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.

Generic howto for Allwinner devices

Legacy or Vanilla kernel ?

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

- Legacy kernel : FEX
- Vanilla kernel : DT (Device Tree)

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 3.4.112-sun8i #10 SMP PREEMPT Wed Jun 1 19:43:08 CEST 2016
armv7l GNU/Linux
```

FEX

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
```

Device Tree

Which file should I edit

I use the following command and try to guess which file to use in `/boot/dtb/` :

```
cat /proc/device-tree/model
```

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.

FEL/NFS boot explanation

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 (https://linux-sunxi.org/FEL/USBBoot) 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

Requirements

- Allwinner device that supports FEL mode. Check wiki (https://linux-sunxi.org/FEL) 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

Build script options

- KERNEL_ONLY=no
- EXTENDED_DEBOOTSTRAP=yes
- ROOTFS_TYPE=fel

Example:

```
./compile.sh KERNEL_ONLY=no BOARD=cubietruck BRANCH=next PROGRESS_DISPLAY=plain
USE_MAINLINE_GOOGLE_MIRROR=yes RELEASE=jessie BUILD_DESKTOP=no
EXTENDED_DEBOOTSTRAP=yes ROOTFS_TYPE=fel
```

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.

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 (https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/plain/Documentation/filesystems/nfs/nfsroot.txt)](https://git.kernel.org/cgit/linux/kernel/git/stable/linux-stable.git/plain/Documentation/filesystems/nfs/nfsroot.txt) for configuring static IP for NFS root

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

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`.

How to contribute to the code?

1. [Fork (http://help.github.com/forking/)](http://help.github.com/forking/) the project
2. Make one or more well commented and clean commits to the repository.
3. Perform a [pull request (http://help.github.com/pull-requests/)](http://help.github.com/pull-requests/) in github's web interface.

If it is a new feature request, don't start the coding first. Remember to [open an issue (https://guides.github.com/features/issues/)](https://guides.github.com/features/issues/) to discuss the new feature.

If you are struggling, check [this detailed step by step guide on contributing (https://www.exchangecore.com/blog/contributing-concrete5-github/)](https://www.exchangecore.com/blog/contributing-concrete5-github/).

How to build Armbian image or custom kernel?

You will need to setup development environment within Ubuntu 14.04 LTS x64 server image (http://releases.ubuntu.com/14.04/) and cca. 20G of free space.

Login as root and run:

```
apt-get -y -qq install git
git clone --depth 1 https://github.com/igorpecovnik/lib
cp lib/compile.sh .
nano compile.sh # alter if necessary
```

Run the script

```
./compile.sh
```

Build options clarification

- **KERNEL_ONLY** (yes|no):
    - set to "yes" to compile only kernel, u-boot and other packages for installing on existing Armbian system
    - set to "no" to build complete OS image for writing to SD card
- **KERNEL_CONFIGURE** (yes|no):
    - set to "yes" to configure kernel (add or remove modules or features). Kernel configuration menu will be brought up before compilation
    - set to "no" to compile kernel without changing default or custom provided configuration
- **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 `output/cache` (rootfs cache),
    - "sources" = delete `sources` (all downloaded sources)
- **KERNEL_KEEP_CONFIG** (yes|no):
    - set to "yes" to use kernel config file from previous compilation for the same branch, device family and version
    - set to "no" to use default or user-provided config file
- **BUILD_DESKTOP** (yes|no):
    - set to "yes" to build image with minimal desktop environment

    - set to "no" to build image with console interface only

- **EXTERNAL** (yes|no):

    - set to "yes" to compile and install some extra applications and drivers (only for **default** kernel branch):

        - USB redirector (http://www.incentivespro.com)
        - Realtek RT8192 wireless driver
        - Mediatek MT7601U wireless - driver
        - Sunxi display control
        - hostapd from sources

- **DEBUG_MODE** (yes|no)

    - set to "yes" will prompt you right before the compilation starts to make changes to the source code. Separate for u-boot and kernel. It will also create a patch out of this. If you want that this patch is included in the normal run, you need to copy it to appropriate directory
    - set to "no" compilation will run uninterrupted

- **FORCE_CHECKOUT** (yes|no):

    - set to "yes" to force overwrite any changed or manually patched kernel, u-boot and other sources
    - set to "no" to keep all changes to sources

- **BUILD_ALL** (yes|no): cycle through all available board and kernel configurations and make images for all combinations

Hidden options to minimize user input for build automation:

- **BOARD** (string): you can set name of board manually to skip dialog prompt
- **BRANCH** (default|next|dev): you can set kernel and u-boot branch manually to skip dialog prompt; some options may not be available for all devices
- **RELEASE** (wheezy|jessie|trusty|xenial): you can set OS release manually to skip dialog prompt; use this option with `KERNEL_ONLY=yes` to create board support package

