ELC Europe 2016

Arturo Rinaldi

LininoOS, LininoIO and ArduinoOS : a suitable ecosystem for Linux and MCUs

ARDUINO | Embedded Linux Conference Europe

Quick Bio

Who am I ?

Senior Software Engineer, working for Arduino Srl since 2014

My main tasks :

- Linino distro (embedded linux distro) maintaining
- Plain Toolchains and cross-platform ones (mips, mingw-w64, arm etc.)
- Kernel Drivers backporting
- Shell scripting
- QA
- Support

LininoOS - What is LininoOS ?

Linino is the combination of two well known words around the world :

Linux + Arduino

LininoOS (rings a bell to you or not ?!?) is in an embedded linux distro relying its core on **OpenWRT** which is a well known embedded linux distro, unleashing to their limits the capabilities of the most popular SOHO market routers.

Its repositories provide the most common unix productivity packages and programming languages such as :

- Python
- Ruby
- Erlang
- PHP
- many more counting…

LininoOS - An ecosystem, why not ?

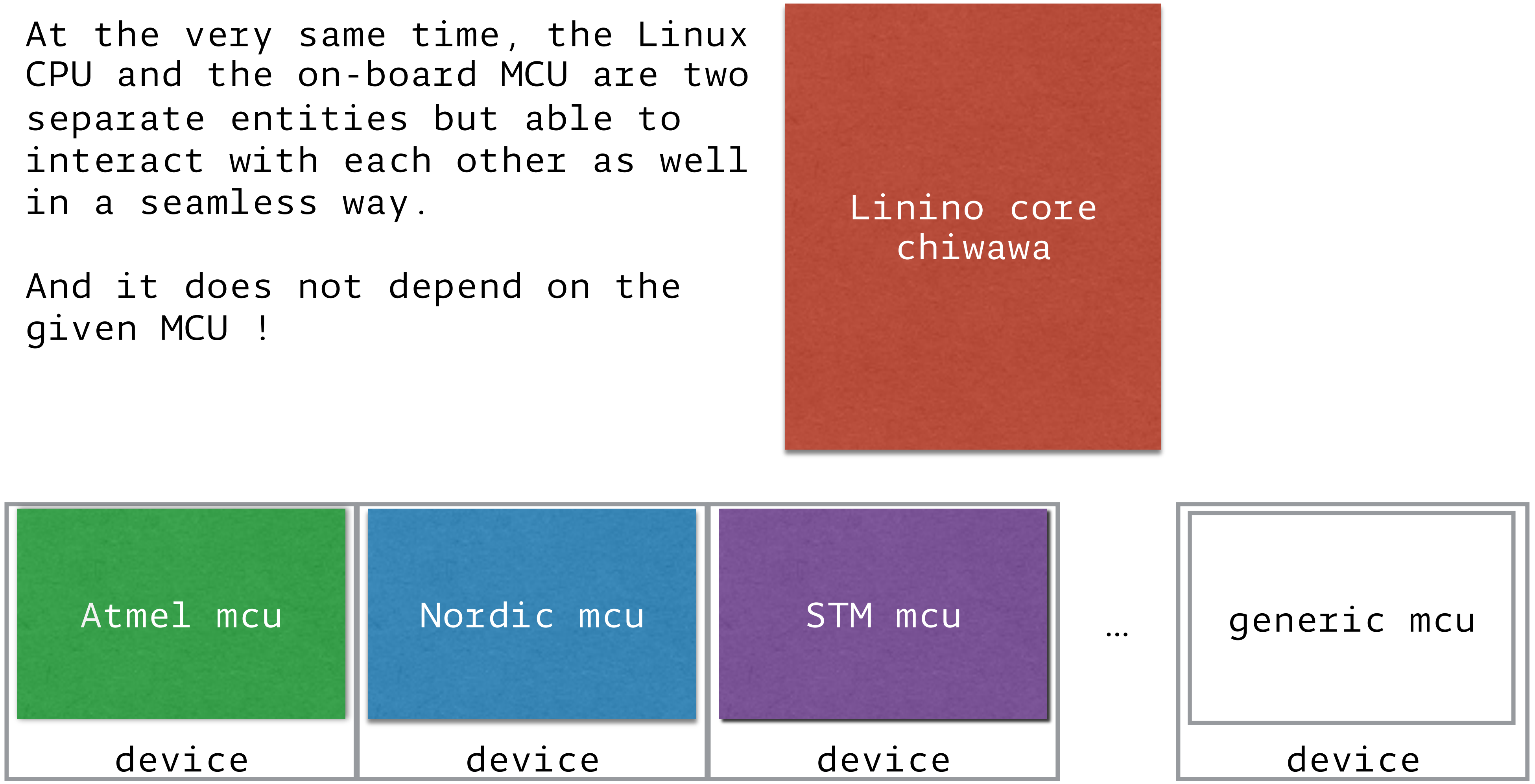
But….we'd like to think about it as more an **ecosystem** composed of:

- a set of wi-fi enabled **open hardware** modules to easily make things;
- an **open software** layer to easily put intelligence on the things;
- an **embedded** development environment;
- an **open cloud** based set of services that make things easily interconnected;
- **people** of course! evangelizers, enthusiasts, makers, hackers, … the ecosystem is for them.

LininoOS - MCU agnostic approach

At the very same time, the Linux CPU and the on-board MCU are two separate entities but able to interact with each other as well in a seamless way.

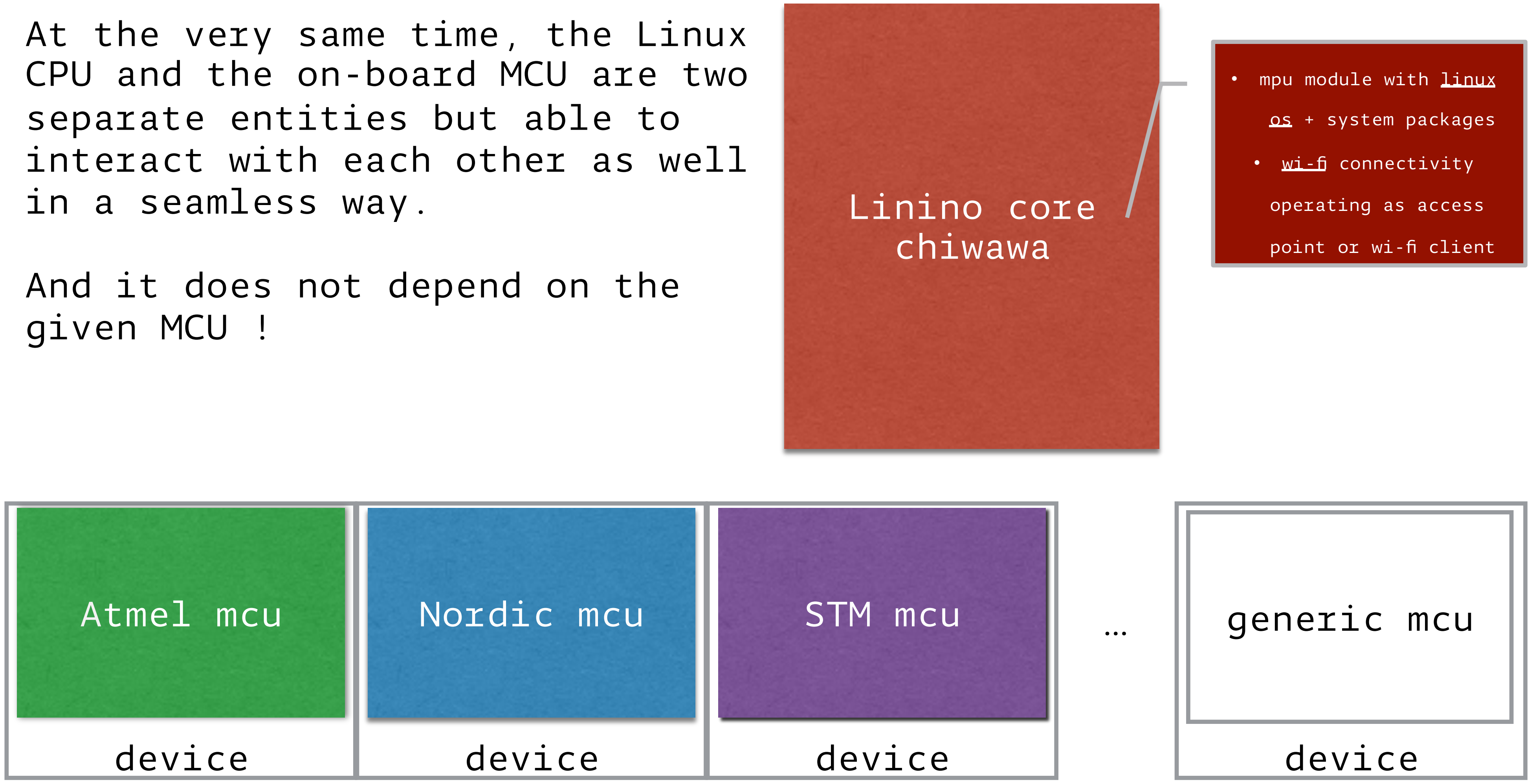
And it does not depend on the given MCU !



LininoOS - MCU agnostic approach

At the very same time, the Linux CPU and the on-board MCU are two separate entities but able to interact with each other as well in a seamless way.

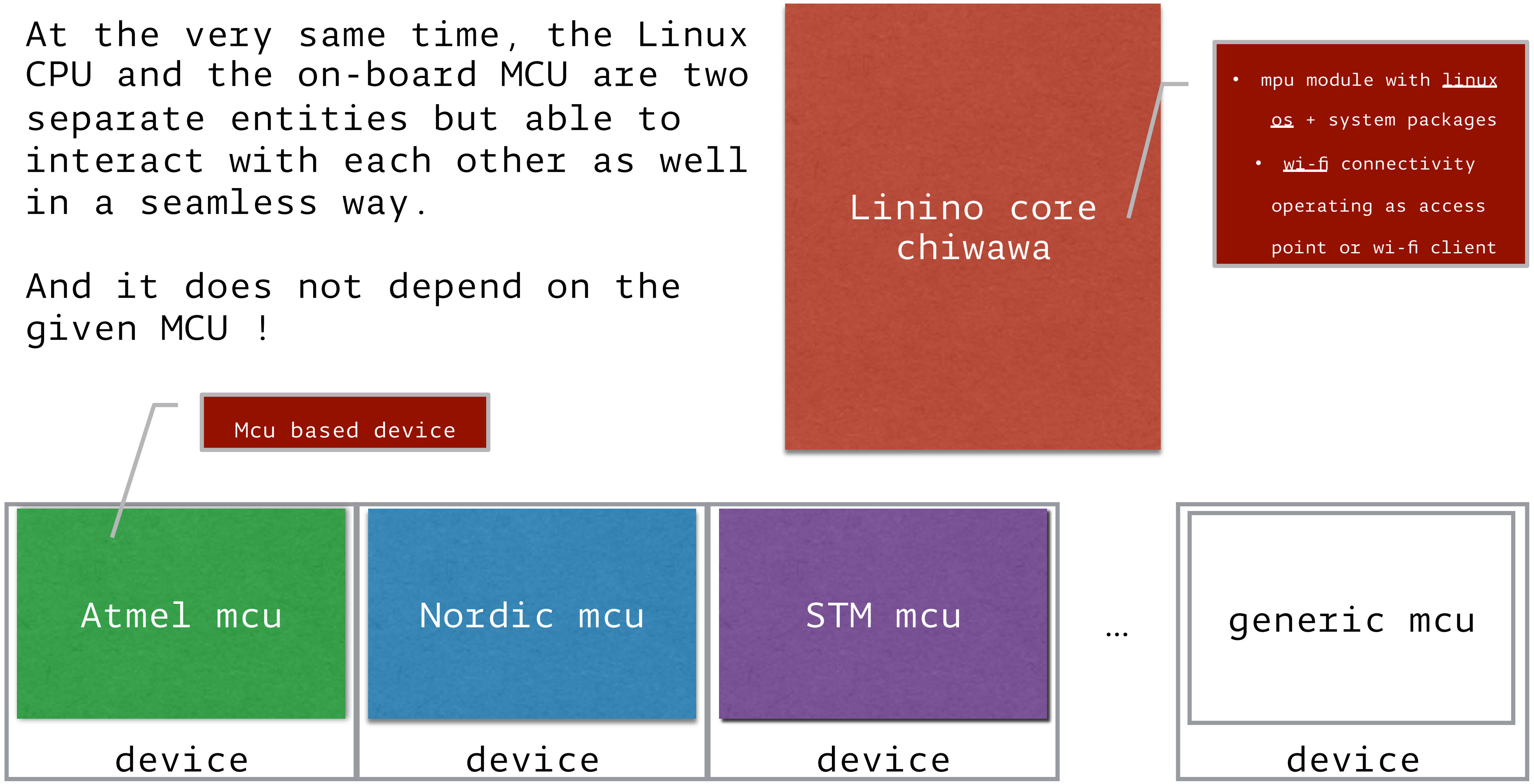
And it does not depend on the given MCU !


Linino core chiwawa
• mpu module with linux os + system packages
• wi-fi connectivity operating as access point or wi-fi client
Atmel mcu
Nordic mcu
STM mcu
…
generic mcu
device
device
device
device

LininoOS - MCU agnostic approach

At the very same time, the Linux CPU and the on-board MCU are two separate entities but able to interact with each other as well in a seamless way.

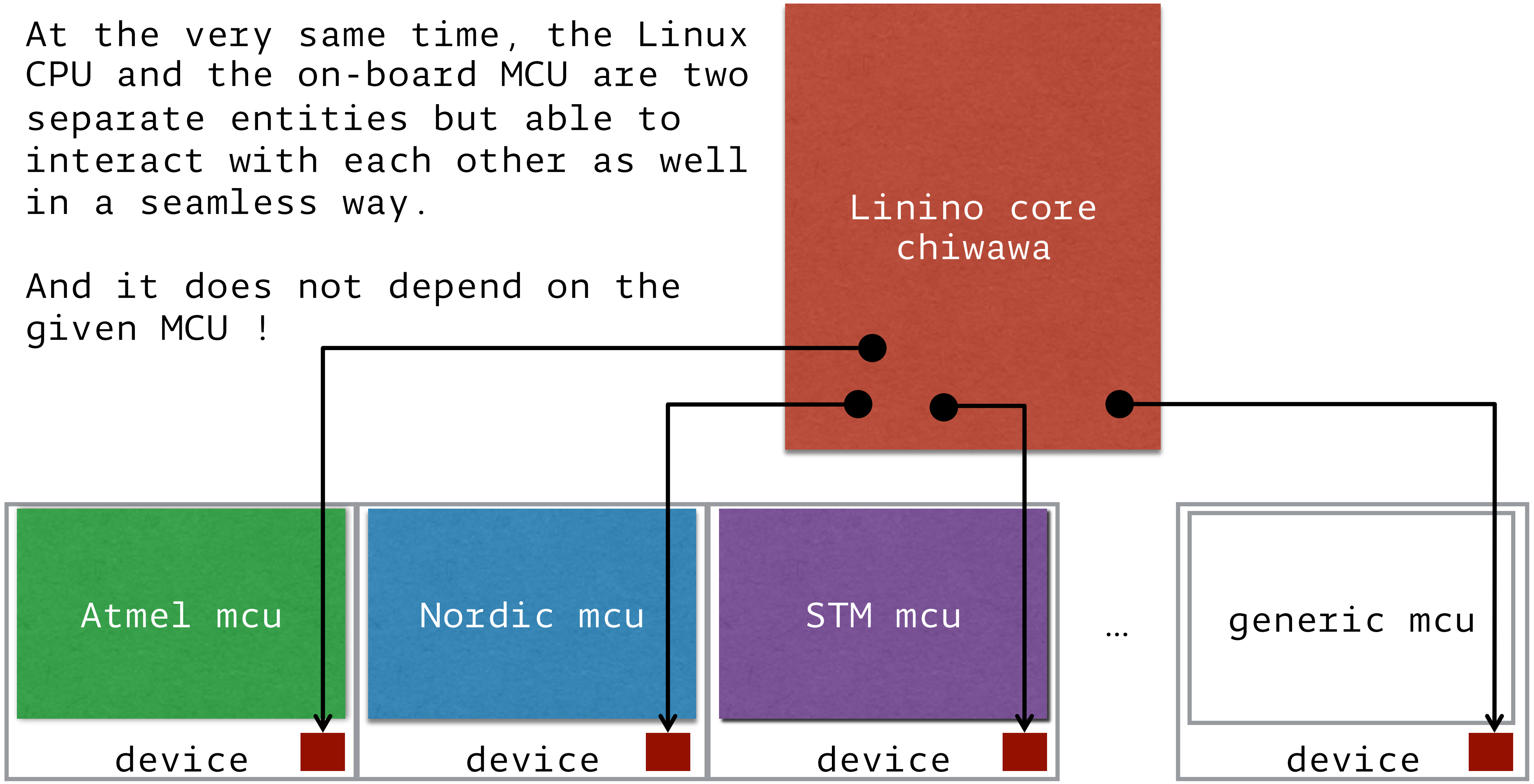
And it does not depend on the given MCU !


Linino core chiwawa
• mpu module with linux os + system packages
• wi-fi connectivity operating as access point or wi-fi client
Mcu based device
Atmel mcu
Nordic mcu
STM mcu
…
generic mcu
device
device
device
device

LininoOS - MCU agnostic approach

At the very same time, the Linux CPU and the on-board MCU are two separate entities but able to interact with each other as well in a seamless way.

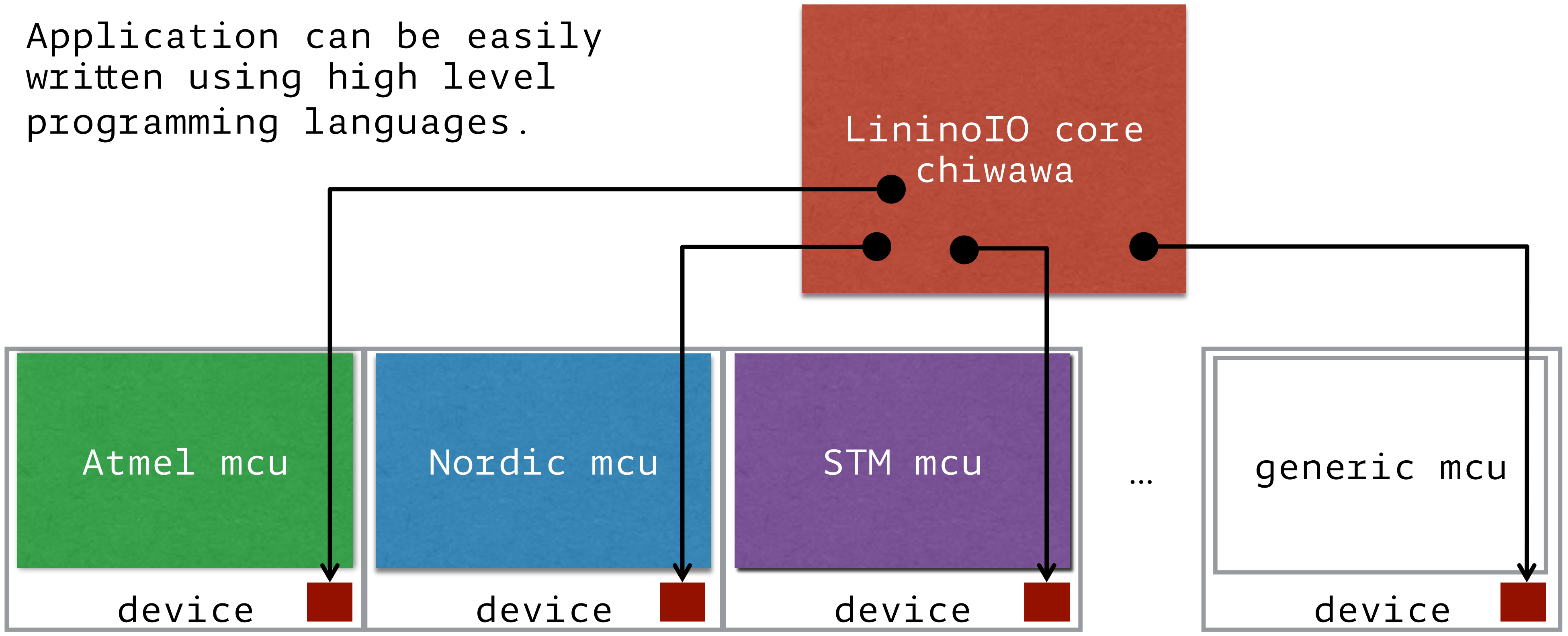
And it does not depend on the given MCU !



