Linux System

Customization with BuildRoot

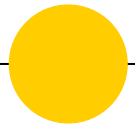
“Du bon code avec le bon sens”

Julien Rollin

_CTO TalanLabs

Dev / CoachCraft / Cloud ready

What are we talking about ?

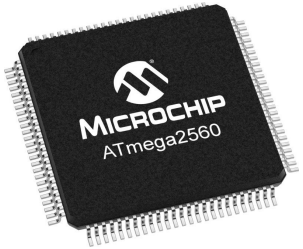


say my name

Hardware system with software that is designed to perform a dedicated function, either as an independent system or as a part of a large system

Two big families

- Microcontroller
  - do not have a memory management unit (MMU)
  - less complex / less power / cheaper
  - industry / automation / reliability



- Microprocessor
  - have a memory management unit (MMU)
  - general purpose / complex / designed for system
  - powerful / more expansive

Both have software with more or less abstraction to interact with hardware

Both have many versions...

Out of scope today

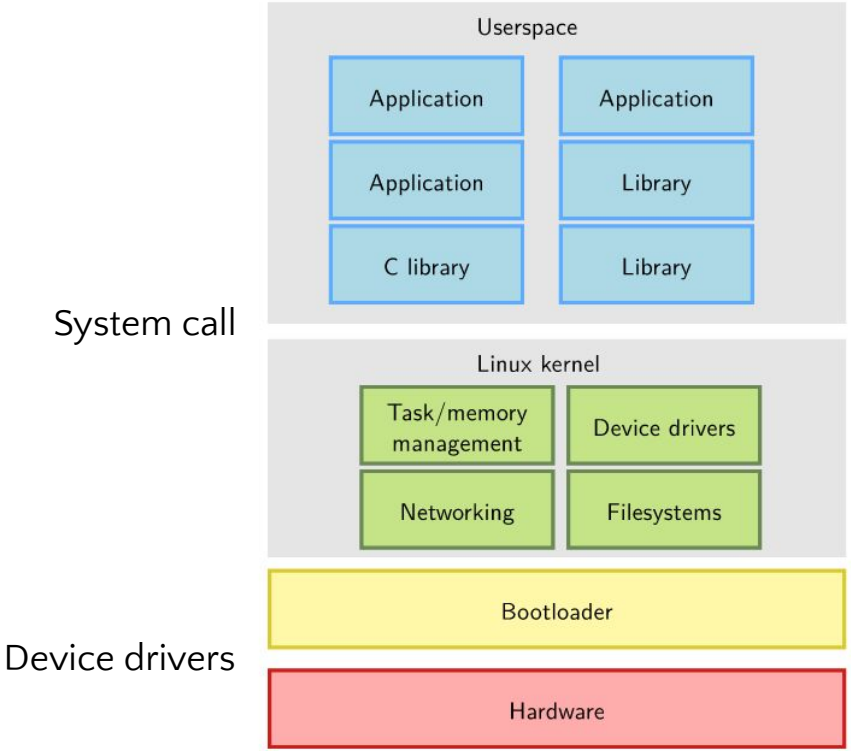
- Arduino, STM family
- RTOS, zephyr, etc
- etc

Linux System

for embedded

Linux history

- Open source system founded in 1991 par Linus Torvald
- huge community and contributor
- large hardware support
- tiny to huge computers
- security
- modularity