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

- **USE_CCACHE** (**yes**|no): use a C compiler cache to speed up the build process
- **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`
- **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
- **EXTENDED_DEBOOTSTRAP** (**yes**|no): use new debootstrap and image creation process
- **FORCE_USE_RAMDISK** (yes|no): overrides autodetect for using tmpfs in new debootstrap and image creation process. Takes effect only if `EXTENDED_DEBOOTSTRAP` is set to "yes"
- **FIXED_IMAGE_SIZE** (integer): create image file of this size (in megabytes) instead of

minimal. Takes effect only if EXTENDED_DEBOOTSTRAP is set to "yes"
- **COMPRESS_OUTPUTIMAGE** (yes|**no**): create compressed archive with image file and GPG signature for redistribution
- **SEVENZIP** (yes|**no**): create .7z archive with extreme compression ratio instead of .zip
- **ROOTFS_TYPE** (**ext4**|f2fs|btrfs|nfs|fel): create image with different root filesystems instead of default ext4. Requires setting FIXED_IMAGE_SIZE to actual size of your SD card for F2FS and BTRFS. Takes effect only if EXTENDED_DEBOOTSTRAP is set to "yes"

Supplying options via command line parameters

Instead of editing compile.sh to set options, you can set them by supplying command line parameters to compile.sh
Example:

```
./compile.sh BRANCH=next BOARD=cubietruck KERNEL_ONLY=yes
PROGRESS_DISPLAY=plain RELEASE=jessie
```

Note: Option BUILD_ALL cannot be set to "yes" via command line parameter.

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 `lib/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-dev 4.4.0-rc6 ]
[ o.k. ] Looking for user patches in [ userpatches/kernel/sunxi-dev ]
```

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

User provided kernel config

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

```
[ o.k. ] Compiling dev kernel [ @host ]
[ o.k. ] Using kernel config file [ lib/config/linux-sunxi-dev.config ]
```

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`

Partitioning of the SD card

In case you define `$FIXED_IMAGE_SIZE` at build time the partition containing the rootfs will be made of this size. Default behaviour when this is not defined and `$ROOTFS_TYPE` is set to *ext4* 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)

What is behind the build process?

Build process summary:

- creates development environment on the top of X86/AMD64 Ubuntu 14.04 LTS,
- downloads proven sources, applies patches and uses tested configurations,
- cross-compiles universal boot loader (u-boot), kernel and other tools and drivers,
- packs kernel, uboot, dtb and root customizations into debs,
- debootstraps minimalistic Debian Wheezy, Jessie and Ubuntu Trusty into SD card image,
- installs additional packets, applies customizations and shrinks image to its actual size.

Image compiling example with partial cache:

[su_youtube_advanced url="https://youtu.be/zeShf12MNLg" controls="yes" showinfo="no" loop="yes" rel="no" modestbranding="yes"]

Creating compile environment

At first run we are downloading all necessary dependencies.

Using board configuration

We need to get some predefined variables about selected board. Which kernel & uboot source to use, modules to load, which is the build number, do we need to have a single partition or dual with boot on fat, which extra drivers to compile out of the kernel tree, ...

Board configuration example:

```
BOOTSIZE="16"                                   # FAT boot partition in
MB, 0 for none
BOOTCONFIG="udoo_neo_config"                    # Which compile config
to use
LINUXFAMILY="udoo"                              # boards share kernel
```

Note that in this case, all main config options (kernel and uboot source) are covered within FAMILY. Check [configuration.sh (https://github.com/igorpecovnik/lib/blob/master/configuration.sh)](https://github.com/igorpecovnik/lib/blob/master/configuration.sh) for more config options.

This **isn't ment to be user configurable** but you can alter variables if you know what you are doing.

Downloading sources

When we know where are the sources and where they need to be the download / update process starts. This might take from several minutes to several hours.

Patching

In patching process we are appling patches to sources. The process is defined in:

```
lib/patch/kernel/sun7i-default
lib/patch/kernel/sunxi-dev
...
lib/patch/u-boot/u-boot-default
lib/patch/u-boot/u-boot-neo-default
...
```

Patch rules for subdirectories are: **KERNEL_FAMILY-BRANCH** for kernel and **U-BOOT-SOURCE-BRANCH** for U-boot.

Debootstrap

Debootstrap creates fresh Debian / Ubuntu root filesystem templates or use cached under:

```
output/cache/rootfs/$DISTRIBUTION.tgz
```

To recreate those files you need to remove them manually.

Kernel install

When root filesystem is ready we need to install kernel image with modules, board definitions, firmwares. Along with this we set the CPU frequency min/max, hostname, modules, network interfaces templates. Here is also the place to install headers and fix + native compile them on the way.

Distribution fixes

Each distributin has it's own way of doing things:

- serial console

- different packets
- configuration locations

Board fixes

Each board has their own tricks: **different device names, firmware loaders, configuration (de)compilers, hardware configurators**

Desktop installation

You can build a desktop withing the image. Consider this feature as experimental. Hardware acceleration on Allwinner boards is working within kernel 3.4.x only.

External applications

This place is reserved for custom applications. There is one example of application - USB redirector.

Closing image

There is an option to add some extra commands just before closing an image which is also automaticaly shrink to it's actual size with some small reserve.

Directory structure

It will be something like this:

```
compile.sh              compile execution script
lib/bin/                blobs, firmwares, static compiled, bootsplash
lib/config/             kernel, board, u-boot, hostapd, package list
lib/documentation/      user and developers manual
lib/patch/              collection of kernel and u-boot patches
lib/scripts/            firstrun, arm hardware info, firmware loaders
lib/LICENSE             licence description
lib/README.md           quick manual
lib/boards.sh           board specfic installation, kernel install, desktop
install
lib/common.sh           creates environment, compiles, shrink image
lib/configuration.sh    boards presets - kernel source, config, modules, ...
lib/debootstrap.sh      basic system template creation
lib/distributions.sh    system specific installation and fixes
lib/main.sh             user input and script calls
lib/makeboarddeb.sh     creates board support package .deb
lib/repo-update.sh      creates and updates your local repository
lib/repo-show-sh        show packets in your local repository
lib/upgrade.sh          script to upgrade older images
sources/                source code for kernel, uboot and other utilities
output/repository       repository
output/cache            cache for root filesystem and headers compilation
output/debs             deb packeges
output/images           zip packed RAW image
userpatches/kernel      put your kernel patches here
userpatches/u-boot      put your u-boot patches here
userpatches/            put your kernel config here
```

Additional info

- Allwinner SBC community (https://linux-sunxi.org/)

Armbian H3 mini FAQ -- 5.15

Important to know

- this will be the last version of the H3 mini FAQ since H3 boards are from now on officially supported
- Updating Armbian images prior to 5.13 to 5.15 or higher is **not** recommended -- see below
- 1st boot takes longer (up to 5 minutes). Please do not interrupt while the red LED is blinking, the board reboots automatically one time and the green LED starts to blink when ready
- our User documentation (http://www.armbian.com/documentation/) (one exception currently: use *h3disp* to adjust display settings)
- our Geek documentation (http://www.armbian.com/using-armbian-tools/) (in case you want to build your own images)
- CPU frequency settings are 240-1200 MHz on BPi M2+, NanoPi M1 and Beelink X2, 480-1200 MHz on OPi One/Lite and 480-1296 MHz on the other boards (cpufreq governor is *interactive* therefore the boards only increase CPU speed and consumption when

needed)
- In case you experience instabilities check your SD card using `armbianmonitor -c $HOME` and think about installing RPi-Monitor for H3 (http://www.cnx-software.com/2016/03/17/rpi-monitor-is-a-web-based-remote-monitor-for-arm-development-boards-such-as-raspberry-pi-and-orange-pi/) to get an idea whether you suffer from overheating (`sudo armbianmonitor -r` will install everything needed)
- In case you're unsure whether to test a desktop or CLI image simply try out the GUI version since you can always get 'CLI behaviour' by running `sudo update-rc.d -f nodm disable` later (this disables the start of X windows and desktop images behave like those made for headless use afterwards. If you're experienced you could also reclaim disk space by removing the `libxfce4util-common` package and doing an `apt-get autoremove` later)
- 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 (http://forum.armbian.com/index.php/topic/793-moving-to-harddisk/))

Upgrading from older images

Upgrading from any of the H3 pre-built images (read as: all between version 5.03 and 5.12) is **not** recommended. We had one fundamental change while developing/improving H3 board support (switching away from providing one OS image for various boards and trying to rely on *board auto detection* at first boot) that might cause loads of problems when upgrading.

Therefore it's strongly recommended to backup settings and `$HOME` contents, start with a fresh 5.15 OS image from scratch and then copy back stuff. We're sorry for any inconveniences caused by this and provide a few tips/tweaks in the forum (http://forum.armbian.com/index.php/topic/1400-upgrading-h3-pre-built-armbian-images-to-515-or-above/)

#OS images with legacy Kernel (3.4.112)

Armbian started beginning with release 5.04 to support all available H3 based Orange Pi boards back then and extended the support to a few more H3 devices (also from other vendors) in the meantime. For a board overview you can look through our buyer's guide (http://forum.armbian.com/index.php/topic/1351-h3-board-buyers-guide/) and/or Jean-Luc's nice comparison table (http://www.cnx-software.com/2016/06/08/allwinner-h3-boards-comparison-tables-with-orange-pi-banana-pi-m2-nanopi-p1-and-h3-olinuxino-nano-boards/#comments).

