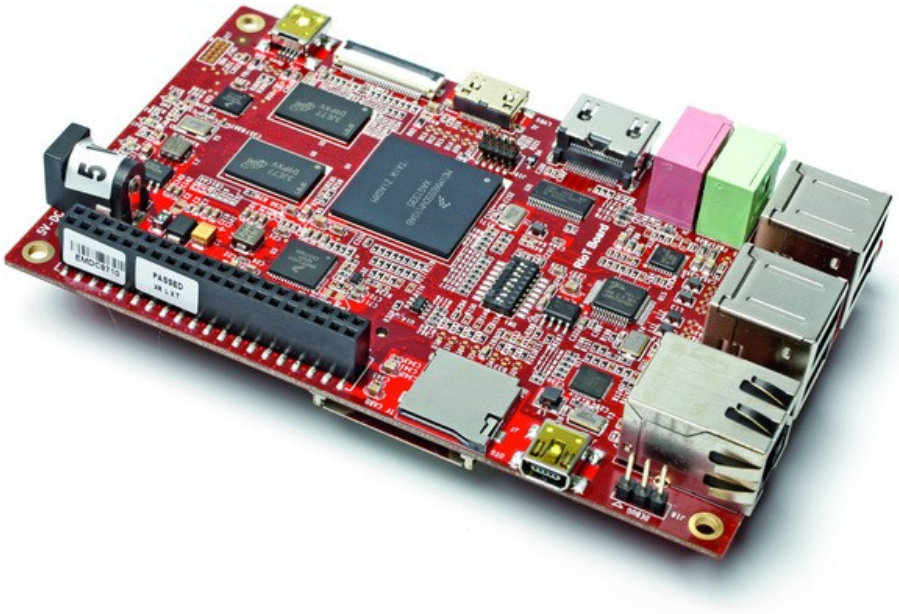
Freie Universität Berlin

Porting RIOT OS to the RioT-Board

by Lennart Dührsen and Leon George

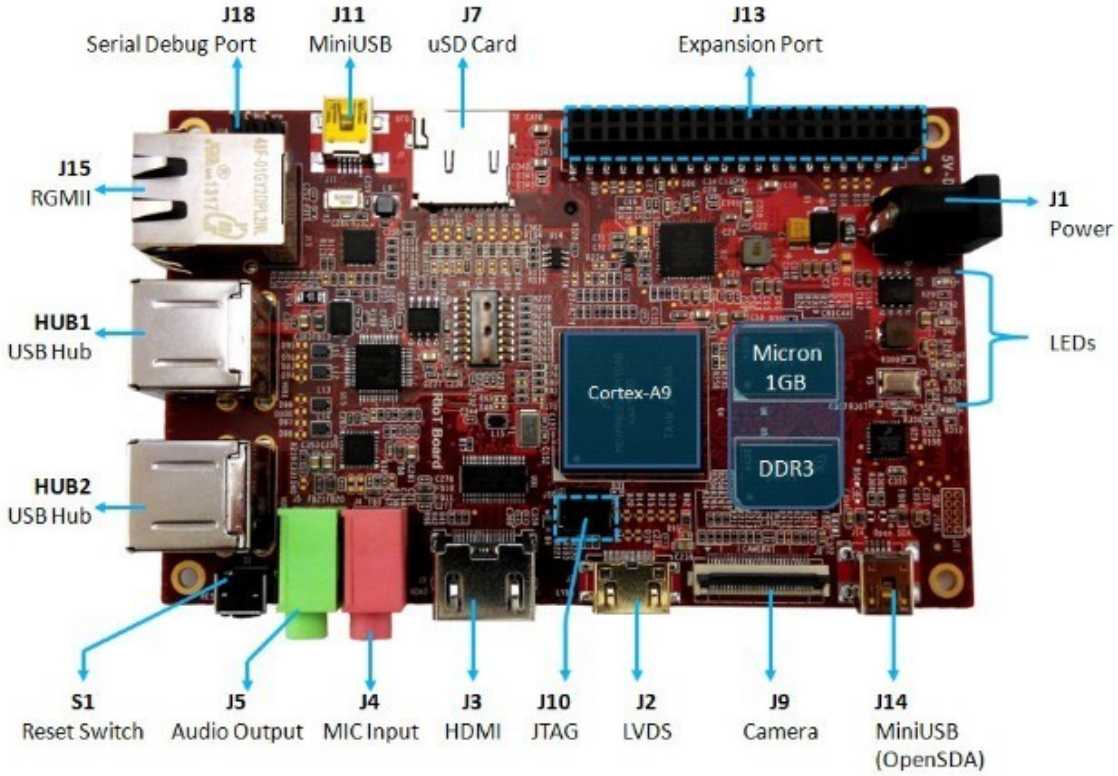
RioT-Board



RioT-Board

- Introduced by Freescale Semiconductor in early 2014
- Based on the i.MX6Solo Application Processor
- Ships with Android installed, Linux (Ubuntu) also available
- Costs ca. 70€
- Low Energy Consumption (if you don't charge your phone using the USB-Ports :-)), although quite powerful

RioT-Board


J18
Serial Debug Port
J11
MiniUSB
J7
uSD Card
J13
Expansion Port
J15
RGMII
J1
Power
HUB1
USB Hub
LEDs
HUB2
USB Hub
Cortex-A9
Micron
1GB
DDR3
S1
Reset Switch
J5
Audio Output
J4
MIC Input
J3
HDMI
J10
JTAG
J2
LVDS
J9
Camera
J14
MiniUSB
(OpenSDA)

RioT-Board

CPU:

- i.MX6Solo
- ARM Cortex-A9 (32Bit)
- 1GHz, Single Core
- 32 KByte L1 Instruction Cache
- 32 KByte L1 Data Cache
- 512 KB unified I/D L2 cache
- Private Timer and Watchdog

RioT-Board

Memory:

- Boot ROM 96KB
- Internal RAM 144KB
- DDR3 RAM 1GB

RioT-Board

Interfaces:

- 3 Pin Debug Port (UART)
- JTAG
- USB (4x Host, 1x OTG)
- 10MBit/100MBit/1GBit Ethernet
- GPIO Expansion Port
- I^2C

RioT-Board

What to do with it?

- Suitable for many tasks, including
  - Personal Computer
  - Multimedia-Station (XBMC)
  - Small Server
  - ...

But what about IoT?

RIoT-Board

