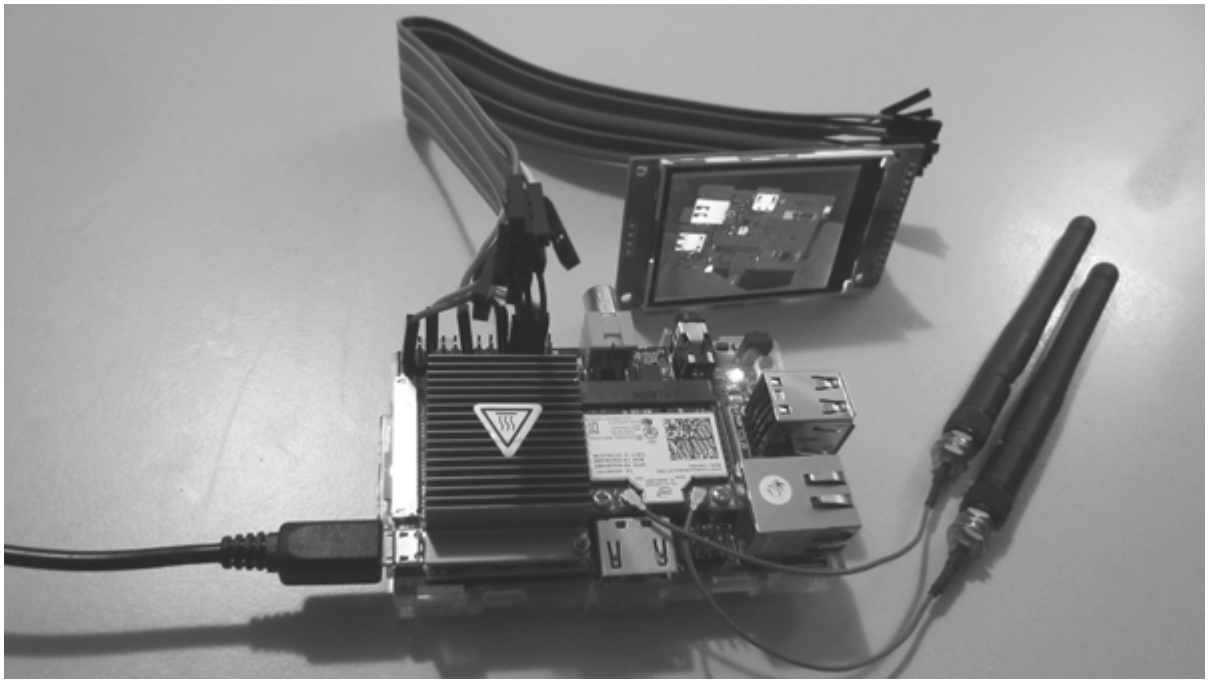
- Gigabit ethernet transfer rate is around 50% of its theoretical max rate (internal chip bus limitation)

4.6.3 Desktop

- Pre-installed: Firefox, LibreOffice Writer, Thunderbird
- Lightweight XFCE desktop
- Autologin, when normal user is created – no login manager (/etc/default/nodm)

4.6.4 Connect your LCD display

I tried two different display connection types: I2C and SPI. Both are working perfectly with my image 2.6 or higher.



- I am using 2.4″ 240×320 SPI TFT LCD Serial Port Module+5/3.3V Pbc Adapter Micro SD ILI9341
- Wire according to this map.
- You have to use Armbian 1.5 or newer. Currently working only under Legacy kernel.
- Add this to your /etc/modules: `fbtft_device name=adafruit22a rotate=90 speed=48000000 fps=50 gpios=reset:67,led:72,dc:195 busnum=1`
- Reboot
- Test – display some picture on the screen: `fbi -d /dev/fb2 -T 1 -noverbose -a yourimage.jpg`
- Troubleshooting and settings for other displays LVDS

4.6.5 GPIO

How to control HummingBoard GPIO from kernel space?

© 2021 by Armbian

4.6.6 Udoo Quad

- Kernel 3.14.x and 4.4.x with some hardware support, headers and some firmware included
- Docker ready – what is Docker?
- Wireless adapter with DHCP ready but disabled (/etc/network/interfaces, WPA2: normal connect, bonding / notebook or AP mode). It can handle between 40-70Mbit/s.
- SATA operational
- Enabled analogue (VT1613) and HDMI audio device

Bugs

SATA & USB install not working on legacy kernel

4.6.7 Udoo Neo

- Kernel 3.14.x with some hardware support, headers and some firmware included
- Wireless adapter with DHCP ready but disabled

© 2021 by Armbian

4.7 Marvell Armada

4.7.1 Helios4

Overview

All builds provide 100% hardware support for Helios4.

Build Version Status

LEGACY

- U-Boot : Mainline 2019.04
- Linux Kernel : Mainline 4.19.y

CURRENT

- U-Boot : Mainline 2019.04
- Linux Kernel : Mainline 5.4.y

Known Limitations

- SDcard High Speed timing have compatibility issue with some brands. Temporary workaround : Disable UHS option/support. Can be manually enable, refer to the following page (2020-11-25 link broken archive).
- During SATA heavy load, accessing SPI NOR Flash will generate ATA errors. Temporary workaround : Disable SPI NOR flash. Can be manually enable, refer to the following page (2020-11-25 link broken archive).

Notes

Find more details about hardware support and configuration on Helios4 Wiki.

4.7.2 Clearfog Pro/Base

Overview

Both builds provide close to 100% hardware support, some slight differences are listed below.

Build Version Status

LEGACY/CURRENT

- Mainline kernel with large hardware support, headers and some firmware included
- Docker ready
- Both mPCIe are operational and convertible to mSATA, M2 operational
- Added patch to unlock Atheros regulatory restrictions which unlock 5Ghz AP mode in cheap Atheros cards (ath9 driver)
- Bluetooth ready (working with supported external keys)
- I2C ready. Basic i2c tools included.
- SPI ready but untested.
- SFP is working at up to 1GB/s even with faster fiber modules
- SFP DDMI is operational (`sudo ethtool -m eth2`)

© 2021 by Armbian

Bugs or limitation

- all builds suffer from minor problems with specific mPCIe combinations. If you run into problems check this test matrix for some known working/not working combinations.

Converting mPCIe to mSATA

- To convert mPCIe to mSATA you have to enable the corresponding patches in u-boot-mvebu. Afterwards rebuild u-boot with our build system and write the new u-boot to your boot medium. If you need assistance ask in the forum.

Notes

- In case you ever run into troubles and ask for help in the forums please ensure to provide a serial console log (UART adapter on board accessible through Micro USB with 115200/8/N/1 settings)
- The boards can boot from various sources that are adjustable with a DIP switch. Comprehensive information about the necessary preparations available here.

© 2021 by Armbian

4.8 Rockchip RK3399 boards

4.8.1 Graphics hardware acceleration (3D and VideoEngine)

Both only work with legacy (4.4.y) kernel using Buster userspace.
Check https://forum.armbian.com/topic/16516-rk3399-legacy-multimedia-framework/

3D acceleration in mainline is still in the works and is covered by Panfrost. There are still other issues like “unusual” screen resolutions like 1920x1200 are neither properly detected nor supported while classic *FullHD* 1920x1080 is fine.

4.8.2 USB-C

Partially works in legacy (4.4.y). Does not work with mainline due to lack of drivers.

4.8.3 Overclock overlay

The RK3399 can be overclocked at own risk. Use `rockchip-rk3399-opp-2ghz.dtbo` this.

4.9 Rockchip RK3328 / RK 3288

A similar multimedia framework is available if using legacy 4.4.y kernel on Buster userspace.
Check https://forum.armbian.com/topic/16517-rk3288rk3328-legacy-multimedia-framework/

© 2021 by Armbian

5. Developer Guide

5.1 Building Armbian

5.1.1 What do I need?

- x86/x64 machine running any OS; at least 4G RAM, SSD, quad core (recommended),
- VirtualBox or similar virtualization software **(highly recommended with a minimum of 25GB hard disk space for the virtual disk image)**
- **The officially supported** compilation environment is Ubuntu Hirsute 21.04.x amd64 **only!**
- Ubuntu Focal can be used for building Bionic, Focal and Buster images as well, unsupported though
- `binfmt_misc` kernel module (some *ubuntu-cloud* images do not have this module. Switch to a generic kernel if that is the case.)
- installed basic system, OpenSSH and Samba (optional)
- no spaces in full path to the build script location allowed
- superuser rights (configured `sudo` or root shell).

Not officially supported build environments from community contributions:

- Setting up VirtualBox and compile environment is easy following our Vagrant tutorial,
- Docker environment is also supported for building kernels and full OS images,
- Multipass

Please note that system requirements (both hardware and OS/software) may differ depending on the build environment (Vagrant, Docker, Virtualbox, native).

5.1.2 How to start?

Native and VirtualBox environments:

Login as root and run:

```
apt-get -y -qq install git
git clone --depth 1 https://github.com/armbian/build
cd build
```

Run the script

```
./compile.sh
```

Make sure that full path to the build script **does not contain spaces**.

5.1.3 Providing build configuration

After the first run of `compile.sh` a new configuration file `config-example.conf` and symlink `config-default.conf` will be created. You may edit it to your needs or create different configuration files using it as a template.

Alternatively you can supply options as command line parameters to compile.sh. Example:

```
./compile.sh BOARD=cubietruck BRANCH=current KERNEL_ONLY=no RELEASE=focal
```

Note: Option `BUILD_ALL` cannot be set to “yes” via command line parameter.
Note: Names for `BOARD` can be found here by looking at file names. Example: OrangePi 4 = **orangepi4.conf** = `BOARD=orangepi4`

© 2021 by Armbian

Base and descendant configuration

You can create one base configuration (`config-base.conf`) and use this in descendant config (`config-edge.conf`). Three parameters (*BRANCH*, *RELEASE*, *COMPRESS_OUTPUTIMAGE*) will be overwritten.

```
. ./config-base.conf

BRANCH="edge"
RELEASE="buster"
COMPRESS_OUTPUTIMAGE="sha,gz"
```

5.1.4 Using our automated build system

If you do not own the proper equipment to build images on your own, you can make use of the automated build system. Packages are recompiled every night (starting at 00:01 CEST) and a few testing images are produced. These images are accessible on the download server under board folder, subfolder "*Nightly*". Packages, when successfully built, are published in the *beta* repository. You can switch to *beta* repository in armbian-config or by changing *apt.armbian.com* to *beta.armbian.com* in /etc/apt/sources.list.d/armbian.list.

Board beta images are defined in board configuration files which are located here. This is a typical board configuration:

```
    # A20 dual core 1Gb SoC
    BOARD_NAME="Banana Pi"
    LINUXFAMILY="sun7i"
    BOOTCONFIG="Bananapi_defconfig"
    MODULES="hci_uart gpio_sunxi rfcomm hidp sunxi-ir bonding spi_sun7i 8021q a20_tp #ap6211"
    MODULES_NEXT="brcmfmac bonding"
    #
    KERNEL_TARGET="legacy,current,edge"
    CLI_TARGET="buster,focal:current"
    DESKTOP_TARGET="buster:legacy,current"

    CLI_BETA_TARGET=""
    DESKTOP_BETA_TARGET=""
    #
    BOARDRATING=""
    CHIP="https://docs.armbian.com/Hardware_Allwinner-A20/"
    HARDWARE="https://linux-sunxi.org/Banana_Pi"
    FORUMS="https://forum.armbian.com/forum/7-allwinner-a10a20/"
    BUY="https://amzn.to/2fToHjR"
```

You can find more information about those variables here.

If you want that our automated system start making images for this particular board, you need to alter parameters `CLI_BETA_TARGET` and `DESKTOP_BETA_TARGET` . Variants are dependent from `KERNEL_TARGET` definitions and supported userlands: `hirsute` , `focal` , `buster` . To edit those parameters you need to push changes to the build script. You need to fork a project and create a pull request and after it is imported by one of the administrators, images will start to show up in appropriate folder.

If you want to enable Debian buster desktop image with *current* kernel choose the following:

```
DESKTOP_BETA_TARGET="buster:current"
```

or for command line interfaces Ubuntu Focal based images with legacy kernel 4.19.x

```
CLI_BETA_TARGET="focal:legacy"
```

or for image with latest upstream development/bleeding edge kernel.

```
DESKTOP_BETA_TARGET="buster:edge"
```

5.1.5 Using alternate armbian builder repos and branches

By default, armbian-builder assumes working from `master` of `https://github.com/armbian/build.git` . If you are working from your own repo / branch, `touch .ignore_changes` will cause armbian-builder to not attempt a repo checkout.

© 2021 by Armbian

5.1.6 Executing any bash statement

Currently, invoking compile.sh will run a monotonous task of building all the components into a final image.

In some situation, especially when developing with Kernel or U-Boot, it is handy to run a portion of that great task like:

```
# using default profile
./compile.sh 'fetch_from_repo "$BOOTSOURCE" "$BOOTDIR" "$BOOTBRANCH" "yes"'
./compile.sh 'compile_uboot'
```

You can also dump the variable:

```
# using profile of `userpatches/config-my.conf`
./compile.sh my 'echo $SRC/cache/sources/$BOOTSOURCEDIR'
```

NOTE: please use single quotes to keep the `$VAR` from early expansion in the command line shell.

© 2021 by Armbian

5.2 Quick Start with Vagrant

5.2.1 Vagrant HOST Steps

The following steps are performed on the *host* that runs Vagrant.

Installing Vagrant and Downloading Armbian

VIRTUALBOX VERSION

WARNING: We'll be using Virtualbox as a virtualization provider for Vagrant. Virtualbox has documented issues running Xenial under heavy disk IO. Please make sure your version of Virtualbox is >= 5.1.12 where the issue, "Storage: fixed a problem with the LsiLogic SCSI controller where requests could be lost with SMP guests", appears to have been resolved.

First, you'll need to install Vagrant on your host box. Next, you'll need to install a plug-in that will enable us to resize the primary storage device. Without it, the default Vagrant images are too small to build Armbian.

```
vagrant plugin install vagrant-disksize
```

Now we'll need to install git and clone the Armbian repo. While this might seem obvious, we rely on it being there when we use Vagrant to bring up our guest-build box.

```
# Clone the project.
git clone --depth 1 https://github.com/armbian/build

# Make the Vagrant box available. This might take a while but only needs to be done once.
vagrant box add ubuntu/focal64

# If the box gets updated by the folks at HashiCorp, we'll want to update our copy too.
# This only needs done once and a while.
vagrant box update
```

Armbian Directory Structure

Before we bring up the box, take note of the directory structure used by the Armbian build tool. When you read the Vagrantfile (which is in the build/config/templates directory) you'll see that Vagrant will mount local *output* and *userpatches* directories. This is helpful as it enables you to easily retrieve your images from the host once built, and customize the build process.

Creating the Vagrant Guest Box Used to Build

Let's bring the box up. This might take a minute or two depending on your bandwidth and hardware.

```
# We have to be in the same directory as the Vagrant file, which is in the build/config/templates directory.

cd build/config/templates

#  Note that you can edit the Vagrant  file to specify the number of cpus and amount of memory you want Vagrant to use.

# And now we simply let vagrant create our box and bring it up.

vagrant up

# When the box has been installed we can get access via ssh.
# (No need for passwords, Vagrant installs the keys we'll need.)
vagrant ssh
```

Shut down, clean up

Wrap up your vagrant box when no longer needed (log out of the guest before running these commands on the *host* system):

```
# Shutdown, but leave the box around for more building at a later time:
vagrant halt

# Trash the box and remove all the related storage devices.
vagrant destroy
```


© 2021 by Armbian

5.2.2 Important note

It is strongly recommended to halt and restart the Vagrant box after building an image. Check this issue for details.

5.2.3 Vagrant GUEST Steps

The following steps are all run on the *guest* Vagrant created for us.

Once it's finally up and you're logged in, it works much like any of the other install methods (NOTE: again, these commands are run on the *guest* box).

```
# Let's get building!
cd armbian
sudo ./compile.sh
```

5.2.4 More Vagrant HOST Steps

Moved here.

© 2021 by Armbian

5.3 Building with Docker

5.3.1 Officially supported and tested method for building with Docker

This method works for building u-boot and kernel packages as well as building full OS images. Note! To write fresh-builded image directly to sdcard or other block device you have to enable Docker run in `privileged` mode. Uncomment line `DOCKER_FLAGS+=(--privileged)` in file `userpatches\config-docker.conf` or your own docker-config file.

Building additional packages (`EXTERNAL_NEW`) is not supported.

Requirements

- x86/x64/aarch64/armhf Linux host that supports running a recent Docker daemon. Refer to Docker documentation for details.
- Docker version 17.06 CE or newer.
- Enough free disk space on the storage used for Docker containers and named volumes. Named volumes path can be changed using standard Docker utilites, refer to Docker documentation for details.

Installation (https://docs.docker.com/engine/install/)

Details

There are 3 options to start build process:

1. By passing configuration file name (`config-<conf_name>.conf`), stored in `userpatches` directory, as an argument:

```
./compile.sh docker <conf_name>
```

2. By passing addtional line arguments to `compile.sh` after `docker` :

```
./compile.sh docker KERNEL_ONLY=yes BOARD=cubietruck BRANCH=current KERNEL_CONFIGURE=yes
```

3. Interactively run inside docker container

```
./compile.sh docker-shell BOARD=rockpi-4a BRANCH=edge RELEASE=focal
```

The process creates and runs a named Docker container `armbian` with two named volumes `armbian-cache` and `armbian-ccache` , and mounts local directories `output` and `userpatches` .

Options 1 and 2 compile the same as without Docker but in separate environment to prevent changes to the base system.

The dockerfile of the created container is placed in `userpatches` directory, and all container-related options can be changed in `userpatches/config-docker.conf` file. Templates of both files are located in the `config/templates` directory.

docker-shell interactive mode

The docker-shell interactive mode is useful for when you need to do more than just “make an image.” This mode allows you to edit U-Boot and kernel sources before and after applying patches, investigate compilation errors, and so on.

This mode also allows you to manually run individual steps of the build process.

First, start docker-shell on the host build system:

```
@droid:~/armbian$ ./compile.sh docker-shell RELEASE=buster BOARD=rockpi-4a BRANCH=edge
```

From there, `RELEASE=buster BOARD=rockpi-4a BRANCH=edge` are passed into shell and will be set into environment variables.

© 2021 by Armbian

Next, we can simply start building an image:

```
root@75ec76203b65:~/armbian# ./compile.sh
```

Alternatively, you can run any function defined in the compile.sh script.

For example, to compile U-Boot, prepare the environment with:

```
./compile.sh default prepare_host compile_sunxi_tools install_rkbin_tools
```

Then, build U-Boot:

```
./compile.sh default compile_uboot
```

To compile only the source code as it is without patching or modifications, run:

```
./compile.sh default COMPILE_ONLY=yes compile_uboot
```

Note that you must enter docker-shell after a docker build, as you must download all of the required toolchains and sourcecodes beforehand.

© 2021 by Armbian

5.4 Build options

These parameters are meant to be applied to the `./compile.sh` command. They are **all** optional. They can also be added to your build configuration file to save time. Default values are marked **bold** if applicable.

5.4.1 Main options

- **KERNEL_ONLY** (yes | no):
- yes: compiles only kernel, U-Boot and other packages for installation on existing Armbian system
- no: build complete OS image for writing to SD card
- leave empty to display selection dialog each time
- **KERNEL_CONFIGURE** (yes | no):
- yes: Automatically call kernel's `make menuconfig` (add or remove modules or features)
- no: Use provided kernel configuration provided by Armbian
- leave empty to display selection dialog each time
- **CLEAN_LEVEL** (comma-separated list): defines what should be cleaned. Default value is `"make,debs"` - clean sources and remove all packages. Changing this option can be useful when rebuilding images or building more than one image
- make: execute `make clean` for selected kernel and U-Boot sources
- images: delete `output/images` (complete OS images)
- debs: delete packages in `output/debs` for current branch and device family
- alldebs: delete all packages in `output/debs`
- cache: delete `cache/rootfs` (rootfs cache)
- oldcache: remove old `cache/rootfs` except for the newest eight files
- sources: delete `cache/sources` (all downloaded sources)
- extras: delete additional packages for current release in `output/debs/extra`
- **REPOSITORY_INSTALL** (comma-separated list): list of core packages which will be installed from repository
- Available options: `u-boot`, `kernel`, `bsp`, `armbian-bsp-cli`, `armbian-bsp-desktop`, `armbian-desktop`, `armbian-config`, `armbian-firmware`
- Set to "" to use packages one from local output or build if not available
- **KERNEL_KEEP_CONFIG** (yes | no):
- yes: use kernel config file from previous compilation for the same branch, device family and version
- no: use default or user-provided config file
- **BUILD_MINIMAL** (yes | no):
- yes: build bare CLI image suitable for application deployment. This option is **not compatible** with `BUILD_DESKTOP="yes"` and `BUILD_EXTERNAL="yes"`
- no: default CLI package selection
- **BUILD_DESKTOP** (yes | no):
- yes: build image with minimal desktop environment
- no: build image with console interface only
- **EXTERNAL** (yes | no):
- yes: compile and install extra applications and firmware
- **BSPFREEZE** (yes | no):
- yes: freeze (from update) armbian packages when building images (U-Boot, kernel, DTB)
- **INSTALL_HEADERS** (**no** | yes):
- yes: install kernel headers

© 2021 by Armbian

- **EXTERNAL_NEW** (no | prebuilt | compile):
- prebuilt: install extra applications from repository
- compile: compile extra applications in chroot
- **CREATE_PATCHES** (yes | **no**):
- yes: prompt right before the compilation starts to make changes to the source code for both U-Boot and kernel. From these changes patch files will be created and placed in `output` directory. If you want these patches included in a normal run (without CREATE_PATCHES to say) these files need to be copied to the appropriate directories. Also see user provided patches.
- **BUILD_ALL** (yes | no | demo): cycle through all available board and kernel configurations and make images for all combinations
- **LIB_TAG** (master | "branchname"):
- set to `master` to compile from the master branch (default)
- set to another "branchname" to compile from any other branch available. Check here for available branches
- **CARD_DEVICE** (/dev/sdX): set to the device of your SD card. The image will be burned and verified using Etcher for CLI.
- **CRYPTROOT_ENABLE** (yes | no): enable LUKS encrypted rootfs
- `CRYPTROOT_PASSPHRASE="MYSECRECTPASS"` mandatory
- `CRYPTROOT_SSH_UNLOCK=yes` Default: `yes`
- `CRYPTROOT_SSH_UNLOCK_PORT=2222` Default: `2022`
- `CRYPTROOT_PARAMETERS="custom cryptsetup options"` Default: `--pbkdf pbkdf2` (May not contain `=`; separate with spaces)
- **Note:** This function might not work well with all distributions. Debian Buster and Stretch were tested. For building under Docker you have to use privileged mode which can be enable in `userpatches/config-docker`.
- **Warning:** This feature was added as community contribution and mostly functional. Under some circumstances though the prompt will not be shown. Therefore it should be considered experimental. Check here and here
- **Hint:** If you want to do the encryption part from scratch check out this forum post.