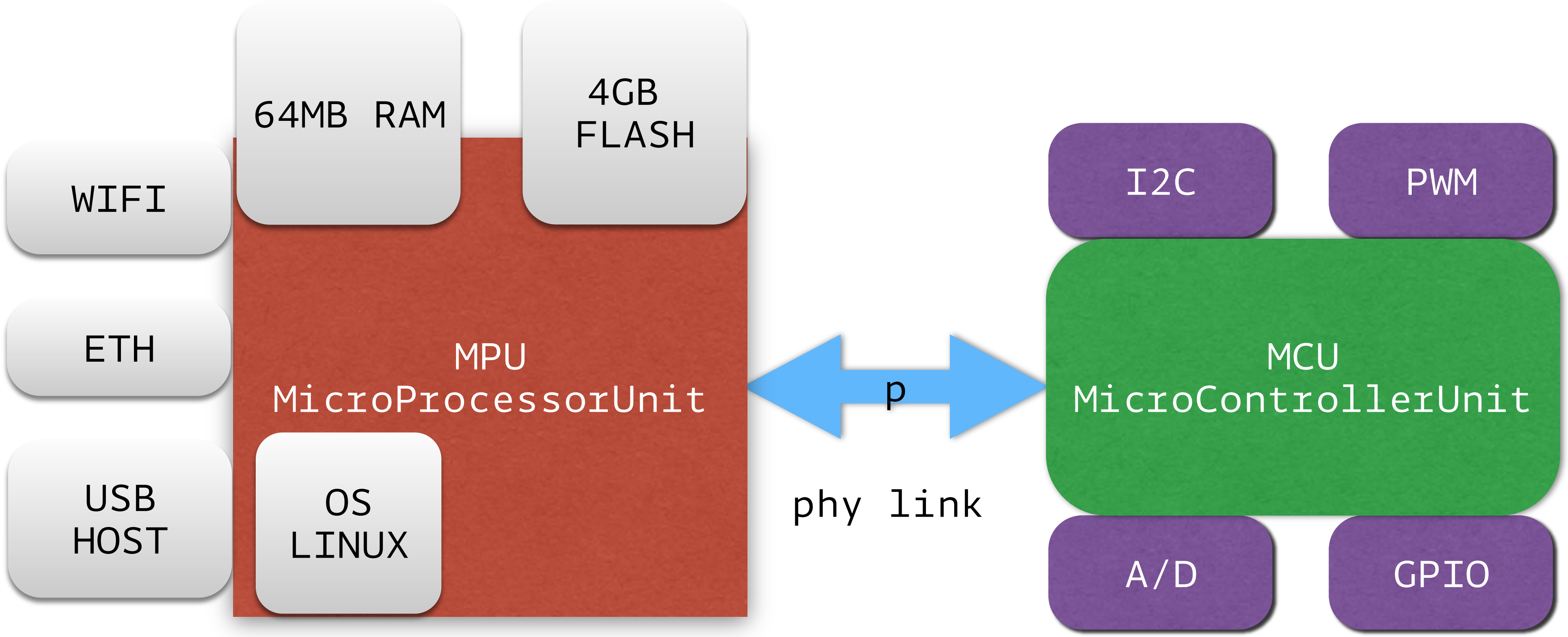
LininoOS – MCUIO approach

LininoIO is a software framework able to expose microcontroller features (such as GPIO, Analog Converters, PWM, I2C, SPI) inside the microprocessor environment in a typical UNIX fashion.

Application can be easily written using high level programming languages.

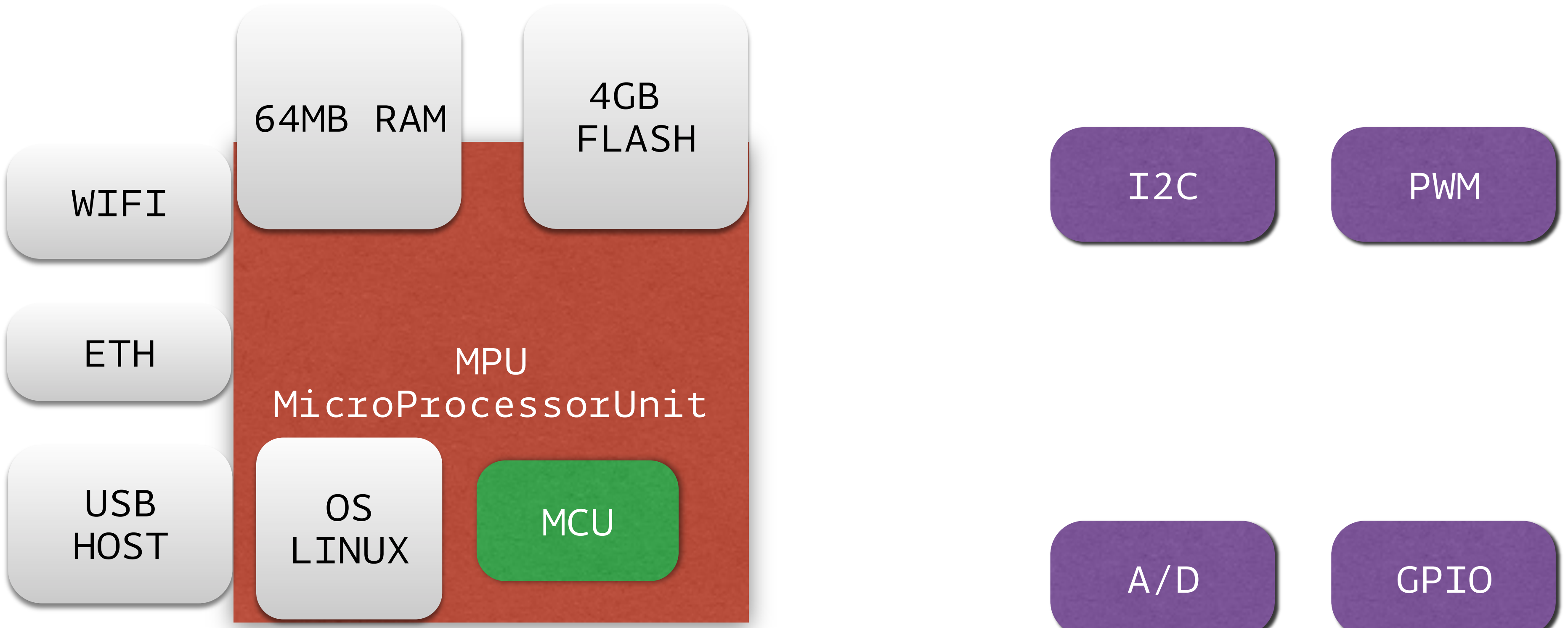


LininoIO – Overview



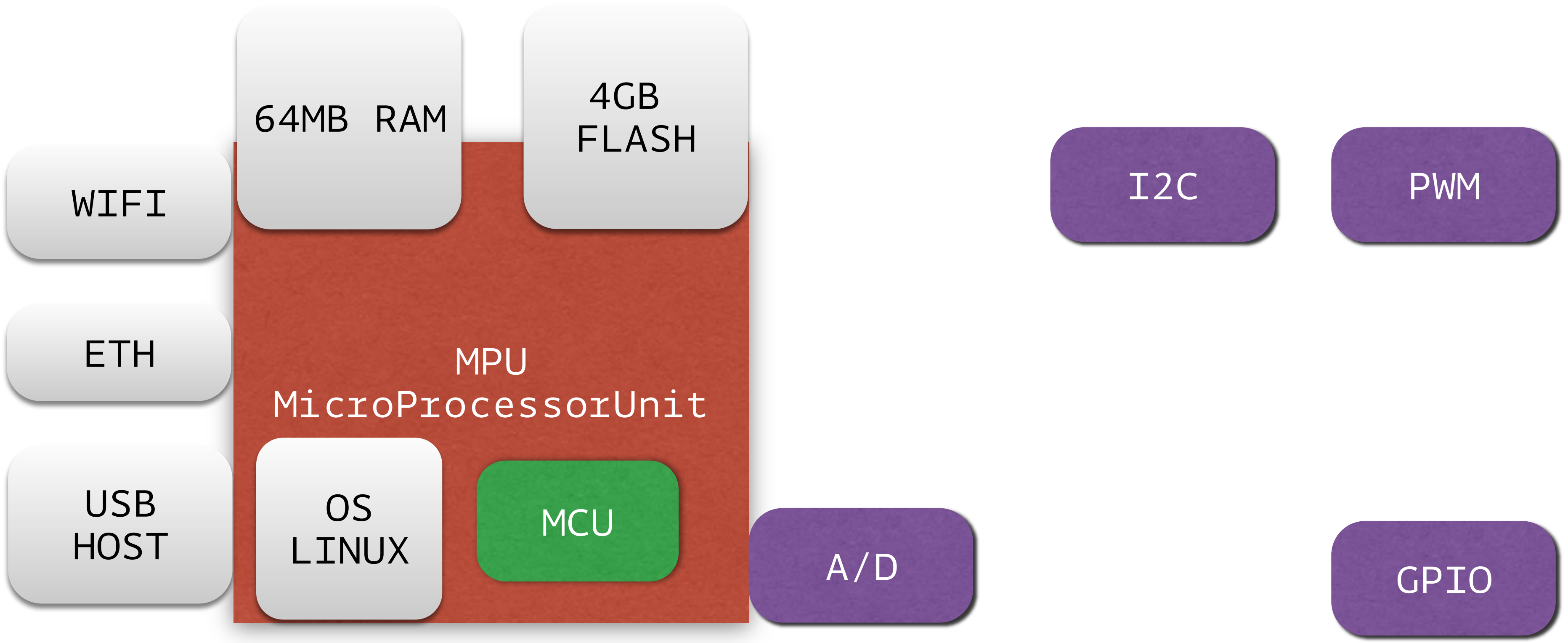
- **MPU**: possibly SMP, large memory, few peripherals, full featured OS.
- **MCU**: single CPU, small memory, many peripherals, special purpose OS.
- **phy link**: any possible on spi, uart, ethernet, wifi, …

LininoIO – Overview



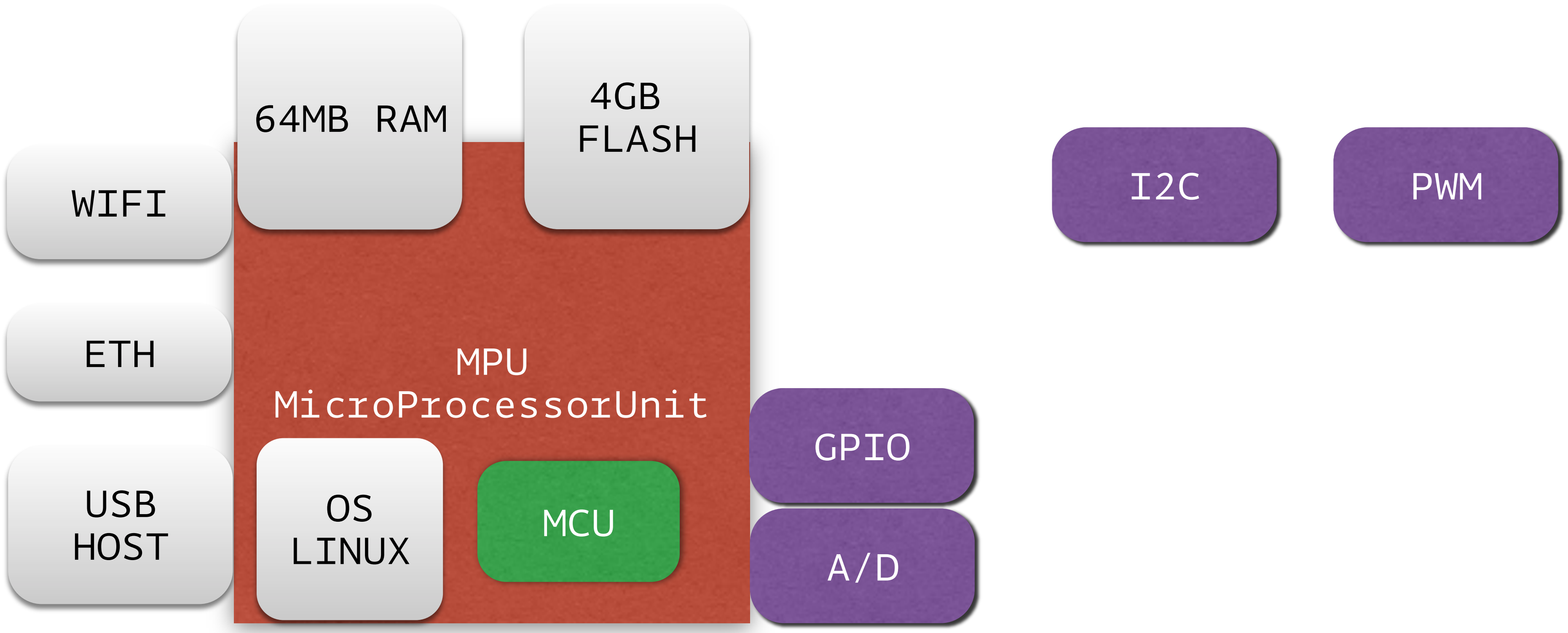
- **MPU**: possibly SMP, large memory, few peripherals, full featured OS.
- **MCU**: single CPU, small memory, many peripherals, special purpose OS.
- **phy link**: any possible on spi, uart, ethernet, wifi, …

LininoIO – Overview



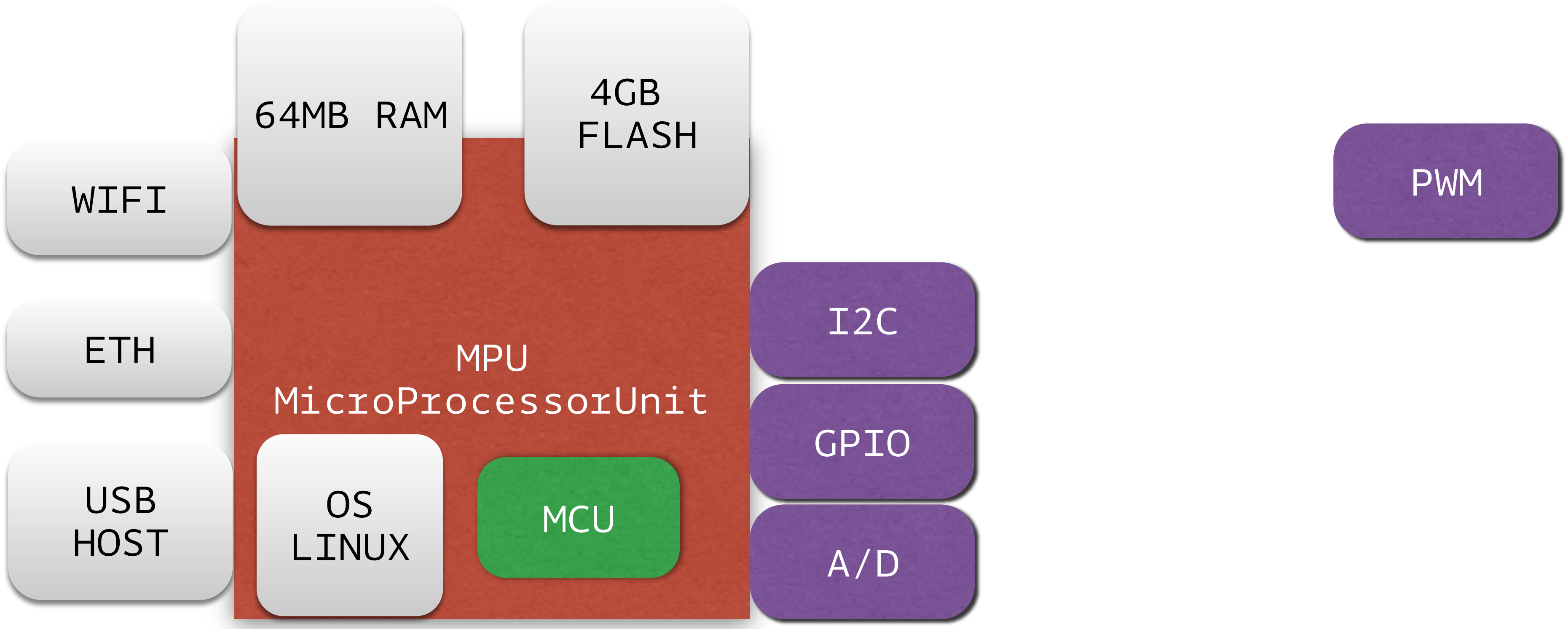
- **MPU**: possibly SMP, large memory, few peripherals, full featured OS.
- **MCU**: single CPU, small memory, many peripherals, special purpose OS.
- **phy link**: any possible on spi, uart, ethernet, wifi, …

LininoIO – Overview



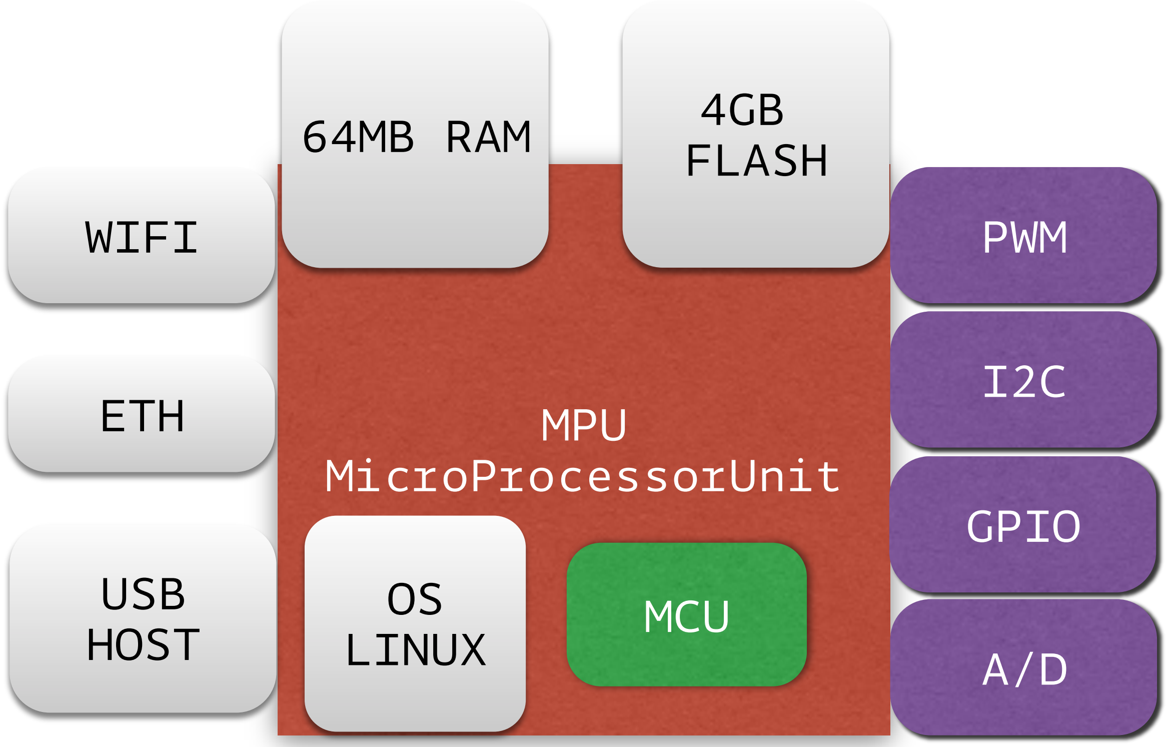
- **MPU**: possibly SMP, large memory, few peripherals, full featured OS.
- **MCU**: single CPU, small memory, many peripherals, special purpose OS.
- **phy link**: any possible on spi, uart, ethernet, wifi, …

LininoIO – Overview



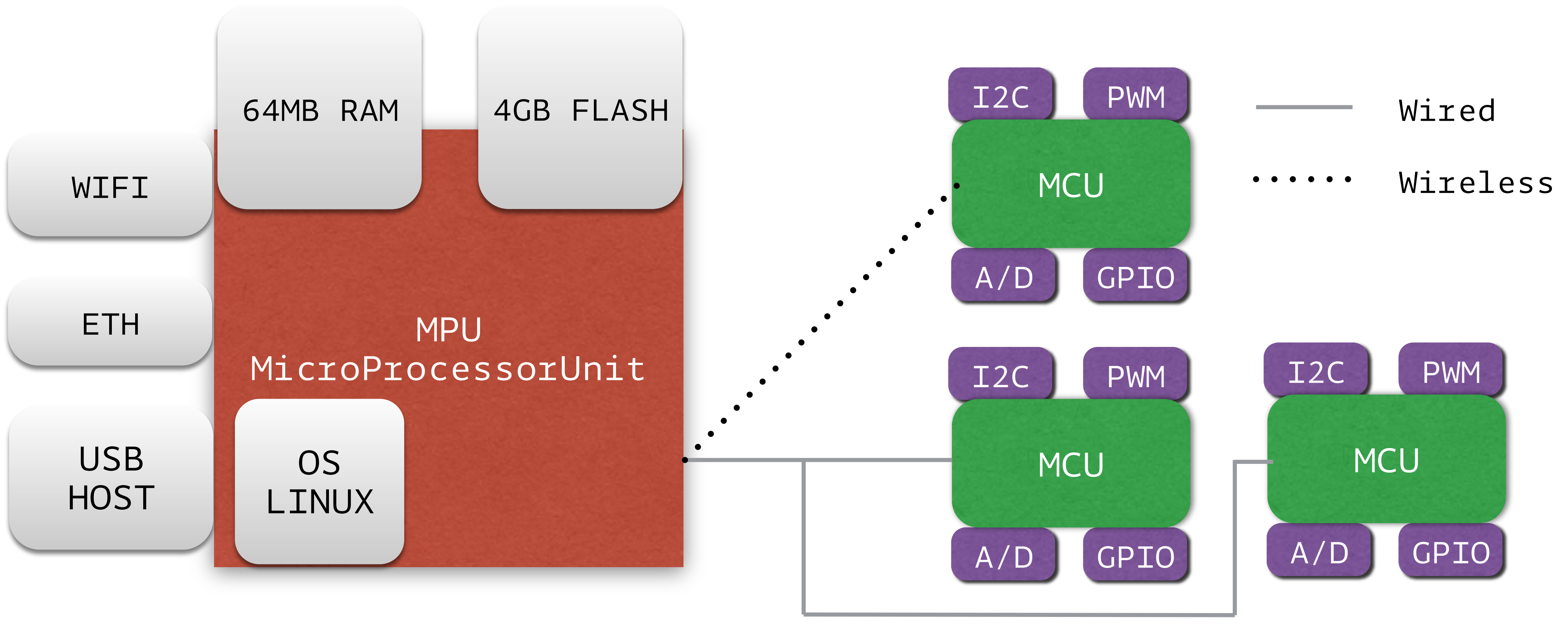
- **MPU**: possibly SMP, large memory, few peripherals, full featured OS.
- **MCU**: single CPU, small memory, many peripherals, special purpose OS.
- **phy link**: any possible on spi, uart, ethernet, wifi, …

LininoIO – Overview



- **MPU**: designed for best average performance (cache levels, virtual memory, etc): real-time is difficult
- **MCU**:much simpler: no cache, no VM: real time is guaranteed.
- *Keep high level applications on MPU and real-time stuff on MCU*

LininoIO – Overview



- Each of the MCU devices is discovered as a local lininoIO service
- They can be controlled directly by Linux side through : Shell, Python, Node.Js and any other high-level programming language

LininoIO – Overview

Please note that :

- **lininoio** (mcuio driver) devices are regular linux kernel devices, so they don't necessarily have to be controlled via user space (some kernel module-subsystem can use them). The most important thing about lininoio is that mcu peripherals become standard linux devices, so that the mpu cannot really tell where they are and how they're implemented (actually some devices such as the interrupt controller are just virtual).