Changes between 5.06 and 5.15 (not released yet)

- Added improved camera driver (https://github.com/avafinger/gc2035) 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)
- 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 (https://github.com/friendlyarm/h3_lichee), 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)
- Many minor fixes

Changes in 5.06

- 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

Changes in 5.05

- 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 (http://forum.armbian.com/index.php/topic/755-tutorial-orange-pi-one-adding-usb-analog-audio-out-tv-out-mic-and-ir-receiver/))
- I2S possible on all Orange Pis (compare with the mini tutorial (http://forum.armbian.com/index.php/topic/759-tutorial-i2s-on-orange-pi-h3/) 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

Changes in 5.04 compared to pre-releases

- HDMI/DVI works (bug in boot.cmd settings)

- 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 (https://github.com/igorpecovnik/lib/blob/6d995e31583e5361c758b401ea44634d406ac
L1286))
- 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 (http://forum.armbian.com/index.php/topic/617-wip-orange-pi-one-support-for-the-upcoming-orange-pi-one/?p=5480) 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

Known issues with 5.15

- Playing HEVC/H.265 video with 10 bit depth not supported since H3 lacks this feature
- (still) not possible to upgrade *server* to *desktop* images -- better go the other way around if unsure

What you can do to improve the situation

- get back to us with feedback regarding our OS images (http://forum.armbian.com/index.php/topic/617-wip-orange-pi-one-support-for-the-upcoming-orange-pi-one/?view=getlastpost)
- fork our repo, fix things and send pull requests

Known to NOT work (reliably) yet

- live display resolution switching. Fixes welcome (anyone willing to port the stuff from the H3 OpenELEC port (https://github.com/jernejsk/OpenELEC-OPi2)?)

#OS images with vanilla Kernel (4.x)

Important: Currently no thermal readouts and no throttling/cpufreq adjustments are implemented in mainline kernel (THS). Therefore it's possible to permanently damage your H3 when running demanding workloads on it. Keep this in mind especially when you use any of the H3 devices not from Xunlong since they all overheat badly and think about reducing maximum clockspeed in u-boot (816 MHz max for example)

Mainlining effort for H3 and Orange Pi's is progressing nicely but since Ethernet is still missing some bits and especially THS support isn't ready we currently do not provide OS images with vanilla kernel. Our build system is already prepared, so as soon as THS and Ethernet are fully

ready we'll release OS images (in the lab an Orange Pi PC is serving files as NAS since months stable with kernel 4.4 and USB-Ethernet and now 4.6.2 with native Ethernet driver).

Release history

v5.14 / 14.6.2016

- added Beelink X2 image

v5.14 / 12.6.2016

- Cubieboard 1 images rebuilt
- Cubox / Hummingboard kernel upgrade to 3.14.72 and 4.6.2
- Trusty was replaced with Xenial on those two boards

v5.14 / 5.6.2016

- Odroid C2 and Orange Pi+ images rebuilt
- fixed eMMC installer, updated images and repository for Opi PC Plus and Opi Plus 2E

v5.12 / 31.5.2016

- updated C1 images
- added wifi driver for new Oranges (modprobe 8189fs)
- added Orange Pi Lite, PC Plus and Plus 2E images

v5.11 / 24.5.2016

- Various bug fixes
- new working images for Actions Semi S500 boards

v5.10 / 1.5.2016

Images:

- all 3.10+ kernels are Docker ready (http://forum.armbian.com/index.php/topic/490-docker-on-armbian/)
- all A10/A20/H3 comes with HW accelerated video playback in desktop build
- fixed root exploit on H3 boards (https://github.com/igorpecovnik/lib/issues/282)
- fixed kswapd 100% bug on H3 boards (https://github.com/igorpecovnik/lib/issues/219)
- 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 (https://github.com/igorpecovnik/lib/commit/94194dc06529529015bfd04767865bbd04d
- 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 (https://github.com/igorpecovnik/lib/issues/258)
- increased MALI clockspeed on sun8i/legacy (https://github.com/igorpecovnik/lib/issues/265)
- added Armbianmonitor (http://forum.armbian.com/index.php/topic/881-prepare-v51-v201604/?p=7095)
- added Odroid C1, C2(arm64), Nanopi M1, Banana M2+, Pcduino 2 and Pcduino 3. CLI and desktop
- added wifi radar to desktop
- added preview vanilla 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

Build script:

- GCC 5 support for vanilla 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

v5.05 / 8.3.2016

- H3 images rebuilt. Updating from older also possible.
- Udoo quad images upgraded to 4.4.4

v5.04 / 1.3.2016

- Banana M1/PRO/M1+ rebuilt
- fixed SATA problem

- set OTG port in HOST mode in vanilla kernel
- wireless is working on PRO out of the box
- added utility to switch between OTG and HOST in vanilla kernel
- Bugs left: OTG mode not working, M1+ wireless not work in vanilla kernel

v5.04 / 28.2.2016

- H3 images rebuilt
- added desktop images for H3 based boards
- 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

v5.03 / 20.2.2016

- H3 images rebuilt

v5.02 / 18.2.2016

- H3 images rebuilt

v5.01 / 17.2.2016

- Bugfix update for Allwinner boards (http://forum.armbian.com/index.php/topic/691-banana-pro-testers-wanted-sata-drive-not-working-on-some-boards/)
- Update for H3 based boards (https://github.com/igorpecovnik/lib/commit/c93d7dfb3538c36739fb8841bd314d75e7d7

v5.00 / 12.2.2016

- Vanilla kernel for Allwinner based boards upgraded to 4.4.1
- Allwinner audio driver playback and capture on kernel 4.4.1, UAS (http://linux-sunxi.org/USB/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 (http://forum.armbian.com/index.php/topic/602-new-motd-for-ubuntudebian/#entry4223) 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 Vanilla