5.4.2 Hidden options to minimize user input for build automation

- **BOARD** (`string`): set name of board manually to skip dialog prompt
- **BRANCH** (`legacy` | `current` | `edge`): set kernel and U-Boot branch manually to skip dialog prompt; some options may not be available for all devices
- **RELEASE** (`stretch` | `buster` | `bullseye` | `bionic` | `focal` | `hirsute`): set OS release manually to skip dialog prompt; use this option with `KERNEL_ONLY=yes` to create board support package
- **ARMBIAN_CACHE_ROOTFS_PATH** (`string`): bind mount cache/rootfs to defined folder
- **ARMBIAN_CACHE_TOOLCHAIN_PATH** (`string`): bind mount cache/toolchain path to defined folder

5.4.3 Hidden options for advanced users (default values are marked **bold**)

- **EXPERT** (yes | **no**): Show development features and boards regardless of its status in interactive mode
- **USERPATCHES_PATH** (**userpatches/**): set alternate path for location of `userpatches` folder
- **USE_CCACHE** (**yes** | no): use a C compiler cache to speed up the build process
- **PRIVATE_CCACHE** (yes | **no**) use `$DEST/ccache` as ccache home directory
- **PROGRESS_DISPLAY** (none | **plain** | dialog): way to display output of verbose processes - compilation, packaging, debootstrap
- **PROGRESS_LOG_TO_FILE** (yes | **no**): duplicate output, affected by previous option, to log files `output/debug/*.log`
- **NO_APT_CACHER** (**yes** | no): disable usage of APT cache. Default `yes` in containers, but can be overridden
- **NO_HOST_RELEASE_CHECK** (yes | **no**): overrides the check for a supported host system

© 2021 by Armbian

- **USE_MAINLINE_GOOGLE_MIRROR** (yes | **no**): use `googlesource.com` mirror for downloading mainline kernel sources, may be faster than `git.kernel.org` depending on your location
- **USE_GITHUB_UBOOT_MIRROR** (yes | **no**): use unofficial Github mirror for downloading mainline U-Boot sources, may be faster than `git.denx.de` depending on your location
- **SYNC_CLOCK** (**yes** | no): sync system clock on builder before start image creation process
- **OFFLINE_WORK** (yes | **no**): skip downloading and updating sources as well as time and host check. Set to "yes" and you can collect packages without accessing the internet
- **FORCE_USE_RAMDISK** (yes | no): overrides autodetect for using tmpfs in new debootstrap and image creation process
- **FIXED_IMAGE_SIZE** (`integer`): create image file of this size (in megabytes) instead of minimal
- **BOOTSIZE** (`integer` **96**): set size (in megabytes) for separate /boot filesystem. Used if **ROOTFS_TYPE** set to non-ext4
- **COMPRESS_OUTPUTIMAGE** (comma-separated list): create compressed archive with image file and GPG signature for redistribution
- sha: generate SHA256 hash for image
- gpg: sign image using gpg
- 7z: compress image, hash and signature to 7z archive
- gz: compress image only using gz format
- yes: compatibility shorcut for `sha,gpg,7z`
- **SEVENZIP** (yes | **no**): create .7z archive with extreme compression ratio instead of .zip
- **BUILD_KSRC** (**yes** | no): create kernel source packages while building...
- **INSTALL_KSRC** (yes | **no**): ... and pre-install these kernel sources on the image
- **ROOTFS_TYPE** (**ext4** | f2fs | btrfs | xfs | nfs | fel): create image with different root filesystems instead of default `ext4`. Requires setting `FIXED_IMAGE_SIZE` to something smaller than the size of your SD card for `F2FS`
- **BTRFS_COMPRESSION** (lzo | none | **zlib** | zstd): when choosing `ROOTFS_TYPE=btrfs` select `btrfs` filesystem compression method and compression level. By default the compression is `zlib`.
  When selecting `zstd` or setting zlib compression level(`zlib:[1-9]`) user must ensure kernel version is **>=4.14.x**.
  When selecting zstd compression level (`zstd:[1-15]`) both the host and the target kernel version must be **>=5.1.x** since kernel started supporting zstd compression ratio only from 5.1 on.
  Note: The script does not check the legality of input variable (compression ratio). Input like `zlib:1234` is legal to the script but illegal to the kernel. Beware that setting this option does affect image creation only (shrinking disk size) and will not adjust `/etc/fstab` so it is up to the user to later edit `/etc/fstab` if compression in daily operation is also wanted (beware of serious performance penalties with random IO patterns and heavy compression algorithms!).
- **FORCE_BOOTSCRIPT_UPDATE** (yes | no):
- yes: force bootscript to get updated during bsp package upgrade
- **NAMESERVER** (`IPv4 address`): the DNS resolver used inside the build chroot. Does not affect the final image. Default: `1.0.0.1`
- **DOWNLOAD_MIRROR** (`china` | `bfsu`): select download mirror for `toolchain` and `debian/ubuntu packages`
- `china`: use `mirrors.tuna.tsinghua.edu.cn`, it will be very fast thanks to Tsinghua University
- `bfsu`: use `mirrors.bfsu.edu.cn`, mirror of Beijing Foreign Studies University
- leave empty to use official source
- **ARMBIAN_MIRROR** (auto): override automated mirror selection, example 'ARMBIAN_MIRROR="https://yourlocalmirror.com"'
- **MAINLINE_MIRROR** (`google` | `tuna` | `bfsu`): select mainline mirror of `linux-stable.git`
- `google`: use mirror provided by Google, the same as `USE_MAINLINE_GOOGLE_MIRROR=yes`
- `tuna`: use mirror provided by Tsinghua University
- `bfsu`: use mirror provided by Beijing Foreign Studies University which is similar to `tuna`
- leave empty to use offical `git.kernel.org`, may be very slow for mainland China users

© 2021 by Armbian

- **UBOOT_MIRROR** `github` | `gitee` : select mainline mirror of `u-boot.git`
- `github` : use mirror provided by github, the same as `USE_GITHUB_UBOOT_MIRROR=yes`
- `gitee` : use mirror provided by Gitee, a Chinese git services
- leave empty to use offical `source.denx.de`, may be very slow for mainland China users
- **GITHUB_MIRROR** (`fastgit` | `gitclone` | `cnpmjs`): select download mirror for GitHub hosted repository
- `fastgit` : use mirror provided by fastgit.org
- `gitclone` : use mirror provided by gitclone.com
- `cnpmjs` : use mirror provided by cnpmjs.org
- leave empty to connect directly to GitHub, may be very slow for mainland China users
- **REGIONAL_MIRROR** (`china`): select mirrors based on regional setting, will not overwrite explicitly specified mirror option
- `china` : MAINLINE_MIRROR= `tuna` , UBOOT_MIRROR= `gitee` , GITHUB_MIRROR= `fastgit` , DOWNLOAD_MIRROR= `china`
- leave empty to use default settings
- **USE_TORRENT** (yes | **no**): use torrent to download toolchains and rootfs
- **ROOT_FS_CREATE_ONLY** (`FORCE`): set to skip rootfs download and create locally
- **EXTRAWIFI** (**yes** | no): include several drivers for WiFi adapters
- **WIREGUARD** (**yes** | no): include Wireguard for kernels before it got upstreamed to mainline. Will lose functionality soon.
- **AUFS** (**yes** | no): Include support for AUFS
- **SKIP_BOOTSPLASH** (yes | **no**): Use kernel bootsplash. Disable in case of troubles

© 2021 by Armbian

5.5 User Configuration

5.5.1 User provided patches

You can add your own patches outside build script. Place your patches inside appropriate directory, for kernel or u-boot. There are no limitations except all patches must have file name extension `.patch`. User patches directory structure mirrors directory structure of `patch`. Look for the hint at the beginning of patching process to select proper directory for patches. Example:

```
[ o.k. ] Started patching process for [ kernel sunxi-edge 4.4.0-rc6 ]
[ o.k. ] Looking for user patches in [ userpatches/kernel/sunxi-edge ]
```

Patch with same file name in `userpatches` directory tree substitutes one in `patch`. To *replace* a patch provided by Armbian maintainers, copy it from `patch` to corresponding directory in `userpatches` and edit it to your needs. To *disable* a patch, create empty file in corresponding directory in `userpatches`.

5.5.2 User provided configuration

If file `userpatches/lib.config` exists, it will be called and can override the particular kernel and u-boot versions. It can also add additional packages to be installed, by adding to `PACKAGE_LIST_ADDITIONAL`. For a comprehensive list of available variables, look through `lib/configuration.sh`. Some examples of what you can change:

```
PACKAGE_LIST_ADDITIONAL="$PACKAGE_LIST_ADDITIONAL python-serial python" # additional packages
[[ $LINUXFAMILY == sunxi64 && $BRANCH == edge ]] && BOOTBRANCH='tag:v2017.09' # conditionally change u-boot git branch/tag
KERNELBRANCH="tag:v5.4.28" #always change to this kernel tag
```

5.5.3 User provided kernel config

If file `userpatches/linux-$LINUXFAMILY-$BRANCH.config` exists, it will be used instead of default one from `config`. Look for the hint at the beginning of kernel compilation process to select proper config file name. Example:

```
[ o.k. ] Compiling current kernel [ 5.10.47 ]
[ o.k. ] Using kernel config provided by user [ userpatches/linux-rockchip64-current.config ]
```

5.5.4 User provided sources config overrides

If file `userpatches/sources/$LINUXFAMILY.conf` exists, it will be used in addition to the default one from `config/sources`. Look for the hint at the beginning of compilation process to select proper config file name. Please note that there are some exceptions for LINUXFAMILY like `sunxi` (32-bit mainline sunxi) and `sunxi64` (64-bit mainline sunxi)

Example:

```
[ o.k. ] Adding user provided sunxi64 overrides
```

5.5.5 User provided image customization script

You can run additional commands to customize created image. Edit file:

```
userpatches/customize-image.sh
```

and place your code here. You may test values of variables noted in the file to use different commands for different configurations. Those commands will be executed in a chroot environment just before closing image.

To add files to image easily, put them in `userpatches/overlay` and access them in `/tmp/overlay` from `customize-image.sh`

Be advised that even though you are compiling an image on an amd64 machine, any additional apt packages you configure or commands you run in customize-image.sh will be automatically installed/executed/virtualized for the architecture of the build target SBC.

© 2021 by Armbian

5.5.6 Partitioning of the SD card

In case you define `$FIXED_IMAGE_SIZE` at build time the partition containing the rootfs will be made of this size. Default behaviour when this is not defined is to shrink the partition to minimum size at build time and expand it to the card's maximum capacity at boot time (leaving an unpartitioned spare area of ~5% when the size is 4GB or less to help the SD card's controller with wear leveling and garbage collection on old/slow cards).

You can prevent the partition expansion from within `customize-image.sh` by a `touch /root/.no_rootfs_resize` or configure the resize operation by either a percentage or a sector count using `/root/.rootfs_resize` (`50%` will use only half of the card's size if the image size doesn't exceed this or `3887103s` for example will use sector 3887103 as partition end. Values without either `%` or `s` will be ignored)

© 2021 by Armbian

5.6 FEL/NFS boot explanation

Warning: Armbian has dropped support for old legacy 3.x kernels. Therefore FEL is no longer supported/available. This page is archived and not maintained.

5.6.1 What is FEL/NFS boot?

FEL/NFS boot mode is a possibility to test freshly created Armbian distribution without using SD card. It is implemented by loading u-boot, kernel, initrd, boot script and .bin/.dtb file via USB FEL mode and providing root filesystem via NFS share.

NOTE: this mode is designed only for testing. To use root on NFS permanently, use `ROOTFS_TYPE=nfs` option. NOTE: “hot” switching between kernel branches (default <-> dev/next) is not supported

5.6.2 Requirements

- Allwinner device that supports FEL mode. Check wiki to find out how to enter FEL mode with your device
- USB connection between build host and board OTG port (VM USB passthrough or USB over IP may work too)
- Network connection between build host and board. For target board **wired** Ethernet connection is required (either via onboard Ethernet or via USB ethernet adapter that has required kernel modules built-in)
- NFS ports on build host should be reachable from board perspective (you may need to open ports in firewall or change network configuration of your VM)
- Selected kernel should have built-in support for DHCP and NFS root filesystem
- `CLEAN_LEVEL="make,debs"` to always update u-boot configuration

Additional requirements (recommended)

- DHCP server in local network
- UART console connected to target board

5.6.3 Build script options

- KERNEL_ONLY=no
- ROOTFS_TYPE=fel

Example:

```
./compile.sh KERNEL_ONLY=no BOARD=cubietruck BRANCH=current PROGRESS_DISPLAY=plain RELEASE=jessie BUILD_DESKTOP=no ROOTFS_TYPE=fel
```

5.6.4 Shutdown and reboot

Once you start FEL boot, you will see this prompt:

```
[ o.k. ] Press any key to boot again, <q> to finish [ FEL ]
```

Pressing `q` deletes current rootfs and finishes build process, so you need to shut down or reboot your board to avoid possible problems unmounting/deleting temporary rootfs. All changes to root filesystem will persist until you exit FEL mode.

To reboot again into testing system, switch your board into FEL mode and press any key other than `q`.

Because kernel and .bin/.dtb file are loaded from rootfs each time, it’s possible to update kernel or its configuration (via `apt-get`, `dtc`, `fex2bin` / `bin2fex`) from within running system.

© 2021 by Armbian

5.6.5 Advanced configuration

If you don't have DHCP server in your local network or if you need to alter kernel command line, use `lib/scripts/fel-boot.cmd.template` as a template and save modified script as `userpatches/fel-boot.cmd`. Check this for configuring static IP for NFS root.

- Note: As of 2020-11-25, the above link is broken. However I am not sure what to replace it with. If you know (or find out) please submit a PR. - TRS-80

Set `FEL_DTB_FILE` to relative path to .dtb or .bin file if it can't be obtained from u-boot config (mainline kernel) or boot/script.bin (legacy kernel)

You may need to set these additional options (it's a good idea to put them in `userpatches/lib.config`:

Set `FEL_NET_IFNAME` to name of your network interface if you have more than one non-loopback interface with assigned IPv4 address on your build host

Set `FEL_LOCAL_IP` to IP address that can be used to reach NFS server on your build host if it can't be obtained from ifconfig (i.e. port forwarding to VM guest)

Set `FEL_AUTO=yes` to skip prompt before trying FEL load

5.6.6 Customization

You can even create `userpatches/fel-hooks.sh` and define there 2 functions: `fel_post_prepare` and `fel_pre_load`. All normal build variables like $BOARD, $BRANCH and so on can be used in these functions to define specific actions.

`fel_post_prepare` is executed once after setting up u-boot script and NFS share, you can use it to add extra stuff to boot.scr (like `gpio set` or `setenv machid`) based on device name.

`fel_pre_load` is executed before calling sunxi-fel, you can use it to implement logic to select one of multiple connected boards; to pass additional arguments to `sunxi-fel` you can use `FEL_EXTRA_ARGS` variable.

An example is provided as `scripts/fel-hooks.sh.example`.

© 2021 by Armbian

5.7 Extensions

"I'm gonna create a `prepare_bootloader` hook [in core] so we can refactor `u-boot` [into an extension]"

The extensions framework allows the board/family developers, extension authors, and users to extend the Armbian build system without overloading the core with specific functionality.

It's a simple framework, written in Bash, that **works based on function naming conventions**. It provides the core and the extensions with tracing and debugging, (error control,?) inline documentation and very simple dependency resolution.

Terminology

- The "core" is everything that's in `lib/` directory plus `compile.sh` and some others. It's the spine of the build system.
- An "extension" is a separate Bash source file that contains exclusively functions. Extensions live in `extensions/` or `userpatches/extensions/` directory, but could one day be in a separate repository too.
- An "extension method" (a.k.a "hook") is used by the core to call extensions, via `call_extension_method()`. This will discover all enabled extension methods implementations, order them, and call them one by one.
- The Armbian core already has quite a few of these in strategic spots.
- More are coming as they're identified.
- An "extension method implementation" is a function that will be called when it's extension method is called. It can be defined in extensions, but also in board config, family config, user config, etc.

EXAMPLE

Core calls extensions

The Armbian core build system has an extension method called `run_after_build`, also known as the "`run_after_build` hook". You can find it in `lib/main.sh` around line 546.

```
# in lib/main.sh:546
call_extension_method "run_after_build" [...]
```

Extension method implementation

Consider the following function:

```
function run_after_build__say_congratulations() {
  echo "Congrats, the build is finished!"
}
```

Such a function is an "extension method implementation" called `say_congratulations` for the extension method `run_after_build`.

Extension file

A file `userpatches/extensions/be-festive.sh` containing the above function is an "extension" called `be-festive`.

Using it

An user of the build system can enable that extension by adding a call to `enable_extension "be-festive"` on his configuration file, or by passing `ENABLE_EXTENSIONS=be-festive` as a parameter to the build.

Naming conventions and ordering

An extension method implementation is just a Bash function that follows the pattern `run_after_build__say_congratulations` where

- `run_after_build` is the name of the extension method.
- `__` is a marker/separator – very important – two underscores, not one, not three.
- `say_congratulations` is the name of the extension method implementation, and should be unique.

© 2021 by Armbian

The system will “magically” compose a single `run_after_build()` function, based on all the hook functions that begin with `run_after_build__`.

Hook functions will be sorted by their numerical value; hook functions that do not begin with a number will receive `500_` prefix automatically.

So the examples `run_after_build__do_this` and `run_after_build__500_do_this` are equivalent, and will run

- sooner than `run_after_build_900_do_smth_else`
- later than `run_after_build_300_do_even_another_thing`

What is an *extension*?

A extension is Bash source file that contains **exclusively**:

- function definitions:
- extension method implementation definitions (with `__` separator)
- other internal functions (for structure and clarity if needed)
- calls to `enable_extension "another-extension"` at the top of the file.
- that’s a very simple dependency system, one extension can enable another.

Specifically, **extension files should not contain any code outside of functions** – they should do nothing when sourced.

Extensions can be official Armbian fragments and live in `/extensions`, or can be user-specific in `/userpatches/extensions`.

An extension could be implemented in any of the following file/dir structures:

- `/extensions/our-ext.sh` - an official, single-file extension.
- `/userpatches/extensions/my-ext.sh` - a user-specific, single-file extension.
- `/extensions/our-dir-ext/our-dir-ext.sh` - an official, directory-based extension.
- `/userpatches/extensions/my-dir-ext/my-dir-ext.sh` - a user-specific, directory-based extensions.

The official extensions can be used by boards, family includes, etc, while the user-specific extensions can only be used by userpatches code or via `ENABLE_EXTENSIONS=my-ext,my-dir-ext` build parameter.

SINGLE-FILE VS DIRECTORY-BASED

They’re the same, except:

- Directory-based extensions will be passed a `${EXTENSION_DIR}` environment variable.
- That is useful if there are other files/assets that belong together with that extension. An example would be a template file, some configuration file, or other static asset that is directly related to the extension.
- Using directory-based extensions and `${EXTENSION_DIR}` allows for easy moving and PR’ing of user extensions.

© 2021 by Armbian

5.8 Extension Hooks

- This file is autogenerated by the armbian/build repository.

5.8.1 Hooks

- Hooks are listed in the order they are called.

`post_family_config`

give the config a chance to override the family/arch defaults

This hook is called after the family configuration (`sources/families/xxx.conf`) is sourced. Since the family can override values from the user configuration and the board configuration, it is often used to in turn override those.

Also known as (for backwards compatibility only):

- `config_tweaks_post_family_config`

`user_config`

Invoke function with user override

Allows for overriding configuration values set anywhere else. It is called after sourcing the `lib.config` file if it exists, but before assembling any package lists.

`extension_prepare_config`

allow extensions to prepare their own config, after user config is done

Implementors should preserve variable values pre-set, but can default values an/or validate them. This runs *after* user_config. Don't change anything not coming from other variables or meant to be configured by the user.

`post_aggregate_packages`

For final user override, using a function, after all aggregations are done

Called after aggregating all package lists, before the end of `compilation.sh`. Packages will still be installed after this is called, so it is the last chance to confirm or change any packages.

Also known as (for backwards compatibility only):

- `user_config_post_aggregate_packages`

`post_determine_cthreads`

give config a chance modify CTHREADS programatically. A build server may work better with hyperthreads-1 for example.

Called early, before any compilation work starts.

Also known as (for backwards compatibility only):

- `config_post_determine_cthreads`

`add_host_dependencies`

run before installing host dependencies

you can add packages to install, space separated, to ${EXTRA_BUILD_DEPS} here.

© 2021 by Armbian

`fetch_sources_tools`

fetch host-side sources needed for tools and build

Run early to fetch_from_repo or otherwise obtain sources for needed tools.

`build_host_tools`

build needed tools for the build, host-side

After sources are fetched, build host-side tools needed for the build.

`pre_install_distribution_specific`

give config a chance to act before install_distribution_specific

Called after `create_rootfs_cache` (*prepare basic rootfs: unpack cache or create from scratch*) but before `install_distribution_specific` (*install distribution and board specific applications*).

Also known as (for backwards compatibility only):

- `config_pre_install_distribution_specific`

`pre_install_kernel_debs`

called before installing the Armbian-built kernel deb packages

It is not too late to `unset KERNELSOURCE` here and avoid kernel install.

`post_install_kernel_debs`

allow config to do more with the installed kernel/headers

Called after packages, u-boot, kernel and headers installed in the chroot, but before the BSP is installed. If `KERNELSOURCE` is (still?) unset after this, Armbian-built firmware will not be installed.

`post_family_tweaks`

customize the tweaks made by $LINUXFAMILY-specific family_tweaks

It is run after packages are installed in the rootfs, but before enabling additional services. It allows implementors access to the rootfs (`${SDCARD}`) in its pristine state after packages are installed.

`pre_customize_image`

run before customize-image.sh

This hook is called after `customize-image-host.sh` is called, but before the overlay is mounted. It thus can be used for the same purposes as `customize-image-host.sh` .

Also known as (for backwards compatibility only):

- `image_tweaks_pre_customize`

`post_customize_image`

post customize-image.sh hook

Run after the customize-image.sh script is run, and the overlay is unmounted.

© 2021 by Armbian

Also known as (for backwards compatibility only):

- `image_tweaks_post_customize`

`post_post_debootstrap_tweaks`

run after removing diversions and qemu with chroot unmounted

Last chance to touch the `${SDCARD}` filesystem before it is copied to the final media. It is too late to run any chrooted commands, since the supporting filesystems are already unmounted.

Also known as (for backwards compatibility only):

- `config_post_debootstrap_tweaks`

`pre_prepare_partitions`

allow custom options for mkfs

Good time to change stuff like mkfs opts, types etc.

Also known as (for backwards compatibility only):

- `prepare_partitions_custom`

`prepare_image_size`

allow dynamically determining the size based on the $rootfs_size

Called after `${rootfs_size}` is known, but before `${FIXED_IMAGE_SIZE}` is taken into account. A good spot to determine `FIXED_IMAGE_SIZE` based on `rootfs_size`. UEFISIZE can be set to 0 for no UEFI partition, or to a size in MiB to include one. Last chance to set `USE_HOOK_FOR_PARTITION` =yes and then implement create_partition_table hook_point.

Also known as (for backwards compatibility only):

- `config_prepare_image_size`

`post_create_partitions`

called after all partitions are created, but not yet formatted

`format_partitions`

if you created your own partitions, this would be a good time to format them

The loop device is mounted, so ${LOOP}p1 is it's first partition etc.

`pre_update_initramfs`

allow config to hack into the initramfs create process

Called after rsync has synced both `/root` and `/root` on the target, but before calling `update_initramfs`.

Also known as (for backwards compatibility only):

- `config_pre_update_initramfs`

`pre_umount_final_image`

allow config to hack into the image before the unmount

Called before unmounting both `/root` and `/boot`.

© 2021 by Armbian

Also known as (for backwards compatibility only):

- `config_pre_umount_final_image`

`post_umount_final_image`

allow config to hack into the image after the unmount

Called after unmounting both `/root` and `/boot`.

Also known as (for backwards compatibility only):

- `config_post_umount_final_image`

`post_build_image`

custom post build hook

Called after the final .img file is built, before it is (possibly) written to an SD writer.

- *NOTE*: this hook used to take an argument ($1) for the final image produced.
- Now it is passed as an environment variable `${FINAL_IMAGE_FILE}` It is the last possible chance to modify `$CARD_DEVICE`.

`run_after_build`

hook for function to run after build, i.e. to change owner of `$SRC`

Really one of the last hooks ever called. The build has ended. Congratulations.

- *NOTE:* this will run only if there were no errors during build process.

`extension_metadata_ready`

meta-Meta time!

Implement this hook to work with/on the meta-data made available by the extension manager. Interesting stuff to process:

- `"${EXTENSION_MANAGER_TMP_DIR}/hook_point_calls.txt"` contains a list of all hook points called, in order.
- For each hook_point in the list, more files will have metadata about that hook point.
- `${EXTENSION_MANAGER_TMP_DIR}/hook_point.orig.md` contains the hook documentation at the call site (inline docs), hopefully in Markdown format.
- `${EXTENSION_MANAGER_TMP_DIR}/hook_point.compat` contains the compatibility names for the hooks.
- `${EXTENSION_MANAGER_TMP_DIR}/hook_point.exports` contains *exported* environment variables.
- `${EXTENSION_MANAGER_TMP_DIR}/hook_point.vars` contains *all* environment variables.
- `${defined_hook_point_functions}` is a map of *all* the defined hook point functions and their extension information.
- `${hook_point_function_trace_sources}` is a map of all the hook point functions *that were really called during the build* and their BASH_SOURCE information.
- `${hook_point_function_trace_lines}` is the same, but BASH_LINENO info. After this hook is done, the `${EXTENSION_MANAGER_TMP_DIR}` will be removed.

© 2021 by Armbian

6. Contributor Process

6.1 Collaborate on the project

6.1.1 How?

1. Fork the project.
2. Make one or more well commented and clean commits to the repository.
3. Perform a pull request in Github's web interface.

   If it is a new feature request, do not start the coding first. Remember to open an issue to discuss the new feature. If you want to add code to someone else pull request. Also check collection of git tips which will make your life easier.

   If you are struggling, check WEB or CLI step-by-step guide on contributing.