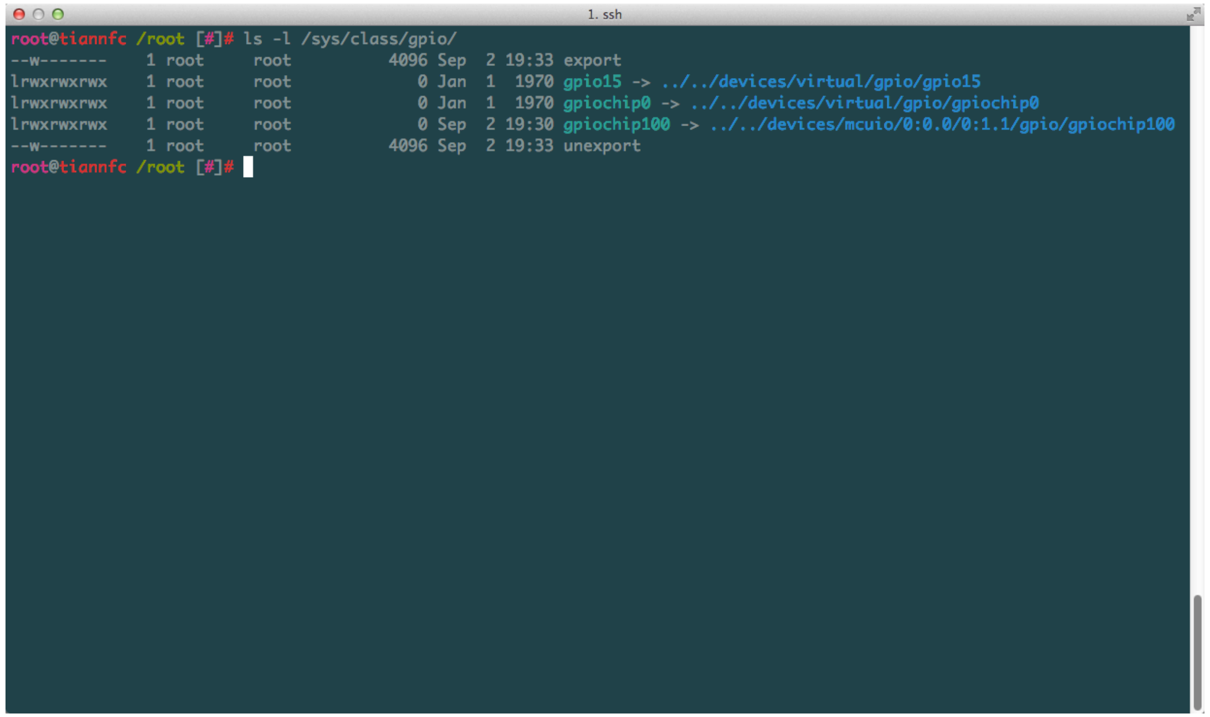
- **mcu devices** have interrupt capability (so you can see gpio interrupts on edge/level from user space or kernel, for instance).

For further reading, please point your browsers here :

Davide Ciminaghi : “Virtualizing MCU Peripherals”

LininoIO – Example 1

Turn on the L13 led with just a couple of shell commands :



```
root@tiannfc /root [#]# ls -l /sys/class/gpio/
--w-------    1 root     root          4096 Sep  2 19:33 export
lrwxrwxrwx    1 root     root             0 Jan  1  1970 gpio15 -> ../../devices/virtual/gpio/gpio15
lrwxrwxrwx    1 root     root             0 Jan  1  1970 gpiochip0 -> ../../devices/virtual/gpio/gpiochip0
lrwxrwxrwx    1 root     root             0 Sep  2 19:30 gpiochip100 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/gpiochip100
--w-------    1 root     root          4096 Sep  2 19:33 unexport
root@tiannfc /root [#]#
```

After running :

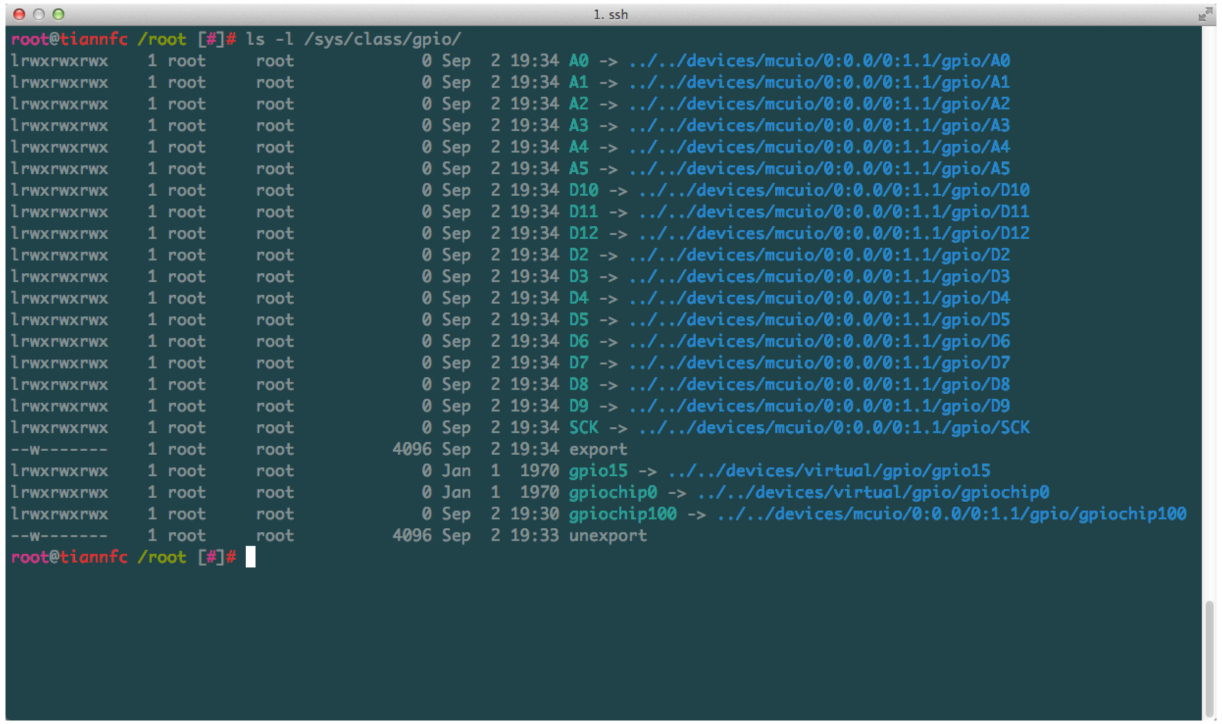
$ lininoio start

and rebooting, the presence of the :

/sys/class/gpio/gpiochip100

folder tells us that lininoIO is up and running as a background service.

LininoIO – Example 1



```
root@tiannfc /root [#]# ls -l /sys/class/gpio/
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 A0 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A0
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 A1 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A1
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 A2 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A2
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 A3 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A3
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 A4 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A4
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 A5 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A5
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 D10 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D10
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 D11 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D11
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 D12 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D12
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 D2 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D2
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 D3 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D3
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 D4 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D4
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 D5 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D5
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 D6 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D6
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 D7 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D7
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 D8 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D8
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 D9 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D9
lrwxrwxrwx    1 root     root             0 Sep  2 19:34 SCK -> ../../devices/mcuio/0:0.0/0:1.1/gpio/SCK
--w-------    1 root     root          4096 Sep  2 19:34 export
lrwxrwxrwx    1 root     root             0 Jan  1  1970 gpio15 -> ../../devices/virtual/gpio/gpio15
lrwxrwxrwx    1 root     root             0 Jan  1  1970 gpiochip0 -> ../../devices/virtual/gpio/gpiochip0
lrwxrwxrwx    1 root     root             0 Sep  2 19:30 gpiochip100 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/gpiochip100
--w-------    1 root     root          4096 Sep  2 19:33 unexport
root@tiannfc /root [#]#
```

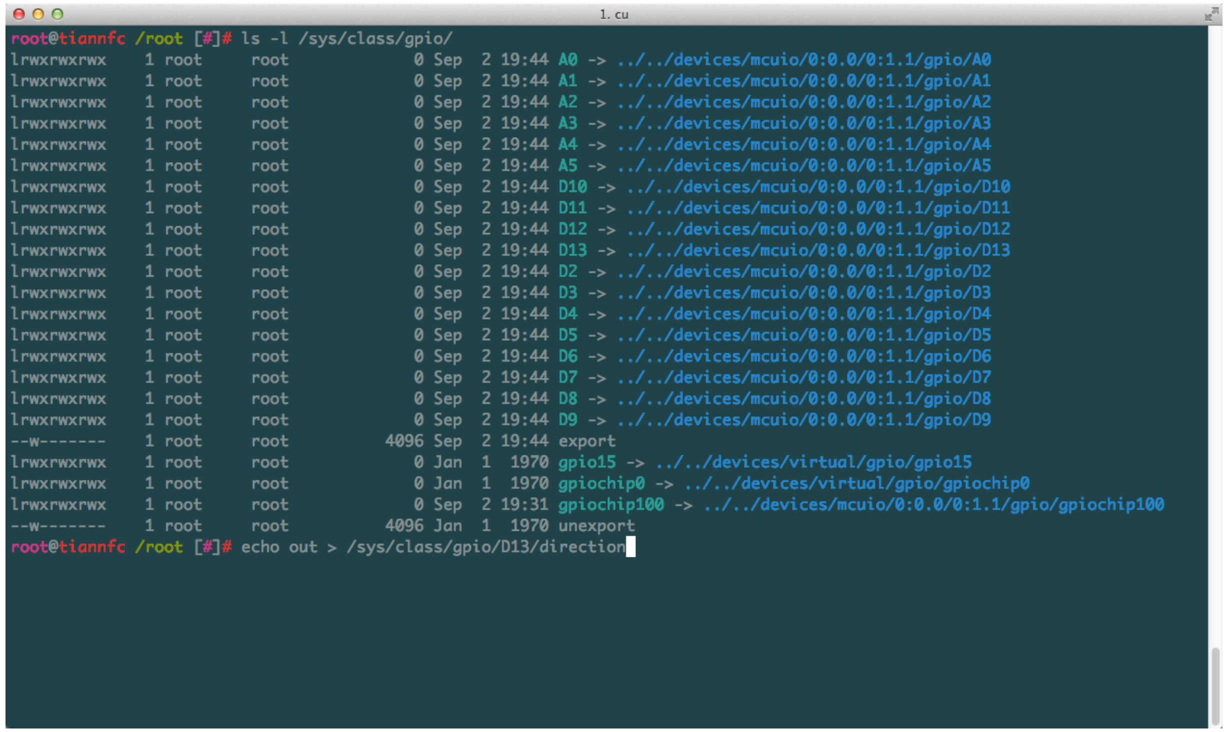
So now we run :

$ lingpio export

To enumerate all the available devices to use.

We are now ready to play around !

LininoIO – Example 1



```
root@tiannfc /root [#]# ls -l /sys/class/gpio/
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 A0 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A0
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 A1 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A1
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 A2 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A2
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 A3 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A3
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 A4 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A4
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 A5 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A5
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D10 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D10
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D11 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D11
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D12 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D12
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D13 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D13
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D2 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D2
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D3 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D3
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D4 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D4
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D5 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D5
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D6 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D6
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D7 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D7
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D8 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D8
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D9 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D9
--w-------    1 root     root          4096 Sep  2 19:44 export
lrwxrwxrwx    1 root     root             0 Jan  1  1970 gpio15 -> ../../devices/virtual/gpio/gpio15
lrwxrwxrwx    1 root     root             0 Jan  1  1970 gpiochip0 -> ../../devices/virtual/gpio/gpiochip0
lrwxrwxrwx    1 root     root             0 Sep  2 19:31 gpiochip100 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/gpiochip100
--w-------    1 root     root          4096 Jan  1  1970 unexport
root@tiannfc /root [#]# echo out > /sys/class/gpio/D13/direction
```

We want to turn on/off the L13 led of the board.

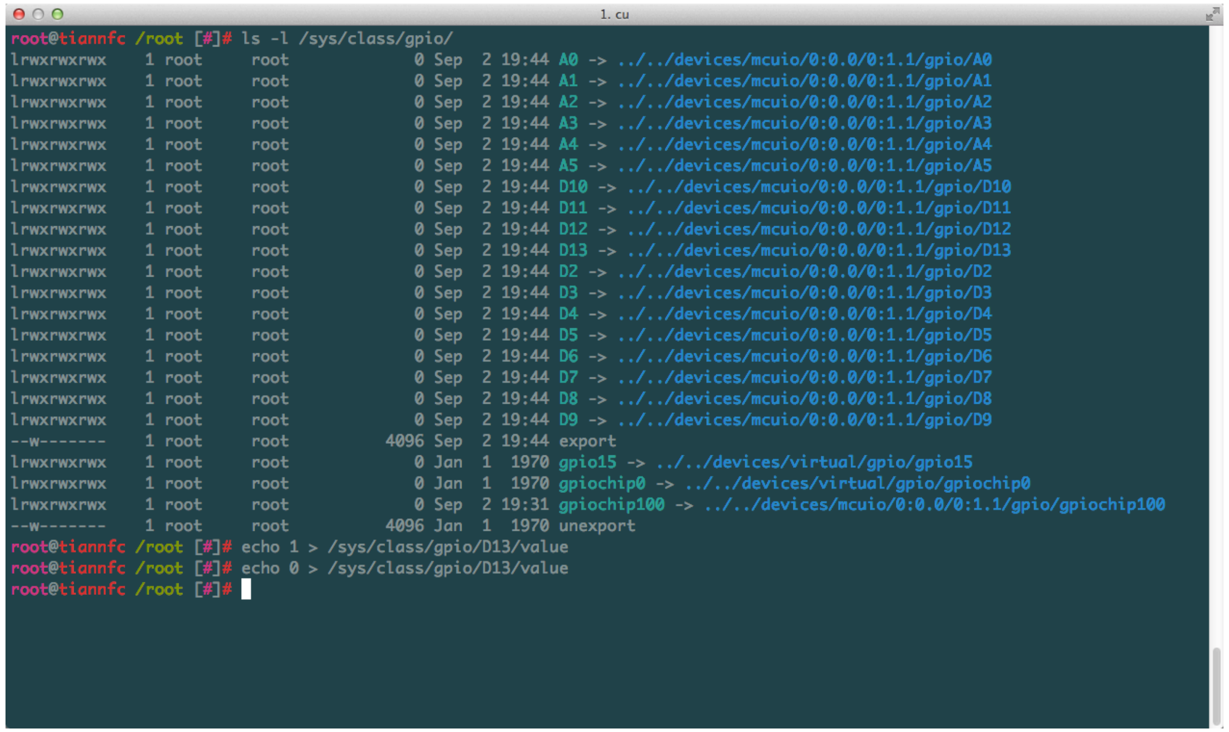
We have to start with exporting its ‘direction’ to **out** :

$ echo out > /sys/class/gpio/D13/direction

You have to do that according to your needs for a given pin.

You might also considering the **in** direction for some use cases.

LininoIO – Example 1



1. cu

```
root@tiannfc /root [#]# ls -l /sys/class/gpio/
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 A0 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A0
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 A1 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A1
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 A2 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A2
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 A3 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A3
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 A4 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A4
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 A5 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/A5
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D10 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D10
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D11 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D11
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D12 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D12
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D13 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D13
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D2 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D2
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D3 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D3
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D4 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D4
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D5 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D5
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D6 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D6
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D7 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D7
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D8 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D8
lrwxrwxrwx    1 root     root             0 Sep  2 19:44 D9 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/D9
--w-------    1 root     root          4096 Sep  2 19:44 export
lrwxrwxrwx    1 root     root             0 Jan  1  1970 gpio15 -> ../../devices/virtual/gpio/gpio15
lrwxrwxrwx    1 root     root             0 Jan  1  1970 gpiochip0 -> ../../devices/virtual/gpio/gpiochip0
lrwxrwxrwx    1 root     root             0 Sep  2 19:31 gpiochip100 -> ../../devices/mcuio/0:0.0/0:1.1/gpio/gpiochip100
--w-------    1 root     root          4096 Jan  1  1970 unexport
root@tiannfc /root [#]# echo 1 > /sys/class/gpio/D13/value
root@tiannfc /root [#]# echo 0 > /sys/class/gpio/D13/value
root@tiannfc /root [#]# 
```

So we'll just use :

$ echo 1 > /sys/class/gpio/D13/value

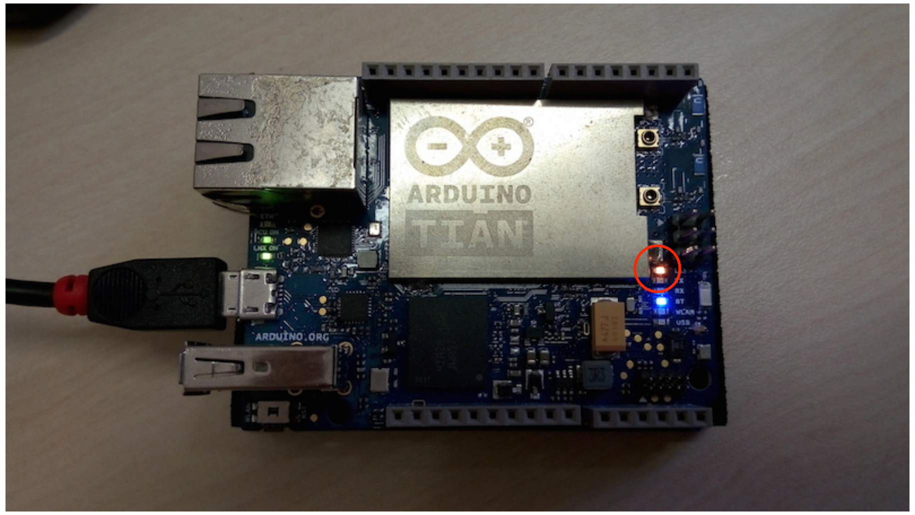
to turn on the L13 led while using :

$ echo 0 > /sys/class/gpio/D13/value

to revert it to its original state.

