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

Linino = 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!
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!
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, ...

Berlin, October 12th 2016
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, ...

Berlin, October 12th 2016
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

Berlin, October 12th 2016
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:

```
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

So now we run:

```
$ lingpio export
```

To enumerate all the available devices to use.

We are now ready to play around!
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

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.
- DIY and maker’s projects
- mcu agnostic (atmel, freescale, stm, etc.)
- professional projects
- mass production
- and of course all linino devices

Berlin, October 12th 2016
Linino “Open Stack”

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

chiwawa module
mips mpu + wi-fi

mcu based device
lininoIO operating system
Linino “Open Stack”

- Linino OpenWRT based Operating System
- Chiwawa module
  - mips mpu + wi-fi
- MCU based device
  - LininoIO operating system

- Linux open source operating system
- Easy package management system
- More than 2000 packages
- Robust, widely adopted
Linino “Open Stack”

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

• 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"

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

- your applications
- mobile applications
- cloud applications

... for your applications
... for mobile applications
... for cloud applications

- monitoring
- robotics
- remote control
- home automation
- vehicle automation
- drones control
- energy management
- ...
ArduinoOS

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

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

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

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

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

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

- Theme
- Desktop
- Panels
- User
- File View
This panel grants you a quick access to the Arduino specific settings:

- General System information
- Network Devices
- System
- WiFi
- Rest API
ArduinoOS – Arduino Settings

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

- General System information
- Network Devices
- System
- WiFi
- Rest API
ArduinoOS – Arduino Settings

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

- General System information
- Network Devices
- System
- WiFi
- Rest API
ArduinoOS – Arduino Settings

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

- General System information
- Network Devices
- System
- WiFi
- Rest API
This panel grants you a quick access to the Arduino specific settings:

- General System information
- Network Devices
- System
- WiFi
- Rest API
ArduinoOS – Desktop Utilities

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

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

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

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...

To install it just type from shell:

```
$ opkg update
$ opkg install gcc
```
The “GCC family” – AVR toolchain

The AVR toolchain has been built with the **Canadian-Cross compilation**:
- is a well known technique for building 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

Follow our guide here:

http://goo.gl/PiLCFV
The “GCC family” – ARM toolchain

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

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 “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