6.1.2 Where are the sources?

Build script:

https://github.com/armbian/build

Documentation:

https://github.com/armbian/documentation

Armbian-config tool:

https://github.com/armbian/config

6.1.3 Help with donations

If you find our project useful, then we'd really appreciate it if you'd consider contributing to the project however you can. Donating is the easiest way to help us – you can use PayPal and Bitcoin or you can buy us something from our Amazon.de wish list.

https://www.armbian.com/donate/

Thanks!

© 2021 by Armbian

6.2 Merge Policy

6.2.1 Overview

Note: This document is a Work In Progress and is subject to change. Definitions may be relocated to a seperate document in the future.

This policy is targeted for Maintainers for Lead Maintainers who have commit access to `master` branch. This document describes the tags needed for different merge types. See Definitions.

The types of code maintained fall into the following categories:

- Kernel
- U-Boot
- Build scripts

Kernel and U-Boot maintainers are usually grouped by SoC Architecture.

Supported boards will have the most scrunity with code review.

6.2.2 U-Boot Patches

- Standard contributors provide *tested-by* submitter (`armbianmonitor -u` with a fresh build).
- SoC maintainer maybe submit a PR with only a reviewed by of the lead SoC maintainer.

6.2.3 Kernel Related Patches

legacy and current branches

- DT changes reviewed by at least one person familiar with this SoC (lead maintainer or deputy), *tested-by* the contributor who sends it (armbianmonitor).
- Trivial module activation doesn't matter.

dev/edge branches

Constraints are at the discretion of the SoC mantainer. This builds are not expected to be stable.

6.2.4 Armbian Build Scripts

This pertains to code used to build system images, OS configuration, and supporting packages (basically anything not u-boot or kernel source).

lib scripts

- Requires at least one *Reviewed-by*.

Configuration

BOARD PROMOTION

Boards have different levels of commitment / support. EOL, CSC, WIP, Supported. To promote a board from WIP to Supported an Acked-by from a Lead Maintainer is required.

KERNEL CONFIG

- Changes in legacy & current kernel config should provide at least *tested-by* with `armbianmonitor -u`.

© 2021 by Armbian

- Changes in edge are at the discretion of maintainer. No constraints.

KERNEL SOURCES

Change kernel source repos, branches, versions can be very disruptive to stable builds. Sufficient communication should occur stable changes.

- U-Boot and kernel version bump for legacy and current requires *tested-by* from Maintainers and/or testers on at least two different boards for the impacted platform.
- Kernel sources switch (legacy current) needs a discussion on forum or github or IRC and documented in PR and *Acked-by* Lead Maintainer. These changes are risky and things can go terrible wrong here…
- edge source changes are at the discretion of the Lead Maintainer.
- Board family tweaks require at least *reviewed-by*.

PACKAGES

- minimum require *Acked-by*

6.2.5 Definitions

Code Review Terms

click here for attribution to terms used below

SIGNED-OFF-BY

Certifies that you wrote it or otherwise have the right to pass it on as a open-source patch.

ACKED-BY

If a person was not directly involved in the preparation or handling of a patch but wishes to signify and record their approval of it then they can arrange to have an Acked-by: line. Acked-by: does not necessarily indicate acknowledgement of the entire patch.

TESTED-BY

A Tested-by: tag indicates that the patch has been successfully tested (in some environment) by the person named. This tag informs maintainers that some testing has been performed, provides a means to locate testers for future patches, and ensures credit for the testers.

REVIEWED-BY

A Reviewed-by tag is a statement of opinion that the patch is an appropriate modification of the kernel without any remaining serious technical issues. Any interested reviewer (who has done the work) can offer a Reviewed-by tag for a patch.

Other

MAINTAINER

An Individual designated to moderate, support and make decisions for a codebase or component. There can be multiple maintainers assigned to any given thing.

LEAD MAINTAINER

A more experienced maintainer that will decide on high-impact and stategic changes and have final say in disputes. A lead maintainer may share or designiate responsibility to some or all components within their domain of responsibility.

SOC

System on-a-Chip.

© 2021 by Armbian

6.3 Automatic rebase of Pull requests

Pull most recent code from master branch and put your work on top within your pull request.

How to use it? Make a comment

/rebase

to trigger the action

- Advantages of Git Rebase,
- Automatic Rebase Action origin.

6.4 Merge request pipelines

On each merge reqest we are running:

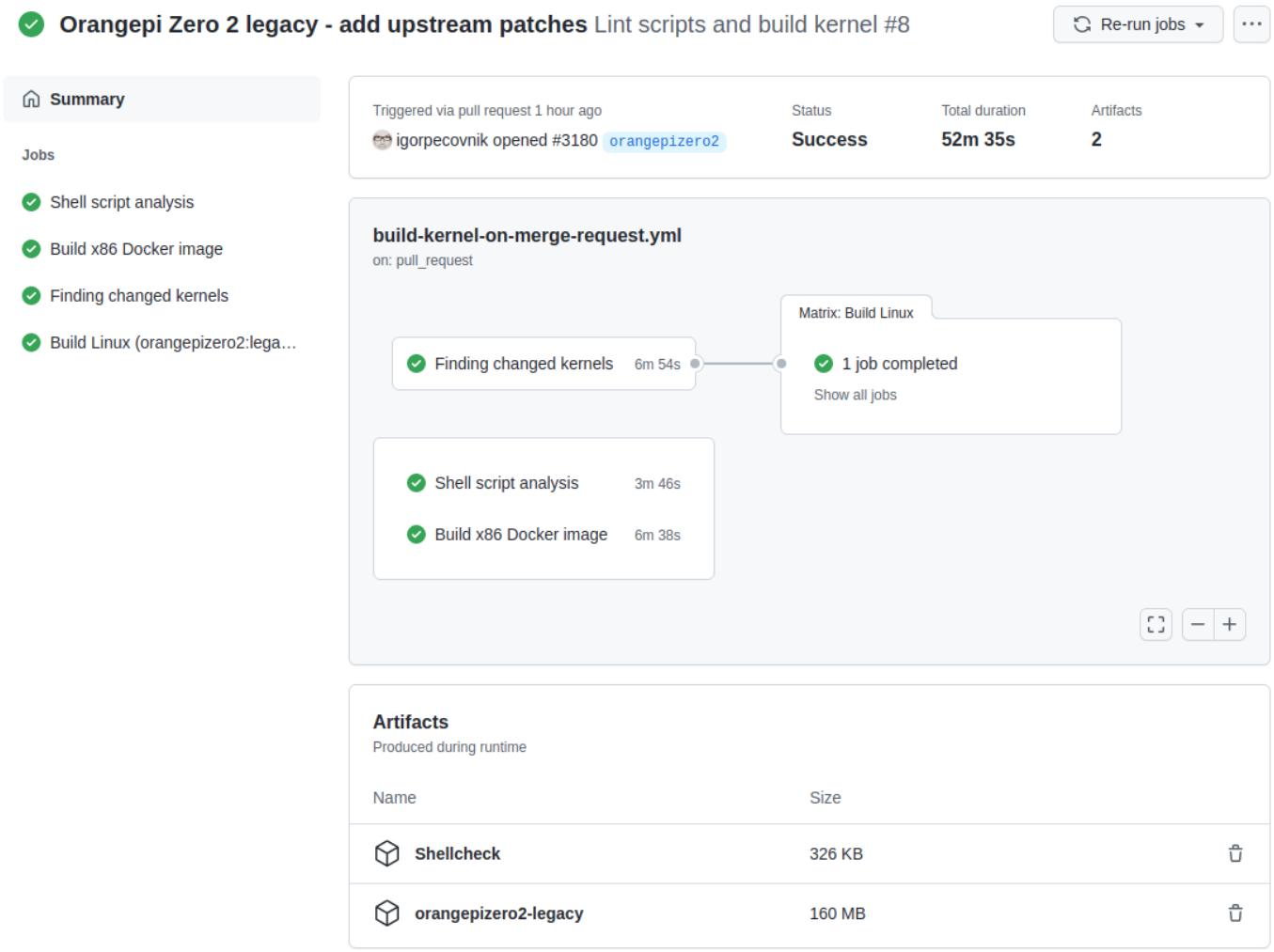
- shell script analysis
- creating Docker image
- creating desktop variants
- if label is set to “desktop”
- when PR is switched between “draft” and “ready for review”
- when PR is approved
- building changed kernels

Those runs are for security reasons executed on public Github runners servers which are very limited. One build cycle takes around one hour and it produces two types of artefacts:

- script anylysis report
- debian packages for kernel, device treee, headers, sources

Those build artefacts are available up to 14 days.

© 2021 by Armbian



6.4.1 Build beta kernel packages

Pipeline is extended version of merge requests pipeline. Pipeline is scheduled to run every day at 6am CET. It builds all changed kernels and update package repository in case it succeeds.

What is affected by this pipeline?

- edge branch in stable repository https://apt.armbian.com
- all branches in beta repository https://beta.armbian.com

Trigger: every day at 6am CET
Condition: change in packages, upstream sources, patches or configuration

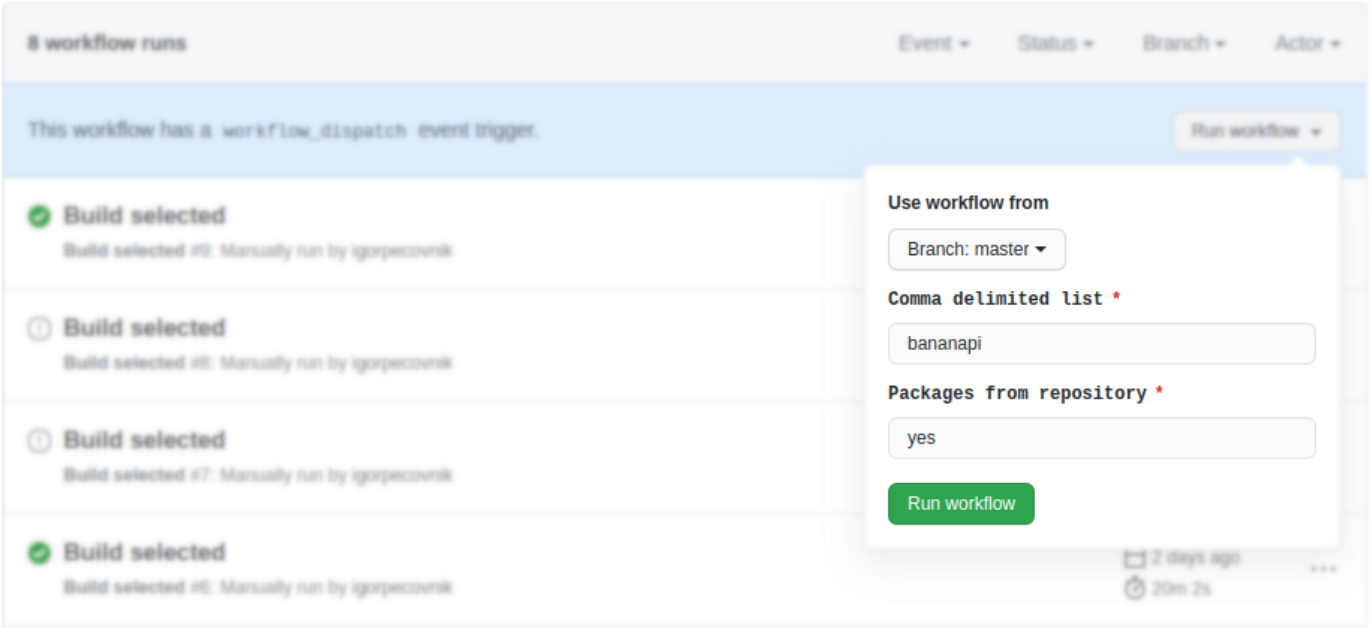
6.4.2 Build all beta images

- triggered manually or uppon completion of nightly / edge builds;
- running the job manual is possible,
- pipeline is always using packages from https://beta.armbian.com repository.

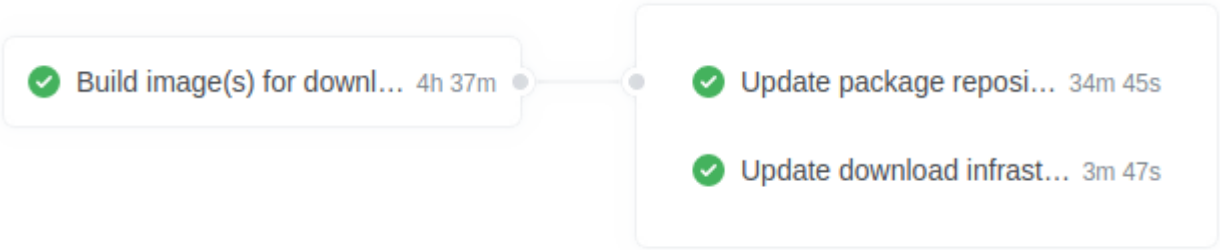
6.4.3 Build selected stable images

If you have a commit rights to the repository, go to Armbian build system actions and select *Build selected*:

© 2021 by Armbian



You can recreate image(s) at main download location from sources - set `packages from repository` to *no* - or from packages that are already in repository (default). In case you choose to build from sources, stable https://apt.armbian.com repository is going to be populated with newly created u-boot, kernel and **BSP packages for all boards** under (patched) stable version (yy.mm.**x+1**) which is incremented automatically if process succeeds.



When new artifacts are created for stable builds, content is uploaded to CDN, then download and repository indexes are updated. The process is typically complete in 1 to 2 days for major releases.

© 2021 by Armbian

6.5 Armbian Documentation

logo

6.5.1 Overview

Documentation is written in markdown and stored in the `docs/` subfolder. Images go in `docs/images`.

This repo is meant for storing and quick glances. Official output is https://docs.armbian.com.

Armbian Documentation is available in the following formats:

- Official document website,
- PDF document

6.5.2 Contributing

This site is built with mkdocs and depends on mkdocs-material.

Armbian Documentation naming of document files follows this rules:

`[Parent-Topic-Example]_[Child-Topic]-example.md`

`Parent-Topic-Name` and `Child-Topic-Name` are separated by an underscore `_`. Hyphens `-` are automatically converted to space.

Please try to avoid creating new parent topics unless absolutely necessary.

Current Parent Topics:

- User Guide
- Hardware notes
- Developer Guide
- Contributor Process
- Release management
- Community

See the document template before you writing any content.

6.5.3 Working on the content

```
git clone https://github.com/armbian/documentation
pip install -r requirements.txt
mkdocs build --clean && mkdocs serve
```

You will be able to make edits to existing files and observe the results in real time.

After changing text in an existing file, use this command to rebuild and view the documentation:

`mkdocs build --clean && mkdocs serve`

After adding a new file, either hand-edit `mkdocs.yml`, or re-run `tools/mkArmbianDocs.py` **unless making changes to the structure of the `docs/` folder**. (See below)

© 2021 by Armbian

6.5.4 Generate tools

mkArmbianDocs.py

Generate `mkdocs.yml` based on the contents of `docs/` folder

- Command-line options for input and output directories
- Requires install requirement
- You don't need to run it every time unless making changes to the structure of the `docs/` folder
- See `mkArmbianDocs.py -h` for help

From the parent folder of the repo, run:

```
python3 tools/mkArmbianDocs.py && mkdocs build
```

This will generate the `mkdocs.yml` and publish built HTML to the `site/` folder.

© 2021 by Armbian

7. Release management

7.1 Release model

7.1.1 Release Dates

Armbian runs “train” based releases. Whatever is ready to board the train, does so. Whatever isn’t ready, has to wait for the next train. This enables us to have a predictable release cycles making it easy to plan. It also puts the responsibility on developers to make sure they have features ready on time.

Armbian releases quarterly at the end of **February, May, August, November**. Offset is because we all know that nothing happens for half of December. At the beginning of a release cycle, we have a planning meeting and **two weeks before the end of the release we freeze integration of new features**.

7.1.2 Release Cycle

Releases last 3 months. Each release starts with a planning meeting. After planning, developers and development teams build their deliverable using whatever methods (scrum, kanban, waterfall, …) they want but shall commit their code frequently, leading **up to the last 2 weeks**. The project does not accept “dumps” of code at the end. **Commit early and often** on master. Two weeks before the release date, we halt feature integration and only allow bug fixes. At some point during those two weeks, we start the release candidate process. This process starts by pulling a branch off master that will become the release branch. That frees up master for development on the next release. On the release candidate branch we work on bug fixes, and choose “release candidate”, RC, tags. The software at that tag is a candidate for release, and it is submitted to automated and manual tests on real hardware. If automated tests are passing, we can officially tag it as the release. If it doesn’t, we enter another bug fix cycle and create a new release candidate. We iterate until we have a candidate that can be the formal release. Usually, this takes 2-3 cycles and 1-3 weeks of time.

Development epics, stories and bugs for each release are tracked through Jira.

7.1.3 Release Branching, Versioning and Tags

Branches in Armbian follow this convention:

- **Master branch (master):** Main development will happen on the master branch. This is the latest and greatest branch, but is always “stable” and “deployable”. All tests always pass on this branch.
- **Maintenance branch (support):** This is the long-term maintenance branch per release.
- **Bleeding edge branch (edge)**: This is a branch created for lengthy and/or involved feature development that could destabilize master.

Each Armbian release will have the following version format:

Format: `<major>.<minor>.<revision>`

`<major>` and `<minor>` version will be incremented at the end of the release cycles while `<revision>` is incremented for a fix (or set of fixes)

Tags are used in ad-hoc manner.

© 2021 by Armbian

7.1.4 Release Naming

version	codename	release month	work
19.11	Vaquita	November	done
20.02	Chiru	February	done
20.05	Kagu	May	done
20.08	Caple	August	done
20.11	Tamandua	November	done
21.02	Urubu	February	done
21.05	Jerboa	May	done
21.08	Caracal	August	done
21.11	Sambar	November	cancelled
22.02	Pig	February	done
22.05	Jade	May	planned

by https://www.codenamegenerator.com from unusual animals

7.1.5 Release Planning

A release planning starts with an public IRC meeting where developers and interested users come together in **first Saturday, one month before the release month**.

Dates for **2020**:

- April 4th
- July 4th
- October 3rd

Dates for **2021**:

- January 2nd
- April 3rd
- July 3rd
- October [16th]

Dates for **2022**:

- January 29th

Agenda:

- 00:00 - 00:02 `#startmeeting Tamandua`: Meeting coordinator (MC) calls meeting to order.
- 00:02 - 00:05 `#topic check-in`: MC gives control to participant to check-in and wait for latecomers. If your handle is not self explanatory, add your forum/github handle and just say hi.
- 00:05 - 00:07 `#topic late topics`: MC points out to agenda and asks if there is any late topic to add.
- 00:07 - 00:09 `#topic FYI`: Stuff to know beforehand - MC presents meeting relevant news and rules of engagement:
- note #1: **IRC translator**: If your English is poor, simply write in your native language. Start your sentence with `--` at the beginning.
- rule #1: When you get a voice, please *be quick and concise* (1-2 min) and make it clear when you stop. (“No more, I’m done”)

© 2021 by Armbian

- rule #2: If meeting is going out of desired agenda, MC will use *"STOP STOP STOP"*, wait to get attention and then proceed with the meeting agenda. **Please stop chatting and listen.**
- rule #3: Please **highlight** important information appropriately by putting a keyword in front of your message: `#info` `#action` `#idea` `#help` check tips below.
- 00:09 - 00:30 **Board maintainers/development**: MC is calling out on by team sections as defined at GitHub
- *Note:* tasks should be tracked by Jira. If they are not there, please add them during or right after the meeting
- *Note:* board maintainers present tasks, bugs or projects they are working on (open discussion for each section if possible, otherwise MC calls people out).
- `#topic development Allwinner`
- `#topic development Amlogic`
- `#topic development Marvell`
- `#topic development Rockchip`
- `#topic development others`
- 00:30 - 00:40 **build system**
- `#topic buildsystem enhancements` : build script enhancements
- `#topic buildsystem desktop` : desktop and multimedia
- 00:40 - 00:45 **Infrastructure**
- `#topic Infrastructure - Servers / CI`
- `#topic Infrastructure - planned/unplanned changes on forums`
- `#topic Infrastructure - IRC: new channel and pushes to commits`
- 00:45 - 00:50 `#topic Jira Backlog` : **Cycle Jira backlog**:
- discuss task / bug (one at a time)
- assign to person / release / tag
- re-prioritise
- 00:50 - 00:55 `#topic open issues`
- Check and cycle open issues and pull requests on a build engine
- 00:55 - 00:56 `#topic board support status updates`
- Discuss (last 10-15) board status update on download pages and build engine (wip, supported, eol) https://www.armbian.com/download/
- 00:56 - 00:57 `#topic release officer and meeting organizer and goverance`
- Choose upcoming release officer and next meeting organizer (1 or 2 roles). We need someone that is not well acquiented with the process to see if our documentation is good enough. He will get full support / backup, so no need to worry about anything.
- Check if https://docs.armbian.com/Community_Governance/ is up to date
- 00:57 - 1:00 `#topic questions about Jira usage`
- Answer questions relating to **how to use Jira** (if any)
- 01:00 - `#topic misc / open discussion`

© 2021 by Armbian

Tips:

- channel is recorded so a summary and adjustments to Jira can made afterwards, ideally along with the meeting
- A *meetbot* will create a meeting summary at the end of the meeting. This however **heavily relys on user input**. So if you have important information to share please put a appropriate keyword in front of your message:

```
#info - Add an info item to the minutes. People should liberally use this for important things they say, so that they can be logged in the minutes.
#action - Document an action item in the minutes. Include any nicknames in the line, and the item will be assigned to them. (nicknames are case-sensitive)
#idea - Add an idea to the minutes.
#help - Add a "Call for Help" to the minutes. Use this command when you need to recruit someone to do a task.
```

Meeting location is IRC channel #armbian on freenode. Meeting starts **at 2pm GMT**.

7.1.6 Release Coordinating

Summary

A release starts as a RC Branch cut from master at freeze time. Once a RC Branch is cut, master can be unfrozen and development can continue. RC Branch is a rolling release that accepts bug fixes. The bug fixes should be cherry-picked back to master branch. Once the RC Is stable, A Final release as a branch named after its version. A release is *never* merged to master. Once a release is complete, it only should be updated for severe bugs and severe security vulnerabilities. A release is only maintained until the next release.

1. Forum Communication

- Create a new thread in the Armbian Build Framework Subforum
- Ex topic name: `Ambian 20.02 (Chiru) Release Thread`
- Tag the post with relase, release version, and codename
- Use the following template to begin the body of the release thread:

```
Release Candidate Code Freeze Date: YYYY-MM-DD
Release Date: YYYY-MM-DD
Release Candidate Branch Link: URL
Release Changelog: URL
Release Coordinator: @yourname
Testing Tracking Sheet: https://example.com/link  (google sheets)

The goal of this thread is to discuss testing, bugfixes, and the overall quality of the release.  Once the release is complete, this thread should be locked and unpinned.
```

- Before Code Freeze – Make note in the thread the incomplete jira issues tagged for the release example - After test images are procuded, engage in community for assistants wih testing.. forums, twitter, etc. share this tool

2. Release Candidate Branch Management

- For code freeze – create a RC branch as `version-rc` ex: `v20.02.0-rc`
- If Possible, create Jira tickets for major changes in github that were not tracked in Jira
- Begin Testing Process. See Release Testing
- Do not modify branch directy. Only accept PRs
- Only accept PRs for Bugfixes. No features
- Update master branch version to the NEXT release version with `-trunk` ex. If RC is v20.02.0-rc Master bacomes v20.05.0-trunk
- CI Testing should pass on PR
- Test images should automatically be built via Igor’s script
- Repeat build, test, and bugfix process until release is stable
- Cherry-pick bugfixes back into master

© 2021 by Armbian

- Create Final Release branch from RC

3. Release

- In Github create a Release from final release branch
- Copy Release notes generated by Jira Release into Github form
- Add other appropriate information into release github release notes
- Point Armbian build system to new release
- Update armbian documentation to reflect current release
- Celebrate

7.1.7 Release Testing

See Opportunties for improvement

7.1.8 Reflection on Prior Releases

Opportunities for Improvement

WIRELESS DRIVER TESTING

- wireless is a particularly sensitive issue. We need to test, fix, or at least be able to inform others of what is broken

BUG TRACKING

TESTING

IMAGE DOWNLOADS

Positive Observations

- Good response from community for testing assistance
- Release was on time

© 2021 by Armbian

7.2 Changelog

- Note: If a new sub-version is released this does not necessarily mean all boards receive a new version number since most of the time these fixes are targeting a specific board or board family only.