That's a simple way to run the "Hello World" of microcontroller units.

LininoIO – Example 1



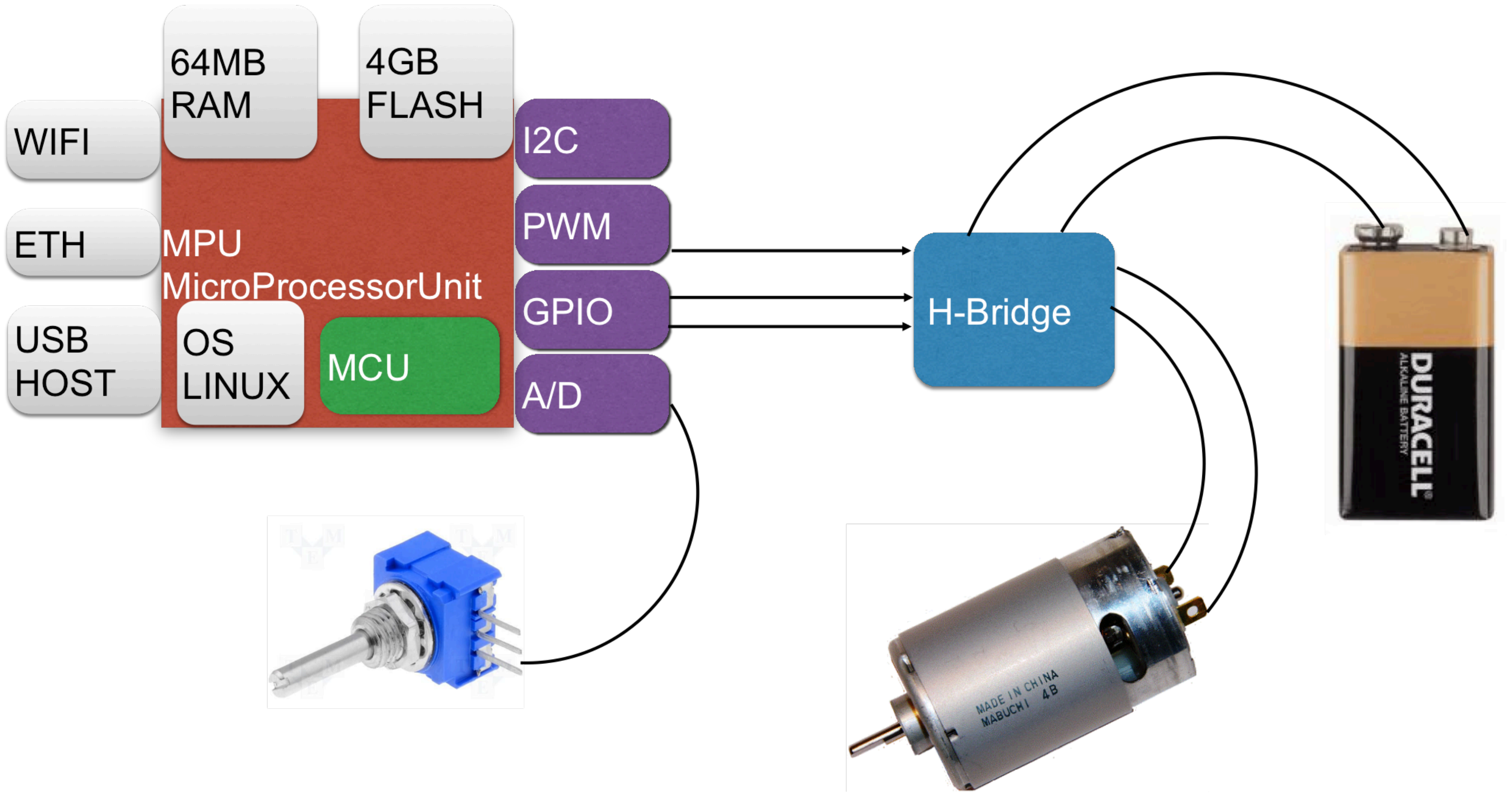
```
$ echo 1 > /sys/class/gpio/D13/value
```

To turn on the L13 led

```
$ echo 0 > /sys/class/gpio/D13/value
```

To turn off the L13 led

LininoIO – Example 2

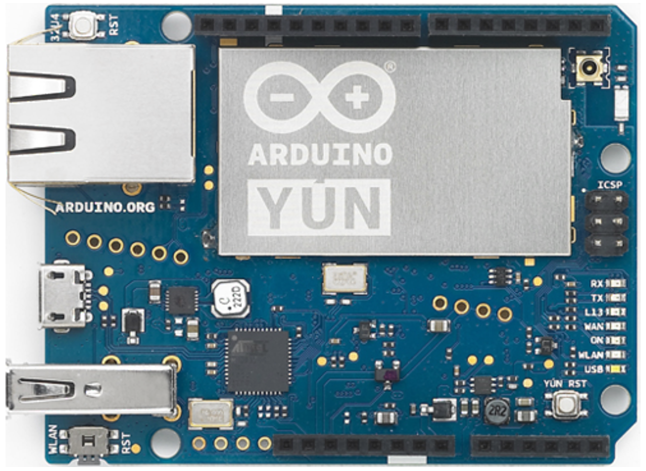


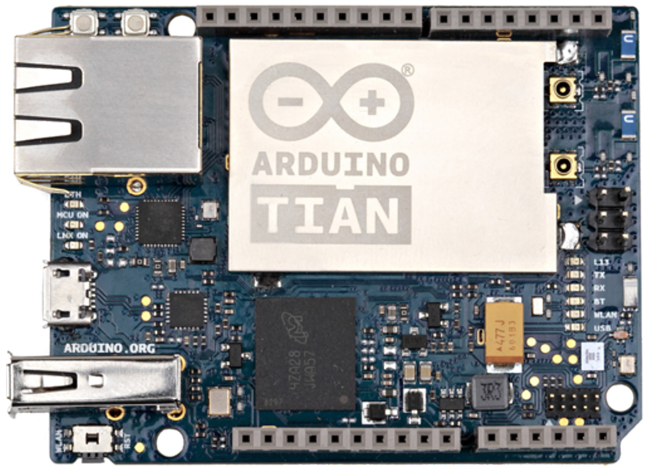
Motor with variable speed regulated from a potentiometer and modulated by PWM signal.

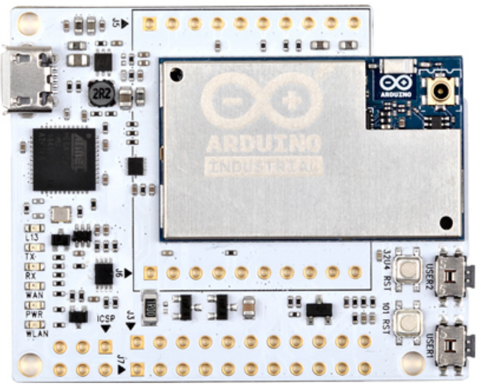
LininoIO – Example 2 (shell code)

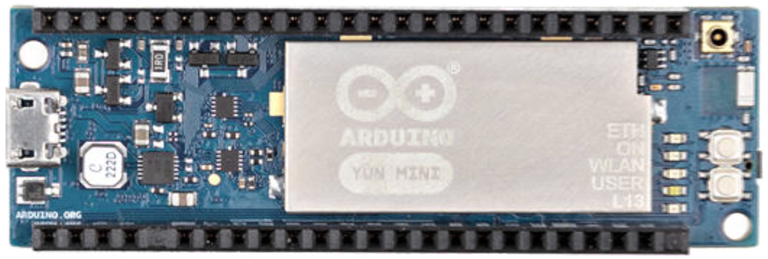
```
#!/bin/bash

#Change the PWM value to change the motor speed:
echo 125 > /sys/class/mcuio_pwm/D11/value

#Read the value measured from ADC:
cat /sys/class/mcuio_adc/A1/value

#Change the motor direction using GPIOs:
echo 1 > /sys/class/gpio/D1/value
echo 0 > /sys/class/gpio/D2/value
```

Linino “Open Stack”









- open hardware platform such as Arduino etc.
- DYI and maker’s projects
- mcu agnostic (atmel, freescale, stm, etc.)
- professional projects
- mass production
- and of course all linino devices

mcu based device
lininoIO operating system

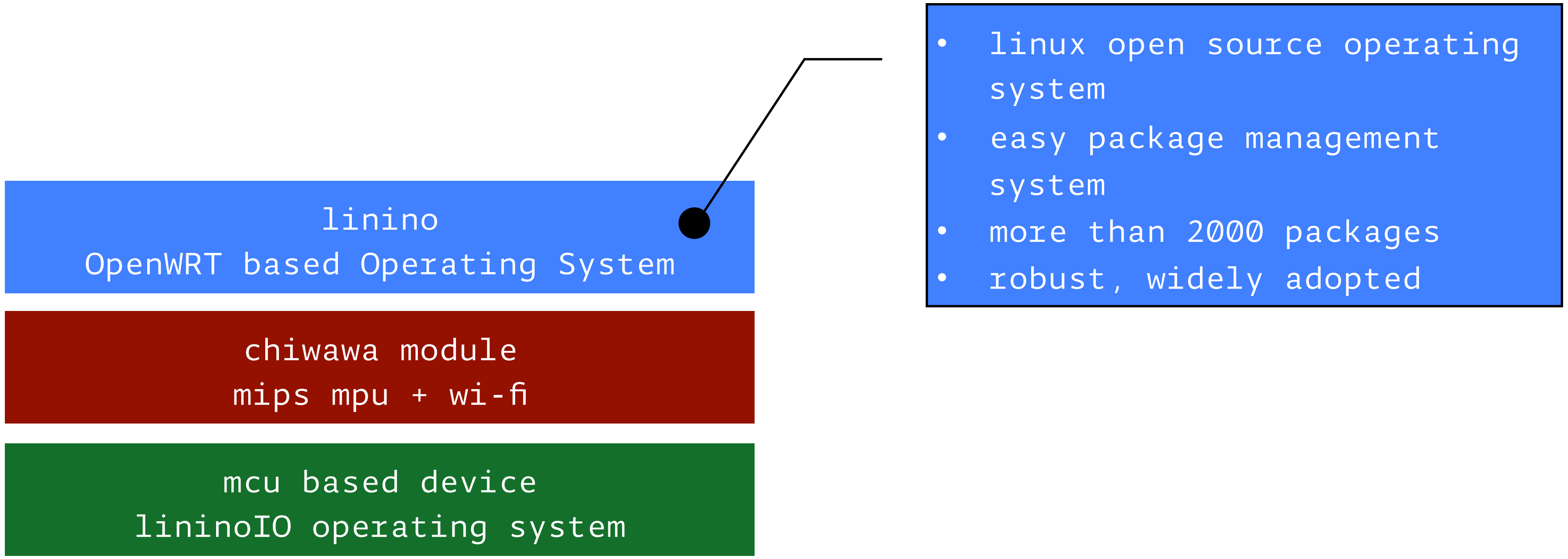
Linino “Open Stack”

chiwawa module
mips mpu + wi-fi

mcu based device
lininoIO operating system

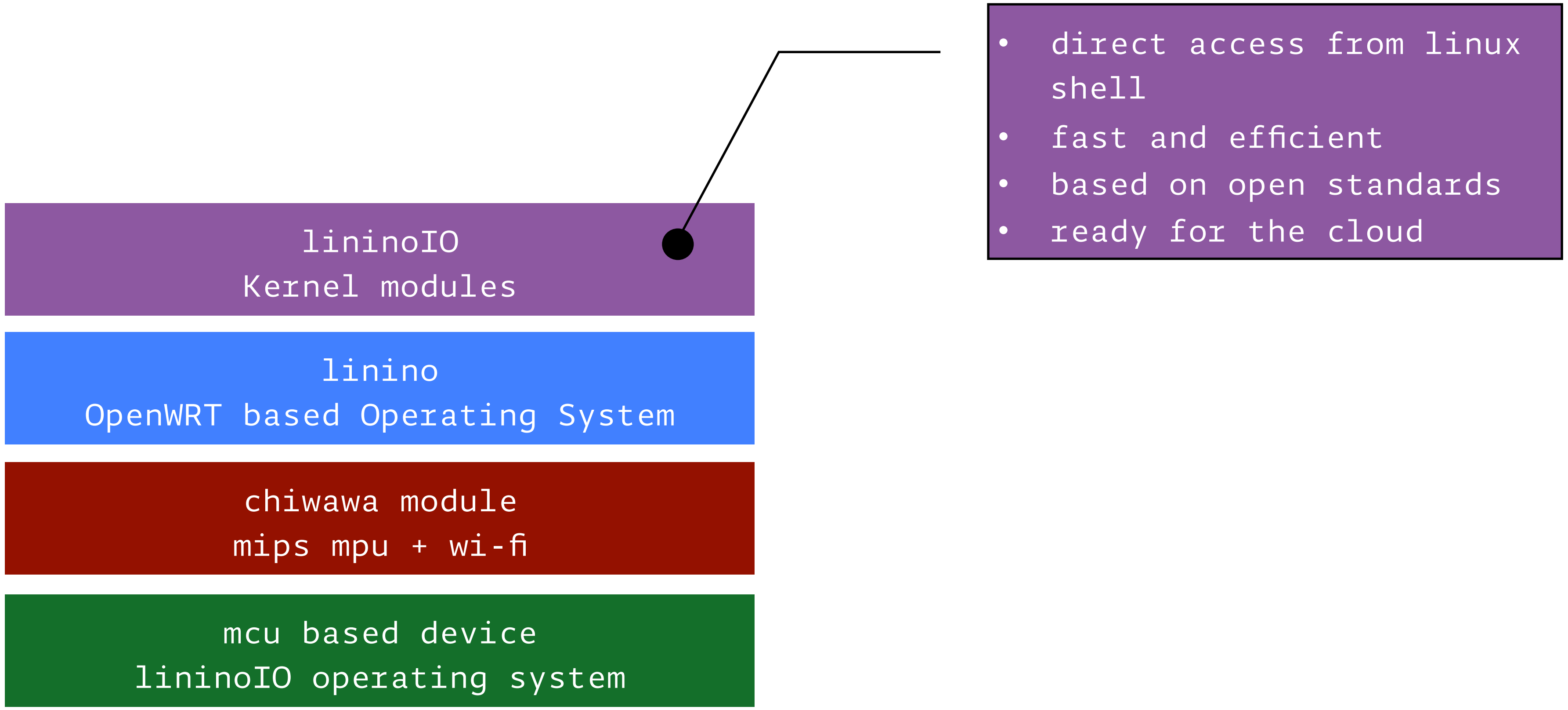
- QCA 9331/9342
- mpu 400/540mhz mips BE based module
- 64mb ram
- uSDcard slot or 4gb flash memory on-board
- usb 2.0 host/device
- wi-fi 802.11b/g/n connectivity, BLE
- 2 lan ports
- 23 x gpio

Linino "Open Stack"



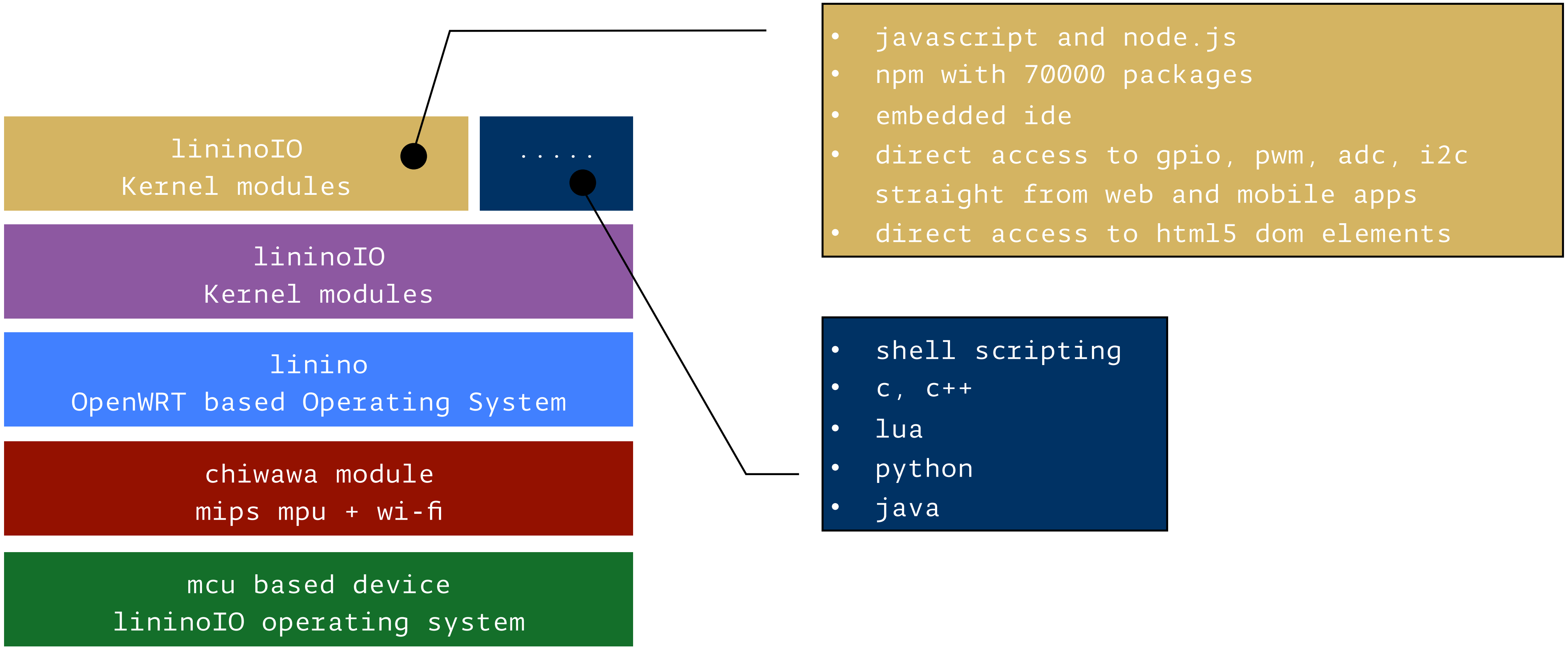
- linux open source operating system
- easy package management system
- more than 2000 packages
- robust, widely adopted

Linino “Open Stack”



- direct access from linux shell
- fast and efficient
- based on open standards
- ready for the cloud

Linino “Open Stack”



lininoIO
Kernel modules

.....

lininoIO
Kernel modules

linino
OpenWRT based Operating System

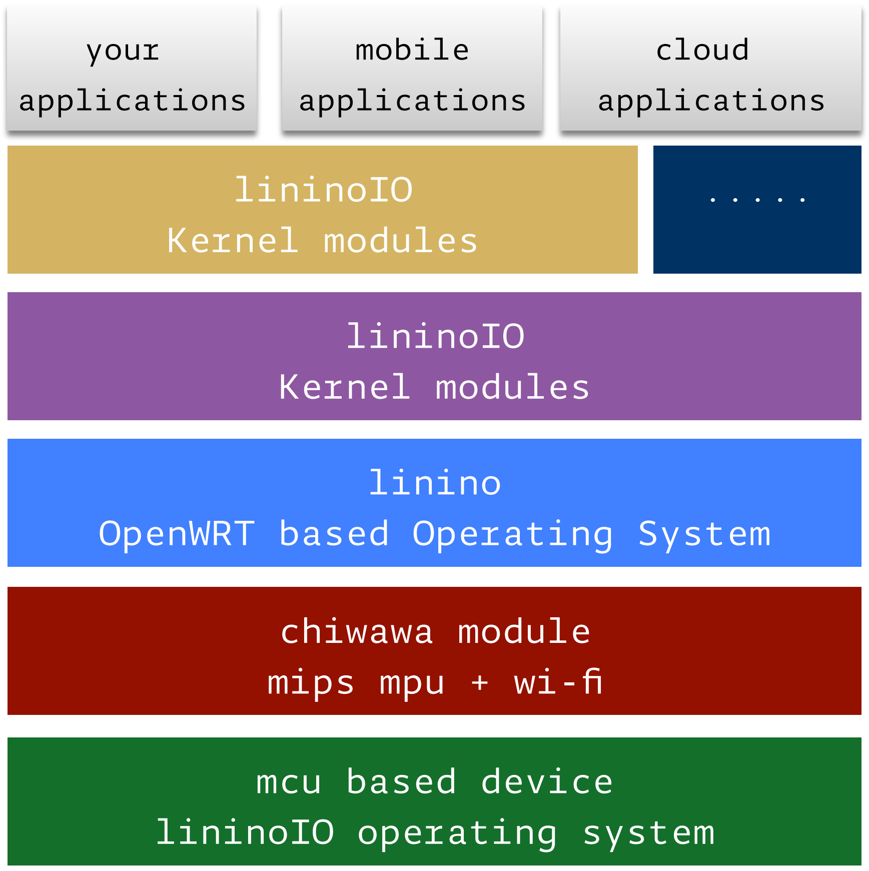
chiwawa module
mips mpu + wi-fi

mcu based device
lininoIO operating system

- javascript and node.js
- npm with 70000 packages
- embedded ide
- direct access to gpio, pwm, adc, i2c straight from web and mobile apps
- direct access to html5 dom elements