- fixed Banana PRO/+ onboard wireless on Vanilla kernel - running with normal Banana DT.
- readded USB sound
- added A13 Olimex SOM (https://www.olimex.com/Products/SOM/A13/A13-SOM-512/)
- added LIRC GPIO receive and send driver (https://github.com/igorpecovnik/lib/issues/135) for legacy Allwinner
- added LED MMC activity to Vanilla 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)

v4.81 / 28.12.2015

- 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 Vanilla 4.2.8
- Vanilla kernel for Allwinner based boards upgraded to 4.3.3
- Udoo vanilla upgraded to 4.2.8, legacy to 3.14.58
- cubox / hummingboard upgraded to 3.14.58, added Vanilla kernel 4.4
- fixed Jessie RTC bug, systemd default on Jessie images

v4.70 / 30.11.2015

- Bugfix update(apt-get update && apt-get upgrade)
- small changes and fixes

v4.6 / 24.11.2015

- Update only (apt-get update && apt-get upgrade)
- Vanilla 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 Vanilla 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

v4.5 / 14.10.2015

- vanilla kernel upgraded to 4.2.3 for Allwinner based boards
- legacy kernel for Allwinner compiled from new sources (linux-sunxi)
- udoo vanilla 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
- 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 vanilla kernel
- fixed dvbsky modules

and a bunch of small fixes.

v4.4 / 1.10.2015

Images:

- vanilla 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) (vanilla Allwinner),
- Udoo quad images with fixed legacy kernel 3.14.28,
- Cubox and Hummingboard kernel upgrade to 3.14.53,
- brcmfmac driver fixes for vanilla 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).

v4.3 / 17.9.2015

- 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.
- 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.

v4.2 / 1.9.2015

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

v4.1 / 5.8.2015

- 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

v4.0 / 12.7.2015

- Fixed stability issues, temperature display in 4.x
- Kernel upgrades to 3.4.108 and 4.1.2

v3.9 / 11.6.2015

- 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

v3.8 / 21.5.2015

- 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

v3.7 / 14.5.2015

- 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

v3.6 / 29.4.2015

- Kernel 3.19.6
- Kernel 3.4.107 with better BT loading solution

v3.5 / 18.4.2015

- 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

v3.4 / 28.3.2015

- 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

v3.3 / 28.2.2015

- Kernel 3.19.0: many new functionality and fixes.
- Bugfixes: CT wireless works in all kernels

v3.2 / 24.1.2015

- Possible to compile external modules on both kernels
- Kernel 3.19.0 RC5
- Bugfixes: install script, headers, bashrc, spi

v3.1 / 16.1.2015

- 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

v3.0 / 29.12.2014

- Kernel 3.18.1 for mainline image
- Added Ubuntu Trusty (14.04 LTS) image
- Bugfixes: auto packages update

v2.9 / 3.12.2014

- 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

v2.8 / 17.10.2014

- 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

v2.7 / 1.10.2014

- 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

v2.6 / 22.8.2014

- Kernel 3.4.103 and 3.17.0-RC1
- Added GPIO patch (only for 3.4.103)

v2.5 / 2.8.2014

- Kernel 3.4.101 and 3.16.0-RC4
- major build script rewrite

v2.4 / 11.7.2014

- 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

v2.3 / 2.7.2014

- 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

v2.2 / 26.6.2014

- 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

v2.1 / 13.6.2014

- Kernel 3.4.93
- Onboard Bluetooth finally works
- Small performance fix
- Allwinner Security System cryptographic accelerator

v2.0 / 2.6.2014

- 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

v1.9 / 27.4.2014

- Kernel headers included
- 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

v1.8 / 27.3.2014

- 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

v1.7 / 26.2.2014

- 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

v1.6 / 9.2.2014

- Added support for Cubieboard 2
- Build script creates separate images for VGA and HDMI
- NAND install script added support for Cubieboard 2

v1.52 / 7.2.2014

- Various kernel tweaks, more modules enabled
- Root filesystem can be moved to USB drive
- Bugfixes: NAND install script

v1.5 / 22.1.2014

- Hotspot Wifi Access Point / Hostapd 2.1
- Bugfixes: MAC creation script, SSH keys creation, removed double packages, ...
- Graphics desktop environment upgrade ready

v1.4 / 12.1.2014

- Patwood's kernel 3.4.75+ with many features
- Optimized CPU frequency scaling 480-1010Mhz with interactive governor
- NAND install script included
- Cubietruck MOTD
- USB redirector – for sharing USB over TCP/IP

v1.3 / 3.1.2014

- CPU frequency scaling 30-1000Mhz
- Patch for gpio

v1.23 / 1.1.2014

- added HDMI version
- added sunxi-tools
- build.sh transfered to Github repository
- disabled LED blinking

v1.2 / 26.12.2013

- changed kernel and hardware config repository
- kernel 3.4.61+
- wi-fi working
- updated manual how-to

v1.0 / 24.12.2013

- 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

What to download?

Each board is fully supported with up to **four basic system** options:

- Debian Wheezy or Jessie

- Ubuntu Trusty or Xenial

Some boards also have a desktop version Debian Jessie.

Legacy or Vanilla?

Both kernels are stable and production ready, but you should use them for different purpuses since their basic support differ:

- legacy: video acceleration, NAND support, connecting displays
- vanilla: headless server, light desktop operations

How to check download authenticity?

All our images are digitally signed and therefore it's possible to check theirs authentication. You need to unzip the download package and issue those commands (Linux):