7.2.1 v22.02 (2022-02-28)

Solved Bugs

- AR-1101 - DRM patch is failing on Rockchip
- AR-1079 - Ubuntu archive redirector is not providing best service
- AR-1077 - Fix RPi4 userland audio
- AR-1069 - First login doesn't pick up correct shell
- AR-1065 - Twitter forum registration is not working
- AR-1063 - X86 desktop images are not enabled in CI
- AR-1062 - When selecting repository install u-boot might not be flashed
- AR-1055 - Aptly repository seems to be out of business
- AR-1048 - Rpi kernel image is not updated on upgrade
- AR-1043 - linux-firmware repository change branch from "master" to "main"
- AR-973 - boot building is failing after update to 2021.10
- AR-871 - Debian SID broken wallpaper

Story

- AR-1074 - Switch all CURRENT to 5.15.y and EDGE to 5.16.y
- AR-1009 - Armbian Framework extensions and UEFI support

Closed tasks

- AR-1100 - Support for Orange Pi R1 Plus LTS
- AR-1084 - 3D support on Debian desktop
- AR-1078 - Add additional forum plugins and adjust settings
- AR-1068 - Add gnome-system-monitor to Focal and Jammy
- AR-1049 - Add ZFS that supports kernel 5.15.y
- AR-1044 - Improve Raspberry Pi support
- AR-1041 - JetHome: fix brcm (AP6255) firmware links
- AR-1040 - Refactor armbian-bsp-cli package creation
- AR-931 - Using Docker image for building kernels
- AR-893 - Cleanup rockchip64 u-boot scenarios
- AR-757 - Adding Raspberry Pi
- AR-586 - Implement fan controller for Nanopi M4V2

7.2.2 v21.08 (2021-08-31)

Solved Bugs

- AR-886 - u-boot package naming

© 2021 by Armbian

- AR-885 - Odroid C4 / HC4 boot failure
- AR-881 - First login shows “degraded”
- AR-879 - RK3328 5.10 + GPU failure
- AR-875 - Homepage fix
- AR-874 - Add SATA fix for eBin
- AR-873 - khadas vim3 no sound
- AR-872 - When reverting u-boot to last known, some boards are broken
- AR-867 - H6 Freezing
- AR-863 - Unstable u-boot also noticed on H3 based boards
- AR-862 - normalize RK3399 xorg configs
- AR-857 - Qemu custom hook executing on non-targeted images
- AR-854 - Pinebook PRO desktop doesn’t build in CI
- AR-853 - Missing folder when making BSP file
- AR-852 - Error when placing wallpaper
- AR-850 - Disable XFWM compositor on XFCE4 desktop to make it run smoother
- AR-844 - CI is picking up wrong kernel
- AR-835 - Amlogic GLX desktop fails
- AR-834 - edge v network weirdness
- AR-833 - Khadas Edge V no HDMI audio
- AR-830 - CI needs to rsync cache before start to building new images.
- AR-829 - CI pipeline could not find some files when signing rootfs cache
- AR-825 - Fixing I2S related errors on RK3399
- AR-824 - Broken wifi on Station boards and kernel 5.12.y
- AR-823 - Pine H64 doesn’t boot kernel 5.12.y
- AR-822 - Motd false reporting of unsupported
- AR-821 - Docker creation failed to load repository keys
- AR-820 - Broken Odroid C2 audio patch
- AR-819 - Wireguard repo errors
- AR-818 - When building selected images via CI, status is changed to user-built
- AR-816 - ZRAM is missing in Jeston Nano legacy
- AR-780 - Nanopi R4S USB broken
- AR-779 - New bsp package is common. Per branch changes doesn’t work anymore
- AR-777 - Docker doesn’t install on Hirsute host
- AR-776 - Tinkerboard legacy have some troubles booting
- AR-774 - Orangepi Lite 2 EDGE is failing
- AR-770 - U-boot fails to install when switching kernel to EDGE
- AR-764 - Htop configuration exploit / vulnerability
- AR-749 - Allwinner A20 bootloops on 5.12.y / 2021.04
- AR-748 - Headers install broken
- AR-747 - Deeping Desktop doesn’t want to reboot / poweroff
- AR-744 - Nanopi K2 S905 network is broken
- AR-741 - Pinebook pro desktop missing tweaks
- AR-740 - Vnstat throws out garbage

© 2021 by Armbian

- AR-737 - Jetson nano throws out some error on boot loader compilation
- AR-736 - Rockpi S u-boot doesn’t build on GCC.-10
- AR-713 - Board specific desktop things are going into common desktop package
- AR-593 - Rockpi S doesn’t boot mainline based kernel

Epic

- AR-788 - Add Official Support for Some Khadas devices

Story

- AR-877 - build_all needs separate logs per image
- AR-847 - Tinkerboard 2 Support
- AR-746 - Upgrade EDGE to 5.12.y
- AR-734 - CSC Support for Avnet MicroZed
- AR-214 - CI improvements
- AR-202 - Drop packaging patches and introduce own packaging
- AR-42 - Merge packaging patches

Closed tasks

- AR-892 - Promoting Bullseye to supported
- AR-890 - Desktop analysis with 3D enabled
- AR-887 - Re-enable Debian Stretch repository update
- AR-882 - Optimise image compression
- AR-876 - Make package lists in one row
- AR-869 - Upgrade ZFS on Linux to v2.1.0 (Focal / Bionic only)
- AR-865 - Updating driver for 2.5G NIC on Helios64
- AR-864 - Upgrading EDGE to K5.13.y
- AR-859 - set - apt.armbian.com - redirect default to http instead of https
- AR-856 - Basic RC Branch build support
- AR-846 - Add Ubuntu 21.10 Impish
- AR-794 - Khadas Edge-V support
- AR-793 - Khadas VIM3L support
- AR-792 - Preliminary Khadas VIM3 support
- AR-791 - Khadas VIM2 support
- AR-790 - Preliminary Khadas VIM1 support
- AR-785 - Move mainline boot console to UART0 on Rockpi S
- AR-784 - Add Nvidia Jetson Nano legacy kernel
- AR-782 - Provides NFS mount functionality out of the box on CLI images
- AR-778 - Do not pre-install obsolete apt-transport-https
- AR-768 - Move Odroid XU4 EDGE to mainline source
- AR-745 - chroot packaging: build script as separate function
- AR-743 - Delay first-run autologin
- AR-732 - Unlock Ubuntu Hirsute as supported target
- AR-714 - Adjusting support status

© 2021 by Armbian

- AR-665 - rk3399 patch failure
- AR-649 - Adding Rockchip VPU support for 5.11.y
- AR-635 - Add legacy kernel for Zero2
- AR-537 - Create Armbian “virtual” build target to run as VM
- AR-519 - Odroid N2 Mainline u-boot for edge kernel
- AR-315 - Add support for GPT table inside nand-sata-install

7.2.3 v21.05.6 (2021-06-21)

Solved Bugs

- Updated images for Orangepi Zero
- [AR-593] - Rockpi S doesn't boot mainline kernel

7.2.4 v21.05.3 (2021-05-24)

Solved Bugs

- [AR-780] - Nanopi R4S USB broken
- [AR-816] - ZRAM is missing in Jeston Nano legacy

7.2.5 v21.05.2 (2021-05-24)

Solved Bugs

- [AR-748] - Headers install broken
- [AR-740] - Vnstat throws out garbage
- [AR-764] - Fixing Htop security issue

Closed Tasks

- ZFS updated to v2.0.4 (tested on 32bit Odroid HC1 and 64bit N2, Focal and Bionic userland)
- Added Hirsute CLI images with EDGE Linux 5.12.y for most of the boards

© 2021 by Armbian

7.2.6 v21.05 (2021-05-09)

Solved Bugs

- [AR-730] - Duplicate packages error when updating repository
- [AR-729] - Fix Partition Alignment for resizes and nand-sata-install
- [AR-711] - Network troubles on Nanopi K2 / Odroids
- [AR-709] - Tinkerboard AP crash on client connect
- [AR-708] - Missing library for compiling u-boot
- [AR-707] - Wrong keyboard code detected
- [AR-705] - Compilation issues when building old u-boot
- [AR-698] - XU4 - current kernel oddness with docker
- [AR-697] - Fix Meson64 Default Governor
- [AR-688] - Firefly boot broken
- [AR-674] - Users can't change desktop wallpaper on Gnome
- [AR-666] - ZSH is disabled on upgrade
- [AR-662] - Distribution support status is not written to the /etc/armbian-release
- [AR-659] - Rootfs image runs out of inodes during build
- [AR-653] - builder issue with gnome
- [AR-647] - Wireless driver 8811CU is broken on 5.11.y
- [AR-646] - Bootsplash breaks compilation on 5.11.y
- [AR-644] - Wireless driver 8188 EU broken and disabled since 5.9.y
- [AR-636] - Odroid N2+ lost additional freq values
- [AR-585] - HDMI-CEC not working on rockchip64 Legacy
- [AR-88] - Banana Pi M3 does not boot

Finished projects

- [AR-694] - Create Jira-based checklist for Desktop Testing
- [AR-457] - Enable native arm/arm64 building
- [AR-454] - Additional Desktop Configurations for use with new framework
- [AR-444] - Improving download infrastructure Phase 2
- [AR-200] - Improving Desktop images

© 2021 by Armbian

Closed Tasks

- [AR-714] - Adjusting support status
- [AR-710] - Create imx edge branch
- [AR-706] - Bump Allwinner u-boot to 2021.04
- [AR-704] - Distinguish between edge and normal image in motd
- [AR-696] - Improve Nvidia Jetson Nano support
- [AR-673] - Add few missing packages
- [AR-670] - Add additonal mirros for linux-firmware beside kernel source
- [AR-667] - Move Meson64 DEV to 5.10.y
- [AR-657] - Add instructions how to manual flash boot loader
- [AR-656] - Implement timeout on cache download
- [AR-654] - Fix stability issues of NanoPi M4V2 in current and dev
- [AR-651] - NanoPC-T4 legacy: enable USB-C DisplayPort & eDP outs
- [AR-648] - Resolve GPIO & PWM patches on mvebu
- [AR-645] - Detach rtl8812au from fixed commit ID if it builds from master
- [AR-643] - Bump DEV kernels to 5.11.y
- [AR-634] - Add Orangepi R1 Plus
- [AR-633] - Enable hardware PRNG/TRNG/SHA on sun8i-ce platform
- [AR-613] - test/beta img auto builder
- [AR-612] - Update pine64 install instructions
- [AR-600] - RK3399's: Add multimedia and OC overlays
- [AR-599] - Enable HDMI-CEC and ISP1 camera support for rk3399 and rockchip64 legacy
- [AR-369] - Check kernel config changes

7.2.7 v21.02.4 (2021-04-04)

- Added Nvidia Jetson Nano (community supported target)
- Rebuild images for Odroid N2, H4, HC4

7.2.8 v21.02.3 (2021-03-09)

- All kernels received upstream updates
- All images has been rebuilt
- Fixed reboot troubles on meson64 family (Odroid N2, C2, H4, HC4)
- ZSH upgrade fixed
- Type-C DP support for the NanoPC T4
- [AR-654] - NanoPi M4V2 stability fix for current and dev
- Allwinner a20 fail to init hdmi in many cases / fixed (all images need to be rebuilt)
- [AR-660] - Attempt to improve stability on Helios64

7.2.9 v21.02.2 (2021-02-16)

- All kernels received upstream updates
- [AR-633] - Enable hardware PRNG/TRNG/SHA on sun8i-ce platform
- [AR-636] - Odroid N2+ lost additional freq values

© 2021 by Armbian

7.2.10 v21.02.1 (2021-02-03)

Finished projects

- [AR-235] - Implement Device Tree Editor
- [AR-476] - Add sound to Odroid N2
- [AR-485] - Improve multicore compilation
- [AR-487] - Rework download pages
- [AR-508] - Add Odroid HC4
- [AR-546] - Added Pine64 Pinecube
- [AR-566] - Add Nanopi R4S
- [AR-568] - Add Orangepizero 2 WIP target
- [AR-571] - Move Meson64 DEV to 5.10.y
- [AR-589] - Add ZShell via armbian-zsh package
- [AR-590] - ZRAM Enhancements - decouple swap config from tmp

Solved bugs

- [AR-365] - 4k not detected properly on Amlogic, Rockchip devices
- [AR-440] - Errors shown at 1st login under certain conditions
- [AR-512] - Fix Ethernet for Opi3 and other devices with phymode for kernel 5.10
- [AR-514] - Download and verify not fully reliable
- [AR-547] - First login: adding a non-existing keyboard variant
- [AR-548] - mvebu DFS seems to cause system hang under high I/O
- [AR-557] - GCC compatibility issues
- [AR-559] - First login script - not all locales have UTF8 encoding
- [AR-565] - SATA on HC4 is not recognized
- [AR-570] - Improper order in getty override.conf
- [AR-584] - Nanopi M4V2 hangs on bluetooth loading
- [AR-595] - Rockpi 4B 1GB not booting
- [AR-605] - Booting troubles on Odroid C4 / HC4
- [AR-606] - Force boot script update throws out some error
- [AR-608] - Broken building out-of-tree modules
- [AR-610] - Nanopi Neo2 black sometimes nic doesn't init
- [AR-615] - Helios64 unstable 2.5Gbps Interface on LK5.x
- [AR-616] - Ubuntu Bionic ZSH / BASH changing issue
- [AR-617] - Locales detection doesn't work properly in some cases
- [AR-627] - Ubuntu update is overwriting our welcome screen
- [AR-629] - Odroid HC4 SATA failure
- [AR-631] - Orangepi Zero2 broken network
- [AR-632] - Desktop fails to load at second run

© 2021 by Armbian

Closed task

- [AR-163] - Systematically cleanup distribution defaults
- [AR-206] - Improve memory performance on Renegade (roc-rk3328-cc) in current
- [AR-399] - Improve Pinebook PRO support
- [AR-467] - Enable AUFS support back
- [AR-472] - Added support for Ubuntu 20.10 Groovy
- [AR-517] - Mark Bionic builds host as deprecated
- [AR-520] - Move Rock64 to CSC in build script
- [AR-525] - Bump Rockchip 32bit to 5.9.y
- [AR-526] - Move mvebu-dev kernel to 5.9+
- [AR-551] - Update fan configuration, enable network LED and enable UPS timer
- [AR-552] - Re-enable UHS SDR104 mode for Helios64 and roc-rk3399-pc
- [AR-553] - Update builder to retrieve web seeds from mirrors api
- [AR-554] - OdroidN2 Ethernet Failure Pt2
- [AR-556] - Adding vnstat and ZFS support to MOTD
- [AR-558] - Switch mvebu current to K5.9
- [AR-563] - Improve headers compilation
- [AR-576] - Enabled debug on RockpiS
- [AR-579] - Improve (oh-my)ZSH loading speed
- [AR-587] - Fix kernel switching for rk3399 family
- [AR-594] - Upgrade Meson64 u-boot to 2020.10
- [AR-598] - Switch rockchip64 u-boot to 2020.10
- [AR-601] - Move sunxi-current to 5.10.y
- [AR-603] - Enable SPI boot option for roc-rk3399-pc
- [AR-607] - Move Meson64 Current to 5.10.y
- [AR-609] - Move Mvebu Current to 5.10.y
- [AR-611] - Switch rockchip64-current to 5.10.y
- [AR-614] - Upgrade ZFS packages
- [AR-618] - Upgrade mvebu64 current to 5.10.y
- [AR-619] - Bump rockchip current to 5.10.y
- [AR-620] - Enable network link leds for NanoPi R4S by default
- [AR-622] - Enable DMC for Station-M1 in current and dev
- [AR-623] - Enable RTC (hym8563) for Station P1 in dev and current
- [AR-624] - Provide an option to skip autodetection at first login
- [AR-628] - Bump Meson64 u-boot to 2021.01
- [AR-630] - Bump Odroid XU4 DEV to 5.10.y

7.2.11 v20.11.10 (2021-01-25)

- All images rebuild due to torrent system corruption

7.2.12 v20.11.9 (2021-01-23)

- Broken Nanopi Neo buster image rebuild, adding Station M1 and P1 legacy images, Odroid XU4 update

© 2021 by Armbian

7.2.13 v20.11.8 (2021-01-17)

- [AR-614] - Upgrade ZFS on Focal and Buster (64bit only) to v2.0.1

7.2.14 v20.11.7 (2021-01-06)

- [AR-605] - Booting troubles on Odroid C4 / HC4
- all images were rebuilt - we had a few corrupted ones in previous build

7.2.15 v20.11.6 (2021-01-03)

- [AR-601] - Move sunxi-current to 5.10.y
- [AR-235] - Implement Device Tree Editor in armbian-config
- [AR-589] - Add armbian-zsh package
- [AR-590] - ZRAM Enhancements - decouple swap config from tmp
- [AR-554] - Fix Odroid N2 Ethernet Failure
- [AR-556] - Adding vnstat and ZFS support to MOTD
- [AR-579] - Improve (oh-my)ZSH loading speed
- [AR-512] - Fix Ethernet for Opi3 and other devices with phymode for kernel 5.10
- [AR-547] - First login: adding a non-existing keyboard variant
- [AR-565] - Fix SATA on HC4 is not recognized
- [AR-595] - Fix Rockpi 4B 1GB not booting

7.2.16 v20.11.5 (2020-12-31)

- [AR-566] - Add Nanopi R4S preview images

7.2.17 v20.11.4 (2020-12-15)

- Re-adding accidentally removed network driver on Helios64
- added OpenHab 3 to the armbian-config software installer
- [AR-587] - Fix kernel switching for rk3399 family

© 2021 by Armbian

7.2.18 v20.11.3 (2020-12-12)

Bugfix release

- [AR-559] - First login script - not all locales have UTF8 encoding
- [AR-163] - Systematically cleanup distribution defaults
- [AR-206] - Improve memory performance on Renegade (roc-rk3328-cc) in current
- [AR-472] - Added support for Ubuntu 20.10 Groovy
- [AR-476] - Add sound to Odroid N2
- [AR-485] - Improve multicore compilation
- [AR-487] - Rework download pages
- [AR-508] - Add Odroid HC4
- [AR-514] - Download and verify not fully reliable
- [AR-517] - Mark Bionic builds host as deprecated
- [AR-525] - Bump Rockchip 32bit to 5.9.y
- [AR-526] - Move mvebu-dev kernel to 5.9+
- [AR-546] - Added Pine64 Pinecube
- [AR-547] - First login: adding a non-existing keyboard variant
- [AR-548] - mvebu DFS seems to cause system hang under high I/O
- [AR-551] - Update fan configuration, enable network LED and enable UPS timer
- [AR-552] - Re-enable UHS SDR104 mode for Helios64 and roc-rk3399-pc
- [AR-553] - Update builder to retrieve web seeds from mirrors api
- [AR-556] - Adding vnstat and ZFS support to MOTD
- [AR-557] - GCC compatibility issues
- [AR-558] - Switch mvebu current to K5.9
- [AR-563] - Improve headers compilation
- [AR-565] - SATA on HC4 is not recognized
- [AR-568] - Add Orangepizero 2 WIP target
- [AR-570] - Improper order in getty override.conf
- [AR-571] - Move Meson64 DEV to 5.10.y

7.2.19 v20.11.1 (2020-12-04)

- [AR-551] - Update fan configuration, enable network LED and enable UPS timer
- [AR-565] - SATA on HC4 is not recognized
- Updated Odroid C4/HC4, Helios64, Rockpi 4* images and rockchip64 kernels

© 2021 by Armbian

7.2.20 v20.11 (2020-11-24)

Finished projects

- [AR-2] - Improving download infrastructure Phase 1
- [AR-151] - Integrate JMCCs Multimedia script
- [AR-230] - Decide what to do with TVboxes
- [AR-412] - Update Odroid XU4 kernels
- [AR-424] - Improve HTOP config
- [AR-456] - Upgrading Allwinner u-boot to 2020.10
- [AR-476] - Add sound to Odroid N2
- [AR-485] - Improve multicore compilation
- [AR-508] - Add Odroid HC4
- [AR-509] - Upgrade meson64 to 5.9.y
- [AR-510] - Move meson (Odroid C1) to 5.9.y
- [AR-532] - Move Odroid C4 from legacy u-boot toward mainline

Solved bugs

- [AR-314] - Links to SHA files at download pages are wrong
- [AR-372] - Meson64 Reboot failure kernel 5.7
- [AR-373] - Rock64 no HDMI (must be unplugged)
- [AR-382] - Fix zram creation on bigger memory devices
- [AR-391] - Warning a reboot is needed doesn't go away after reboot
- [AR-407] - Bug in first login script
- [AR-417] - HTOP in Bullseye needs higher package version
- [AR-420] - GPIO SPI patch is failing on Rockchip64
- [AR-422] - Improper version showing at upgrade
- [AR-425] - Resize is finished but message doesn't disappear
- [AR-428] - Firefox initial config has different location then ESR variant
- [AR-436] - Rockpi S reports some error in postinst scripts
- [AR-437] - MOTD cosmetic issue
- [AR-439] - Automated rebuilds set image status to USER_BUILT
- [AR-441] - Odroid C4 legacy bootscript problem
- [AR-452] - Fix first boot locales selection and add desktop lang switching
- [AR-459] - Missing package libreoffice-style-tango from Bullseye desktop
- [AR-471] - Mitigate Git server failures
- [AR-482] - Htop doesn't show CPU speed to normal user but shows properly to root
- [AR-484] - Odroid C4 refuse to boot
- [AR-491] - LEDs on Helios4 not working
- [AR-493] - Patches are not creating
- [AR-494] - Fix armbian-hardware-opitimization not being run
- [AR-505] - armbian-hardware-optimization: eth0 tweak applied before it is appear on /proc/interrupts
- [AR-527] - Rockchip 32bit sources were removed
- [AR-528] - Improve creating images from repository
- [AR-529] - Z28 PRO device tree doesn't exists in mainline

© 2021 by Armbian

Closed tasks

- [AR-284] - Discuss if there is a cleaner way to install Chromium
- [AR-350] - Switch rock64 to mainline u-boot
- [AR-351] - Switch rockpro64 to mainline u-boot
- [AR-363] - Switch mvebu current to K5.8.y
- [AR-380] - Revisit RTL8812AU driver
- [AR-387] - Switch from rk3399-bluetooth service to btbcm for loading firmware/patchram in dev/current
- [AR-388] - XU4 - Introduce new Mem freq scaling patch and re-enable
- [AR-390] - Add Radxa Rockpi 4C
- [AR-400] - Enable overlays in rockchip64-legacy
- [AR-401] - Enable creation of SPI flash u-boot image for ROCK Pi 4
- [AR-403] - Enable overlays in rk3399-legacy
- [AR-404] - Switch renegade to mainline u-boot
- [AR-409] - Move imx6 current kernels to 5.8.y
- [AR-413] - Improve reliability of Helios64's eMMC module
- [AR-415] - Improve reboot reliability for Helios64
- [AR-416] - Move Rockchip 32bit to 5.8.y
- [AR-419] - Add dedicated DT for Nanopi Neo3
- [AR-445] - systemd-journal not rotated with armbian-ramlog
- [AR-458] - Update board support statuses
- [AR-461] - Add Armbian to Neofetch
- [AR-462] - Adapt helios64 device tree name to match upstream Linux
- [AR-464] - Move Libre Computer Renegade to mainline u-boot
- [AR-472] - Added support for Ubuntu 20.10 Groovy
- [AR-473] - Add interactive option to use precompiled packages from Armbian repository
- [AR-477] - Advanced recovery options for rockchip64 boards
- [AR-483] - Fix analog (3.5 jack) audio on ROCK Pi 4C
- [AR-490] - Enable RTC on Odroid N2
- [AR-495] - Allow building images with kernels 5.8.17+ and 5.9.2+
- [AR-499] - Enable Watchdog for G12/Odroidn2
- [AR-504] - Helios64: Switch fusb302 driver to mainline and enable DP over TypeC
- [AR-511] - Switch rockchip64-current to linux 5.9.y
- [AR-513] - Move Odroid XU4 kernels up
- [AR-515] - Upgrade imx6 to 5.9.y
- [AR-521] - Exchange mv with rsync
- [AR-522] - Allow setting MTU for Rockchip64's dwmac interface
- [AR-523] - enable CONFIG_TARGET_CORE for iSCSI target support
- [AR-524] - Upgrade rockpis legacy kernel
- [AR-531] - Check why disabling one update-initramfs breaks Ubuntu initrd making on update

7.2.21 v20.08.22 (2020-11-8)

- Added WIP images for Odroid HC4

© 2021 by Armbian

- Updated images for Odroid C4, N2, C2, Lafrite, Lepotato, KVIM1

7.2.22 v20.08.13 (2020-10-19)

- [AR-363] - Switch mvebu current to K5.8.y
- [AR-466] - Enable Recovery button on Helios64
- [AR-416] - Move Rockchip 32bit to 5.8.y
- [AR-476] - Add sound to Odroid N2
- update all kernels
- Rebuild images for Helios4, Helios64 and Odroid N2