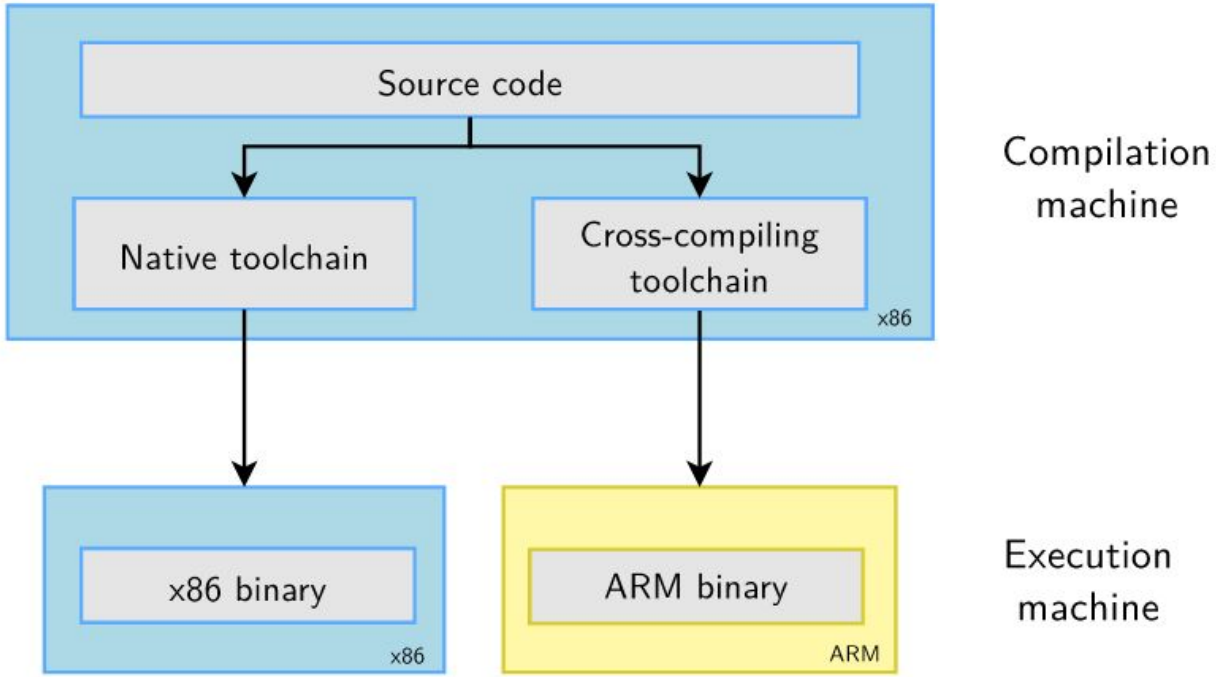
Linux architecture

Pillars of embedded Linux

- **Toolchain:**
  - The compiler and other tools needed to create code for your target device.
- **Bootloader:**
  - The program that initializes the board and loads the Linux kernel.
- **Kernel**:
  - This is the heart of the system, managing system resources and interfacing with hardware.
- **Root filesystem:**
  - Contains the libraries and programs that are run once the kernel has completed its initialization.

Toolchain

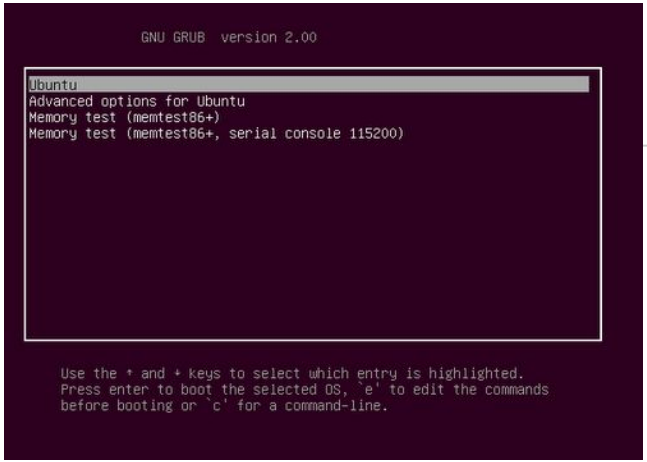
- Native:
  - This toolchain runs on the same type of system as the programs it generates.
- Cross:
  - runs on a different type of system than the target
  - the development to be done on a fast desktop

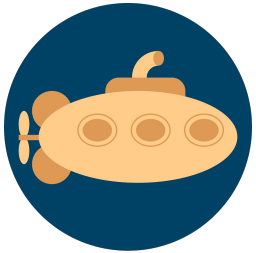


sources image : Bootlin

Bootloader

- load hardware config
  - cpu, memory, etc
- load kernel
- execute kernel





Kernel Linux component

- **Architecture code :**
  - provides low-level routines for very basic (register manipulation, synchronization primitives, etc.)
- **Drivers** :
  - ships drivers for every device supported by Linux in one source code tree.
  - some are not useful for your usage
- **Device tree** :
  - explains to the kernel how hardware is actually connected to the system.
  - are the “config files” for Linux drivers
  - driver will find the entry and set hardware