- shell scripting
- c, c++
- lua
- python
- java

Linino "Open Stack"


your applications
mobile applications
cloud applications
lininoIO Kernel modules
.....
lininoIO Kernel modules
linino OpenWRT based Operating System
chiwawa module mips mpu + wi-fi
mcu based device lininoIO operating system

robotics

monitoring

remote control

home automation

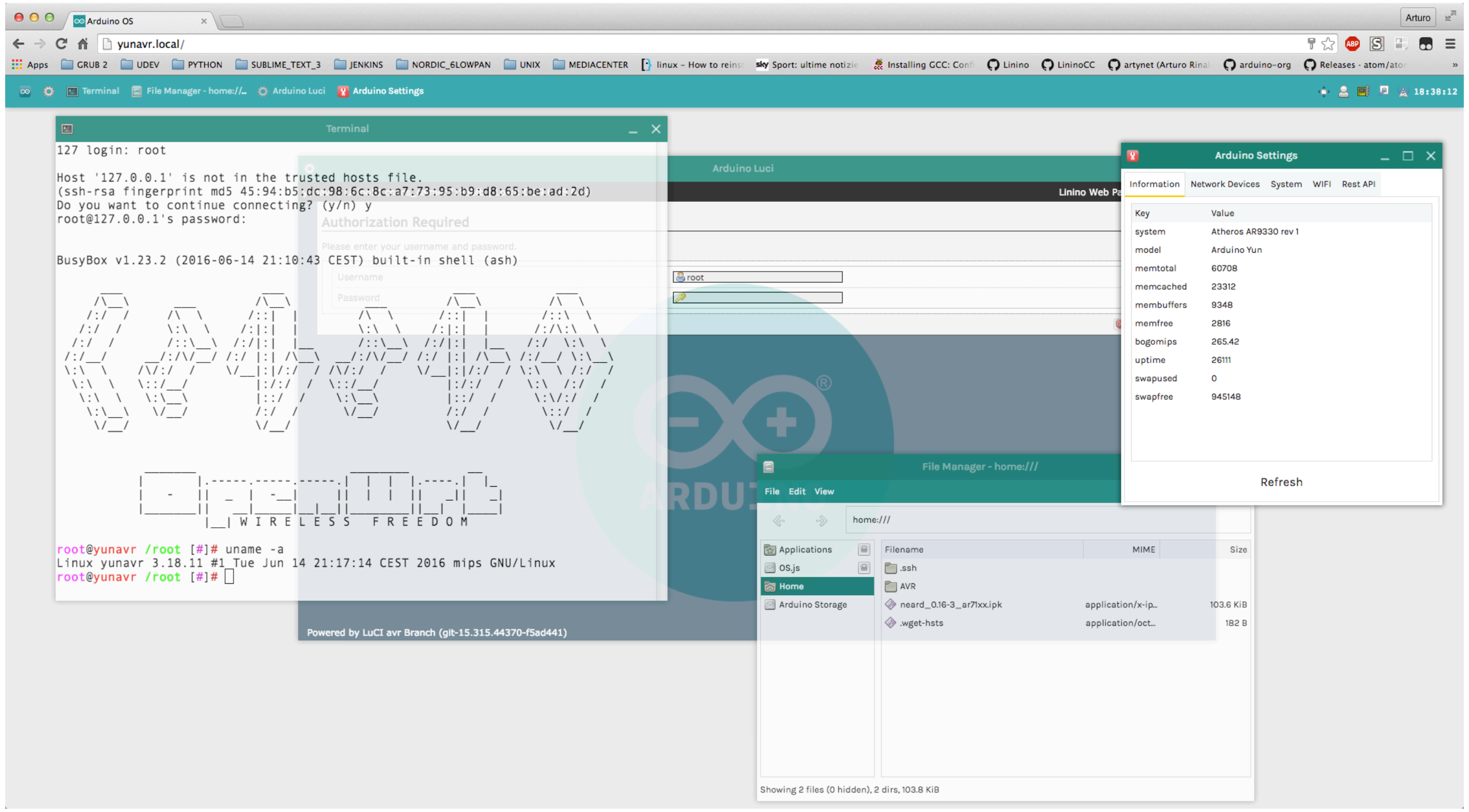
vehicle automation

drones control

energy management

...

ArduinoOS



Terminal

```
127 login: root

Host '127.0.0.1' is not in the trusted hosts file.
(ssh-rsa fingerprint md5 45:94:b5:dc:98:6c:8c:a7:73:95:b9:d8:65:be:ad:2d)
Do you want to continue connecting? (y/n) y
root@127.0.0.1's password:


BusyBox v1.23.2 (2016-06-14 21:10:43 CEST) built-in shell (ash)

     ___                ___           ___           ___
    /\__\     ___      /\__\   ___   /\__\         /\  \
   /:/  /    /\  \    /::|  | /\  \ /::|  |       /::\  \
  /:/  /     \:\  \  /:|:|  | \:\  \/:|:|  |      /:/\:\  \
 /:/  /      /::\__\/:/|:|  |__/::\__\/:/|:|  |__  /:/  \:\  \
/:/__/    __/:/\/__/:/ |:| /\__\/:/\/__/:/ |:| /\__\/:/__/ \:\__\
\:\  \   /\/:/  /  \/__|:|/:/  /\/:/  /  \/__|:|/:/  /\:\  \ /:/  /
 \:\  \  \::/__/       |:/:/  /\::/__/       |:/:/  /  \:\  /:/  /
  \:\  \  \:\__\       |::/  /  \:\__\       |::/  /    \:\/:/  /
   \:\__\  \/__/       /:/  /    \/__/       /:/  /      \::/  /
    \/__/              \/__/                 \/__/        \/__/


  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M

root@yunavr /root [#]# uname -a
Linux yunavr 3.18.11 #1 Tue Jun 14 21:17:14 CEST 2016 mips GNU/Linux
root@yunavr /root [#]#
```

Arduino Settings

Information | Network Devices | System | WIFI | Rest API

Key	Value
system	Atheros AR9330 rev 1
model	Arduino Yun
memtotal	60708
memcached	23312
membuffers	9348
memfree	2816
bogomips	265.42
uptime	26111
swapused	0
swapfree	945148

Refresh

File Manager - home:///

File Edit View

home:///

Applications
OS.js
Home
Arduino Storage

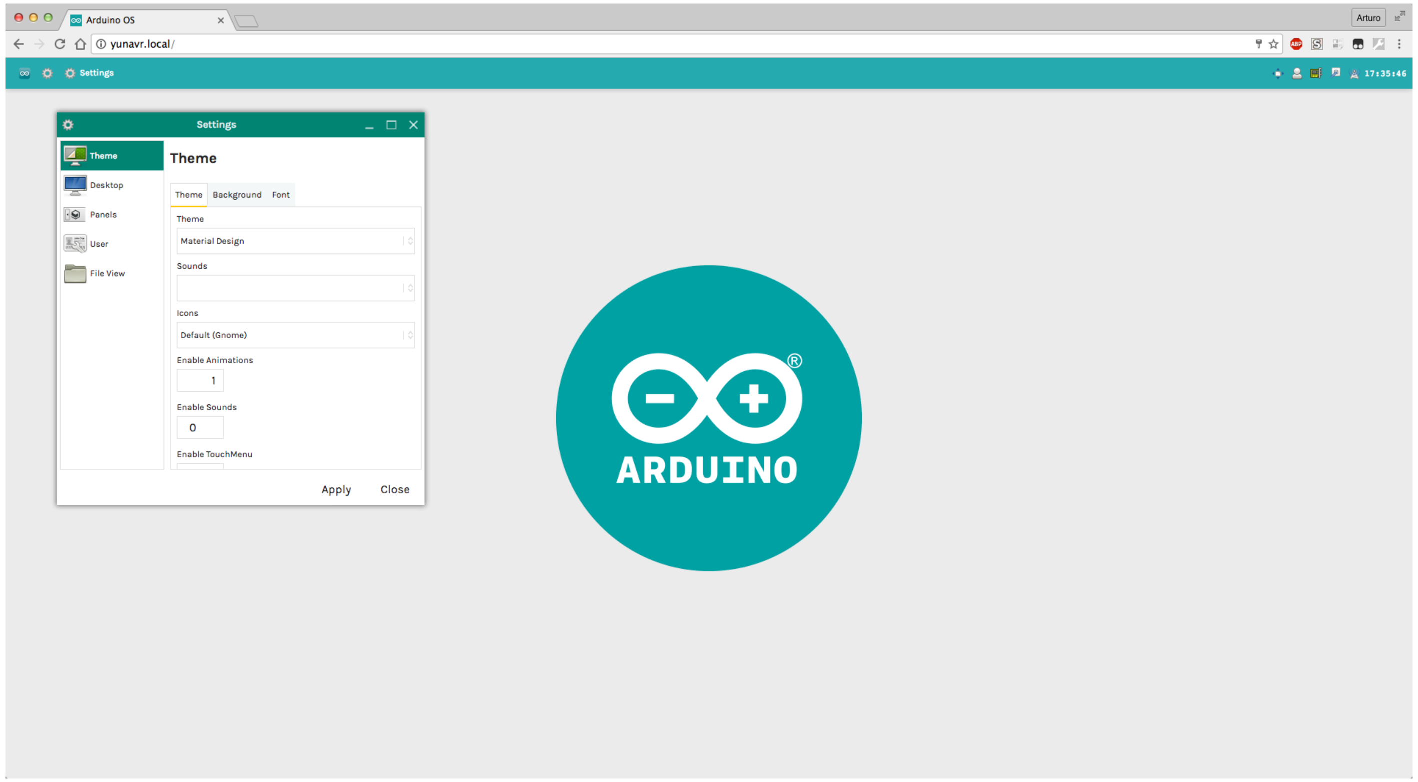
Filename	MIME	Size
.ssh		
AVR		
neard_0.16-3_ar71xx.ipk	application/x-ip...	103.6 KiB
.wget-hsts	application/oct...	182 B

Showing 2 files (0 hidden), 2 dirs, 103.8 KiB

Features at a glance :

- SSH web terminal based on **shellinabox**
- File Manager
- Quick settings panel
- Web **LuCI** configurator as standalone application
- Basic desktop utilities

ArduinoOS - Settings Panel



Theme

Theme: Material Design

Sounds:

Icons: Default (Gnome)

Enable Animations: 1

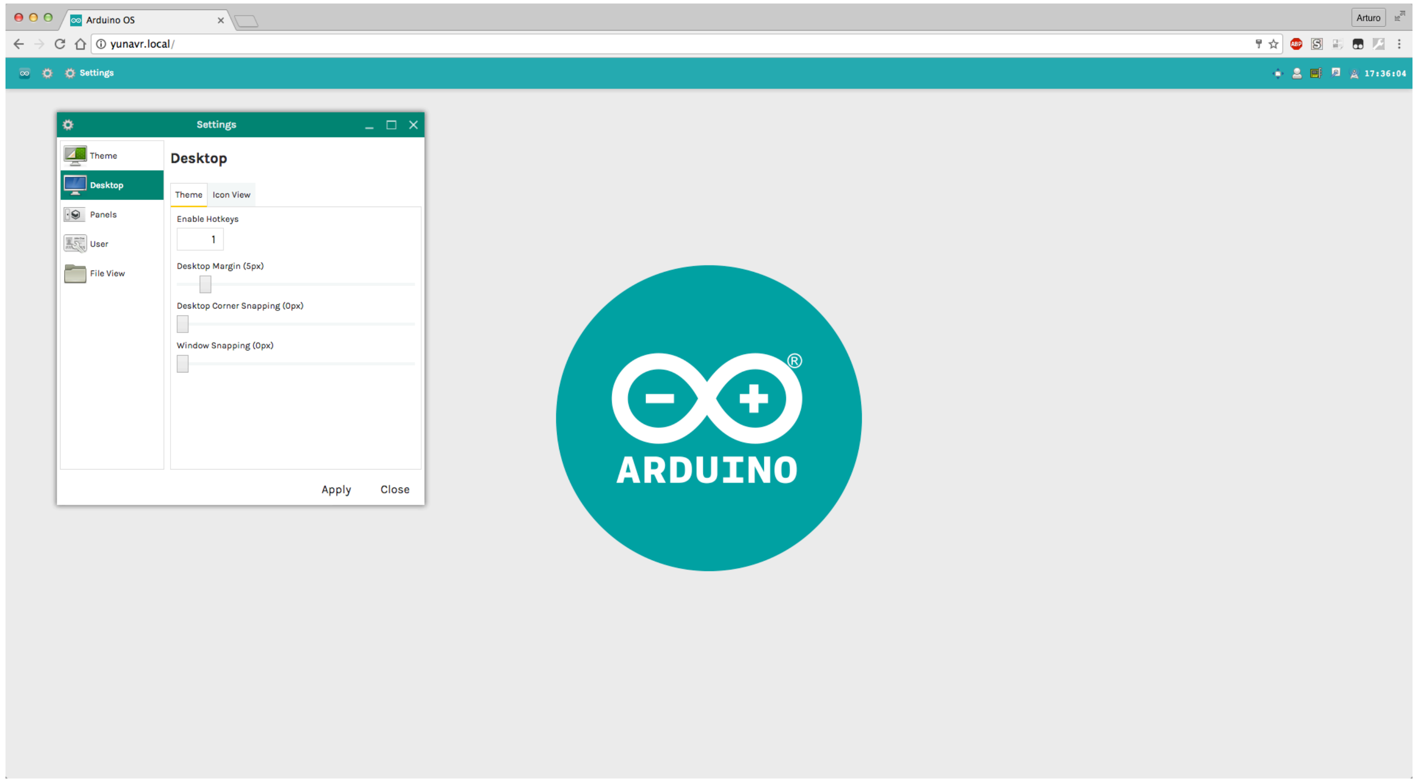
Enable Sounds: 0

Enable TouchMenu

This panel grants you a quick access to customize your desktop experience with the following entries :

- Theme
- Desktop
- Panels
- User
- File View

ArduinoOS - Settings Panel

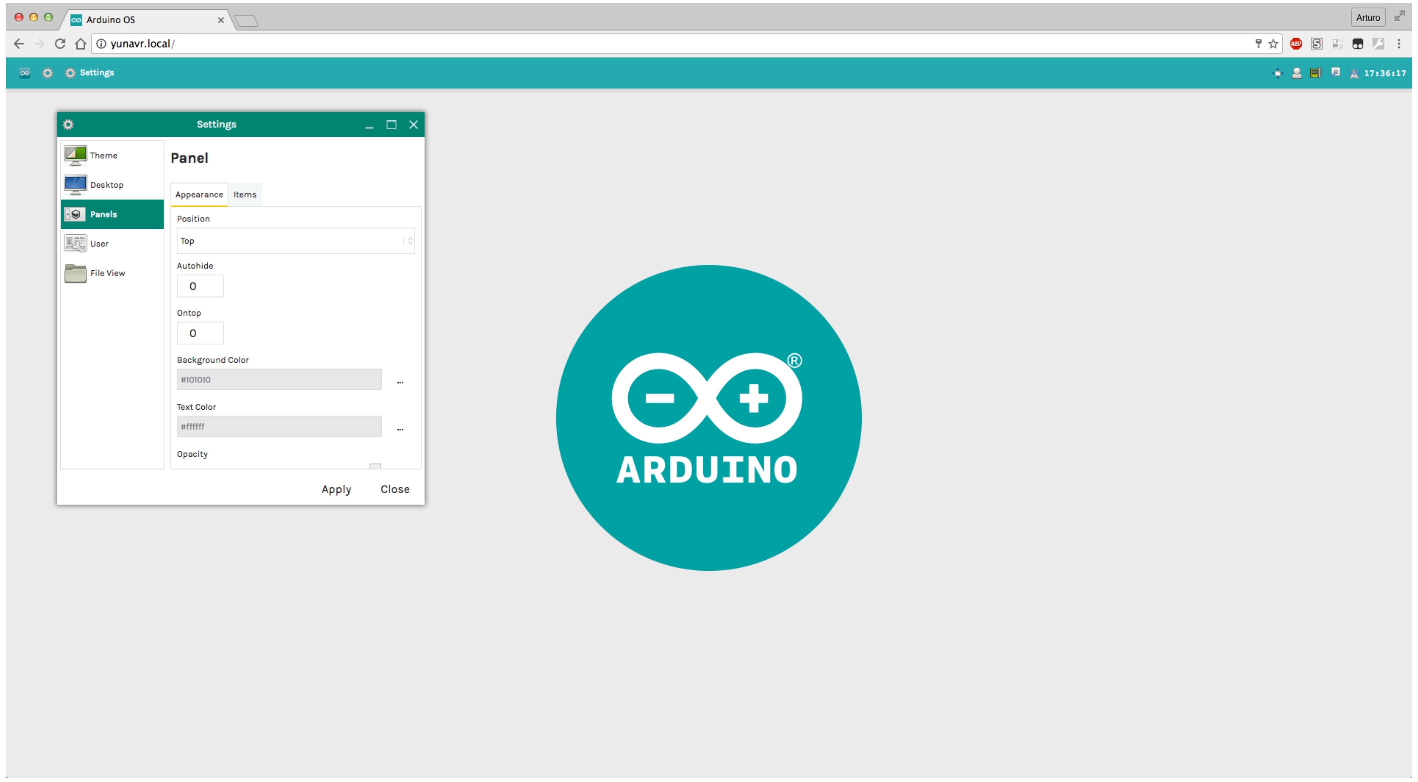


Enable Hotkeys: 1
Desktop Margin (5px)
Desktop Corner Snapping (0px)
Window Snapping (0px)

This panel grants you a quick access to customize your desktop experience with the following entries :

- Theme
- Desktop
- Panels
- User
- File View

ArduinoOS - Settings Panel



Panel

Position: Top

Autohide: 0

Ontop: 0

Background Color: #101010

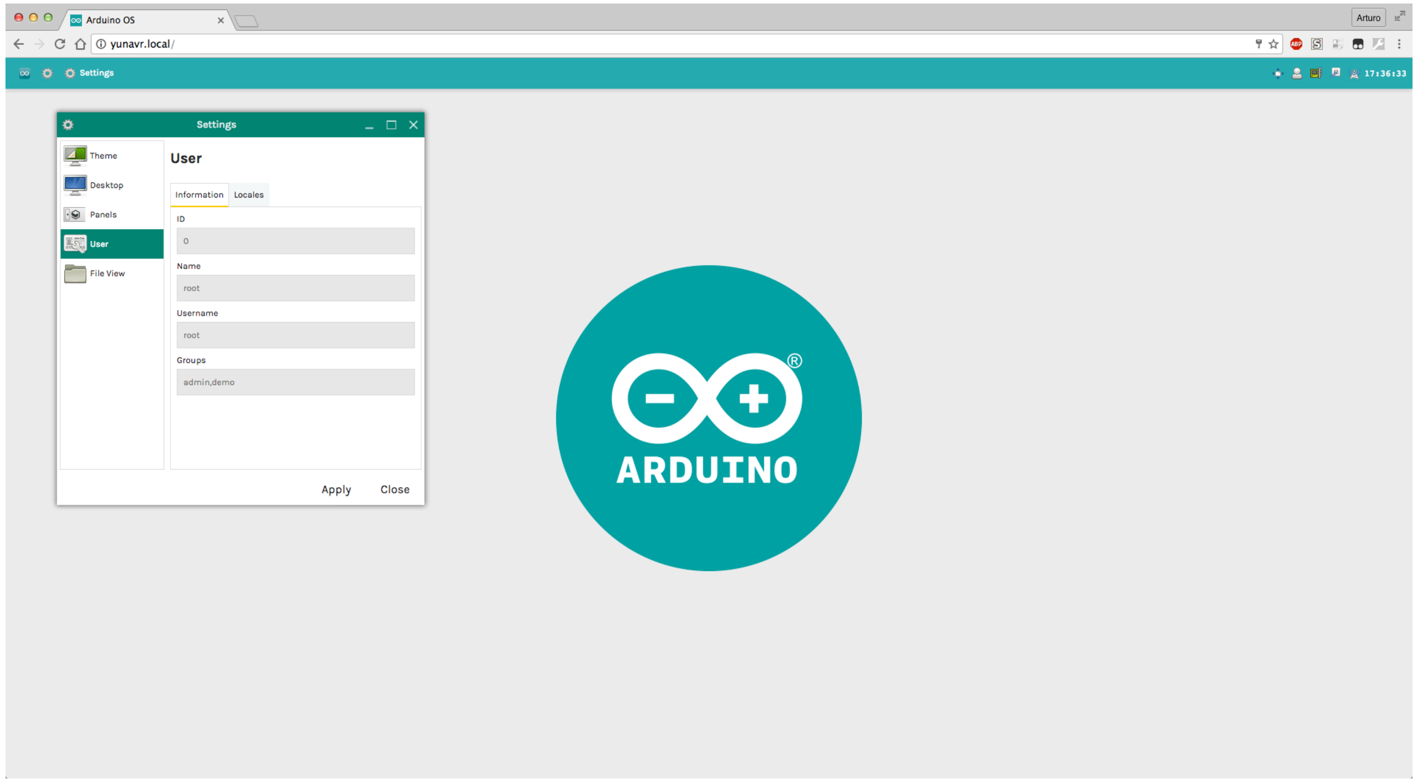
Text Color: #ffffff

Opacity

This panel grants you a quick access to customize your desktop experience with the following entries :