7.2.23 v20.08.11 (2020-10-16)

- [AR-465] - Helios64 cannot boot from eMMC
- enable Ubuntu 20.10 Groovy as a CSC build option (need build parameter EXPERT="yes")
- update u-boot loader to 2020.10 on Allwinner platform
- update all kernels
- update images for Helios 64
- add option to build images from prebuild packages from repository which drastically improves build time in case you don't need to rebuild kernel

7.2.24 v20.08.8 (2020-10-05)

- [AR-463] - Improve Helios64 Stability, updated images
- Adapt Helios64 devicetree name to match upstream Linux

7.2.25 v20.08.4 (2020-09-23)

- [AR-399] - Improve Pinebook PRO support, updated images
- Updated Helios64 images

7.2.26 v20.08.3 (2020-09-21)

- updated mainline kernel based images to Linux 5.8.10
- all other kernels updated to respective latest version
- improved htop with showing network status dynamically, GPU temp, improved CPU speed display
- fixed usbip for sharing usb over network
- fixed Odroid C4 boot script bug; adding normal and higher CPU speeds
- added many improvements for Helios64
- enabled GPU temperatures in htop for XU4, meson64 (Odroid N2/N2+) and rockchip64/32
- fixed initial configuration for Firefox
- fixed tx offloading for Rockchip64 NIC’s
- move rockchip 32bit to 5.8.y (Tinkerboard / MiQi)
- improved RK3399 stability by mingling OPP
- fixed a bunch of bugs related to encrypted root filesystem
- enabled hardware watchdog support for mvebu64 / Espressobin
- cosmetic fixes to motd

© 2021 by Armbian

- enabling I2S and spdif on Nanopi Neo3 by default
- fixes wrong memory calculation on ZRAM display
- fixing firstlogin bug preventing running xrdp
- move Espressobin kernel to 5.8.y
- adjust / fix Kali Linux wifi injections patches

Known bugs:

- Some Rockpro64 boards have troubles with upgrade
- Bananapi M3 eMMC can’t boot from eMMC (solution is available)
- H6 stability issues on some boards
- RockpiS shows some error on upgrade but upgrade suceeds
- 4K and audio on mainline based meson64 boards

7.2.27 v20.08 (2020-08-20)

Finished projects

- [AR-45] - Make first login more user friendly
- [AR-71] - Create a document: How we will use Jira
- [AR-182] - Unify / merge kernel configs
- [AR-201] - Introduce CI autotest facility
- [AR-227] - Move Espressobin current to K5.6
- [AR-313] - Ability to work in offline mode
- [AR-320] - Initial support for Rockpi E
- [AR-324] - Add Rockchip RK322X SoC support
- [AR-328] - Meson64 move current to 5.7.y
- [AR-329] - Switch sunxi dev target to kernel 5.7
- [AR-331] - Enable kernel boot splash as an option
- [AR-335] - Improve patch making
- [AR-392] - Add Odroid N2+
- [AR-402] - Add Helios64

© 2021 by Armbian

Solved bugs

- [AR-282] - Rockpi 4B 1Gb doesn't boot modern kernel / u-boot
- [AR-295] - Odroid C2: no more USB devices after upgrade
- [AR-298] - Missing default SElinux policy
- [AR-303] - Create a download page for BPI M2 zero
- [AR-305] - K-worker creates load on Allwinner devices
- [AR-319] - Armbian config failed to switch kernels
- [AR-330] - Shell check bugs
- [AR-332] - When making all kernels - building sometimes fails
- [AR-337] - Odroid XU4 Memcopy Slow on all Kernel 5.x 80MB/sec instead of 370+MB/sec
- [AR-338] - Bananapi R2 does not boot at all
- [AR-340] - Fix WiFi on Nanopi M4V2
- [AR-348] - Confirm RK3399 TcpOffloading bug
- [AR-352] - Fix Random MAC on H3 boards
- [AR-354] - Support User Provided EDID Firmware
- [AR-355] - backport Linux v5.8 fbtft/fb_st7789v invert colors, proper gamma
- [AR-356] - Building multiple u-boot targets breaks
- [AR-371] - CPU frequency scaling broken on H6
- [AR-378] - Increase address room for initial ramdisk
- [AR-381] - selinux-policy-default missing on Debian Bullseye
- [AR-393] - Ask for setting locale at first run

© 2021 by Armbian

Closed tasks

- [AR-28] - Added new GCC compilers
- [AR-225] - Introduce PACKAGE_LIST for BOARD and FAMILY
- [AR-300] - Enable HDMI audio for OrangePi 4
- [AR-317] - Move Odroid XU4 dev to mainline + patches
- [AR-318] - Upgrade Odroid XU4 legacy kernel
- [AR-321] - Upgrade Meson (C1) current to 5.7.y
- [AR-323] - Allow install to SD NAND for Rockpi S
- [AR-326] - Make USB3 support of ROCK Pi E on par with other rk3328 boards
- [AR-327] - Bump imx6 current kernel to 5.7.y
- [AR-333] - Update Odroid XU4 kernels
- [AR-334] - Cleanup boot environment files
- [AR-336] - Add support for cheap 2.5GB USB network dongles
- [AR-341] - Follow-up N2 CPU Affinity
- [AR-349] - Update mainline u-boot to v2020.07 for rockchip64 and rk3399
- [AR-357] - IRQ affinity improvements for G12B
- [AR-358] - Added initial support for Neo 3
- [AR-361] - Update Odroid XU4 boot.ini
- [AR-362] - HDMI sound support for Allwinner A10, A20, A31
- [AR-364] - Change sunxi legacy to 5.4.y, current to 5.7.y
- [AR-366] - Move rockchip/64 current to 5.7.y
- [AR-383] - Upgrades for Tapatalk plugin
- [AR-389] - Add PACKAGE_LIST_BOARD_REMOVE option

7.2.28 v20.05.7 (2020-07-02)

- [AR-308] - Disable HDMI in u-boot for rk3399 boards
- [AR-338] - Bananapi R2 does not boot at all
- [AR-337] - Odroid XU4 Memcopy Slow on all Kernel 5.x 80MB/sec instead of 370+MB/sec
- Update images for: NanoPC T4, Nanopi M4,Nanopi M4v2, Nanopi Neo4, Orangepi 4, Firefly RK3399, Bananapi R2, Odroid XU4

7.2.29 v20.05.6 (2020-06-19)

- [AR-324] - Add Rockchip RK322X SoC support
- [AR-320] - Initial support for Rockpi E
- [AR-323] - Allow install to SD NAND for Rockpi S

7.2.30 v20.05.5

never released/skipped

7.2.31 v20.05.4 (2020-06-16)

- [AR-311] - Initrd on Focal can get corrupted followup fix

© 2021 by Armbian

7.2.32 v20.05.3 (2020-06-10)

- [AR-300] - Enable HDMI audio for OrangePi 4
- [AR-305] - K-worker creates load on Allwinner devices
- [AR-282] - Rockpi 4B 1Gb doesn't boot modern kernel / u-boot

7.2.33 v20.05.2 (2020-06-05)

- [AR-294] - Initrd on Focal can get corrupted

7.2.34 v20.05.1 (2020-05-31)

Kagu

Finished projects

- [AR-108] - Upgrade remaining kernels to 5.4.y
- [AR-158] - Update 3rd party wireless drivers
- [AR-159] - Switch fake-hwclock to hardware RTC on mvebu family
- [AR-168] - Add NanoPi R2S board support
- [AR-180] - Update Wireguard drivers on kernels below 5.4.y
- [AR-184] - Improve slow booting on Rockchip RK3399 devices
- [AR-185] - Change download images compression format
- [AR-190] - Update wireless driver for RTL88x2BU
- [AR-196] - Upgrade u-boot to 2020.04 where possible
- [AR-201] - Introduce CI autotest facility
- [AR-207] - Merge rockpis-dev into rockchip64
- [AR-208] - Consolidate u-boot variants for mvebu family
- [AR-210] - Add support for more HDMI resolutions on Rockchip RK3288 devices
- [AR-215] - Move meson64 dev branch to 5.6.y
- [AR-221] - Upgrade imx6 current to 5.6.y
- [AR-222] - Port Docker image building to Ubuntu 20.04
- [AR-226] - Add Hardkernel Odroid C4 mainline u-boot / kernel
- [AR-236] - Attach Meson64 CURRENT to 5.6.y
- [AR-238] - Updating hostapd, PSD and theme package in repository
- [AR-247] - Revitalise Udoo board
- [AR-250] - Improve usage of external patches
- [AR-253] - Add prerm script for headers
- [AR-254] - Add Banana Pi M2 Zero
- [AR-257] - Bring Odroid C1 back from the EOL with latest upstream kernel
- [AR-260] - Add Nanopi A64 board support
- [AR-261] - Add Rockpi S mainline board support
- [AR-262] - Move Allwinner development branch to 5.6.y
- [AR-278] - Add snap free Chromium to Ubuntu Focal
- [AR-279] - Add Hardkernel Odroid C4 stock kernel

© 2021 by Armbian

Solved bugs

- [AR-109] - Upgrade is not done properly on some boards
- [AR-165] - Instability with Rock64 and Rock PRO
- [AR-177] - No serial gadget console on Nanopi Neo2 black
- [AR-181] - Odroid N2 crashes during USB rsync backups
- [AR-198] - Olimex Lime 2 doesn't boot from eMMC
- [AR-204] - CPUfreq defaults missing on update
- [AR-205] - No sound output with OrangePi 4 in dev and current
- [AR-211] - Chrony fails to start on Ubuntu Focal
- [AR-212] - Random MAC on Nanopi R2S
- [AR-220] - Disable 3D support in Bionic due to broken mesa packages
- [AR-231] - Unstable stmmac network driver on Meson64
- [AR-237] - Desktop install on Ubuntu Focal installs Gnome3 desktop
- [AR-239] - Chrony fails to start on Focal
- [AR-240] - Broken VFAT kernel upgrade
- [AR-244] - Thermal throttling on H5 doesn't work properly
- [AR-245] - Hostapd doesn't go up
- [AR-248] - Odroid C4 CPU speed is limited to 1.5Ghz
- [AR-249] - Problems with CI testings
- [AR-251] - Fix kernel 5.7.y packages patch
- [AR-255] - Fix Debian mirrors URL
- [AR-263] - Fix audio on Renegade
- [AR-269] - Add correct CPU regulator configuration for the NanoPI R1
- [AR-274] - Add missing iozone3 package to the minimal image
- [AR-277] - Distinguish nightly and stable images at the download pages
- [AR-286] - Armbian-resize-filesystem fails on first run due to missing fdisk in Bullseye
- [AR-287] - Make sure cryptsetup-initramfs is installed in any case

© 2021 by Armbian

Closed tasks

- [AR-150] - Disable Stretch image creation for Helios4 and Clearfog
- [AR-157] - Add Ubuntu Focal 20.04 as a supported build host
- [AR-186] - Blacklist 3D engine on headless images
- [AR-189] - Move wireless driver for 8189ES from patch to git
- [AR-195] - Adding Ubuntu 20.04 to all builds
- [AR-209] - Disable CONFIG_VIDEO_DE2 in u-boot for Allwinner devices
- [AR-213] - Make manual for .xz images and check their authentication
- [AR-228] - Enable audio and USB on Nanopi A64
- [AR-229] - Bump with AUFS for DEV kernels
- [AR-232] - Switch Odroid XU4 DEV branch to Libreelec branch
- [AR-234] - Disable ZSH update prompt on every two weeks
- [AR-242] - Enable SELinux
- [AR-252] - Improve source code cleaning
- [AR-258] - Enables PCIE PHY with Mezzanine NVME
- [AR-259] - Add mp8859 regulator to current for RK3399-ROC-PC
- [AR-264] - Enable RTL8723DS WiFI driver
- [AR-265] - Remove Xenial from supported host OS
- [AR-266] - Fix dependency for native building on Linux Mint and Debian Buster
- [AR-267] - Enable Cedrus video acceleration on Allwinner kernels
- [AR-268] - Add higher clock for Allwinner H6
- [AR-270] - Add support for alternate console UARTs in Allwinner H3 u-boot
- [AR-271] - Lower DDR clock rate to 504MHz for H5 boards
- [AR-280] - Update CONF, CSC and WIP statuses according to support level
- [AR-285] - Improve thermal throttling on Allwinner H6
- [AR-288] - Add vendor name to the board config files

7.2.35 v20.02.12 (2020-04-27)

- Added preview images for Odroid C4

7.2.36 v20.02.8 (2020-03-26)

- update kernels with upstream versions, synchronise and test kernel sources download

7.2.37 v20.02.7 (2020-03-26)

- Updated images for Rockpi S, Odroid XU4 and FriendlyARM Nanopc T3, T3+, M3, Fire3

7.2.38 v20.02.6 (2020-03-23)

- Updated images for Rockpi S and Orangepi 4
- Updated armbian-config (fixed OMV installer)

7.2.39 v20.02.5 (2020-03-19)

- Updated images for Orangepi 4, Bananapi and Rockpi S

© 2021 by Armbian

7.2.40 v20.02.4 (2020-03-18)

- Added images for Nanopi R2S and Bananapi M2 Zero
- Kernel update for Odroud XU4

7.2.41 v20.02.3 (2020-02-21)

- Updating images for Le potato, Khadas Vim1,La Frite, Nanopik2 S905, Odroid N2/C2 - fixing audio
- Updating images for Orangepi 4 - boot loader problem

7.2.42 v20.02.2 (2020-02-18)

Chiru

© 2021 by Armbian

Tasks

- [AR-46] - Support for single function run
- [AR-47] - Adding Docker shell support
- [AR-49] - Move sunxi kernel to 5.4.y
- [AR-79] - Check and adjust AUFS patch for 5.4.y
- [AR-80] - Move imx6 to 5.4.y
- [AR-81] - Enable Meson64 DEV at 5.4.y
- [AR-82] - Move Mvebu64 / Espressobin dev kernel to 5.4.y
- [AR-84] - Move rockchip64 current to 5.4.y
- [AR-85] - Adjusted Sunvell R69
- [AR-90] - Add support for Nanopi M4 v2
- [AR-92] - Enable stable MAC address from cpuid on rk3399
- [AR-96] - Update Xradio wireless driver
- [AR-97] - Tag supported builds properly at download pages
- [AR-98] - Enable missing Kuberenetes kernel dependency
- [AR-100] - Add Debian Bullseye and Ubuntu Focal
- [AR-112] - Enabled internal WLAN on RockPi S
- [AR-113] - Install wireguard tools only when selected
- [AR-114] - Enable audio codec on Orangepi Win
- [AR-115] - Add drivers for Realtek RTL8811CU and RTL8821C chipsets
- [AR-116] - Remove annoying debug message filling logs on 8189es
- [AR-117] - Add Pine H64 model B
- [AR-124] - Enable wireless on Rockpi-S
- [AR-127] - Refactoring wifi patches
- [AR-128] - Adding WIP support for Pinebook PRO
- [AR-129] - Move NanopiM4 V2 and Pine H64 under supported
- [AR-134] - Update AUFS version on Odroid XU4 and Nanopi Fire3/T3/T3+
- [AR-138] - Update RK3399 legacy kernel (Nanopi M4, T4, Neo4) to latest upstream version
- [AR-139] - Nanpi R1 - move primary serial console to ttyS1 which is on the chassis
- [AR-143] - Create OpenHab installation instructions for their official documentation
- [AR-146] - Update rockchip-legacy to most recent upstream kernel version
- [AR-147] - Enable analogue audio on Allwinner H6
- [AR-148] - [mvebu-current] Fix cpufreq (dynamic frequency scaling)
- [AR-149] - [mvebu-current] Fix pcie issues
- [AR-153] - Enable USB3 for Rock64/Renegade with RK3328 on mainline kernel
- [AR-154] - Add analogue audio driver to Allwinner H6
- [AR-155] - Enable Cedrus video acceleration support on Allwinner boards
- [AR-167] - Add upstream patches for Odroid XU4
- [AR-172] - USB3 Support for Rockchip

© 2021 by Armbian

Bugs

- [AR-74] - User patches directories not created
- [AR-76] - Rockchip64 missing CPU_MIN variable
- [AR-77] - Wrong board name variable for Orangepi RK 3399
- [AR-83] - Packaging patch broken for kernel 5.4.y
- [AR-86] - CPU freq scaling for H6 doesn't work in K5.4
- [AR-88] - Banana Pi M3 does not boot
- [AR-89] - Tinkerboard S doesn't start from eMMC
- [AR-91] - Broken Allwinner overlays
- [AR-94] - Espressobin v7 with 2gb of ram fail to boot
- [AR-102] - Missing packaging patch for Rockpis legacy kernel
- [AR-103] - PPA way of adding sources are failing on Ubuntu
- [AR-104] - 32bit rust compiler doesn't run new kernels
- [AR-105] - Orangepi Zero Plus 2 doesn't boot
- [AR-106] - Wireguard breaks building on 5.4.y
- [AR-107] - Improve compiler and rootfs download process
- [AR-110] - Missing Bionic image for Nanopi Neo Plus2
- [AR-111] - Some versions of Orangepi Win does not boot modern kernel
- [AR-118] - NanoPi M4V2 ethernet partialy broken in one side
- [AR-123] - OpenHAB2 installation is failing
- [AR-125] - Wireless driver for 8188EUS breaks on K4.14
- [AR-126] - Nanopi M3/Fire3/PC3 compilation breaks
- [AR-130] - Instability with various A64 boards
- [AR-131] - Add support for 3rd version of Pinebook A64 panel
- [AR-133] - Odroid XU4 legacy kernel images instability
- [AR-141] - Odroid XU4 current with kernel 5.4.y seems unstable
- [AR-142] - Cryptsetup disk encryption build feature broken
- [AR-144] - Tinkerboard break booting
- [AR-145] - Missing HDMI audio on H3 boards
- [AR-152] - Display issues with Bionic Mesa update
- [AR-164] - Htop package does not build in qemu under Ubuntu Focal 20.04
- [AR-166] - Rootfs cache number creates a window of 12h when users are forced to rebuild cache
- [AR-170] - Wireless not connecting for SBCs
- [AR-171] - Fix broken loading process on MiQi
- [AR-173] - Fix makefile of kernel headers 4.4.210 for rk3399
- [AR-174] - Teres Keyboard Not Working

Stories

- [AR-48] - Bump u-boot to 2020.01 on RK3399 boards
- [AR-156] - WIP orangepi 4 preliminary support

© 2021 by Armbian

7.2.43 v19.11.3 (2019-11-20)

Tasks

- [AR-1] - Adding support category for distributions
- [AR-4] - Remove Allwinner legacy
- [AR-5] - Drop Udoo family and move Udoo board into newly created imx6 family
- [AR-9] - Rename sunxi-next to sunxi-legacy
- [AR-10] - Rename sunxi-dev to sunxi-current
- [AR-11] - Adding Radxa Rockpi S support
- [AR-13] - Rename rockchip64-default to rockchip64-legacy
- [AR-14] - Add rockchip64-current as mainline source
- [AR-15] - Drop Rockchip 4.19.y NEXT, current become 5.3.y
- [AR-16] - Rename RK3399 default to legacy
- [AR-17] - Rename Odroid XU4 next and default to legacy 4.14.y, add DEV 5.4.y
- [AR-18] - Add Odroid N2 current mainline
- [AR-19] - Move Odroid C1 to meson family
- [AR-20] - Rename mvebu64-default to mvebu64-legacy
- [AR-21] - Rename mvebu-default to mvebu-legacy
- [AR-22] - Rename mvebu-next to mvebu-current
- [AR-23] - Drop meson64 default and next, current becomes former DEV 5.3.y
- [AR-24] - Drop cubox family and move Cubox/Hummingboard boards under imx6
- [AR-26] - Adjust motd
- [AR-27] - Enabling distribution release status
- [AR-28] - Added new GCC compilers
- [AR-29] - Implementing Ubuntu Eoan
- [AR-30] - Add desktop packages per board or family
- [AR-31] - Remove (Ubuntu/Debian) distribution name from image filename
- [AR-32] - Move arch configs from configuration.sh to separate arm64 and armhf config files
- [AR-33] - Revision numbers for beta builds changed to day_in_the_year
- [AR-34] - Patches support linked patches
- [AR-35] - Break meson64 family into gxbb and gxl
- [AR-36] - Add Nanopineo2 Black
- [AR-38] - Upgrade option from old branches to new one via armbian-config
- [AR-41] - Show full timezone info
- [AR-43] - Merge Odroid N2 to meson64
- [AR-44] - Enable FORCE_BOOTSCRIPT_UPDATE for all builds
- [AR-57] - New kernel feature requested CONFIG_BLK_DEV_DRBD
- [AR-60] - Modified logrotate.service
- [AR-63] - Docker maintenance features

© 2021 by Armbian

Bugs

- [AR-25] - Armbian resize stopped working in Ubuntu 19.10 or higher
- [AR-40] - When changing console layout it does not change
- [AR-51] - Prevent configuring locale
- [AR-52] - Broken desktop install
- [AR-54] - Upstream package name changed
- [AR-55] - Wireless driver remove patch for Odroid XU4 broke down
- [AR-56] - Missing CPU regulator
- [AR-58] - Troubles with wireless on Nanopi DUO & Opi Zero
- [AR-59] - Compressed files are getting back to /var/log
- [AR-62] - No HDMI sound on various meson64 boards
- [AR-64] - Docker require root
- [AR-68] - Broken Ethernet on Pine64+

Stories

- [AR-61] - Adding support for LOCAL_MIRROR
- [AR-65] - Moving configs under userpatches
- [AR-66] - Enable build system torrent download by default
- [AR-67] - Install Docker when we want to build under Docker
- [AR-69] - Use kernel command line instead of a patch
- [AR-70] - Enable Lima kernel driver on meson64
- [AR-73] - Enable PCI on Rockpi 4 and overlay for GEN2 speed

7.2.44 v5.98 (2019-10-09)

- changed ntptime with chrony
- fixed serial console on several hosts
- added FriendlyARM ZeroPi
- enabled gadgets on rockchip64
- bumped RK3399 boards to latest kernel, recreate images and repository
- merged odroidxu4 down to default since we only have one kernel
- fixed Cubox images, move them to stock kernel
- fixed low Synaptic search speed

© 2021 by Armbian

Build script:

- script configurations were migrated to userpatches
- added option to create minimal images with around 500Mb in size BUILD_MINIMAL=”yes”
- added initial support of MCIMX7SABRE board (CSC)
- updates for xt-q8l-v10 (CSC)
- vagrant-disksize is beeing determined automatically
- Docker is installed automatically if one want use it (Debian based build host only)
- refactor build all images scripting that images can be build in full parallel mode
- added one file for storing which combinations shell be made for each board
- replaced Etcher with dd + verify for directly burning images when done
- cleaned initial config and remove confusing advanced options out

7.2.45 v5.92 (2019-08-02)

- updated sunxi NEXT (4.19.63) and DEV (5.2.5) kernels
- updated htop application to show cpu speed and temperature (buster / disco)

7.2.46 v5.91 (2019-07-31)

- created new images for Helios4 and Clearfog Pro/Base
- moved mvebu DEFAULT, NEXT and DEV branch to next kernel (LTS) and U-boot version
- fixed armada_thermal sensor reading, adjusted Helios4 fancontrol configuration
- fixed ODT on data signals of DDR RAM for Armada A388 SOMs
- recreated Armbian Buster images due to a bug in Network manager which in some cases failed to initiate network connection

Armbian-config:

- added Emby installation
- updated Plex install to use official repo
- added netmask-to-CIDR function for manual IP configuration

7.2.47 v5.90 (2019-07-07)

- added Armbian Buster images for all boards
- added Macchiatobin Doubleshot CSS target and images
- added images with test kernel v5.1.y for: Orangepi3, Lite2, One+, PineH64, Odroid C1, Teres, Pinebook
- added wireless drivers for 88x2bu
- added eMMC support for Nanopi K2 (booting from doesn’t work yet)
- added dual w1 overlay for meson64 family
- updated wireless drivers for Realtek 8811, 8812, 8814 and 8821
- updated wireless drivers for rtl8188eus & rtl8188eu & rtl8188etv
- added latest Wireguard driver
- enable eMMC on Orangepi Win Plus
- enable Bluetooth on Tinkerboard, Nanopi4, Rockpi 4 CLI images
- improved ALSA config on Tinkerboard
- fixed Bluetooth on Nanopi M4/Neo4/T4 and Rockpi4
- fixed wireless drivers on OPi3 & Lite2