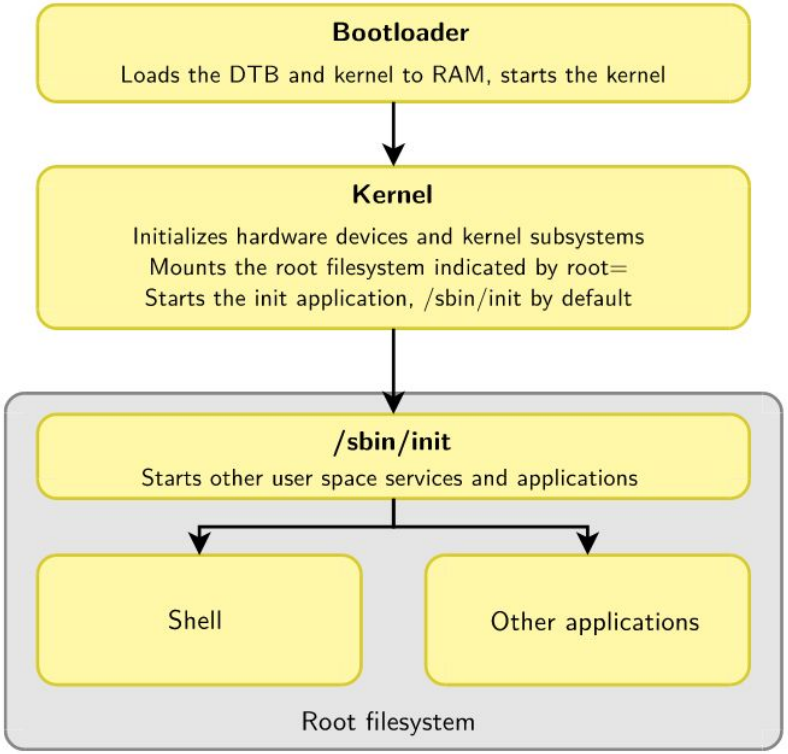
FileSystem

- directories and files are organized as a hierarchy
- organize data in directories and files on storage devices
- allows applications to access files and directories easily

Filesystem directory hierarchy

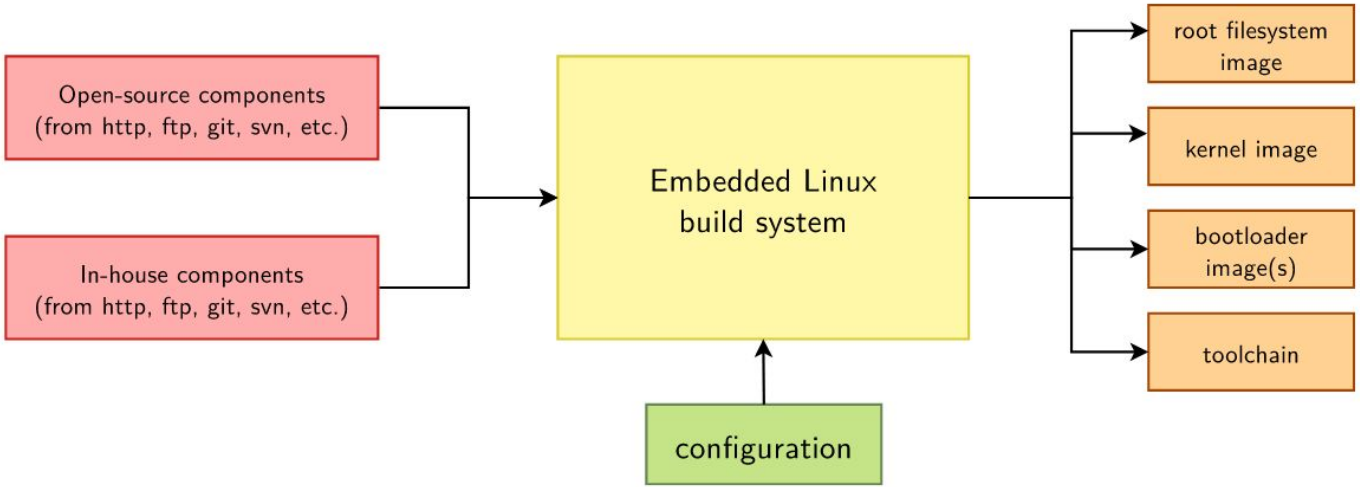
- **/ : root filesystem (first mounted)**

- **/bin:** Basic programs
- **/boot:** Kernel images, configurations and initramfs
- **/dev** Device files
- **/etc** System-wide configuration
- **/home** Directory for the users home directories
- **/lib** Basic libraries
- **/media** Mount points for removable media
- **/mnt** Mount points for static media
- **/proc** Mount point for the proc virtual filesystem
- **/root** Home directory of the root user
- **/sbin** Basic system programs
- **/sys** Mount point of the sysfs virtual filesystem
- **/tmp** Temporary files
- **/usr**
  - **/usr/bin** Non-basic programs
  - **/usr/lib** Non-basic libraries
  - **/usr/sbin** Non-basic system programs
- **/var** Variable data files, for system services.



All blocks needed

How to build a custom system ?