- Theme
- Desktop
- Panels
- User
- File View

ArduinoOS - Settings Panel



Settings

Theme
Desktop
Panels
User
File View

User

Information | Locales

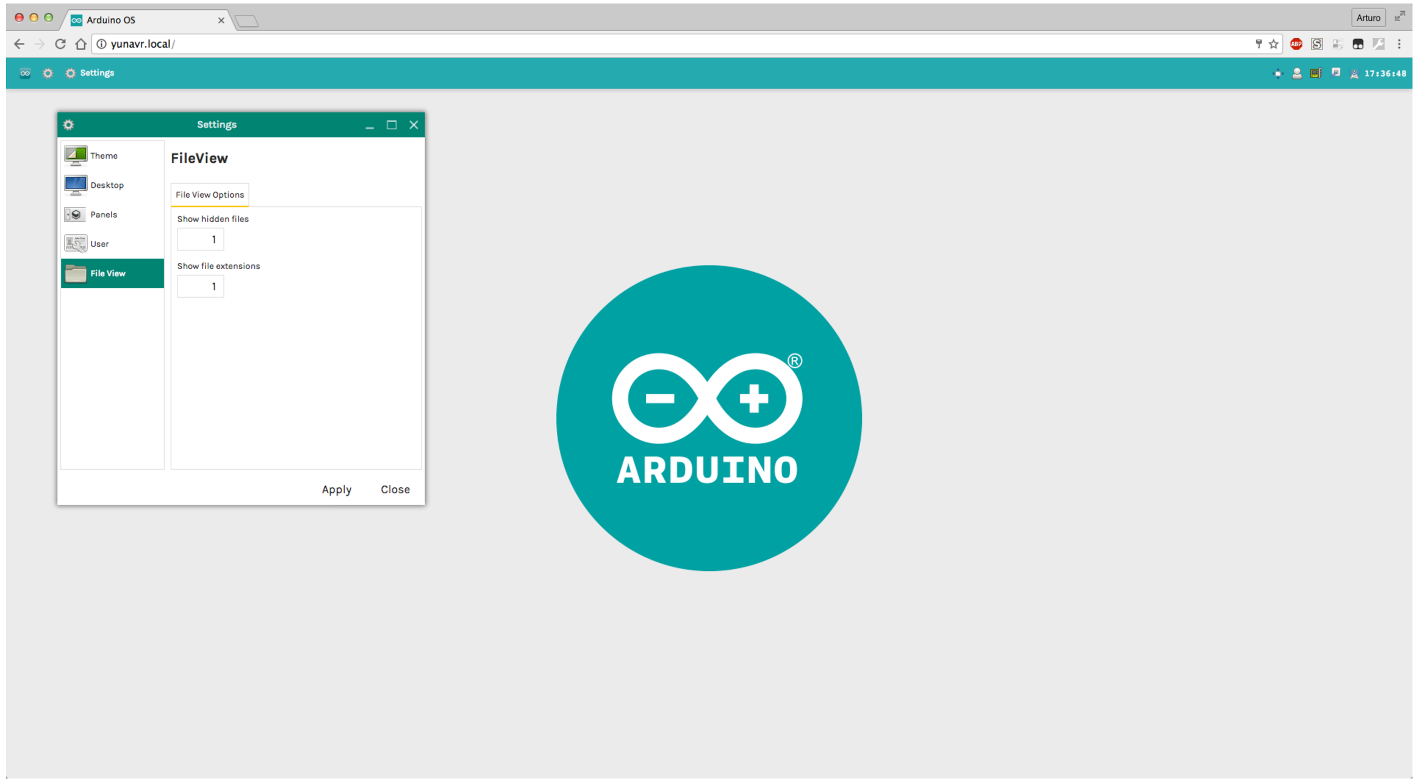
ID: 0
Name: root
Username: root
Groups: admin,demo

Apply Close

This panel grants you a quick access to customize your desktop experience with the following entries :

- Theme
- Desktop
- Panels
- User
- File View

ArduinoOS - Settings Panel



FileView

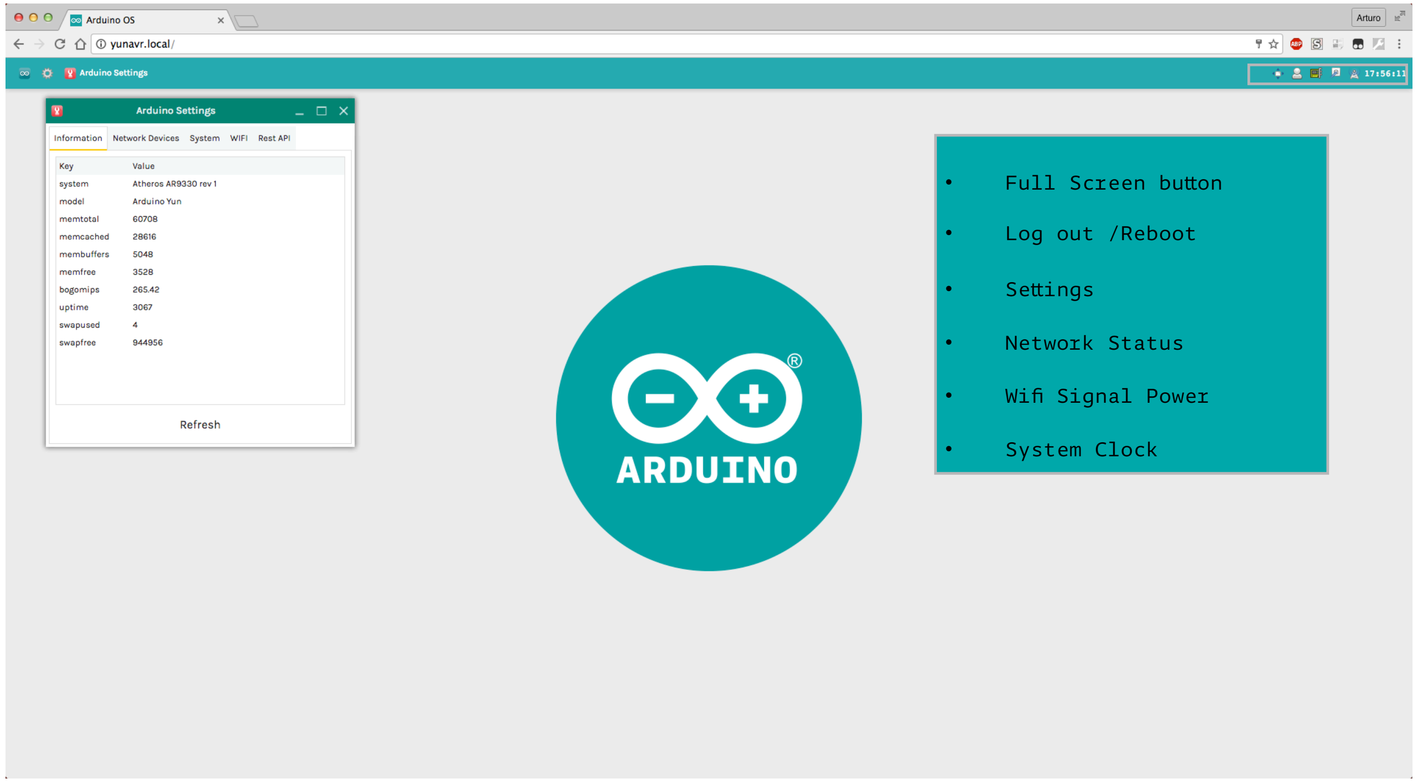
Show hidden files
1

Show file extensions
1

This panel grants you a quick access to customize your desktop experience with the following entries :

- Theme
- Desktop
- Panels
- User
- File View

ArduinoOS – Arduino Settings



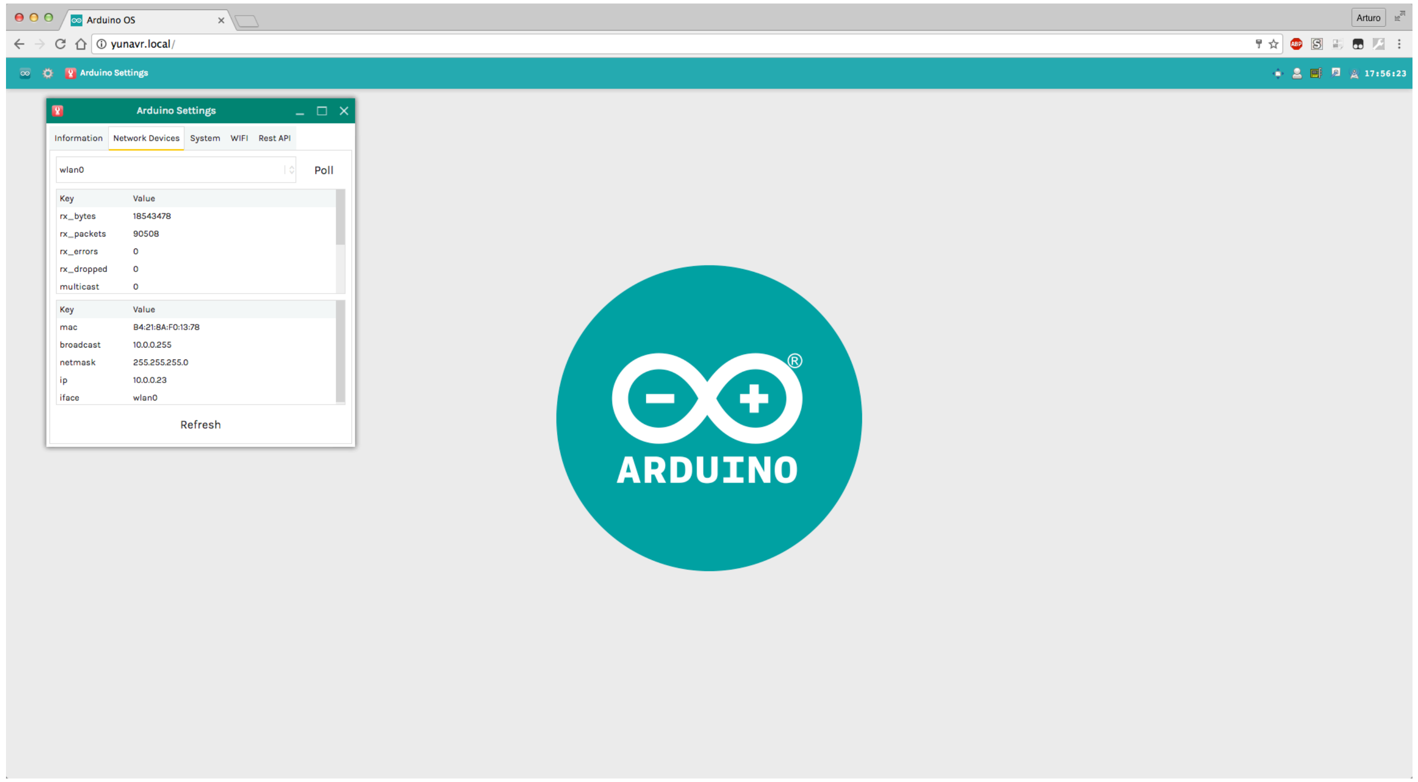
Key	Value
system	Atheros AR9330 rev 1
model	Arduino Yun
memtotal	60708
memcached	28616
membuffers	5048
memfree	3528
bogomips	265.42
uptime	3067
swapused	4
swapfree	944956

- Full Screen button
- Log out /Reboot
- Settings
- Network Status
- Wifi Signal Power
- System Clock

This panel grants you a quick access to the Arduino specific settings :

- General System information
- Network Devices
- System
- WiFi
- Rest API

ArduinoOS – Arduino Settings



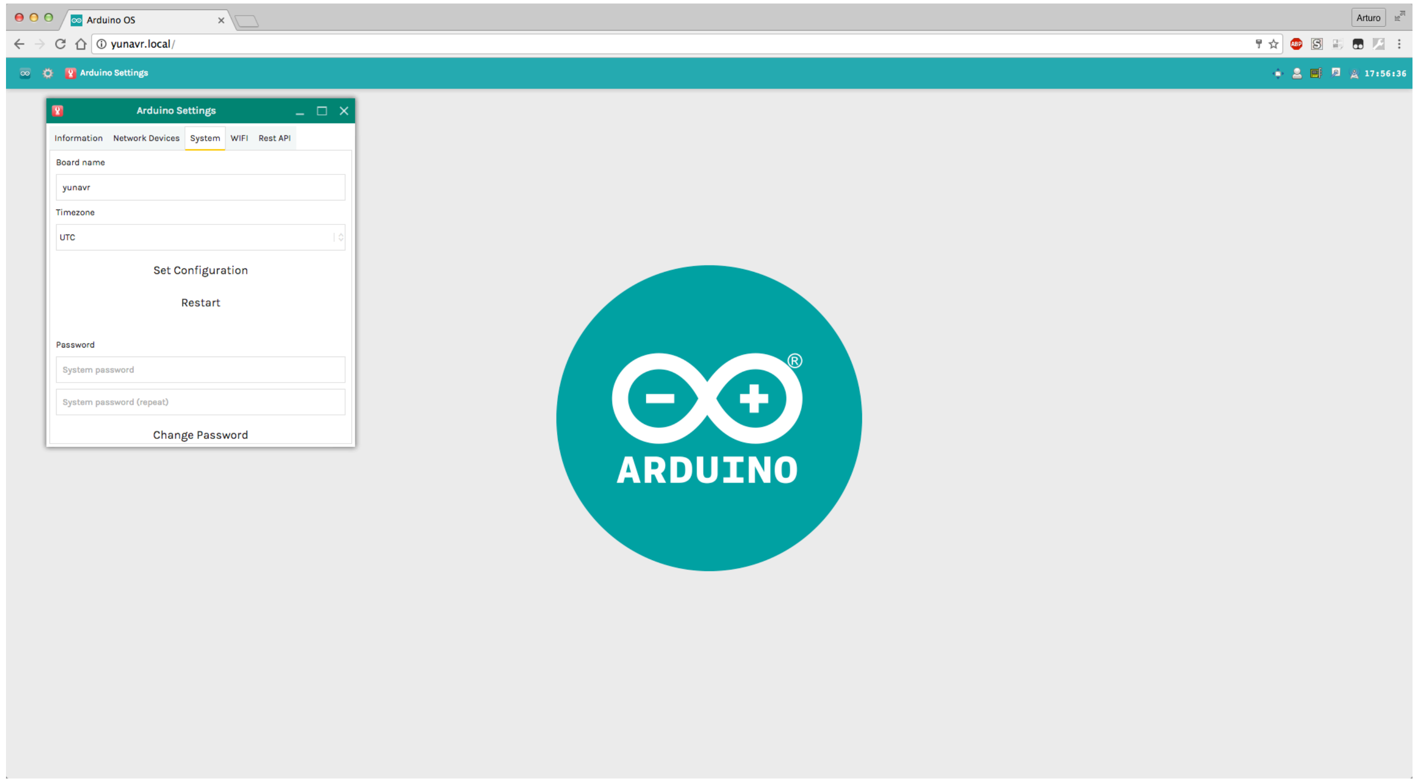
Key	Value
rx_bytes	18543478
rx_packets	90508
rx_errors	0
rx_dropped	0
multicast	0

Key	Value
mac	B4:21:8A:F0:13:78
broadcast	10.0.0.255
netmask	255.255.255.0
ip	10.0.0.23
iface	wlan0

This panel grants you a quick access to the Arduino specific settings :

- General System information
- Network Devices
- System
- WiFi
- Rest API

ArduinoOS – Arduino Settings


Arduino Settings

Information | Network Devices | System | WIFI | Rest API

Board name: yunavr

Timezone: UTC

Set Configuration

Restart

Password

System password

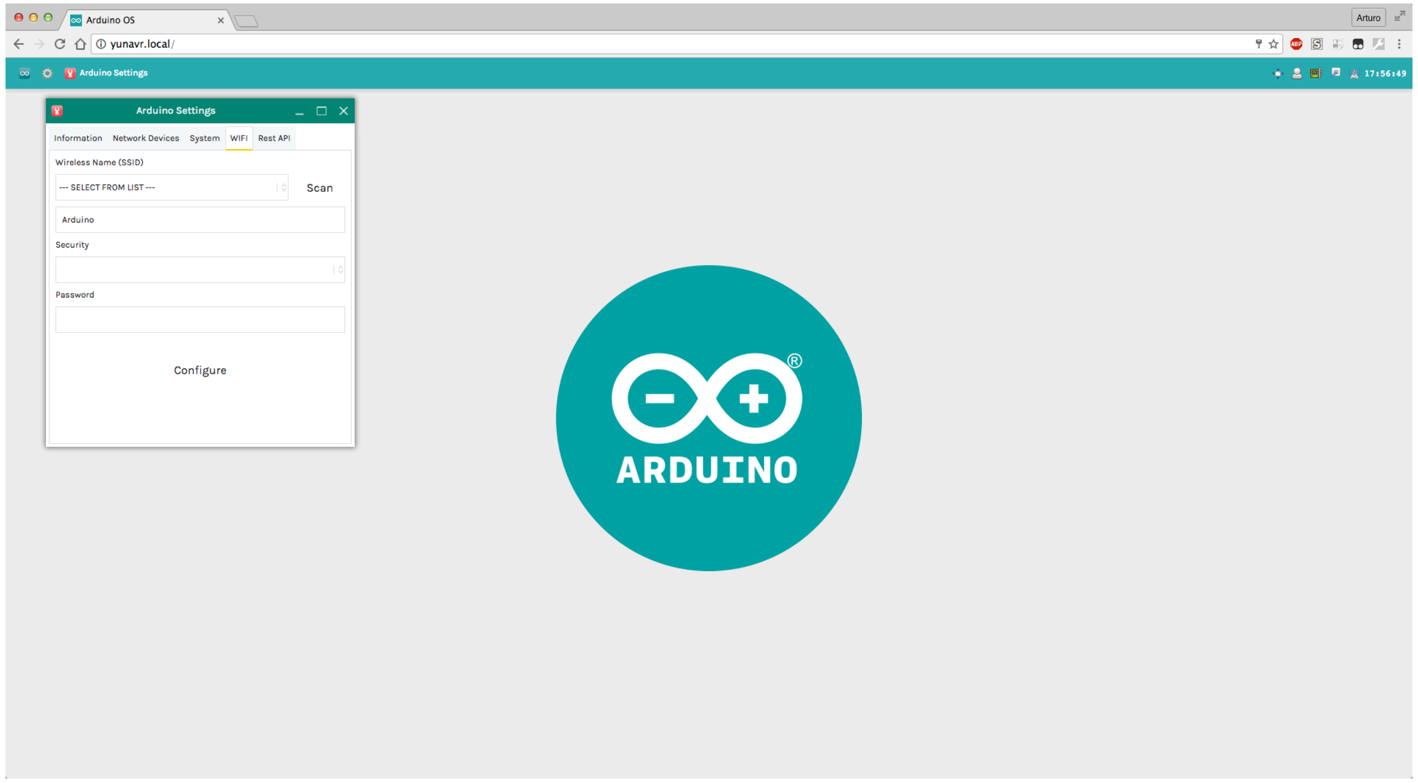
System password (repeat)

Change Password

This panel grants you a quick access to the Arduino specific settings :

- General System information
- Network Devices
- System
- WiFi
- Rest API

ArduinoOS – Arduino Settings



Arduino Settings

Wireless Name (SSID): --- SELECT FROM LIST ---

Arduino

Security:

Password:

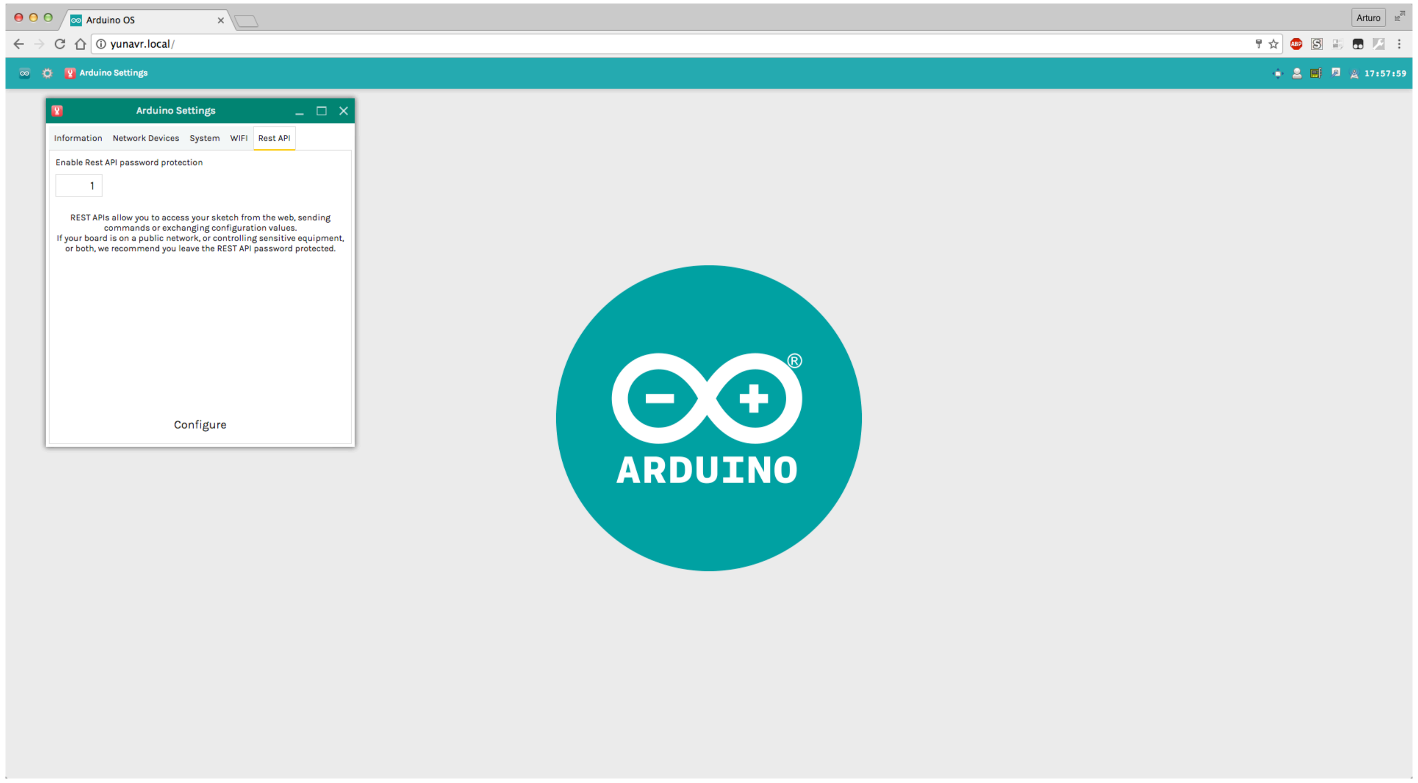
This panel grants you a quick access to the Arduino specific settings :

- General System information
- Network Devices
- System
- WiFi
- Rest API

Berlin, October 12th 2016

ELC EUROPE 2016

ArduinoOS – Arduino Settings



Arduino Settings

Information | Network Devices | System | WIFI | Rest API

Enable Rest API password protection

1

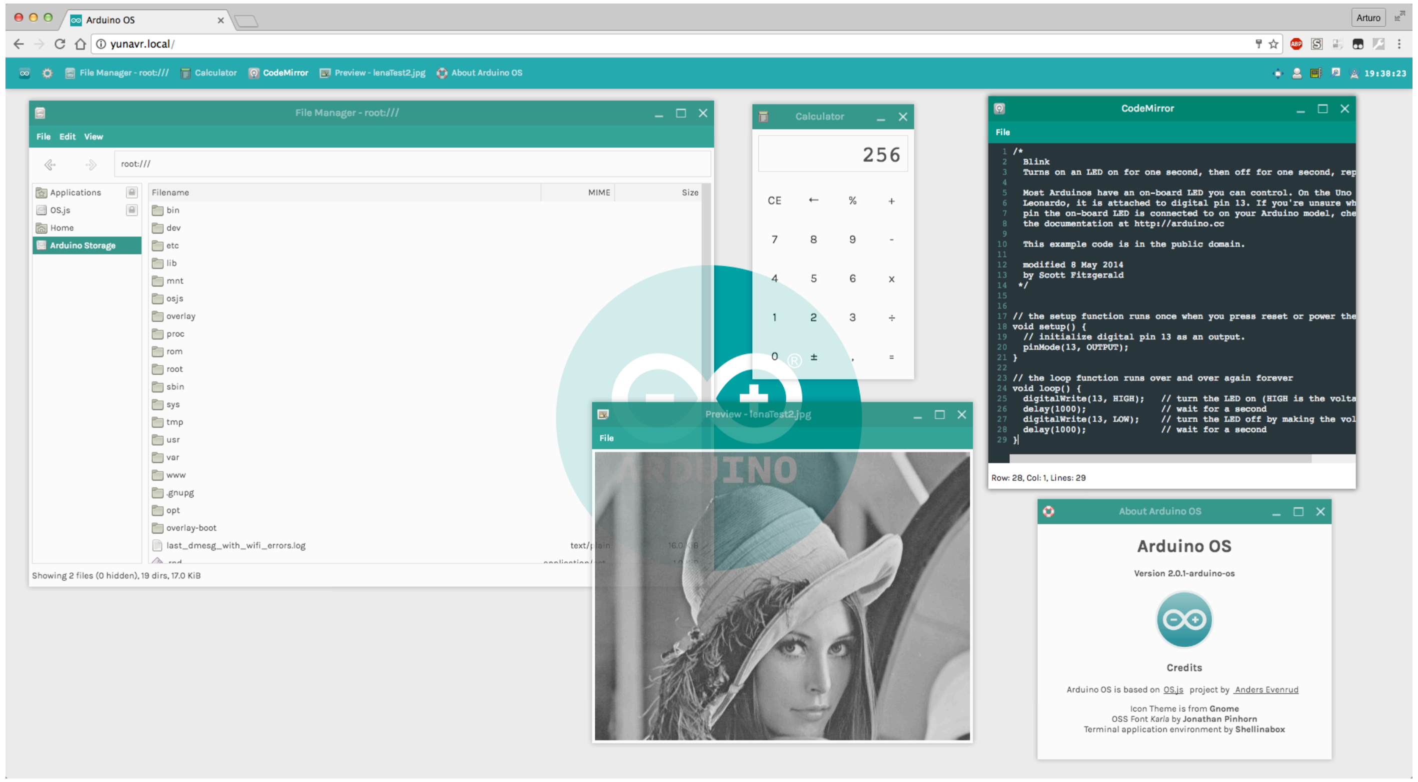
REST APIs allow you to access your sketch from the web, sending commands or exchanging configuration values.
If your board is on a public network, or controlling sensitive equipment, or both, we recommend you leave the REST API password protected.

Configure

This panel grants you a quick access to the Arduino specific settings :

- General System information
- Network Devices
- System
- WiFi
- Rest API

ArduinoOS – Desktop Utilities



bin
dev
etc
lib
mnt
osjs
overlay
proc
rom
root
sbin
sys
tmp
usr
var
www
.gnupg
opt
overlay-boot
last_dmesg_with_wifi_errors.log

Showing 2 files (0 hidden), 19 dirs, 17.0 KiB

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, rep
  
  Most Arduinos have an on-board LED you can control. On the Uno
  Leonardo, it is attached to digital pin 13. If you're unsure wh
  pin the on-board LED is connected to on your Arduino model, che
  the documentation at http://arduino.cc
  
  This example code is in the public domain.
  
  modified 8 May 2014
  by Scott Fitzgerald
 */


// the setup function runs once when you press reset or power the
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the volta
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the vol
  delay(1000);              // wait for a second
}
```

Row: 28, Col: 1, Lines: 29

Arduino OS

Version 2.0.1-arduino-os

Credits

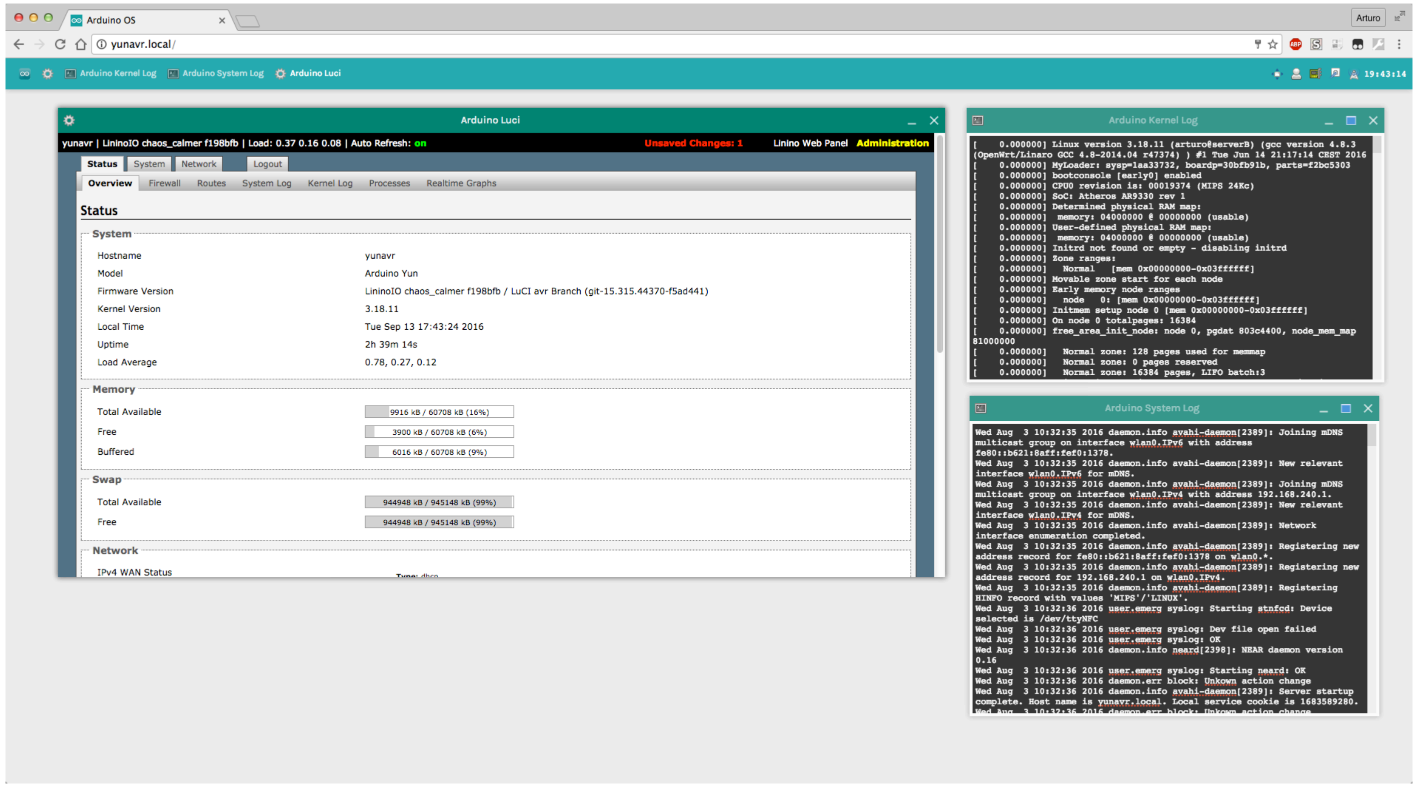
Arduino OS is based on OS.js project by Anders Evenrud

Icon Theme is from **Gnome**
OSS Font *Karla* by **Jonathan Pinhorn**
Terminal application environment by **Shellinabox**

The main desktop utilities are shown here:

- Complete and handy file manager supporting *drag'n'drop* functionality
- Calculator
- Text Editor
- Photo Viewer
- Many more coming….

ArduinoOS – Luci Panel



Arduino Luci

yunavr | LininoIO chaos_calmer f198bfb | Load: 0.37 0.16 0.08 | Auto Refresh: on   Unsaved Changes: 1   Linino Web Panel Administration

Status

System

Hostname	yunavr
Model	Arduino Yun
Firmware Version	LininoIO chaos_calmer f198bfb / LuCI avr Branch (git-15.315.44370-f5ad441)
Kernel Version	3.18.11
Local Time	Tue Sep 13 17:43:24 2016
Uptime	2h 39m 14s
Load Average	0.78, 0.27, 0.12

Memory

Total Available	9916 kB / 60708 kB (16%)
Free	3900 kB / 60708 kB (6%)
Buffered	6016 kB / 60708 kB (9%)

Swap

Total Available	944948 kB / 945148 kB (99%)
Free	944948 kB / 945148 kB (99%)

Network

IPv4 WAN Status   Type: dhcp

Arduino Kernel Log

```
[    0.000000] Linux version 3.18.11 (arturo@serverB) (gcc version 4.8.3
(OpenWrt/Linaro GCC 4.8-2014.04 r47374) ) #1 Tue Jun 14 21:17:14 CEST 2016
[    0.000000] MyLoader: sysp=1aa33732, boardp=30bfb91b, parts=f2bc5303
[    0.000000] bootconsole [early0] enabled
[    0.000000] CPU0 revision is: 00019374 (MIPS 24Kc)
[    0.000000] SoC: Atheros AR9330 rev 1
[    0.000000] Determined physical RAM map:
[    0.000000]  memory: 04000000 @ 00000000 (usable)
[    0.000000] User-defined physical RAM map:
[    0.000000]  memory: 04000000 @ 00000000 (usable)
[    0.000000] Initrd not found or empty - disabling initrd
[    0.000000] Zone ranges:
[    0.000000]   Normal   [mem 0x00000000-0x03ffffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x00000000-0x03ffffff]
[    0.000000] Initmem setup node 0 [mem 0x00000000-0x03ffffff]
[    0.000000] On node 0 totalpages: 16384
[    0.000000] free_area_init_node: node 0, pgdat 803c4400, node_mem_map
81000000
[    0.000000]   Normal zone: 128 pages used for memmap
[    0.000000]   Normal zone: 0 pages reserved
[    0.000000]   Normal zone: 16384 pages, LIFO batch:3
```

Arduino System Log

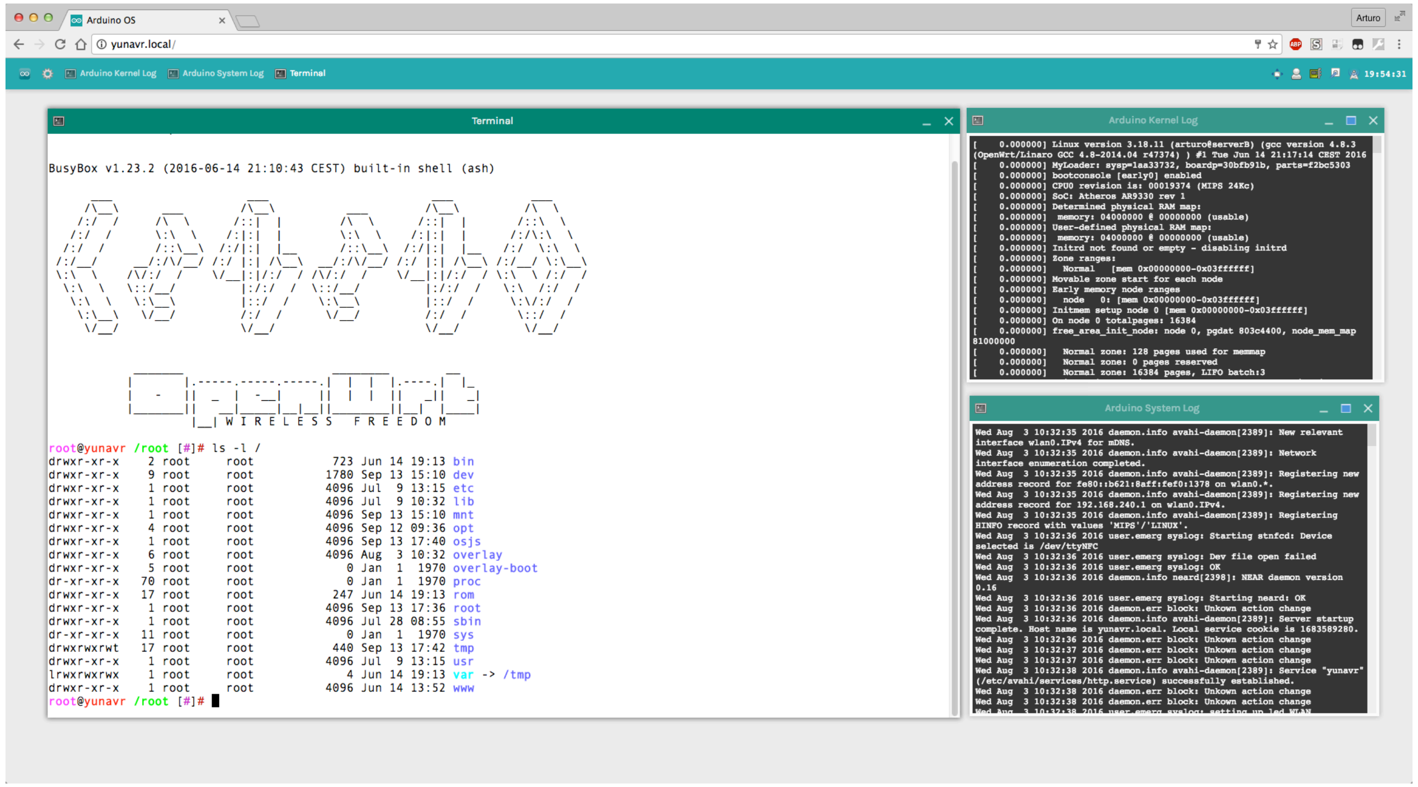
```
Wed Aug  3 10:32:35 2016 daemon.info avahi-daemon[2389]: Joining mDNS
multicast group on interface wlan0.IPv6 with address
fe80::b621:8aff:fef0:1378.
Wed Aug  3 10:32:35 2016 daemon.info avahi-daemon[2389]: New relevant
interface wlan0.IPv6 for mDNS.
Wed Aug  3 10:32:35 2016 daemon.info avahi-daemon[2389]: Joining mDNS
multicast group on interface wlan0.IPv4 with address 192.168.240.1.
Wed Aug  3 10:32:35 2016 daemon.info avahi-daemon[2389]: New relevant
interface wlan0.IPv4 for mDNS.
Wed Aug  3 10:32:35 2016 daemon.info avahi-daemon[2389]: Network
interface enumeration completed.
Wed Aug  3 10:32:35 2016 daemon.info avahi-daemon[2389]: Registering new
address record for fe80::b621:8aff:fef0:1378 on wlan0.*.
Wed Aug  3 10:32:35 2016 daemon.info avahi-daemon[2389]: Registering new
address record for 192.168.240.1 on wlan0.IPv4.
Wed Aug  3 10:32:35 2016 daemon.info avahi-daemon[2389]: Registering
HINFO record with values 'MIPS'/'LINUX'.
Wed Aug  3 10:32:36 2016 user.emerg syslog: Starting stnfcd: Device
selected is /dev/ttyNFC
Wed Aug  3 10:32:36 2016 user.emerg syslog: Dev file open failed
Wed Aug  3 10:32:36 2016 user.emerg syslog: OK
Wed Aug  3 10:32:36 2016 daemon.info neard[2398]: NEAR daemon version
0.16
Wed Aug  3 10:32:36 2016 user.emerg syslog: Starting neard: OK
Wed Aug  3 10:32:36 2016 daemon.err block: Unkown action change
Wed Aug  3 10:32:36 2016 daemon.info avahi-daemon[2389]: Server startup
complete. Host name is yunavr.local. Local service cookie is 1683589280.
Wed Aug  3 10:32:36 2016 daemon.err block: Unkown action change
```

The original Luci panel has not been lost at all !

It is provided as a separate web frame and launched from the main program menu as you're used to in your home desktop.

Two additional frames for displaying the kernel and system log are provided as well.

ArduinoOS – Shell Terminal



```
BusyBox v1.23.2 (2016-06-14 21:10:43 CEST) built-in shell (ash)