© 2021 by Armbian

- fixed temperature readout on Allwinner H5 boards
- fixed SPI related bug on Allwinner 5.1.y kernel
- fixed HDMI output and bump kernel to 5.1.y on imx6 boards
- fixed eMMC install, add rootdev= to armbianEnv if missing
- fixed A10/A20 SATA write speed
- set default build target from Debian Stretch to Buster for all boards
- changed CPU clock back to 1.5/1.8Ghz defaults on boards with RK3399 to minimise thermal throttling
- changed motd console welcome text to: “Welcome to Debian Stretch with Armbian Linux 5.1.6-sunxi”
- changed display manager to lightdm by default and remove nodm completely
- changed u-boot for A64 to upstream sources
- changed RK3399 to U-boot 2019.04
- added URL to the build script and commit hash to /etc/armbian-release file
- added synaptic package manager and on-board keyboard to the desktop base
- added “logout” to the panel/menu
- added normal users to additional groups: disk tty users games
- updated all kernels with upstream
- updated ATF and bootloader on Espressobin, supporting all versions

Build script:

- added mirrors for speed-up building in China mainland
- added support for download compilers and rootfs cache via torrent network
- added new output image compression option (xz)
- enabled Debian Buster and Ubuntu Disco (unsupported) targets
- few Docker building improvements, caching image
- replace curl with aria2
- Linaro compilers update to 2019.02

Armbian-config:

- added Gimp installation
- added enable/disable Avahi
- updated OMV installer, OMV5 preparations
- enable screen resolution changer for Odroid N2
- enable CPU speed and governor adjustment

7.2.48 v5.87 (2019-05-26)

- added support for Odroid N2, Nanopi R1, Nanopi Duo2
- enabled nightly images for Orangepi3, One+, Lite2, PineH64, Rock64pro, RockPi4b
- enabled nigtly Buster and Disco images for Le Potato
- recompiled all images and pushed update where updates are known to work (sunxi, sunxi64, meson64, ...)
- improved SATA write speed on A20 chips for up to 300%
- fixed thermal throtling for H5 devices
- mainline u-boot moved 2019.04
- most development kernels moved to 5.1.y
- added separate DT for espressobinv7, updated boot loader

© 2021 by Armbian

- enable WoL for eth0 on Helios4

Build script:

- added Debian Buster and Ubuntu Disco (WIP)
- improved building under Docker. Source code is not copied to docker image, caching image
- Linaro compilers update to 2019.02
- fixed incomplete cleaning of the source code

Armbian-config:

- fixed kernel changing
- fixed sources download
- fixed Hass.IO and TVheadend install
- added menu driven CPU frequency/governor adjustement
- improved two-factor authentication
- added meson64 and rockchip to overlay/hardware configuration
- improved hostapd management

Infrastructure:

- main download server has been hooked to 10GbE connection.
- added web/http seeds to torrent download. Torrent download could/should fully utilize your download capacity.
- major forum upgrade (v4.4.3)
- added another IPV6 capable EU mirror https://mirrors.dotsrc.org

7.2.49 v5.76 (2019-02-11)

- remove Exagear Desktop

7.2.50 v5.75 (2019-02-10)

- added updated driver for Realtek 8811, 8812, 8814 and 8821 chipsets
- added Wireguard support to remaining kernels (except lower than 3.10)
- images rebuild with latest upstream sources, mainline u-boot was bumped to 2018.11

7.2.51 v5.74 (2019-01-31)

- fixing systemd related bug found in sunxi legacy 3.4.y kernels

7.2.52 v5.73 (2019-01-29)

- much faster nand-sata-install operations. Thanks to @dedalodaelus
- added support for @wireguard on all kernels higher than 3.10.y
- fixed drivers for popular DVB tuner S960 (all kernels)
- fixed bug in wireless drivers on Cubietruck, BananpiPRO, Bananapi+
- fixed AP mode on Orangepi PC+, Prime, One, .. when using kernel 4.19.y
- added prolific USB-to-USB bridges in mvebu-next/dev
- added nftables masquerade in mvebu64-next
- added MD raid support for SUNXI64
- upgrade bugfix for Helios4

© 2021 by Armbian

- updated hostapd to 2.7
- fixed 1512MHz OPP on Renegade
- fixed DRM crashing for rockchip64
- mainline u-boot bumped to 2018.11 (update goes manually from nand-sata-install utility)
- added testing images for Orangepi RK3399 and Radxa Rockpi 4B

7.2.53 v5.72 (2019-01-16)

- added additional repository mirror (updated armbian-config)
- fixed Tinkerboard DTB in repository and images rebuild

7.2.54 v5.71 (2019-01-16)

- updated images for Odroid C2, Lepotato and Nanopik2-S905 due to this bug

7.2.55 v5.70 (2019-01-12)

- sunxi-next and sunxi64-next were moved from 4.14.y to 4.19.y (remake of all AW images)
- better DVFS on H3/H5/A64, enabled higher cpu speed.
- added overlay support for Tinkerboard/rockchip next and kernel upped to 4.19.y
- updated next kernel for Odroid XU4 to 4.19.y
- updated next kernel for Odroid C2, Lepotato and Nanopik2-S905 to 4.19.y with overlay support
- fixed poweroff on H5
- H5/A64 lost experimental status,
- upgraded images and upstream/bugfix kernel upgrade for Rock64, Renegade,
- u-boot update is moved from automated to manual (armbian-config) to minimize boot related troubles
- added two repository mirrors: China and France (armbian-config -> Personal -> Mirror)
- changed switching to alternative kernels from armbian-config. It is possible to select a direct version and it only replaces kernel (safer)
- first official build for Olimex Teres
- mainline kernel builds for: Pine64, Pine64so, Olinuxino A64, OrangepiWin
- added more download variants for Rock64, Renegade, Tritium H3&H5
- updated images for Z28PRO, Bananapi PRO, Espressobin, Olimex Micro, Lime, Udoo, Bananapi M2, Bananapi M2U,

7.2.56 v5.68 (2018-12-30)

- updated Espressobin images, kernel updated to 4.19.y

7.2.57 v5.67 (2018-11-26)

- updated Helios4 images
- added experimental mainline kernel images for Pinebook and Pinebook 1080p

7.2.58 v5.67 (2018-11-12)

- updated images for Bananapi R2 with eMMC install support.

© 2021 by Armbian

7.2.59 v5.66 (2018-11-08)

- added Mediatek MT7623 family.
- added images for Bananapi R2 with kernel 4.19.y without official support.

7.2.60 v5.66 (2018-11-07)

- removing Odroid C2 official support, drop its default 3.16.y kernel from build engine and merge with the meson64 family.
- attach meson64 dev to 4.19.y
- drop Udoo Neo completly, drop Udoo Quad default and dev kernel.
- Odroid XU4: drop kernel 3.10.y, default branch is upgraded to offical 4.14.y, next becomes vanilla 4.19.y

7.2.61 v5.65 (2018-11-06)

- Cubox-i/Hummigboard: drop kernel 3.14.y and move 4.14.y to default, next becomes 4.19.y, dev 4.19.y with a mainline u-boot

7.2.62 v5.64 (2018-10-09)

- updated images and packages for Helios4.
- added images for Nanopi Neo4.

7.2.63 v5.63 (2018-10-08)

- updated images for Helios4 with SPI booting support.
- updated armbian-config. Added advanced ZSH shell install with most used plugins and tmux.

7.2.64 v5.62 (2018-10-01)

- updated armbian-config

7.2.65 v5.61 (2018-09-26)

- updated armbian-config,
- fixed Chromium on Debian builds with a workaround. We are overwriting package with last known working one. It will show some error on startup which is safe to ignore. This workaround will fade out with Chromium upstream update.

7.2.66 v5.60 (2018-09-19)

Changes overview:

© 2021 by Armbian

General:

- Ubuntu Xenial was replaced with Bionic unless kernel was too old for the change,
- Debian Jessie becomes EOL and its building is not maintained anymore while you will still receive kernel updates,
- Emergency swap file creation is disabled by default since we use compressed memory (ZRAM) as an alternative,
- `vm.swappiness` has been changed from 0 to 100 (if you run databases on your board you might want to revert this change in `/etc/sysctl.conf`),
- RAM logging also uses ZRAM now and rotates logs automagically,
- all images were rebuilt, except boards for which support ended,
- significantly lighter - browser only - desktop images (< 1.5G),
- fixed hanging on headers installation,
- install boot script (BSP package) if not present. This fixes upgrade or kernel switching problems,
- Proper bind mount directory when installing to SATA/USB and booting from SD,
- update for wireless drivers 8812/11/14AU, 8188EU and AUFS,
- Bugfix when a temperature is not present or readings are invalid,
- Also showing bridge IP addresses in MOTD,
- storing package list compressed - saves 50-70Mb,
- enlarging automated apt-get update and purge intervals,
- smaller overhead for CLI images,
- improved alternative kernel switching,
- stop setting Google’s DNS server as default for privacy reasons.

Family:

- sunxi and sunxi64, u-boot was bumped to 2018.05, NEXT branch was updated to the latest 4.14.y, DEV is attached to 4.18.y + fixed overlay support,
- mvebu64, default BSP kernel was upgraded to 4.14.y, NEXT to 4.18.y,
- odroidc1, experimental NEXT kernel branch was attached to 4.18.y,
- odroidc2 kernel was merged with meson64 on the source level,
- meson64 u-boot was pushed to 2018.05, a default was updated to the latest 4.14.y, NEXT to 4.18.y,
- rk3288, u-boot was pushed to 2018.05, legacy kernel cleaned and fixed after upstream troubles, NEXT branch was updated to the latest 4.14.y,
- rockchip64, rk3399 was split into rk3399 for Friendly ARM boards and rockchip64 for Rock64 and RockPro, Ayufan repository. Merging is postponed for the future,
- s5p6818 family support added NEXT branch was updated to the latest 4.14.y,
- mvebu, NEXT branch was updated to latest 4.14.y, DEV attached to 4.18.y,
- fixed randomly failing X server on imx6 family,

© 2021 by Armbian

Board:

- added WIP support for Firefly RK3399, Lime A64, Renegade, Rockpro64, Olimex Teres
- added experimental images for Bananapi M3 and Cubietruck+,
- adeed support for: Tinkerboard S, Rock64, Orangepi Zero Plus, Nanopi Neo Core2, Nanopi M4,
- added NEO 1.1 regulator overlay,
- added Helios4 device tree with FAN control for modern kernel,
- enabled SPI access on Espressobin,
- updated SPI boot firmware on Espressobin (18.09.1) with many fixes and support for booting from USB, SATA, eMMC or SD,
- added Tinkerboard S DC-IN voltage to armbianmonitor,
- fixed network interface initialization,
- fixed clock drift on Bananapi boards,
- enabled concurrent AP/STA mode on Tinkerboard,
- improved support for NanoPi Fire 3 (added SPU1705, DVFS, thermal tables, etc.),
- fixed network crashing on high load. Affected: Odroid C1/C2, Le Potato kernel 4.18.y,
- fixed wireless, eMMC and Bluetooth on (unsupported) Z28 PRO and changed boot order,
- fixed eMMC install on NanoPC T3+ and Docker dependencies on Fire3, M3, NanoPC T3+,
- added eMMC and DVFS support on Espressobin mainline kernel,
- ported Tinkerboard UMS to modern u-boot,
- enabled 1392 MHz cpufreq OPP on all RK3328 devices,
- enabled 1992/1512MHz cpufreq OPP on all RK3399 devices,
- added eMMC to OlinuXino A64 kernel and u-boot,
- added Sunvell R69 CSC target,
- OrangepiWin: fixed BT,
- fixed ethernet on (unsupported) Bananapi M64.

Build script:

- changed recommended build host to Bionic, Xenial still supported for everything except building Bionic images,
- added support for burning image directly to SD card when your build is done by using Etcher for CLI,
- added support for making LUKS encrypted root images, parameters: CRYPTROOT_ENABLE=yes, CRYPTROOT_PASSPHRASE=unlockpass,
- fixed building under Docker, bumped to Bionic host,
- added building Bionic and block building it for images with too old kernels,
- added multibranch support (LIB_TAG).

Infrastructure:

- build machine main SSD and memory upgrade, switched from bare metal Ubuntu Xenial to fully optimsed Debian KVM server, free build capacity is avaliable for any armbian related activity upon request,
- download server drive capacity and download speed upgrade, IPV6,
- geo load balancing for repository and download server is under testing,
- improved repository management. Possibility to add packages via Github,
- introducing new internal parameter, example: BUILD_ALL=”yes” REBUILD_IMAGES=”bananapi,udoo,rock64” to specify which images need rebuilding,
- main torrent server cleanup, removed deprecated images,
- creating report https://beta.armbian.com/buildlogs/report.html when building all kernels. Prepared to include simple per board testing report where exists https://github.com/armbian/testings.

© 2021 by Armbian

Known bugs:

- modern kernel support on A64 boards is mainly broken.

7.2.67 v5.59 (2018-08-18)

- rebuilt images for Espressobin with kernel 4.18.y, Nanopc T4

7.2.68 v5.58 (2018-08-13)

- rebuilt images for Bananapi, Bananapi Pro, Bananapi+, Odroid C2, Odroid XU4
- updated repository for Odroid C2/XU4, changed NEXT from 4.9.y to 4.14.y

7.2.69 v5.58 (2018-08-13)

- rebuilt images for Bananapi, Bananapi Pro, Bananapi+

7.2.70 v5.57 (2018-08-11)

- added Bionic desktop and Stretch CLI images for RK3399 powered Nanopc T4

7.2.71 v5.56 (2018-08-10)

- rebuilt images for Pinebook. Added Bionic build

7.2.72 v5.55 (2018-08-09)

- rebuilt images for Orangepi One+, Orangepi Lite 2 and Pine H64. Enabled USB3, network, THS, DVFS, higher frequencies, HDMI on 4.18.y DEV branch images.

7.2.73 v5.55 (2018-08-03)

- added Stretch and Bionic mainline kernel images for Odroid C1 (testing),
- rebuilt images for Bananapi M3 (fixed ethernet)

7.2.74 v5.54 (2018-07-25)

- updated images for Odroid C2, Nanopi M3, Nanopi Fire 3 and NanoPC T3+, Espressobin, Cubox-i/HB and Le potato
- added preview images without end user support for Bananapi M3,Cubietruck+ and Bananapi M2 Berry.

7.2.75 v5.53 (2018-07-23)

- Z28PRO images updated. Fixed wireless and Bluetooth
- FriendlyARM Nanopi K2 S905 images updated. Fixed ethernet problems.
- FriendlyARM Nanopi K1+ images updated. Fixed HDMI out and wireless

7.2.76 v5.51 (2018-07-04)

- Helios4 Stretch and Bionic images update

7.2.77 v5.50 (2018-06-28)

- Espressobin images rebuild and repository update, default 4.4.138, next 4.17.3, dev 4.18.RC, hardware crypto support in 4.17.y, zram and zswap

© 2021 by Armbian

- Odroid C2 bugfix update

7.2.78 v5.49 (2018-06-28)

- Amlogic Meson64 family (Odroid C2, Lepotato and FriendlyARM K2 S905) were merged into one kernel. Default images comes with kernel 4.14.52, next with 4.17.3 and DEV with 4.18.RC, updated boot scripts, implemented latest kernel bug fixes
- updated kernels, desktop packages and armbian config on the stable repository (apt update & upgrade)

7.2.79 v5.48 (2018-06-26)

- added nightly images for Odroid C2 with 4.16.y (NEXT) and 4.18.y (DEV) and hopefully fixed ethernet driver

7.2.80 v5.47 (2018-06-22)

- Odroid C2 images rebuild. Legacy kernel was upgraded to 3.16.57, next to 4.14.51, u-boot to 2018.05
- Added Tritium H5

7.2.81 v5.46 (2018-06-20)

- Added Olimex Teres nightly builds
- Added FriendlyARM Nanopi K1 plus

7.2.82 v5.46 (2018-06-06)

- Added Orange Pi Lite 2 and One plus nightly builds

7.2.83 v5.45 (2018-05-23)

- Orangepi Zero+ images rebuild

7.2.84 v5.44 (2018-05-10)

- Espressobin images were rebuilt and moved under stable. Kernel 4.14.40, Stretch, Xenial and Bionic. Fixed bootloader, ath10 wireless card support
- added initial Bionic storage to the main apt repository
- Cubox-i / Hummingboard bugfix update to 4.16.y and images rebuild
- Odroid C2 images rebuild

7.2.85 v5.41 (2018-02-10)

- fixed LED driver on Helios4
- bugfix update on sunxi/sunxi64 kernel. Updated to 4.14.18
- kernel update for MVEBU next (4.14.18 and default 4.4.115) for Clearfog and Helios4. Upstream fixes,AUFS and Realtek 881yAU drivers update

7.2.86 v5.40 (2018-02-05)

- fixed eMMC support on Odroid C2 NEXT, kernel 4.14.y
- updated PWM driver on Helios4
- kernel update for MVEBU next (Clearfog, Helios4)

© 2021 by Armbian

7.2.87 v5.38 (2018-01-29)

- updated all images
- added H3/H5 testing images with kernel 4.14.y
- added Nanopi M3/T3+/Fire testing image
- fixed Bluetooth on Orangepi Win
- main repository update with recent kernel on all NEXT builds

7.2.88 v5.37 (2018-01-23)

- bugfix release
- armbianmonitor -u fix
- setting cronjob permissions
- replace broken u-boot packages on A20 boards
- updated utilities: hostapd, sunxi-tools, armbian-config
- updated images: Bananapi, PRO, M2, BeelinkX2, Clearfog,Cubieboard2, Cubietruck, Cubox-i/HB, Espressobin, Helios4

7.2.89 v5.36 (2017-12-03)

- bugfix release

7.2.90 v5.35 (2017-11-25)

- mainline kernel updated to 4.13.y
- mainline u-boot updated to v2017.09
- added new sunxi Device Tree overlays, fixed and improved old overlays
- Micro-USB g_serial console is enabled by default on most small Allwiner based boards
- Olimex Lime2 and Micro: merging eMMC and normal versions
- Odroid C2: next and dev branches migrated to mainline u-boot
- Odroid XU4: added dev branch, next branch migrated to mainline u-boot
- Clearfog: added dev branch with mainline u-boot
- added supports for 7” RPi display to Tinkerboard with legacy kernel
- All mainline kernels: added Realtek 8811AU/8812AU/8814AU USB wireless driver with monitor mode and frame injection
- All boards: added kernel source packages to the repository (Package names `linux-source-${BRANCH}-${LINUXFAMILY}`, i.e. `linux-source-sunxi-next`)
- Kernel headers are no longer installed by default to new images
- Additional out of tree drivers and USB Redirector are no longer installed by default to new images
- Switching from emergency swap to zram on new Ubuntu Xenial images
- New hardware support (stable/supported images): NanoPi Duo, Orange Pi R1, Pinebook
- New hardware support (experimental): Le Potato, NanoPi NEO 2, Orange Pi Zero Plus, Orange Pi Zero Plus 2 (H5)
- sunxi mainline u-boot: reenabled USB keyboard support and disabled stopping the boot sequence with any key - autoboot now can be aborted with

© 2021 by Armbian

Desktop images:

- xterm was replaced with full featured xfce terminal,
- added memory profile caching for Chromium,
- added OpenVPN connector,
- shortcuts to armbian-config, support and donate were moved to menu,
- default icon theme was changed to lighter one (Numix),
- fixed login greeter theme,
- changed wallpaper.
- changed CMA handling on Allwinner legacy kernels

armbian-config:

- was splitted from board support packages to a new package `armbian-config`
- managing board hardware configurations, hotspot, Bluetooth, SSH server
- freezing/unfreezing kernel upgrade
- switching between stable and beta builds,
- switching between alternative kernels,
- installing/uninstalling kernel headers,
- changing timezone, locales, hostname,
- running diagnostic tools,
- enabling/disabling RDP server,
- 3rd party software installer: Samba, OMV, Pi hole, Transmission, ...

Build script:

- added Debian Stretch
- most tweaks moved from inline files to separate files in board support package
- firmware blobs moved to a separate repository
- disabled distcc in extra software compilation process due to toolchain compatibility issues

Known problems

- Allwinner A20/sun7i legacy boards. Changed CMA settings prevents playing video. You need to add cma=96M to kernel command line

7.2.91 v5.34 (2017-10-18)

- bugfix Odroid XU4/HC1 image rebuild due to broken USB install on kernel 4.9.x
- added Le Potato and Orange Pi Zero testing image (mainline kernel)
- Tinkerboard, MiQi and Pinebook images rebuilt

7.2.92 v5.33 (2017-09-24)

- Odroid XU4/HC1 images were rebuilt.

7.2.93 v5.33 (2017-09-21)

- Tinkerboard and MiQi images were rebuilt. Rockchip legacy kernel was updated to 4.4.88 and mainline (NEXT) to 4.13.3.

© 2021 by Armbian

7.2.94 v5.32 (2017-06-23)

- bugfix release due to broken crypto functions on kernel 4.11.x

7.2.95 v5.31 (2017-06-15)

- bugfix release due to network failure on some A10 / A20 boards

End of support notice

Following boards are no longer receiving support and updates since this version:

- Cubieboard (Allwinner A10) - not enough hardware samples to maintain support
- Lamobo R1 (Allwinner A20) - hardware design flaws, poor software support for the onboard switch
- Olimex Lime A10

7.2.96 v5.30 (2017-06-14)

- mainline kernel updated to 4.11
- mainline u-boot updated to v2017.05
- Firefox was replaced with Chromium (desktop images)
- sunxi mainline configuration: added Device Tree overlays support (new images only)
- sunxi mainline configuration: added `armbian-add-overlay` helper for compiling and activating DT overlays (new images only)
- log2ram: fixed saving `/var/log` contents on shutdown
- new hardware support (stable/supported images): Xunlong Orange Pi Zero Plus 2 (H3), ASUS TinkerBoard, MiQi
- reworked package updates MOTD script to speed up the login process
- added config file `/etc/default/armbian-motd` for disabling MOTD components
- added `armbian-config` dialog-based configuration program (WIP)
- Banana Pi M2: fixed HDMI video output
- Clearfog: adjusted temperature readout
- i.MX6 mainline: enabled support for HDMI audio and PCIe bus

End of support notice

Following boards are no longer receiving support and updates since this version:

- Orange Pi (Allwinner A20) - no hardware samples, out of stock
- Orange Pi Mini (Allwinner A20) - no hardware samples, out of stock
- LeMaker Guitar (Actions S500)
- Roseapple Pi (Actions S500)

7.2.97 v5.26, v5.27 (2017-02-24)

- security update for most kernels (packages only)
- fixes for hostapd configuration

7.2.98 v5.25 (2017-02-02)

- nand-sata-install expanded functionality: you can partition destination and choose file-system type: ext2, ext3, ext4 and BTRFS (BTRFS requires kernel 4.4+)
- added new boards: Clearfog Base, Lime2 eMMC, Lime A33, NanoPi M1+, OrangePi Zero, OrangePi PC2 (mainline only, experimental)

© 2021 by Armbian

- new default kernel for Clearfog(s), changed kernel family to “mvebu” to avoid conflicts
- disabled wireless power management by default to improve performance with certain drivers
- added wireless drivers to mainline kernels: OrangePi Zero, Neo Air
- implemented initrd loading support for all boards
- moved all images to single ext4 partition scheme
- changed default wallpaper, startup icon, shadows to windows on desktop builds
- Firefox web cache moved to memory
- added g_serial driver to boards without a network connector, working on both kernel (Opi Zero,Opi Lite,FA Neo Air)
- added “Software boutique” application installer on desktop builds (currently not working properly on arm64)
- added per board patching option
- added u-boot video driver and boot logo to H3 based boards
- added simplefb video driver (HDMI only) to mainline H3 kernel
- updated MALI driver on H3 platform, fixed problems on 2GB boards
- changed Ethernet switch driver on Lamobo R1 to DSA based one (mainline kernel)
- fixed soft cursor (CLI) for H3 legacy and Odroid C2
- expand and adjust multiple kernel configurations based on user requests
- adjusted sunxi boot script to support booting in SPI flash + USB storage scenario (w/o the SD card)
- dropped support for Debian Wheezy and Ubuntu Trusty releases
- sunxi mainline kernel was updated to 4.9.x, some dev kernels to 4.10
- added log2ram (Ramlog alternative) to default installation
- changed first run logic, disabled forced automatic reboot
- changed new user account creation logic, disabled forced reboot on user creation failure

7.2.99 v5.24

- this version is not released, it was used for the nightly or user-built images

7.2.100 v5.23 (2016-10-23)

- fixed bug in nand-sata-install
- fixed u-boot update bug on Allwinner platform

Known problems:

- Lamobo R1 fails to boot upon upgrade

7.2.101 v5.22 (2016-10-22)

- fixed eMMC install on Odroid C2
- firmware package was splitted into minimal (default) and full versions
- patched Dirty COW exploit on all kernels
- added Odroid XU4 mainline kernel image
- added Olimex A33 mainline kernel image
- added Overlay FS for Cubox, Udoo and Udoo Neo
- booting problems fixed on more boards
- updated wireless driver on M2+ (dhd)
- updated driver for OV5640 on sun8i default kernel

© 2021 by Armbian

- sunxi-next kernel version updated to 4.8.4
- BananaPi M1+ now uses upstream DTB file `sun7i-a20-bananapi-m1-plus.dtb`, boot script adjusting may be required for existing images

Desktop images:

- prebuilt mpv and FFmpeg were removed in favor of providing only configuration files
- fixed an issue with video brightness on A10/A20 based boards

Build script:

- DEBUG_MODE was renamed to CREATE_PATCHES
- GLshim was moved to a private directory, it can be activated for selected applications by changing `LD_LIBRARY_PATH`

Known problems:

- eMMC install fails (will be fixed in bugfix update)
- H3 development kernel (4.8.4) update fails to boot
- C2 upgrade hangs on compiling headers (Jessie)

7.2.102 v5.20 (2016-09-16)

- added FriendlyARM Neo legacy and mainline images (experimental)
- added Orange Pi PC+ mainline kernel (experimental)
- added Pine64 / Pine64+ images with legacy kernel
- added UUID support for NAND/SATA/USB installer
- added desktop images for Cubox(s) / Hummingboard(s) with mainline kernel
- enabled MIDI sequencer and snd-rawmidi-seq in H3 legacy kernel
- added H3 consumption tool to control board consumtion level on legacy kernel
- fixed and enabled Bluetooth on Cubietruck and Cubox(s) / Hummigboard(s) desktop, both kernels
- masked p2p0 wifi direct device on Bluetooth legacy kernel
- Odroid C1/C2 upgrade fail fixed
- wireless enabled by default on Banana Pi PRO
- added new screen resolutions to H3 boards with legacy kernel
- DeviceTree Overlay ConfigFS interface for H3 mainline kernel
- update of mainline u-boot to 2016.09 should fix boot failures on H3 boards with eMMC
- disabled USB keyboard support in mainline u-boot should fix boot failures with connected USB devices

Desktop images:

- WICD was replaced with NetworkManager
- ALSA was replaced with PulseAudio
- sunxi boards: GLshim was added to desktop images with Mali support (except for Orange Pi Plus and Orange Pi Plus 2e)
- sunxi boards: prebuilt mpv now supports OSD and subtitles, activated by setting environment variable `VDPAU_OSD=1`

© 2021 by Armbian

Build script:

- complete desktop building rework - now packages are built from sources
- added Lime 2 eMMC as build target (WIP)
- added Pine64 / Pine64+ mainline (dev) target (experimental)
- added FriendlyArm Neo as build target
- fixed MT7601 wifi driver building
- github download rework
- external toolchain rework

Added additional packages, not installed by default:

- hostapd-realtek: replacement for hostapd with support for several Realtek Wi-Fi adapters
- fswebcam-gc2035: replacement for fswebcam with support for GC2035 camera driver for H3 based boards
- guvcview: replacement for stock guvcview with support for H3-based Orange Pi CMOS cameras

Known problems:

- Mali OpenGL ES does not work on H3 boards with 2GB RAM (Orange Pi Plus 2, Orange Pi Plus 2e)
- Hardware video decoding on A10/A20 based boards produces dark video
- Some applications that depend on livav libraries (i.e. minidlna) may not work on Jessie images

7.2.103 v5.17 (2016-07-07)

- bugfix release on some boards.