Building workflow

Top Down

- start with desktop Distribution
- remove unnecessary libs / apps

- complex (many shared library)
- still big

Bottom Up

- start with minimalist root filesystem
- add what needed

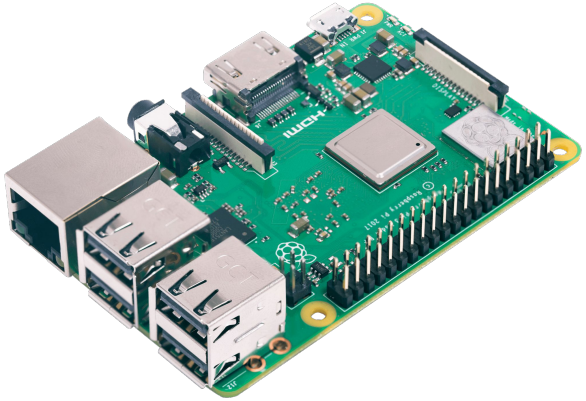
- easier maintenance
- small image

Many Options

- Manually with custom scripts
- Building tools : BuildRoot, Yocto
- distributions

BuildRoot

custom config



Our famous RaspberryPI

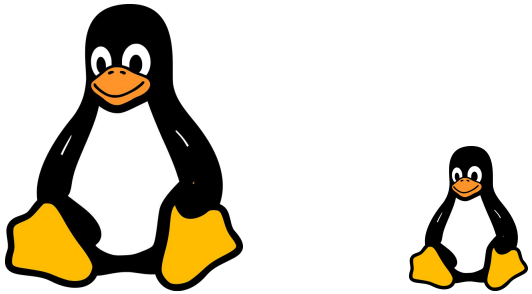
“

Desktop version

- Release date: January 28th 2022
- System: 64-bit
- Kernel version: 5.10
- Debian version: 11 (bullseye)
- Size: **1,135 GB**

OS Lite

- Release date: January 28th 2022
- System: 64-bit
- Kernel version: 5.10
- Debian version: 11 (bullseye)
- Size: **435MB**

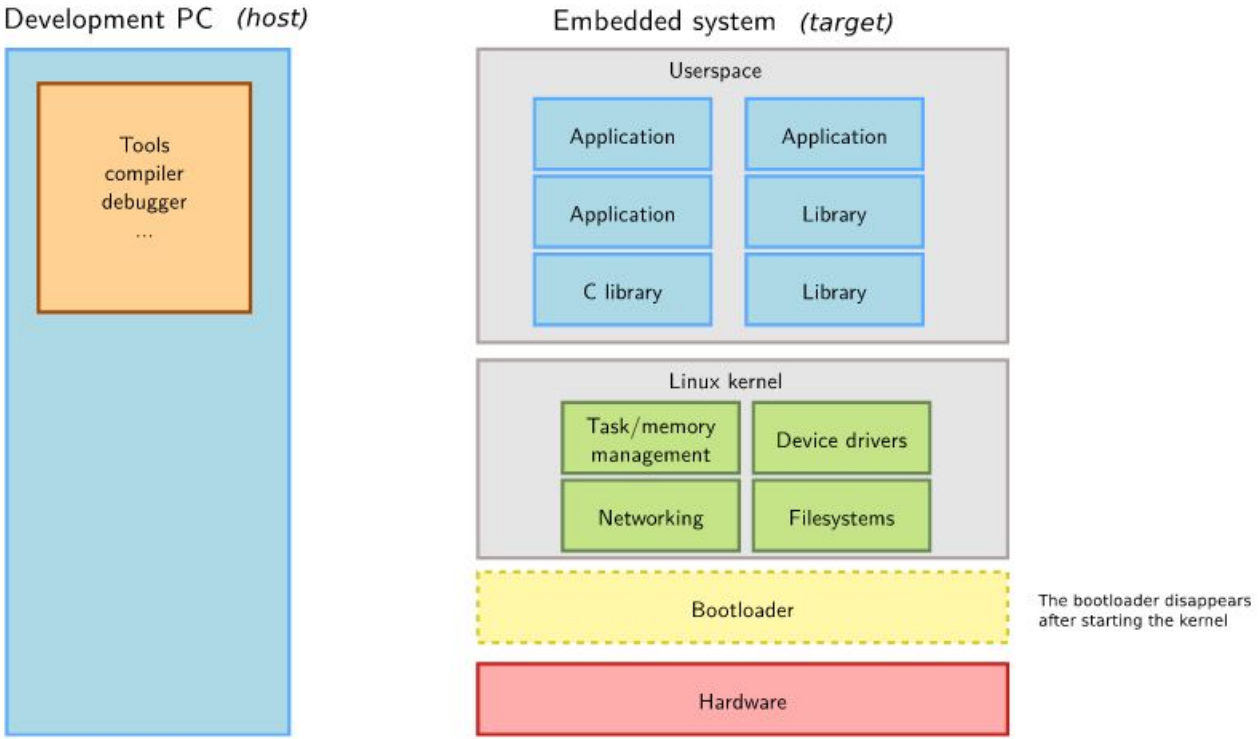


How to reduce size ?