```
# download my public key from the database
gpg --keyserver pgp.mit.edu --recv-key 9F0E78D5
gpg --verify Armbian_4.83_Armada_Debian_jessie_3.10.94.raw.asc

# proper respond
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 repond. 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 WARNING: This key is not certified with a trusted signature!

How to prepare SD card?

7z and zip archives can be uncompressed with 7-Zip (http://www.7-zip.org/) on Windows, Keka (http://www.kekaosx.com/en/) on Mac and 7z on Linux (apt-get install p7zip-full). RAW images can be written with Rufus (https://rufus.akeo.ie/) (Win) or DD in Linux/Mac:

```
# Linux example: /dev/sdx is your sd card device
dd bs=1M if=filename.raw of=/dev/sdx
# OS X example: /dev/[r]diskx is your sd card device:
diskutil unmountDisk diskx && dd bs=1m if=filename.raw of=/dev/rdiskx &&
diskutil eject diskx
```

Image writing takes around 3 minutes on a slow, class 6 SD card.

Important: Make sure you use a **good & reliable** SD card. If you encounter boot troubles or simply as a measure of precaution, check them with F3 (http://oss.digirati.com.br/f3/) or H2testw (http://www.heise.de/download/h2testw.html).

Also important: SD cards are optimised for sequential reads/writes as it's common in digital cameras. This is what the *speed class* is about. And while you shouldn't buy any card rated less than *class 10* today you should especially take care to choose one that is known to show high random I/O performance since this is way more performance relevant when used with any SBC.

You won't be wrong picking one of these:

(http://www.amazon.com/dp/B00IVPU7KE)

(http://www.amazon.com/gp/product/B00BLHWYWS)

(http://www.amazon.com/dp/B008HK1YAA)

Detailed informations regarding SD cards performance:

- SD card performance with Armbian - Thomas Kaiser (http://forum.armbian.com/index.php/topic/954-sd-card-performance/)
- Raspberry Pi microSD card performance comparison - Jeff Geerling (http://www.jeffgeerling.com/blogs/jeff-geerling/raspberry-pi-microsd-card)
- The Best microSD Card - Kimber Streams (http://thewirecutter.com/reviews/best-microsd-card/)

How to boot?