7.2.104 v5.16 (2016-07-05)

- bugfix release. In 5.15 we accidentaly overwrote default network settings. Check /etc/network/interfaces if you use advanced network settings or fixed ip.
- small changes.

7.2.105 v5.15 (2016-07-01)

- Added improved camera driver for Xunlong’s cheap 2MP GC2035 camera
- Improved throttling/DRAM settings for the new 3 overheating H3 devices (BPi M2+, NanoPi M1, Beelink X2)
- Added official support for Beelink X2, NanoPi M1, Banana Pi M2+
- Improved console output (serial + display)
- Finally got rid of (broken) board auto detection. We do not ship any more one image for several devices that tries to detect/fix things on 1st boot but provide one dedicated image per board (Plus and Plus 2 and both NanoPi M1 variants being handled as the same device since only size of DRAM/eMMC differs)
- Tried to improve user experience with better/unified led handling (light directly after boot, communicate booting states through blinking)
- Improve partitioning and filesystem resize on 1st boot making it easier to clone every installation media afterwards
- fully support installation on eMMC on all H3 devices (`u-boot` and `nand-sata-install.sh` fixes)
- Improved performance/thermal/throttling behaviour on all H3 boards (especially newer Oranges)
- Prevent HDMI screen artefacts (disabling interfering TV Out by default)
- Enhanced 8189ETV driver for older Oranges
- Added support for OPi Lite, PC Plus and Plus 2E including new 8189FTV Wi-Fi (client, AP and monitoring mode, added fix for random MAC address)

© 2021 by Armbian

- Added in-kernel corekeeper patch (bringing back killed CPU cores after heavy overheating situations when thermal situation is ok again)
- Added TV Out patch for Orange Pi PC
- Further improve driver compilation due to improved kernel headers scripts compilation
- Initrd support
- increased kernel version to 3.4.112
- Exchanged whole kernel source tree to newer BSP variant, cleaned up sources, rebased all +100 patches (fixed display issues and kswapd bug, new and more performant GPU driver, increase Mali400MP2 clock to 600MHz)
- Added RTL2832U drivers to kernel (DVB-T)
- Fixed Docker on Odroid XU4
- Added overlay fs to Clearfog and Odroid XU4
- Many minor fixes

7.2.106 v5.14 (2016-06-14)

- all images rebuilt, most of them were manually tested
- added Beelink X2 image
- Cubox / Hummingboard kernel upgrade to 3.14.72 and 4.6.2
- Trusty was replaced with Xenial

7.2.107 v5.12 (2016-05-31)

- updated C1 images
- added wifi driver for new Oranges (modprobe 8189fs)
- added Orange Pi Lite, PC Plus and Plus 2E images

7.2.108 v5.11 (2016-05-24)

- Various bug fixes
- new working images for Actions Semi S500 boards

© 2021 by Armbian

7.2.109 v5.10 (2016-05-01)

Images:

- all 3.10+ kernels are Docker ready
- all A10/A20/H3 comes with HW accelerated video playback in desktop build
- fixed root exploit on H3 boards
- fixed kswapd 100% bug on H3 boards
- fixed SPDIF / I2S audio driver in legacy kernel
- fixed Udoo Neo wireless
- fixed slow SD cards boot
- fixed Allwinner SS driver
- fixed bluetooth on Cubietruck, both kernels
- fixed wireless driver on H3 boards
- fixed R1 switch driver
- kernel for Allwinner boards was upgraded to 3.4.112 & 4.5.2
- kernel for iMx6 boards was upgraded to 3.14.67 & 4.5.2
- kernel for Armada (Clearfog) was upgraded to 3.10.101 & 4.5.2
- kernel for Udoo boards was updated to 3.14.67 & 4.4.8
- kernel for Guitar (Lemaker) was upgraded to 3.10.101
- kernel for H3/sun8i legacy come from new Allwinner updated source (friendlyarm)
- added support for Olimex Lime2 eMMC
- increased MALI clockspeed on sun8i/legacy
- added Armbianmonitor
- added Odroid C1, C2(arm64), Nanopi M1, Banana M2+, Pcduino 2 and Pcduino 3. CLI and desktop
- added wifi radar to desktop
- added preview mainline kernel images for H3 boards (4.6.RC1)
- added initrd creation on all Allwinner images
- added Hummigboard 2 with working PCI and onboard wireless with legacy kernel 3.14.65
- added eMMC installer for H3
- added support for IFB and net scheduling for sun7i-legacy
- added ax88179_178a USB 3.0 Ethernet driver for sun7i-legacy
- hostapd comes as separate package (armbian-hostapd)
- changed first boot procedure and force user creation
- verbose / no verbose boot works almost on all boards
- enabled I2S on sun8i
- removed Debian Wheezy from auto build
- installing headers autocompile scripts
- all images come compressed with 7zip

© 2021 by Armbian

Build script:

- GCC 5 support for mainline and allwinner legacy
- RAW images are not compressed by default
- added arm64 building support
- added docker as host
- Added Belink X2 (H3 based media player), and Roseapple (S500) as WIP target
- introducted CLI_TARGET per board
- prepared FEL boot
- prepared Xenial target
- fixed USB redirector building on all kernels
- support for Xenial as a build host is 95% ready.
- implemented automatic toolchain selection
- come cleanup, configurations are subfoldered
- extended_deboostrap becomes default

Known bugs:

- Udoo Neo reboots takes a while, 1min+
- headers within sun8i needs some fixing
- H3 board autodetection fail under certain conditions

7.2.110 v5.06 (2016-03-18)

- increase kernel version to 3.4.111
- headers auto creation while install (eases kernel/driver compilation)
- improved SD card partitioning to help old/slow cards with wear leveling and garbage collection
- Possible to use *Ubuntu Xenial Xerus* as target
- changed behaviour of board leds (green == power, red == warning)
- speed improvements for 1st automated reboot
- Integrates OverlayFS backport

7.2.111 v5.05 (2016-03-08)

- Auto detection for the Orange Pi 2 does work now
- Mali acceleration works for all users not only root
- verbose boot logging on 1st boot and after crashes (you can toggle verbose logging using `sudo armbianmonitor -b`)
- more WiFi dongles supported due to backported firmware loader patch
- all 3 USB ports on Orange Pi One (Lite) available (2 of them need soldering)
- I2S possible on all Orange Pis (compare with the mini tutorial since you need to tweak script.bin)
- default display resolution set to 720p60 to fix possible overscan issues on 1st boot
- HW accelerated video decoding works for most formats
- Booting from eMMC on OPi Plus now possible
- Udoo quad images upgraded to 4.4.4

7.2.112 v5.04 (2016-03-01)

- HDMI/DVI works (bug in boot.cmd settings)

© 2021 by Armbian

- Reboot issues fixed (bug in fex settings)
- 1-Wire useable (we chose to stay compatible to loboris' images so the data pin is 37 by default. You're able to change this in the fex file)
- changing display resolution and choosing between HDMI and DVI is now possible with the included *h3disp* tool (should also work in the stand-alone version with Debian based OS images from loboris/Xunlong). Use `sudo h3disp` in a terminal to get the idea.
- Ethernet issues fixed (combination of kernel and fex fixes)
- USB-to-SATA bridge on the Orange Pi Plus works
- stability problems on Orange Pi One fixed (due to undervoltage based on wrong fex settings)
- problems with 2 USB ports on the PC fixed (wrong kernel config)
- Mali400MP acceleration (EGL/GLES) works now
- suspend to RAM and resume by power button works now (consumption less than 0.4W without peripherals)
- Enforce user account creation before starting the GUI
- USB and Ethernet IRQs distributed nicely accross CPU cores
- Full HDMI colour-range adjustable/accessible through *h3disp* utility
- already useable as stable headless/server board
- rebuilt Cubieboard 1 & 2 with 3.4.110 and 4.4.3
- fixed Bluetooth on Cubietruck + rebuild with 3.4.110 and 4.4.3
- all new images has no login policy: forced user generation

7.2.113 v5.03 (2016-02-20)

- H3 images rebuilt

7.2.114 v5.02 (2016-02-18)

- H3 images rebuilt

7.2.115 v5.01 (2016-02-17)

- Bugfix update for Allwinner boards
- Update for H3 based boards

7.2.116 v5.00 (2016-02-12)

- mainline kernel for Allwinner based boards upgraded to 4.4.1
- Allwinner audio driver playback and capture on kernel 4.4.1, UAS, USB OTG, battery readings,
- added Marvel Armada kernel 3.10.96, 4.4.1 and patches for changing mPCI to SATA
- added Cubox / Hummingboard kernel 4.4.1 (serial console only)
- firstrun does autoreboot only if needed: wheezy and some legacy kernels.
- added motd to /etc/updated.motd ... redesign, added battery info for Allwinner boards, bugfix, coloring
- fixed temperature reading on Cubox / Hummingboard legacy kernel
- fixed FB turbo building on Allwinner
- fixed NAND install on A10 boards (Legacy kernel only)
- fixed USB boot, added PWM on mainline
- fixed Banana PRO/+ onboard wireless on mainline kernel - running with normal Banana DT.
- readded USB sound

© 2021 by Armbian

- added A13 Olimex SOM
- added LIRC GPIO receive and send driver for legacy Allwinner
- added LED MMC activity to mainline kernels for Cubietruck and Cubieboard A10
- build script: option to build images with F2FS root filesystem for Allwinner boards
- build script: added alternative kernel for Lemaker Guitar (NEXT), Cubox (DEV)

7.2.117 v4.81 (2015-12-28)

- complete build script rework
- new development kernel package linux-image-dev-sunxi (4.4RC6) for Allwinner boards
- added Lemaker Guitar, kernel 3.10.55
- added Odroid XU3/4, kernel 3.10.94 and mainline 4.2.8
- mainline kernel for Allwinner based boards upgraded to 4.3.3
- Udoo mainline upgraded to 4.2.8, legacy to 3.14.58
- cubox / hummingboard upgraded to 3.14.58, added mainline kernel 4.4
- fixed Jessie RTC bug, systemd default on Jessie images

7.2.118 v4.70 (2015-11-30)

- Bugfix update(apt-get update && apt-get upgrade)
- small changes and fixes

7.2.119 v4.6 (2015-11-24)

- Update only (apt-get update && apt-get upgrade)
- mainline kernel for Allwinner based boards upgraded to 4.2.6
- Legacy kernel for Allwinner based boards upgraded to 3.4.110
- added new board: Udoo Neo
- added USB printer, CAN, CMA, ZSWAP, USB video class, CDROM fs, sensor classs, … to Allwinner mainline kernel
- nand-sata-install scripts rewrite. Now it’s possible to install to any partition.
- fixed nand install for Allwinner A10 based boards: Cubieboard 1 / Lime A10
- universal upgrade script bugfix / rewrite.
- 8 channel HDMI support for legacy Allwinner kernel
- unattended upgrade fixed
- sunxi tools fixed
- added two new options to build script: keep kernel config and use_ccache
- added kernel version to motd

7.2.120 v4.5 (2015-10-14)

- mainline kernel upgraded to 4.2.3 for Allwinner based boards
- legacy kernel for Allwinner compiled from new sources (linux-sunxi)
- udoo mainline upgraded to 4.2.3
- cubox / hummingboard upgraded to 3.14.54
- changed kernel naming: A10 = linux-image-sun4i, A20 = linux-image-sun7i
- new boards: Banana M2, Orange+(A31S), Cubieboard 1, Cubieboard 2 Dual SD, Lime A10

© 2021 by Armbian

- fixed Udoo legacy wireless problems
- fixed Jessie boot problems by disabling systemd. It’s possible to re-enable within boot scripts
- added ramlog to Jessie because we don’t have systemd anymore
- changed wireless driver for Cubietruck and Banana PRO (now it’s ap6210)
- added ZRAM to mainline kernel
- fixed dvbsky modules

and a bunch of small fixes.

7.2.121 v4.4 (2015-10-01)

Images:

- mainline kernel upgrade to 4.2.2 (Allwinner, Udoo Quad),
- legacy kernel upgraded to 3.4.109 (Allwinner),
- added I2C support and bunch of multimedia modules (DVB) (mainline Allwinner),
- Udoo quad images with fixed legacy kernel 3.14.28,
- Cubox and Hummingboard kernel upgrade to 3.14.53,
- brcmfmac driver fixes for mainline kernel (Banana PRO / Cubietruck)
- performance tweak: choosing a closest Debian mirror (Debian images)
- added Astrometa DVB firmware and dvb-tools
- added Nikkov SPDIF / I2S recent patch (legacy Allwinner)
- added patch for rtl8192cu: Add missing case in rtl92cu_get_hw_reg (Lamobo R1)
- bigger NAND boot partition on install
- install script bug fixes

Script:

- force apt-get update on older rootfs cache,
- image harden manipulation security,
- packages NAND/FAT/same version install faling fixed,
- image shrinking function rework,
- better packages installation install checking,
- added Debian keys to suppress warnings in debootstrap process,
- added fancy progress bars,
- added whiptail downloading prior to usage (bugfix).

7.2.122 v4.3 (2015-09-17)

- kernel 4.2 for Allwinner based boards
- kernel 4.2 for Udoo Quad
- walk-around if ethernet is not detected on some boards due to RTC not set(?)
- update is done (semi) automatic if you are using Armbian 4.2. You only need to issue command: apt-get update && apt-get upgrade. If you are coming from older system, check Documentation
- U-boot on R1 is now updated to latest stable version (2015.07)
- Fixed AW SOM. Working with latest u-boot but you need to build image by yourself.
- Enabled whole USB net and HID section in kernel for Allwinner boards v4.2
- Fixed upgrade script – only some minor bugs remains.

© 2021 by Armbian

- Fixes to build script that it’s working under Ubuntu 15.04
- Adding Bananapi Wireless driver (ap6210) back to legacy kernel
- Udoo official kernel (3.14.28) not updated due too many troubles.

7.2.123 v4.2 (2015-09-01)

Images:

- Upgraded NAND / SATA installer. Possible to install to SATA/NAND boot in one step.
- Easy kernel switching between old 3.4 and 4.x
- Automatic kernel updating (to disable comment armbian repo /etc/apt/sources.list)
- Allwinner boards share one 4.x kernel and two 3.4
- All boards share the same revision number
- One minimal Ubuntu Desktop per board (Wicd, Firefox, Word)
- u-boot v2015.07 for most boards
- Aufs file system support
- kernel 4.1.6 and 3.4.108
- Added Orangepi Mini, Cubieboard 1 (4.x only), Udoo with official kernel
- Repository for Wheezy, Jessie and Trusty
- enabled USB audio in kernel 4.x
- kernel headers fixed. No need to rebuild when you update the kernel.
- fixed boot scripts that can load from FAT partition too
- removed Cubox binnary repository because of troubles
- Docker support (kernel 4.x). Already here for a while / forget to mention.
- nodm change default login

Build script:

- changed structure: sources now in folder sources, output is what we produce, deb in one folder
- expanded desktop part
- possible to build all images at once, create package repository
- SD card initial size is 4Gb, variable transfered into configuration.sh
- Avaliable board list is now created from file configuration.sh
- Fixed image shrinking problem
- Patching part rework
- Using first FAT boot partition now fixes boot scripts
- Uboot TAG moved to configuration.sh and differs for some boards
- new variables for source branches. Only too remove errors when checking out

7.2.124 v4.1 (2015-08-05)

- Added desktop image
- U-Boot 2015.07 with many new features
- Added auto system update via repository apt.armbian.com
- Root password change is initialized at first boot.
- 3.4.108 kernel fixes, 4.1.4 Allwinner Security System

© 2021 by Armbian

7.2.125 v4.0 (2015-07-12)

- Fixed stability issues, temperature display in 4.x
- Kernel upgrades to 3.4.108 and 4.1.2

7.2.126 v3.9 (2015-06-11)

- Bugfix release
- Kernel 4.0.5 traffic control support
- SATA / USB install fixed on kernel 4.x
- Added 256Mb emergency swap area, created automatically @first boot

7.2.127 v3.8 (2015-05-21)

- Bugfix release: Cubietruck images successfully booted on Cubietruck. I waited for automatic reboot than tested remote login.
- Kernel 4.0.4 added support for power on/off button
- Both: Jessie fixed, Ethernet init fixed (uboot)
- armbian.com introduction

7.2.128 v3.7 (2015-05-14)

- Kernel 4.0.3 some new functionality
- Kernel 3.4.107 added sunxi display manager to change FB on demand
- Both: Ubuntu and jessie install errors fixed, removed busybox-syslogd and changed to default logger due to problems in Jessie and Ubuntu, apt-get upgrade fixed, documentations update, Uboot fixed to 2015.4 – no more from dev branch
- Build script rework - image size shrink to actual size, possible to have fat boot partition on SD card, several script bug fixes

7.2.129 v3.6 (2015-04-29)

- Kernel 3.19.6
- Kernel 3.4.107 with better BT loading solution

7.2.130 v3.5 (2015-04-18)

- Kernel 3.19.4: fixed AP mode, fixed USB, added 8192CU module
- Common: apt-get upgrade ready but not enabled yet, serial console fixed, fixed hostapd under jessie, easy kernel switching, latest patched hostapd for best performance – normal and for realtek adaptors, auto IO scheduler script
- Build script: everything packed as DEB

7.2.131 v3.4 (2015-03-28)

- Kernel 3.19.3: docker support, apple hid, pmp, nfsd, sata peformance fix
- Kernel 3.4.106: pmp, a20_tp - soc temp sensor
- Common: console setup fixed, headers bugfix, nand install fix
- Build script: kernel build only, custom packets install, hardware accelerated desktop build as option

7.2.132 v3.3 (2015-02-28)

- Kernel 3.19.0: many new functionality and fixes.
- Bugfixes: CT wireless works in all kernels

© 2021 by Armbian

7.2.133 v3.2 (2015-01-24)

- Possible to compile external modules on both kernels
- Kernel 3.19.0 RC5
- Bugfixes: install script, headers, bashrc, spi

7.2.134 v3.1 (2015-01-16)

- Kernel 3.19.0 RC4
- Added Cubieboard 1 images
- Dualboot for CB2 and CT dropped due to u-boot change. Now separate images.
- New user friendly SATA + USB installer, also on mainline

7.2.135 v3.0 (2014-12-29)

- Kernel 3.18.1 for mainline image
- Added Ubuntu Trusty (14.04 LTS) image
- Bugfixes: auto packages update

7.2.136 v2.9 (2014-12-03)

- Kernel 3.4.105 with new MALI driver and other fixes
- Added: Jessie image
- Major build script rewrite - much faster image building
- Fixed: failed MIN/MAX settings

7.2.137 v2.8 (2014-10-17)

- Added: ondemand governor, fhandle, squashfs and btrfs
- Removed: bootsplash, lvm, version numbering in issue
- Fixed: custom scripts, Jessie upgrade
- Disabled: BT firmware loading, enable back with: insserv brcm40183-patch
- Added working driver for RT 8188C, 8192C

7.2.138 v2.7 (2014-10-01)

- Kernel 3.4.104
- Automatic Debian system updates
- VGA output is now default but if HDMI is attached at first boot than it switch to HDMI for good. After first restart!
- Fixed NAND install script. /boot is mounted by default. Kernel upgrade is now the same as on SD systems.
- Cubieboard2 - disabled Cubietruck dedicated scripts (BT firmware, LED disable)
- Added network bonding and configuration for “notebook” mode (/etc/network/interfaces.bonding)
- IR receiver is preconfigured with default driver and LG remote (/etc/lirc/lircd.conf), advanced driver is present but disabled
- Added SPI and LVM functionality
- Added Debian logo boot splash image
- Added build essentials package

© 2021 by Armbian

7.2.139 v2.6 (2014-08-22)

- Kernel 3.4.103 and 3.17.0-RC1
- Added GPIO patch (only for 3.4.103)

7.2.140 v2.5 (2014-08-02)

- Kernel 3.4.101 and 3.16.0-RC4
- major build script rewrite

7.2.141 v2.4 (2014-07-11)

- Kernel 3.4.98
- default root password (1234) expires at first login
- build script rewrite, now 100% non-interactive process, time zone as config option
- bug fixes: removed non-existing links in /lib/modules

7.2.142 v2.3 (2014-07-02)

- Kernel 3.4.96
- cpuinfo serial number added
- bug fixes: stability issues - downclocked to factory defaults, root SSH login enabled in Jessie, dedicated core for eth0 fix
- disp_vsync kernel patch

7.2.143 v2.2 (2014-06-26)

- Kernel 3.4.94
- Added Jessie distro image
- Updated hostapd, bashrc, build script
- bug fixes: disabled upgrade and best mirror search @firstboot, bluetooth enabler fix
- MD5 hash image protection

7.2.144 v2.1 (2014-06-13)

- Kernel 3.4.93
- Onboard Bluetooth finally works
- Small performance fix
- Allwinner Security System cryptographic accelerator

7.2.145 v2.0 (2014-06-02)

- Kernel 3.4.91 with many fixes
- Cubieboard 2 stability issues fix
- eth0 interrupts are using dedicated core
- Global bashrc /etc/bash.bashrc
- Verbose output and package upgrade @ first run

7.2.146 v1.9 (2014-04-27)