 _______                     ________        __
|       |.-----.-----.-----.|  |  |  |.----.|  |_
|   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
|_______||   __|_____|__|__||________||__|  |____|
         |__| W I R E L E S S   F R E E D O M

root@yunavr /root [#]# ls -l /
drwxr-xr-x    2 root     root           723 Jun 14 19:13 bin
drwxr-xr-x    9 root     root          1780 Sep 13 15:10 dev
drwxr-xr-x    1 root     root          4096 Jul  9 13:15 etc
drwxr-xr-x    1 root     root          4096 Jul  9 10:32 lib
drwxr-xr-x    1 root     root          4096 Sep 13 15:10 mnt
drwxr-xr-x    4 root     root          4096 Sep 12 09:36 opt
drwxr-xr-x    1 root     root          4096 Sep 13 17:40 osjs
drwxr-xr-x    6 root     root          4096 Aug  3 10:32 overlay
drwxr-xr-x    5 root     root             0 Jan  1  1970 overlay-boot
dr-xr-xr-x   70 root     root             0 Jan  1  1970 proc
drwxr-xr-x   17 root     root           247 Jun 14 19:13 rom
drwxr-xr-x    1 root     root          4096 Sep 13 17:36 root
drwxr-xr-x    1 root     root          4096 Jul 28 08:55 sbin
dr-xr-xr-x   11 root     root             0 Jan  1  1970 sys
drwxrwxrwt   17 root     root           440 Sep 13 17:42 tmp
drwxr-xr-x    1 root     root          4096 Jul  9 13:15 usr
lrwxrwxrwx    1 root     root             4 Jun 14 19:13 var -> /tmp
drwxr-xr-x    1 root     root          4096 Jun 14 13:52 www
root@yunavr /root [#]#
```

```
[    0.000000] Linux version 3.18.11 (arturo@serverB) (gcc version 4.8.3
(OpenWrt/Linaro GCC 4.8-2014.04 r47374) ) #1 Tue Jun 14 21:17:14 CEST 2016
[    0.000000] MyLoader: sysp=1aa33732, boardp=30bfb91b, parts=f2bc5303
[    0.000000] bootconsole [early0] enabled
[    0.000000] CPU0 revision is: 00019374 (MIPS 24Kc)
[    0.000000] SoC: Atheros AR9330 rev 1
[    0.000000] Determined physical RAM map:
[    0.000000]  memory: 04000000 @ 00000000 (usable)
[    0.000000] User-defined physical RAM map:
[    0.000000]  memory: 04000000 @ 00000000 (usable)
[    0.000000] Initrd not found or empty - disabling initrd
[    0.000000] Zone ranges:
[    0.000000]   Normal   [mem 0x00000000-0x03ffffff]
[    0.000000] Movable zone start for each node
[    0.000000] Early memory node ranges
[    0.000000]   node   0: [mem 0x00000000-0x03ffffff]
[    0.000000] Initmem setup node 0 [mem 0x00000000-0x03ffffff]
[    0.000000] On node 0 totalpages: 16384
[    0.000000] free_area_init_node: node 0, pgdat 803c4400, node_mem_map
81000000
[    0.000000]   Normal zone: 128 pages used for memmap
[    0.000000]   Normal zone: 0 pages reserved
[    0.000000]   Normal zone: 16384 pages, LIFO batch:3
```

```
Wed Aug  3 10:32:35 2016 daemon.info avahi-daemon[2389]: New relevant
interface wlan0.IPv4 for mDNS.
Wed Aug  3 10:32:35 2016 daemon.info avahi-daemon[2389]: Network
interface enumeration completed.
Wed Aug  3 10:32:35 2016 daemon.info avahi-daemon[2389]: Registering new
address record for fe80::b621:8aff:fef0:1378 on wlan0.*.
Wed Aug  3 10:32:35 2016 daemon.info avahi-daemon[2389]: Registering new
address record for 192.168.240.1 on wlan0.IPv4.
Wed Aug  3 10:32:35 2016 daemon.info avahi-daemon[2389]: Registering
HINFO record with values 'MIPS'/'LINUX'.
Wed Aug  3 10:32:36 2016 user.emerg syslog: Starting stnfcd: Device
selected is /dev/ttyNFC
Wed Aug  3 10:32:36 2016 user.emerg syslog: Dev file open failed
Wed Aug  3 10:32:36 2016 user.emerg syslog: OK
Wed Aug  3 10:32:36 2016 daemon.info neard[2398]: NEAR daemon version
0.16
Wed Aug  3 10:32:36 2016 user.emerg syslog: Starting neard: OK
Wed Aug  3 10:32:36 2016 daemon.err block: Unkown action change
Wed Aug  3 10:32:36 2016 daemon.info avahi-daemon[2389]: Server startup
complete. Host name is yunavr.local. Local service cookie is 1683589280.
Wed Aug  3 10:32:36 2016 daemon.err block: Unkown action change
Wed Aug  3 10:32:37 2016 daemon.err block: Unkown action change
Wed Aug  3 10:32:37 2016 daemon.err block: Unkown action change
Wed Aug  3 10:32:38 2016 daemon.info avahi-daemon[2389]: Service "yunavr"
(/etc/avahi/services/http.service) successfully established.
Wed Aug  3 10:32:38 2016 daemon.err block: Unkown action change
Wed Aug  3 10:32:38 2016 daemon.err block: Unkown action change
Wed Aug  3 10:32:38 2016 user.emerg syslog: setting up led WLAN
```

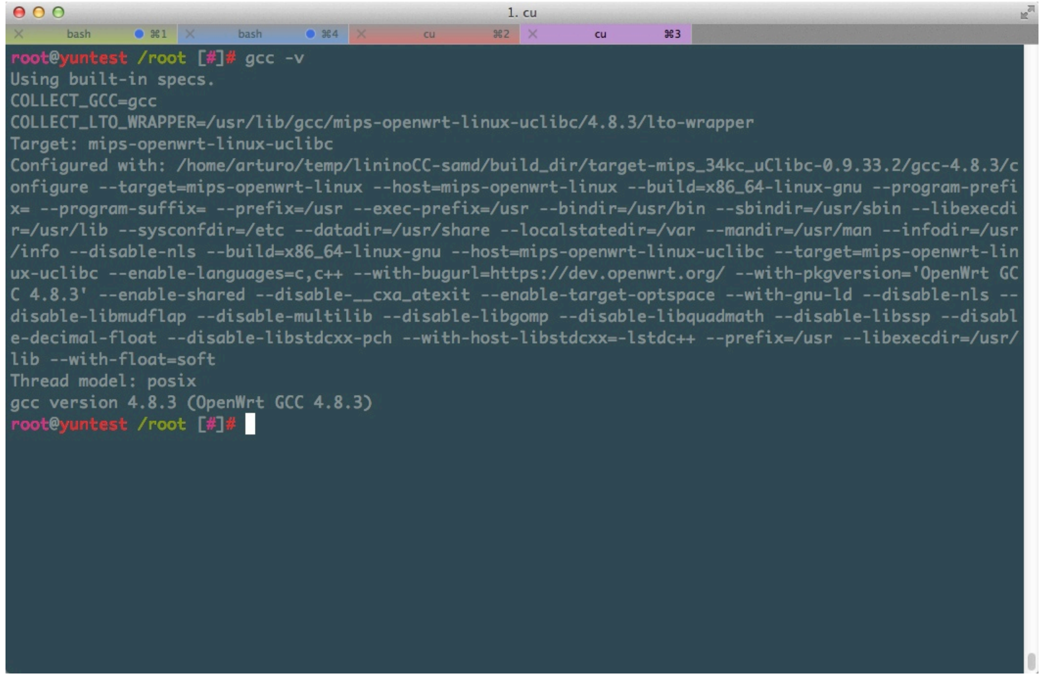
A fully-fledged terminal based on shellinabox residing in a pop-up web frame.

You can access all the linux functionalities from the very same web page.

It relies on SSH of course for basic login.

Possibility to open the terminal in a new browser tab as well.

The “GCC family” – the “plain” GCC



```
root@yuntest /root [#]# gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/mips-openwrt-linux-uclibc/4.8.3/lto-wrapper
Target: mips-openwrt-linux-uclibc
Configured with: /home/arturo/temp/lininoCC-samd/build_dir/target-mips_34kc_uClibc-0.9.33.2/gcc-4.8.3/c
onfigure --target=mips-openwrt-linux --host=mips-openwrt-linux --build=x86_64-linux-gnu --program-prefi
x= --program-suffix= --prefix=/usr --exec-prefix=/usr --bindir=/usr/bin --sbindir=/usr/sbin --libexecdi
r=/usr/lib --sysconfdir=/etc --datadir=/usr/share --localstatedir=/var --mandir=/usr/man --infodir=/usr
/info --disable-nls --build=x86_64-linux-gnu --host=mips-openwrt-linux-uclibc --target=mips-openwrt-lin
ux-uclibc --enable-languages=c,c++ --with-bugurl=https://dev.openwrt.org/ --with-pkgversion='OpenWrt GC
C 4.8.3' --enable-shared --disable-__cxa_atexit --enable-target-optspace --with-gnu-ld --disable-nls --
disable-libmudflap --disable-multilib --disable-libgomp --disable-libquadmath --disable-libssp --disabl
e-decimal-float --disable-libstdcxx-pch --with-host-libstdcxx=-lstdc++ --prefix=/usr --libexecdir=/usr/
lib --with-float=soft
Thread model: posix
gcc version 4.8.3 (OpenWrt GCC 4.8.3)
root@yuntest /root [#]#
```

To install it just type from shell:

```
$ opkg update
$ opkg install gcc
```

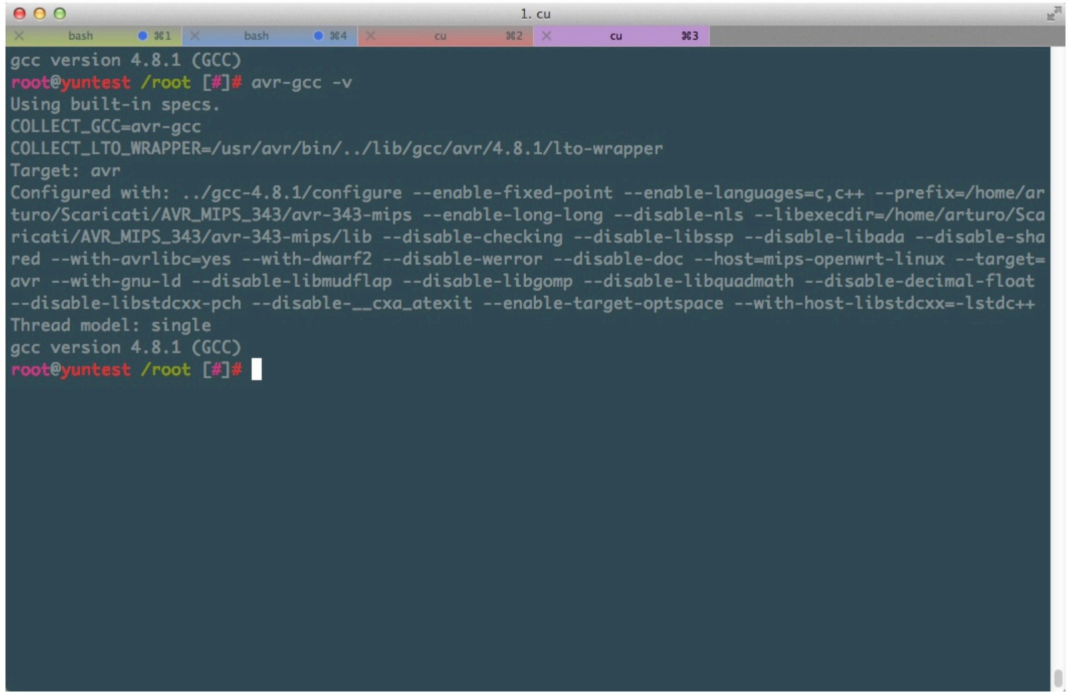
The GCC is officially residing in the OpenWRT repositories since the latest stable release of the distro : Chaos Calmer 15.05

But….we were also able to provide GCC v4.6.3 for our old distro based on Attitude Adjustment 12.09 leveraging the work made for Chaos Calmer 15.05

Please keep in mind the limitations of the current C library (uClibc) when building an executable, or provide patches to enhance its overall functionalities.

That means rebuilding the whole linux image of course…

The "GCC family" – AVR toolchain



```
gcc version 4.8.1 (GCC)
root@yuntest /root [#]# avr-gcc -v
Using built-in specs.
COLLECT_GCC=avr-gcc
COLLECT_LTO_WRAPPER=/usr/avr/bin/../lib/gcc/avr/4.8.1/lto-wrapper
Target: avr
Configured with: ../gcc-4.8.1/configure --enable-fixed-point --enable-languages=c,c++ --prefix=/home/ar
turo/Scaricati/AVR_MIPS_343/avr-343-mips --enable-long-long --disable-nls --libexecdir=/home/arturo/Sca
ricati/AVR_MIPS_343/avr-343-mips/lib --disable-checking --disable-libssp --disable-libada --disable-sha
red --with-avrlibc=yes --with-dwarf2 --disable-werror --disable-doc --host=mips-openwrt-linux --target=
avr --with-gnu-ld --disable-libmudflap --disable-libgomp --disable-libquadmath --disable-decimal-float
--disable-libstdcxx-pch --disable-__cxa_atexit --enable-target-optspace --with-host-libstdcxx=-lstdc++
Thread model: single
gcc version 4.8.1 (GCC)
root@yuntest /root [#]#
```

Follow our guide here :

http://goo.gl/PiLCFV

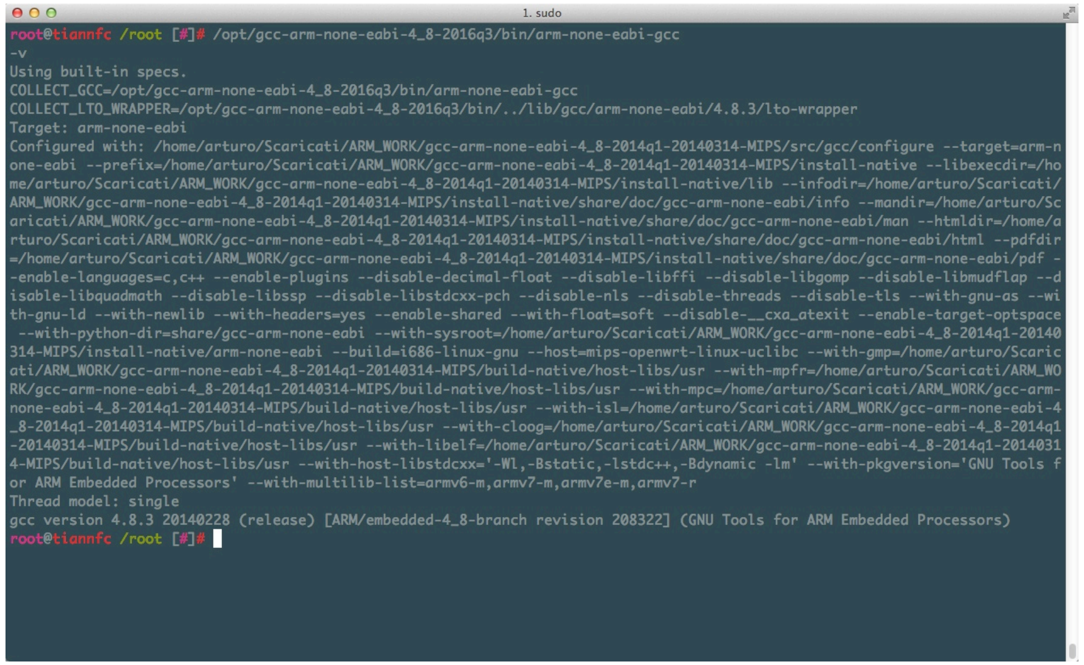
The AVR toolchain has been built with the **Canadian-Cross** compilation :

- is a well known technique for build ing cross-compilers for other machines especially if the given target is a particular architecture such as **AVR**, **ARM** or **xtensa**.
- It implies the existence of a native 'target' compiler in your search **PATH**.

To build the firmware on our board we need :

- the arduino libraries (ipk provided)
- An Arduino *.ino sketch
- The **make** command
- A prebuilt makefile to set the necessary paths

The “GCC family” – ARM toolchain


1. sudo

```
root@tiannfc /root [#]# /opt/gcc-arm-none-eabi-4_8-2016q3/bin/arm-none-eabi-gcc
-v
Using built-in specs.
COLLECT_GCC=/opt/gcc-arm-none-eabi-4_8-2016q3/bin/arm-none-eabi-gcc
COLLECT_LTO_WRAPPER=/opt/gcc-arm-none-eabi-4_8-2016q3/bin/../lib/gcc/arm-none-eabi/4.8.3/lto-wrapper
Target: arm-none-eabi
Configured with: /home/arturo/Scaricati/ARM_WORK/gcc-arm-none-eabi-4_8-2014q1-20140314-MIPS/src/gcc/configure --target=arm-n
one-eabi --prefix=/home/arturo/Scaricati/ARM_WORK/gcc-arm-none-eabi-4_8-2014q1-20140314-MIPS/install-native --libexecdir=/ho
me/arturo/Scaricati/ARM_WORK/gcc-arm-none-eabi-4_8-2014q1-20140314-MIPS/install-native/lib --infodir=/home/arturo/Scaricati/
ARM_WORK/gcc-arm-none-eabi-4_8-2014q1-20140314-MIPS/install-native/share/doc/gcc-arm-none-eabi/info --mandir=/home/arturo/Sc
aricati/ARM_WORK/gcc-arm-none-eabi-4_8-2014q1-20140314-MIPS/install-native/share/doc/gcc-arm-none-eabi/man --htmldir=/home/a
rturo/Scaricati/ARM_WORK/gcc-arm-none-eabi-4_8-2014q1-20140314-MIPS/install-native/share/doc/gcc-arm-none-eabi/html --pdfdir
=/home/arturo/Scaricati/ARM_WORK/gcc-arm-none-eabi-4_8-2014q1-20140314-MIPS/install-native/share/doc/gcc-arm-none-eabi/pdf -
-enable-languages=c,c++ --enable-plugins --disable-decimal-float --disable-libffi --disable-libgomp --disable-libmudflap --d
isable-libquadmath --disable-libssp --disable-libstdcxx-pch --disable-nls --disable-threads --disable-tls --with-gnu-as --wi
th-gnu-ld --with-newlib --with-headers=yes --enable-shared --with-float=soft --disable-__cxa_atexit --enable-target-optspace
 --with-python-dir=share/gcc-arm-none-eabi --with-sysroot=/home/arturo/Scaricati/ARM_WORK/gcc-arm-none-eabi-4_8-2014q1-20140
314-MIPS/install-native/arm-none-eabi --build=i686-linux-gnu --host=mips-openwrt-linux-uclibc --with-gmp=/home/arturo/Scaric
ati/ARM_WORK/gcc-arm-none-eabi-4_8-2014q1-20140314-MIPS/build-native/host-libs/usr --with-mpfr=/home/arturo/Scaricati/ARM_WO
RK/gcc-arm-none-eabi-4_8-2014q1-20140314-MIPS/build-native/host-libs/usr --with-mpc=/home/arturo/Scaricati/ARM_WORK/gcc-arm-
none-eabi-4_8-2014q1-20140314-MIPS/build-native/host-libs/usr --with-isl=/home/arturo/Scaricati/ARM_WORK/gcc-arm-none-eabi-4
_8-2014q1-20140314-MIPS/build-native/host-libs/usr --with-cloog=/home/arturo/Scaricati/ARM_WORK/gcc-arm-none-eabi-4_8-2014q1
-20140314-MIPS/build-native/host-libs/usr --with-libelf=/home/arturo/Scaricati/ARM_WORK/gcc-arm-none-eabi-4_8-2014q1-2014031
4-MIPS/build-native/host-libs/usr --with-host-libstdcxx='-Wl,-Bstatic,-lstdc++,-Bdynamic -lm' --with-pkgversion='GNU Tools f
or ARM Embedded Processors' --with-multilib-list=armv6-m,armv7-m,armv7e-m,armv7-r
Thread model: single
gcc version 4.8.3 20140228 (release) [ARM/embedded-4_8-branch revision 208322] (GNU Tools for ARM Embedded Processors)
root@tiannfc /root [#]#
```

Then it’s just a matter to replace the original entries with the cross ones in the prebuilt bash scripts shipped with the original sources.

The ARM toolchain has been built with the **Canadian-Cross** technique.

We need to leverage the well-known **Crosstools-NG suite** to generate a sysroot-ed cross-toolchain for building the actual one.

To build the firmware we need, as the **AVR** platform :

- the arduino libraries
- An Arduino *.ino sketch
- The **make** command
- A prebuilt makefile to set the necessary paths for the **ARM** architecture

The "GCC family" – Demonstration

The toolchains have also been built for a testbed **x86** machine for running as VM on VirtualBox.

It has been proven a reliable system to test the software before deploying it on our boards.

To build the firmware for **AVR** or **ARM** platform we again need :

- the arduino libraries
- An Arduino.ino sketch
- The **make** command
- A prebuilt makefile to set the necessary paths for the architecture

So…. .let's run the demo !

THANK YOU

For further information please get in touch with me at :

arturo@arduino.org

Embedded Linux Conference Europe