“

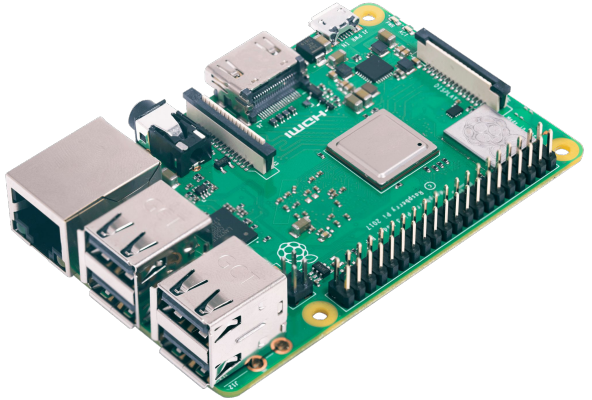
Buildroot steps

- Choose target device
- Customize configuration
- Compiling the Linux kernel
- Copy Root filesystem to card
- Boot with new image



Reminder Embedded Linux System architecture

X86



ARM

Not same architecture… need toolchain for cross building

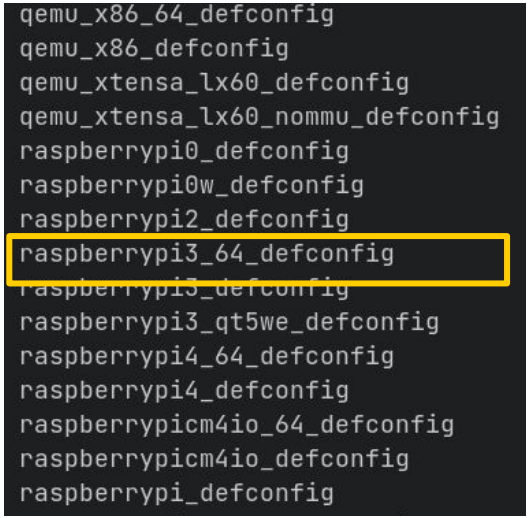
Toolchain with BuilRoot

- BuildRoot provides own toolchain
- Lib to compile C
  - uClibc is selected by default
  - other available:
    - glibc, musl, etc
    - depends on your apps

Nb: you can download binaries too if wanted

Choose Board config

- choose among thousands configs... (271)
  - raspberrypi3_64_defconfig
- generate config
  - make O=../build-rpi3 \
    raspberrypi3_64_defconfig
- config is written to .config file



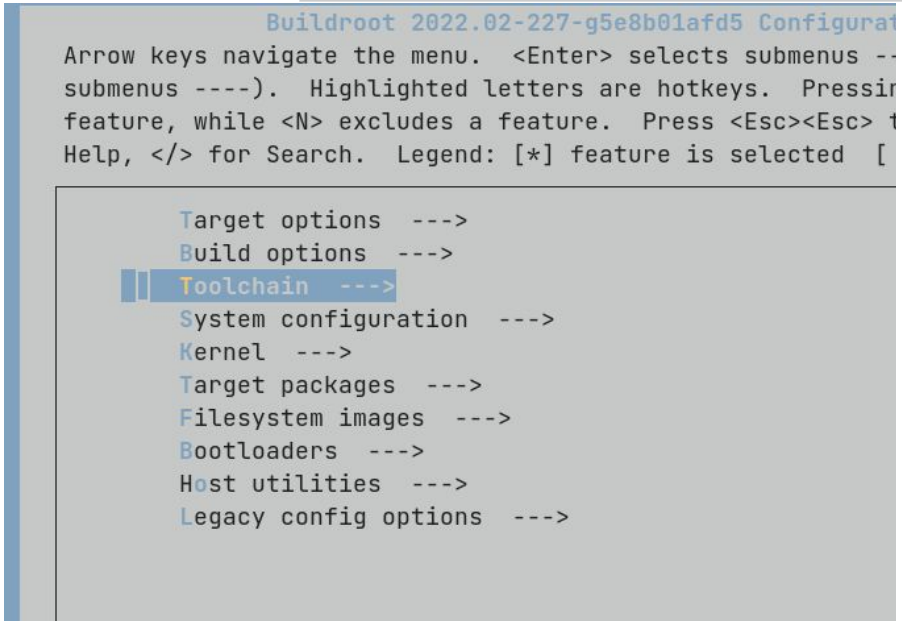
```
#
# Automatically generated file; DO NOT EDIT.
# Buildroot 2022.02-227-g5e8b01afd5 Configuration
#
BR2_HAVE_DOT_CONFIG=y
BR2_HOST_GCC_AT_LEAST_4_9=y
BR2_HOST_GCC_AT_LEAST_5=y
BR2_HOST_GCC_AT_LEAST_6=y
BR2_HOST_GCC_AT_LEAST_7=y
BR2_HOST_GCC_AT_LEAST_8=y
BR2_HOST_GCC_AT_LEAST_9=y

#
# Target options
#
BR2_ARCH_IS_64=y
BR2_ARCH_HAS_MMU_MANDATORY=y
BR2_ARCH_HAS_MMU_OPTIONAL=y
# BR2_arcle is not set
# BR2_arceb is not set
# BR2_arm is not set
# BR2_armeb is not set
```