- Kernel headers included

© 2021 by Armbian

- Clustering support
- Advanced IR driver with RAW RX and TX
- Bluetooth ready (working only with supported USB devices)
- Bugfixes: VLAN, login script, build script
- New packages: lirc, bluetooth

7.2.147 v1.8 (2014-03-27)

- Kernel 3.4.79
- Alsa I2S patch + basic ALSA utils
- Performance tweaks: CPU O.C. to 1.2Ghz, IO scheduler NOOP for SD, CFQ for sda, journal data writeback enabled
- Avaliable memory = 2000MB
- Minimized console output at boot
- MAC address from chip ID, manual optional
- Latest (Access point) hostapd, 2.1 final release
- Login script shows current CPU temp, hard drive temp & actual free memory
- Fastest Debian mirror auto selection @first boot
- New packages: alsa-utils netselect-apt sysfsutils hddtemp bc

7.2.148 v1.7 (2014-02-26)

- Flash media performance tweaks, reduced writings, tmp & logging to RAM with ramlog app – sync logs on shutdown
- SATA install script
- Dynamic MOTD: Cubieboard / Cubietruck
- Disabled Debian logo at startup
- New packages: figlet toilet screen hdparm libfuse2 ntfs-3g bash-completion

7.2.149 v1.6 (2014-02-09)

- Added support for Cubieboard 2
- Build script creates separate images for VGA and HDMI
- NAND install script added support for Cubieboard 2

7.2.150 v1.52 (2014-02-07)

- Various kernel tweaks, more modules enabled
- Root filesystem can be moved to USB drive
- Bugfixes: NAND install script

7.2.151 v1.5 (2014-01-22)

- Hotspot Wifi Access Point / Hostapd 2.1
- Bugfixes: MAC creation script, SSH keys creation, removed double packages, …
- Graphics desktop environment upgrade ready

7.2.152 v1.4 (2014-01-12)

- Patwood’s kernel 3.4.75+ with many features

© 2021 by Armbian

- Optimized CPU frequency scaling 480-1010Mhz with interactive governor
- NAND install script included
- Cubietruck MOTD
- USB redirector – for sharing USB over TCP/IP

7.2.153 v1.3 (2014-01-03)

- CPU frequency scaling 30-1000Mhz
- Patch for gpio

7.2.154 v1.23 (2014-01-01)

- added HDMI version
- added sunxi-tools
- build.sh transfered to Github repository
- disabled LED blinking

7.2.155 v1.2 (2013-12-26)

- changed kernel and hardware config repository
- kernel 3.4.61+
- wi-fi working
- updated manual how-to

7.2.156 v1.0 (2013-12-24)

- total memory available is 2G (disabled memory for GPU by default)
- gigabit ethernet is fully operational
- sata driver enabled
- root filesystem autoresize
- MAC address fixed at first boot
- Kernel 3.4.75
- root password=1234
- Bugs: wifi and BT not working

© 2021 by Armbian

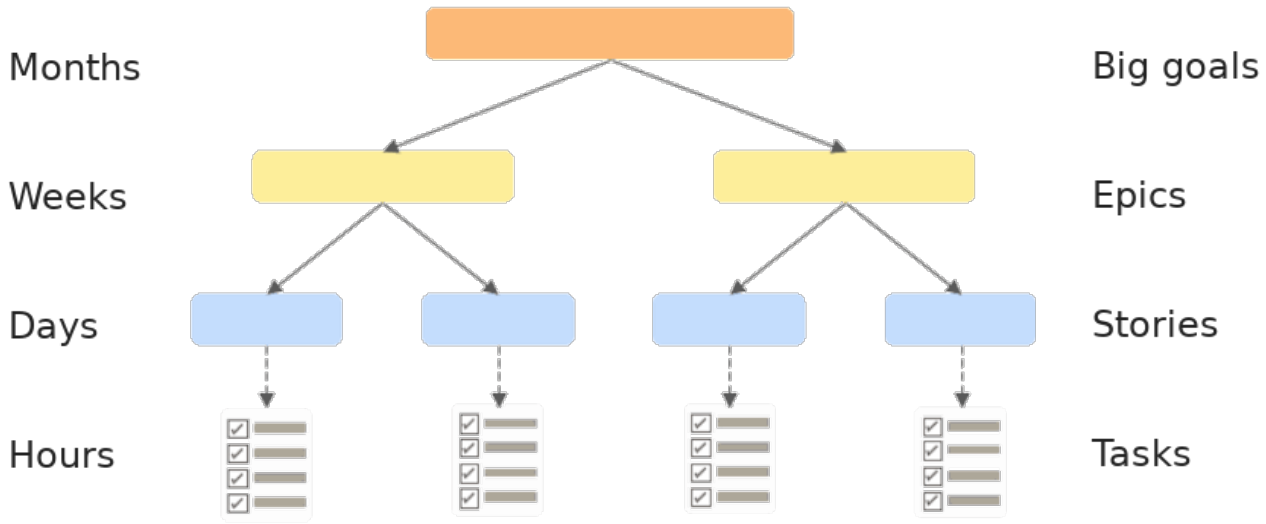
7.3 Jira

Jira where development work is entered and prioritized. https://armbian.atlassian.net/

7.3.1 Issue Types

When creating issues, try to assign issue type most appropriate. Issue type *can* be changed later so don't worry too much. If possible assign to a "Fix Version" aka Release.

- **Epic** - useful as placeholders for large requirements. Common objective, overall goals, contains several stories.
- **Story** - Smallest units of functionality that can be achieved in one or two weeks. Non-technical language.
- **Task** - Work that is clearly defined usually by people that will do the work. Specific, technical language.



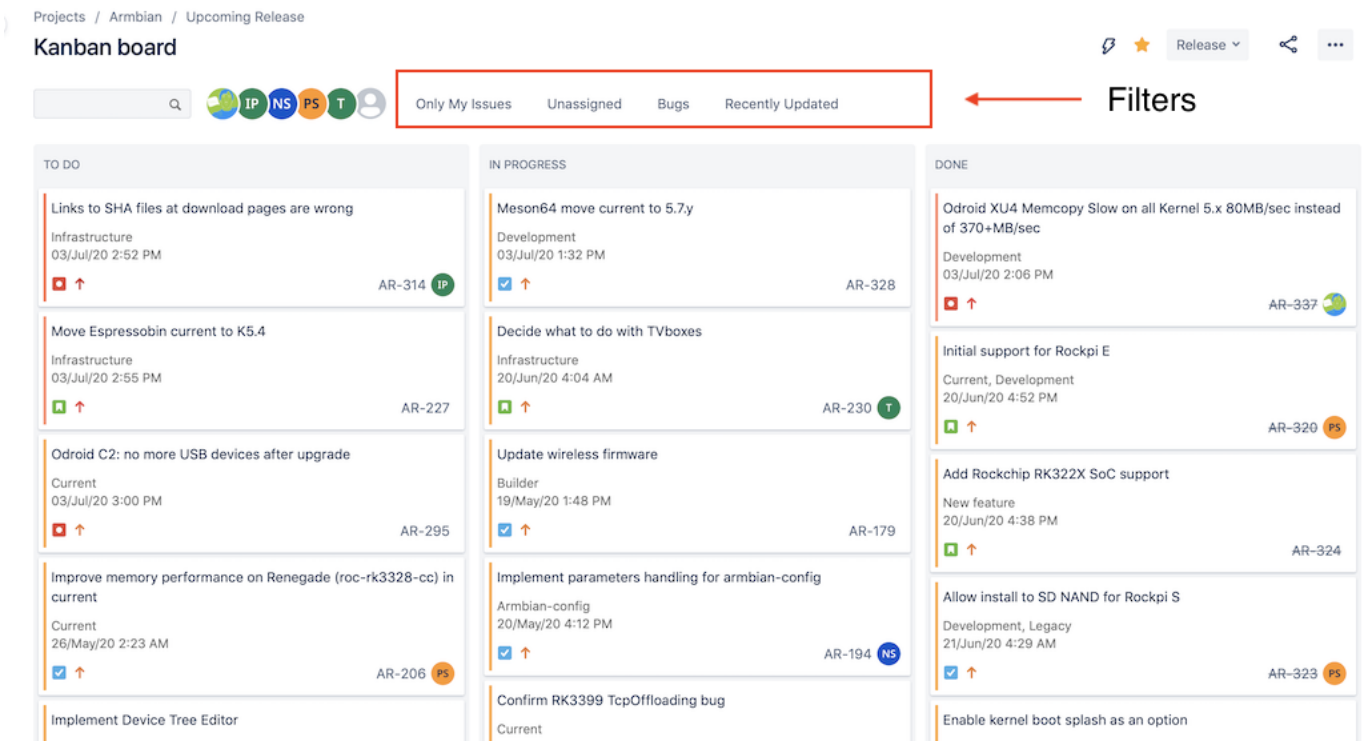
7.3.2 Special Issue Type

- **Bug** - malfunction of the system, an error, flaw, or a default in the system, that causes an incorrect result.

7.3.3 Work Queue

The easiest way to follow the work queue Upcoming Release Kanban Board. This board lists only work select for the upcoming release.

© 2021 by Armbian



Use the filter buttons at top to quickly see unassigned work, work assigned to you, bugs, and work recently updated.

Work is listed in 3 columns, and sorted by priority.

Columns: * **Todo** * Work prioritized to be done next * Pick up any task from this column * **In Progress** * Work In Progress * **Done** * Shows **recently** completed work. Has time limit to keep board clean

7.3.4 Managing Work

All issues for an upcoming release are assigned a “Fix Version” to indicate release number.

Backlog

With the Kanban Board, there are 2 states for the Upcoming Release backlog.

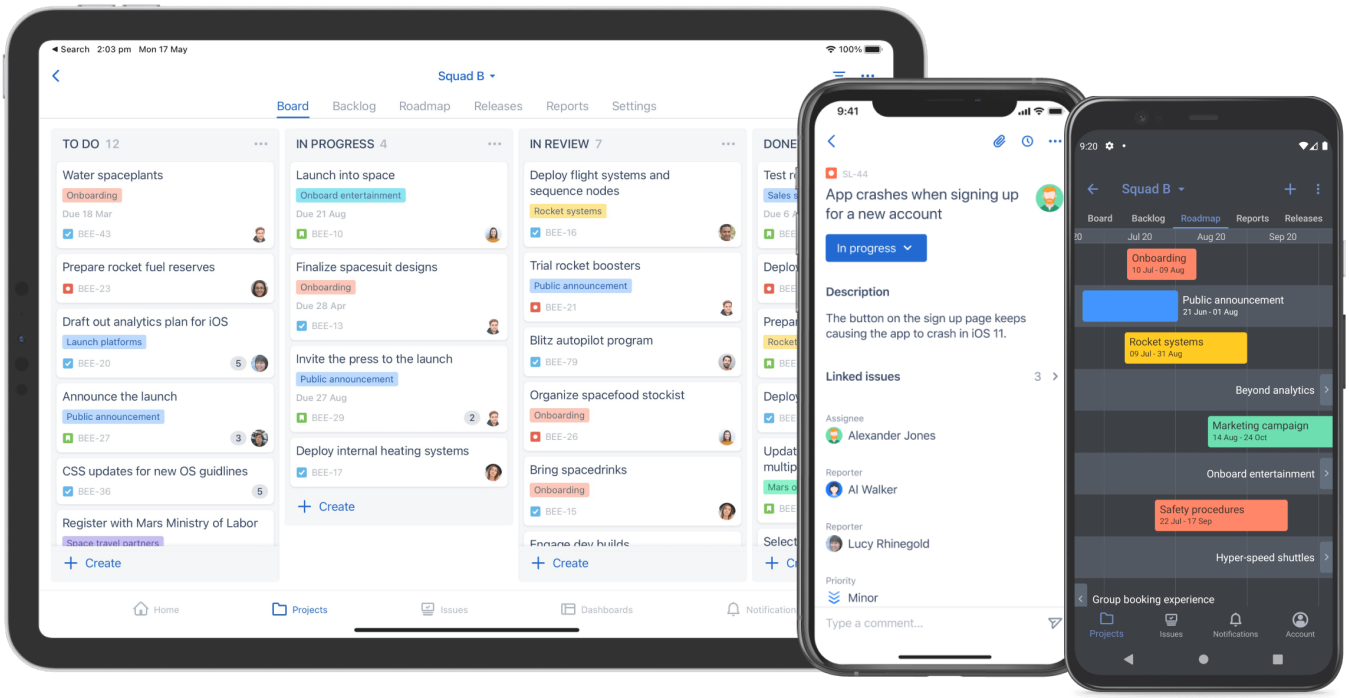
- **Todo** - These tasks are visible the **Todo** column of the Kanban board. To keep things simple, there shouldnt be more than 5-10 issues in Todo
- **Backlog** – Other tasks seleceted for release, but not are not visible on the Kanban board. The purpose of this is to keep the **Todo** column clean and easy to work from. As the **Todo** column clears, prioritize next tasks in backlog by moving them to todo.

All issues for an upcoming release are assigned a “Fix Version” to indicate release number.

© 2021 by Armbian

7.3.5 Mobile access

You can download the app for Android or iOS.



© 2021 by Armbian

7.4 Board Maintainers

Where more than one Maintainer is listed, the top line person may be considered the ‘primary’ and the other(s) on following line(s) as co-maintainers / helpers.

Maintainer column, in this table, is GitHub user name. For cross reference to forum handle, see Forum Name Cross Reference (below).

© 2021 by Armbian

The authoratitative list of board.conf can be found here. Eventually we may automatically generate this list from there, in the mean time we are still maintaining it manually.

© 2021 by Armbian

board.conf	Maintainer	Note(s)
bananapi	janprunk	
bananapim2plus	igorpecovnik	
bananapim64	devdotnetorg	
bananapipro	teknoid	
clearfogbase	heisath	
clearfogpro	heisath	
clockworkpi	littlecxm	
cubieboard		
cubietruck		
cubox-i	igorpecovnik	
firefly-rk3399	150balbes	
helios4	heisath	
jethubj100	jethome-ru	
jethubj80	jethome-ru	
jetson-nano	150balbes	
khadas-edge	igorpecovnik	
khadas-vim1		1
khadas-vim2		1
khadas-vim3	NicoD-SBC	
khadas-vim3l	rpardini	
lafrite	Tonymac32	
lepotato	Tonymac32	
lime-a64	igorpecovnik	
lime2	igorpecovnik	
nanopct4	150balbes	
nanopi-r1	igorpecovnik	
nanopi-r2s	igorpecovnik	
nanopi-r2c	igorpecovnik	
nanopi-r4s	littlecxm	
“	piter75	
nanopiair		
nanopiduo		
nanopiduo2		
nanopik1plus	igorpecovnik	
nanopik2-s905		

© 2021 by Armbian

board.conf	Maintainer	Note(s)
nanopim4	piter75	
nanopim4v2	piter75	
nanopineo	igorpecovnik	
nanopineo2	igorpecovnik	
nanopineo2black	igorpecovnik	
nanopineo3		
nanopineo4		
nanopineocore2		
nanopineoplus2	teknoid	
odroidc2	NicoD-SBC	1
odroidc4	Technicavolous	1
odroidhc4	rpardini	
	Technicavolous	
odroidn2	rpardini	
odroidxu4	igorpecovnik	
“	joekhoobyar	
orangepi-r1	schwar3kat	
orangepi-r1plus		
orangepi-r1plus-lts	schwar3kat	
orangepi-rk3399		
orangepi2		
orangepi3	igorpecovnik	
orangepi4	igorpecovnik	
orangepilite	igorpecovnik	
orangepilite2	igorpecovnik	
orangepione	igorpecovnik	
orangepioneplus	igorpecovnik	
orangepipc	lbmendes	
orangepipc2	igorpecovnik	
orangepipcplus	igorpecovnik	
orangepiplus		
orangepiplus2e	igorpecovnik	
orangepiprime	igorpecovnik	
orangepiwin		
orangepizero	igorpecovnik	
orangepizero2	krachlatte	

© 2021 by Armbian

board.conf	Maintainer	Note(s)
orangepizeroplus	schwar3kat	
orangepizeroplus2-h3	Kalobok	2
orangepizeroplus2-h5		
pine64	janprunk	
pinebook-a64	igorpecovnik	
pinebookpro	seclorum	
pineh64-b		
radxa-n10		
radxa-zero	engineer-80	
radxa-zero2		
renegade	Tonymac32	
rock-3a	catalinii	
“	ZazaBr	2
rockpi-4a		
rockpi-4b		
rockpi-4c		
rockpi-e		
rockpi-s		
rockpro64	joekhoobyar	
station-m1	150balbes	
station-p1	150balbes	
teres-a64		
tinkerboard	Tonymac32	
tritium-h3		
tritium-h5		
udoo		
zeropi	igorpecovnik	
station-m2	150balbes	
station-p2	150balbes	

Notes (for above):

1. rpardini would take this if he could get a board ;-)
2. Maintainer is forum name, we need to update to GitHub name! Please submit PR or contact TRS-80 in IRC or via PM on forums.

7.4.1 Forum Name Cross Reference

Rather than update many different cells above, please maintain GitHub <-> forum handle cross-reference below.

© 2021 by Armbian

If you have a question mark next to, or in place of, your name, please update by sending a PR or contact TRS-80 on IRC or via PM on forums.

Important note for people seeking support:

Do not contact Maintainers via Personal Messages (PMs) on the forums for private support. Such behavior is considered harassment and may result in a ban.

Please make a forum post in the appropriate place instead, where more people may see it, and ultimately everyone may benefit.

GitHub	Forum Name
devdotnetorg	antondeveloper
150balbes	balbes150
catalinii	catalinii
heisath	Heisath
igorpecovnik	Igor
janprunk	yang
jethome-ru	jethome
joekhoobyar	joekhoobyar
?	Kalobok
krachlatte	krachlatte
lbmendes	LucasM
littlecxm	littlecxm
NicoD-SBC	NicoD
piter75	piter75
schwar3kat	schwar3kat
seclorum	seclorum
Technicavolous	Technicavolous
teknoid	teknoid
Tonymac32	TonyMac32
TRSx80	TRS-80
?	ZazaBr
engineer-80	Yakov

© 2021 by Armbian

8. Community

8.1 IRC Channel / Matrix / Discord

8.1.1 👋 ---------- Overview

As announced in the forums everyone interested can communicate in realtime using the internet relay chat (or *IRC*). Well known clients for CLI are Weechat or Irssi and for GUI Hexchat.

Besides that communication is also possible via Matrix or our (unofficial) Discord server.

8.1.2 🔌 ---------- Connect

IRC

- Server: `irc.libera.chat`
- Ports: `6697` / non-encrypted: `6667`
- Channel: `#armbian`

In order to enter main #armbian channel registration with Nickserv is mandatory. Check Libera Chat documentation for further information.

Discord

Simply click here: https://discord.gg/gNJ2fPZKvc
Some of these channels are relayed to our IRC channel so it does not matter if you join IRC or Discord as both received your messages. Check `#welcome-and-rules` for more info.

Matrix

Simply join this room with our Matrix client: `#armbian:libera.chat`
Channel names are the same as for IRC mentioned below.

8.1.3 🛑 ---------- Rules

Forums registration terms and rules apply for our chats: https://forum.armbian.com/terms

8.1.4 💬 ---------- Channels

- **#armbian** is the project's main channel. As for now all user interaction happens there, regardless if chit-chat, issue tracking, peer-to-peer user support or even upcoming release planning talks.
- **#armbian-devel** is the project's channel dedicated for build engine development topics.
- **#armbian-commits** is a moderated channel. Whenever a new interaction with a repository on Github happens it will be announced. Also newly added issues on Jira will be pasted. User chat is not possible.
- **#armbian-rss** is a live forum and Twitter feed. Whenever a new post in the Armbian Forums is made or somebody mentions Armbian on Twitter it will be announced here. User chat is not possible. Of course you can also enable desktop notification in your favorite browser for the forums.

Everybody is free to join any of these channels. We may or may not add more channels in future depending on the needs.

© 2021 by Armbian

8.1.5 👮 ---------- Services

Besides the services offered by Libera (like Nickserv or Chanserv) Armbian has set up some own services.

- **ArmbianGithub**
- Has the purpose to fill #armbian-commits and #armbian-rss like described above
- **Armbian-Discord**
- Has the purpose to relay messages from and to our Discord server in certain channels
- **ArmbianTwitter**
- Recurringly searches on Twitter for new Tweets from *@armbian* and when people are actually mentioning *Armbian*
- **ArmbianHelper**
- Allows to search for Issues and Task on Jira
- If you know the actual task id simply write it to the channel and the bot will look it up. Like `AR-123`
- Search issue by keyword/s in the summary. Like `,searchissue Allwinner H6`
  Take note of the `,`. Will output up to three results.
- Allows to search forums via Google API (not very precise though)
- Example: `,g Allwinner H6 panfrost`
- A few more minor commands, mostly used by staff or do not need introduction
- `.nonprofit`, `.sed`, `.contribute`, `.rtfm`, `.fortune`, `.sunxi`, `.meson`, `help`, `help irc`, `.tvboxes`
- Translation for non-native English speakers
- Simply start your sentence with `--` at the beginning and the bot will translate your message regardless of the source language into English.
  Note: This services will be activated manually on demand (like planned meetings for example) since its backend generates cost.

8.1.6 ❔ ---------- FAQ

- *Why are there so many people in the channel and nobody is talking?*
- It is pretty common for community IRC channels for people to simply *idle* there. Many also using so called IRC bouncers https://en.wikipedia.org/wiki/BNC_(software) that keeps their connection to the channel alive.
- *I wrote 'Hi' but nobody answered. How do I get support there?*
- Probably there is nobody around at the time. Keep in mind that all users are spread around the globe and therefore living in many different time zones.
  It is a common habit to simply state your question or issue and then wait patiently for an answer. Depending how complex this may take up to a few hours because most Armbian contributors have detailed knowledge in a specific board family only.
- *Is the chat history public as well?*
- Yes. All conversation is redundantly logged. These logs are open to the public. You can find them here: http://irc.armbian.com
- *Why do some people have odd hostnames like '@armbian/staff/lanefu' or '@user/username'?*
- These *hostnames* are so called project affiliation cloaks. These are meant to show a users affiliation to a specific project and their role there.
- *Can I have that too?*
- Yes. An Armbian affiliation cloak can be requested from *Werner* either via forums or IRC. They usually will be granted if you are a well known member in forums, a contributor via Github or donated to the project. Make sure you identified yourself to Nickserv beforehand. If you cannot find yourself in the list above you are free to request an unaffiliated cloak from Libera staff. Join #libera for that.

© 2021 by Armbian

- *How can I protect my nickname so nobody can spoof me?*
- Register your nick with Libera's Nickserv service. Check https://libera.chat/guides/registration
  Even though it is not mandatory you should register and identify with the services as other channels for example may not allow unregistered users to chat or join at all as anti-spam measure. If the situation demands Armbian will enforce this as well.
- *Why do some users have voice (+v) in channel?*
- As mentioned in forums "all contributors to the project, regardless if forums staff, contributor on Github or well known and longtime active user" may get voice on request.
- *Should I add **away** to my nick if I am AFK? Like **Werner|away***
- No. Please use the `/away [reason]` command as intended. For an explanation please have a look at the ZNC Wiki.

8.1.7 👉 ---------- Bottom line

If you have any questions, comments regarding the IRC channels and/or services or found an issue in this documentation for think you can enhance it get in touch with *Werner* either via forums or IRC.

By the way if you like to have a free IRC bouncer to always be up-to-date and never miss a conversation again you should check out this thread.

© 2021 by Armbian

8.2 Maintainers

8.2.1 Board / Platform / Maintainers

See individual board configuration files for named maintainer of SBC.

8.2.2 Armbian Base Maintainers

Area	Lead Maintainer	Maintainers	Acronyms, Codenames	additional info
Build Scripts	@igorpecovnik	@lanefu	`/lib/*.sh`	code responsible for building images
Armbian-Tools	@igorpecovnik		armbian-config, armbian-monitor	userland tools provided by Armbian
Armbian-Tools: armbian-config			armbian-config	
Multimedia/ Desktops		@JMCC, @jernejsk, @Miouyouyou		

8.2.3 Other Roles

Area	Lead Maintainer	Maintainers	Acronyms, Codenames	additional info
Release Management	@igorpecovnik			
Testing and Code Quality, CI	@lanefu	@igorpecovnik		
Security				
Documentation		@Werner		
Community Engagement		@NicoD, @tido		
Legal and Financial	@igorpecovnik			
Web and Infrastucture	@lanefu	@[TheBug]		
Forums moderation	@Werner	click		
IRC & Services	@Werner			

8.2.4 Hackers Emeritus

Members who have stepped away from the project, but had a huge impact. We always welcome their contributions and wisdom.

- @tkaiser
- @zador.blood.stained
- @rneese

© 2021 by Armbian