Insert SD card into a slot and power the board. First boot takes around 3 minutes then it reboots and you will need to wait another one minute to login. This delay is because system updates package list and creates 128Mb emergency SWAP on the SD card.

Normal boot (with DHCP) takes up to 35 seconds with a class 6 SD CARD and cheapest board.

How to login?

Login as **root** on console or via SSH and use password **1234**. You will be prompted to change this password at first login. You will then be asked to create a normal user account that is sudo enabled (beware of default QWERTY keyboard settings at this stage).

Desktop images starts into desktop without asking for password. To change this add some display manager:

```
apt-get install lightdm
```

... or edit the contents of file:

```
/etc/default/nodm
```

and change the autologin user.

How to update?

```
apt-get update
apt-get upgrade
```

This will not only update distribution packages (Debian/Ubuntu) but also updates Armbian kernel, u-boot and board support package if available. So if you've seen in the list of updated packages the names *u-boot* or *linux* the following command is required for changes to take effect:

```
reboot
```

How to switch kernels or upgrade from other systems?

If you are running **legacy kernel** and you want to switch to **vanilla**, **development** or vice versa, you can do it this way:

```
    wget -q -O - http://upgrade.armbian.com | bash
```

You will be prompted to select and confirm some actions. It's possible to upgrade **from any other distribution**. Note that this procedure upgrades only kernel with hardware definitions (bin, dtb, firmware and headers. Operating system and modifications remain as is.

Check [this for manual way (http://www.armbian.com/kernel/)](http://www.armbian.com/kernel/) and more info.

[su_youtube_advanced url="https://youtu.be/iPAlPW3sv3I" controls="yes" showinfo="no" loop="yes" rel="no" modestbranding="yes"]

How to troubleshoot?

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

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 (USB support in u-boot 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 these commands, replacing root device path if necessary. Select setenv line with ttyS0 for serial, tty1 for keyboard+monitor (these are for booting with mainline kernel, check boot.cmd for your device for commands related to legacy kernel):

```
setenv bootargs init=/bin/bash root=/dev/mmcblk0p1 rootwait
console=ttyS0,115200
# or
setenv bootargs init=/bin/bash root=/dev/mmcblk0p1 rootwait console=tty1

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

System should eventually boot to bash shell:

```
root@(none):/#
```

Now you can try to fix your broken system.

- [Fix a Jessie systemd problem due to upgrade from 3.4 to 4.x (https://github.com/igorpecovnik/lib/issues/111)](https://github.com/igorpecovnik/lib/issues/111)

How to unbrick the system?

When something goes terribly wrong and you are not able to boot the system, this is the way to proceed. You need some linux machine, where you can mount the failed SD card. With this procedure you will reinstall the u-boot, kernel and hardware settings. In most cases this should be enought to unbrick the board. It's recommended to issue a filesystem check before mounting:

```
fsck /dev/sdX -f
```

Than mount the SD card and download those files (This example is only for Banana R1):

```
http://apt.armbian.com/pool/main/l/linux-trusty-root-next-lamobo-r1/linux-
trusty-root-next-lamobo-r1_4.5_armhf.deb
http://apt.armbian.com/pool/main/l/linux-upstream/linux-image-next-
sunxi_4.5_armhf.deb
http://apt.armbian.com/pool/main/l/linux-upstream/linux-firmware-image-next-
sunxi_4.5_armhf.deb
http://apt.armbian.com/pool/main/l/linux-upstream/linux-dtb-next-
sunxi_4.5_armhf.deb
```

This is just an example for: **Ubuntu Trusty, Lamobo R1, Vanilla kernel** (next). Alter packages naming according to [this (http://forum.armbian.com/index.php/topic/211-kernel-update-procedure-has-been-changed/)](http://forum.armbian.com/index.php/topic/211-kernel-update-procedure-has-been-changed/).

Mount SD card and extract all those deb files to it's mount point.

```
dpkg -x DEB_FILE /mnt
```

Go to /mnt/boot and link (or copy) **vmlinuz-4.x.x-sunxi** kernel file to **zImage**.

Unmount SD card, move it to the board and power on.

How to add users?

To create a normal user do this:

```
adduser MyNewUsername
```

Put user to sudo group:

```
usermod -aG sudo MyNewUsername
```

How to customize keyboard, time zone?

keyboard:

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

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
```

console font, codepage:

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

time zone:

```
dpkg-reconfigure tzdata
```

screen settings on H3 devices:

```
# Example to set resolution to 1920 x 1080, full colour-range and DVI
h3disp -m 1080p60 -d -c 1
```

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
```

screen resolution interactive - only Allwinner boards with A10 and A20 with legacy kernel:

```
# Example to set console framebuffer resolution to 1280 x 720
a10disp changehdmimodeforce 4
```

Other modes:

```
0 480i
1 576i
2 480p
3 576p
4 720p 50Hz
5 720p 60Hz
6 1080i 50 Hz
7 1080i 60 Hz
8 1080p 24 Hz
9 1080p 50 Hz
10 1080p 60 Hz
```

How to alter CPU frequency?

Some boards allow to adjust CPU speed.

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

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

```
service cpufrequtils restart
```

How to upgrade into simple desktop environment?