.config content file with preconfigured constants

Configuration

- kconfig format
  - same as linux kernel
- Store changes in .config



```
➜  build-rpi3 make -j 8
/usr/bin/make -j1 O=/data/projects/mine/embedded/build-rpi3 HOSTCC="/usr/bin/gc
CXX="/usr/bin/g++" syncconfig
make[2]: warning: -j1 forced in submake: resetting jobserver mode.
  GEN     /data/projects/mine/embedded/build-rpi3/Makefile
>>> host-skeleton  Extracting
>>> host-skeleton  Patching
>>> host-skeleton  Configuring
```

Very long time first time ....
(activate cache in buildroot options)

```
-firmware/overlays' '::'" (stderr):
INFO: vfat(boot.vfat): adding file 'Image' as 'Image' ...
INFO: vfat(boot.vfat): cmd: "MTOOLS_SKIP_CHECK=1 mcopy -sp -i '/data/projects/mine/em
edded/build-rpi3/images/boot.vfat' '/data/projects/mine/embedded/build-rpi3/images/Ima
ge' '::'" (stderr):
INFO: hdimage(sdcard.img): adding partition 'boot' (in MBR) from 'boot.vfat' ...
INFO: hdimage(sdcard.img): adding partition 'rootfs' (in MBR) from 'rootfs.ext4' ...
INFO: hdimage(sdcard.img): adding partition '[MBR]' ...
INFO: hdimage(sdcard.img): writing MBR
```

Compile with make

- device tree
  - *.dtb
- Kernel
  - Image
- bootloader :
  - rpi-firmware
- partitions
  - boot.vfat
  - rootfs.ext4
- Image to flash
  - sdcard.img

```
→  build-rpi3 l images
total 219M
drwxr-xr-x 3 jrollin jrollin 4,0K mars  23 18:54 .
drwxrwxr-x 6 jrollin jrollin 4,0K mars  23 18:54 ..
-rwxr-xr-x 1 jrollin jrollin  30K mars  23 18:54 bcm2710-rpi-3-b.dtb
-rwxr-xr-x 1 jrollin jrollin  31K mars  23 18:54 bcm2710-rpi-3-b-plus.dtb
-rwxr-xr-x 1 jrollin jrollin  21K mars  23 18:54 bcm2837-rpi-3-b.dtb
-rw-r--r-- 1 jrollin jrollin  32M mars  23 18:54 boot.vfat
-rw-r--r-- 1 jrollin jrollin  20M mars  23 18:54 Image
-rw-r--r-- 1 jrollin jrollin 120M mars  23 18:54 rootfs.ext2
lrwxrwxrwx 1 jrollin jrollin   11 mars  23 18:54 rootfs.ext4 -> rootfs.ext2
drwxr-xr-x 3 jrollin jrollin 4,0K mars  23 18:47 rpi-firmware
-rw-r--r-- 1 jrollin jrollin 153M mars  23 18:54 sdcard.img
```

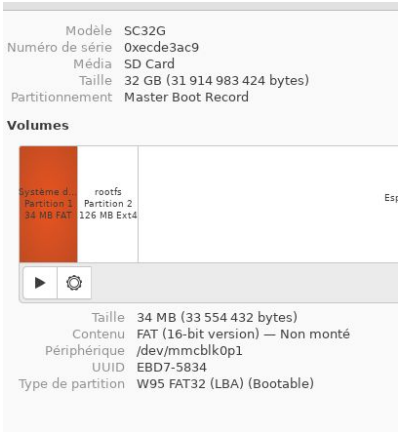
compiled files

```
build-rpi3 sudo dd if=images/sdcard.img of=/dev/mmcblk0 bs=1M status=progress
152+1 enregistrements lus
152+1 enregistrements écrits
159384064 octets (159 MB, 152 MiB) copiés, 8,55373 s, 18,6 MB/s
```

see partitions on card with lsblk

```
mmcblk0     179:0    0  29,7G  0 disk
├─mmcblk0p1 179:1    0    32M  0 part
└─mmcblk0p2 179:2    0   120M  0 part
```