Adding some sort of wireless Transceiver/Receiver (and sensors maybe), it could be

- Part of a typical sensor network (you'd need an AC-Adaptor or a really strong battery)
- A Router (Home-Wifi, AODV, OLSR, RPL...)
- Central instance of a sensor network that collects all the data and processes it
- ...

RIOT-OS

...well you know about RIOT, obviously ;-)

But the question is: **How to port it?**

RIOT-OS

- Operating Systems are (mostly) written in C
- Most parts are hardware generic
- But some have to be written exactly for the underlying metal!
- e.g. the booting process, context switch, timer etc.

RIOT-OS

```
#include "irq.h"
#include <stdio.h>
#include "kernel_internal.h"

extern void board_init(void);

__attribute__((constructor)) static void startup(void)
{
    /* use putchar so the linker links it in: */
    putchar('\n');

    board_init();

    puts("RIOT MSP430 hardware initialization complete.\n");

    kernel_init();
}
```

RIOT-OS

To get RIOT to run, the following has to work:

- UART I/O for debugging and shell communication
- Timer(s) so the kernel can run
- Interrupts
- Set up a stack
- Build it successfully (probably the hardest part :-))

Strategy

- Read documentation
- Read more documentation
- Adjust folder structure and create Makefiles
- Split work in an ad hoc way, as soon as

  we know what we're doing

Strategy

Good luck to ourselfes, thank you for listening!