```
apt-get -y install xorg lightdm xfce4 tango-icon-theme gnome-icon-theme
reboot
```

Check [this site (http://namhuy.net/1085/install-gui-on-debian-7-wheezy.html)](http://namhuy.net/1085/install-gui-on-debian-7-wheezy.html) for others and be prepared that some desktop image features currently might not work afterwards (eg. 2D/3D/video HW acceleration, so downgrading a *desktop* image, removing the `libxfce4util-common` package and doing an `apt-get autoremove` later might be the better idea in such cases)

How to upgrade Debian from Wheezy to Jessie?

```
dpkg -r ramlog
cp /etc/apt/sources.list{,.bak}
sed -i -e 's/ \(old-stable\|wheezy\)/ jessie/ig' /etc/apt/sources.list
apt-get update
apt-get --download-only dist-upgrade
apt-get dist-upgrade
```

How to upgrade from Ubuntu Trusty to 16.04 LTS (Xenial Xerus)?

```
apt-get install update-manager-core
do-release-upgrade -d
# further to xenial
apt-get dist-upgrade
```

How to toggle boot output?

Edit and change [boot parameters (http://redsymbol.net/linux-kernel-boot-parameters/)](http://redsymbol.net/linux-kernel-boot-parameters/) in /boot/boot.cmd:

```
- console=ttyS0,115200
+ console=tty1
```

and convert it to boot.scr with this command:

```
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).

How to toogle verbose boot?

```
touch /boot/.force-verbose # enable
```

You need to reboot to conduct changes.

```
rm /boot/.force-verbose # disable
```

How to provide boot logs for inspection?

When computer behaves strange first step is to look into kernel logs. We made a tool that grabs info and paste it to the website.

```
sudo armbianmonitor -b
reboot
sudo armbianmonitor -u
```

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

How to build a wireless driver ?

Recreate kernel headers scripts (optional)

```
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
```

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

[su_youtube_advanced url="https://youtu.be/6So8MA-qru8" controls="yes" showinfo="no" loop="yes" rel="no" modestbranding="yes"]

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 or permanently 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

How to change network configuration?

There are five predefined configurations, you can find them in those files:

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

By default **/etc/network/interfaces** is symlinked to **/etc/network/interfaces.default**

1. DEFAULT: your network adapters are connected classical way.
2. 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.
3. BONDING: your network adapters are bonded in fail safe / "notebook" way.
4. Router configuration for Lamobo R1 / Banana R1.
5. Switch configuration for Lamobo R1 / Banana R1.

You can switch configuration with re-linking.

```
cd /etc/network
ln -sf interfaces.x interfaces
```

(x = default,hostapd,bonding,r1)

Than check / alter your interfaces:

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

How to set fixed IP?

By default your main network adapter's IP is assigned by your router DHCP server.

```
iface eth0 inet dhcp
```

change to - for example:

```
iface eth0 inet static
    address 192.168.1.100
    netmask 255.255.255.0
    gateway 192.168.1.1
```

How to froze your filesystem?

In certain situations, it is desirable to have an 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 A10, A20 or H3 board with legacy kernel (3.4.x) where we added support for overlayfs. We tested it on Ubuntu Xenial but it should work elsewhere too. Login as root and execute:

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

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.

How to run Docker?

Preinstallation requirements:

- Armian 5.1 or newer with Kernel 3.10 or higher
- Debian Jessie (might work elsewhere with some modifications)
- root access

Execute this as root:

```
echo "deb https://packagecloud.io/Hypriot/Schatzkiste/debian/ wheezy main" >
/etc/apt/sources.list.d/hypriot.list
curl https://packagecloud.io/gpg.key | sudo apt-key add -
apt-get update
apt-get -y install --no-install-recommends docker-hypriot
apt-get -y install cgroupfs-mount
reboot
```

Test example:

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

More info in this forum topic (http://forum.armbian.com/index.php/topic/490-docker-on-armbian/)

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 (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
```

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. Link **/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

How to connect IR remote?

Required conditions:

- IR hardware
- loaded driver

Get your remote configuration (http://lirc.sourceforge.net/remotes/) (lircd.conf) or learn (http://kodi.wiki/view/HOW-TO:Setup_Lirc#Learning_Commands). 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
```