or with GUI

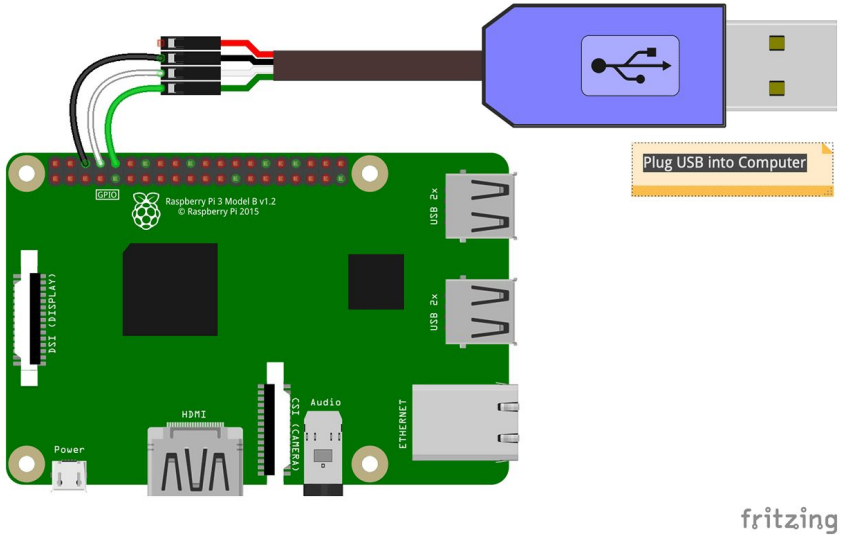


Copy image to flash card

sudo screen -fn /dev/ttyUSB0 115200

```
dnsmasq: failed to open pidfile /var/run/dnsmasq.pid: No such file or directory
Jan  1 00:00:08 buildroot daemon.crit dnsmasq[185]: failed to open pidfile /var/run/d
smasq.pid: No such file or directory
Jan  1 00:00:08 buildroot daemon.crit dnsmasq[185]: FAILED to start up
FAIL
Starting hostapd: wlan0: interface state UNINITIALIZED[    8.856105] IPv6: ADDRCONF(N
TDEV_CHANGE): wlan0: link becomes ready
->ENABLED
wlan0: AP-ENABLED
OK
Jan  1 00:00:08 buildroot daemon.info : starting pid 191, tty '/dev/console': '/sbin/
etty -L  console 0 vt100 '
Jan  1 00:00:08 buildroot kern.info kernel: [    8.856105] IPv6: ADDRCONF(NETDEV_CHAN
E): wlan0: link becomes ready
Jan  1 00:00:08 buildroot daemon.info : starting pid 192, tty '/dev/tty1': '/sbin/get
y -L  tty1 0 vt100 '

Welcome to Buildroot
buildroot login:
```



screen to Connect to the Serial Console

Demo time !

“

Tiny and Fast Linux system !

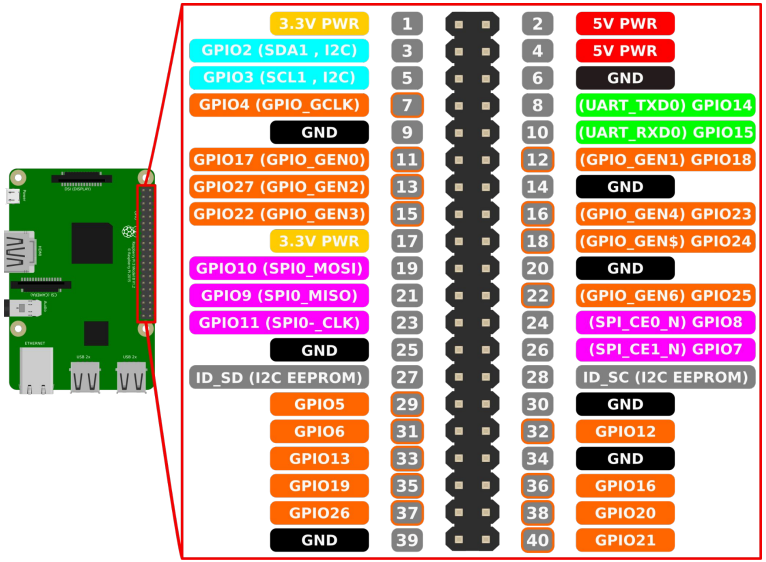
“

Need more customization

- change hostname
- add root password...
- add users
- add packages (more than 2600)
- add custom partitions
- ...

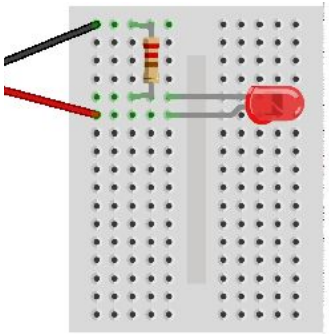
Bonus !

“



Ground

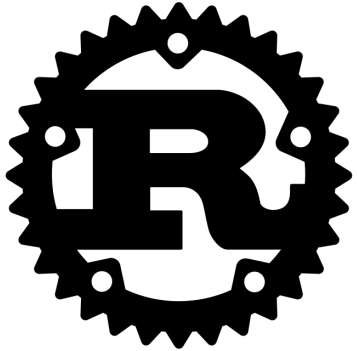
pin 26



The famous blink project ！

Rust to interact with Gpio

- crates HAL
  - Hardware Abstraction Layer (HAL) for embedded systems
- crates RPPAL
  - Raspberry Pi Peripheral Access Library
  - controls the GPIO peripheral by directly accessing the registers



Rust Cross Build

$ cargo install cross (docker based tool)

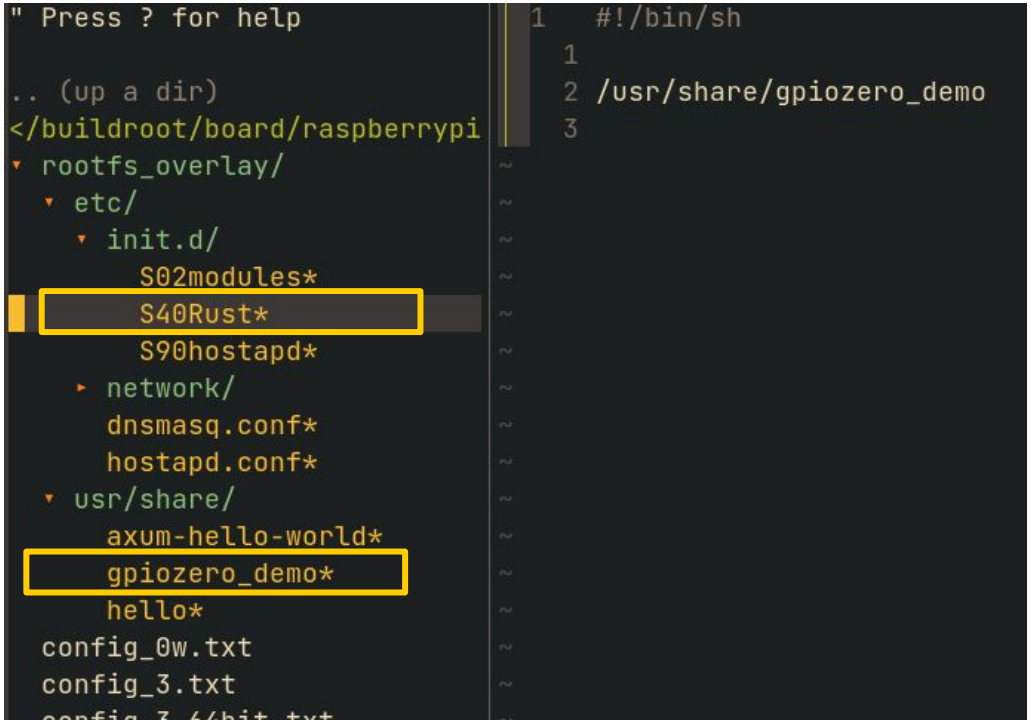
$ cross build --target=armv7-unknown-linux-musleabihf
--release

Not in BuildRoot...
How to add my App ?

script content

script launched at startime

rust app release for arm



custom root filesystem with custom scripts

Make to rebuild image
Copy to Sd card

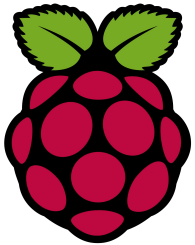
“

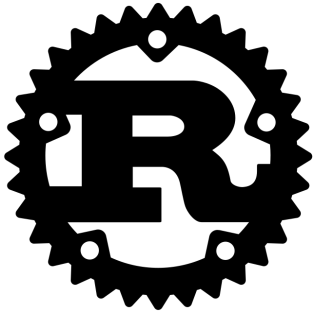
Démo !

“

What next ?

- create custom package for BuildRoot
- Yocto as replacement for buildroot ?
- update over the air with certificates
- more performance and power efficiency
- more Rust :)





Merci !

“

[Mastering Embedded Linux, Part 1: Concepts • &> /dev/null]

[Mastering Embedded Linux, Part 3: Buildroot • &> /dev/null]

[Créer un système complet avec Buildroot]

[Bootlin training Buildroot]

Framboise 314

bootlin

Sources / références