

OCaml on the ESP32 chip

Well typed lightbulbs await

Lucas Pluinage – ENS Paris

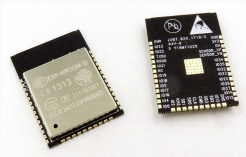
OCaml Workshop – ICFP 2018



- A language: OCaml



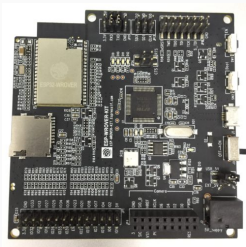
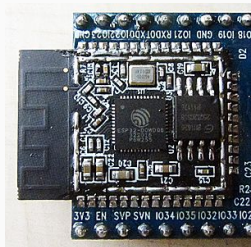
- A language: OCaml
- A platform: ESP32



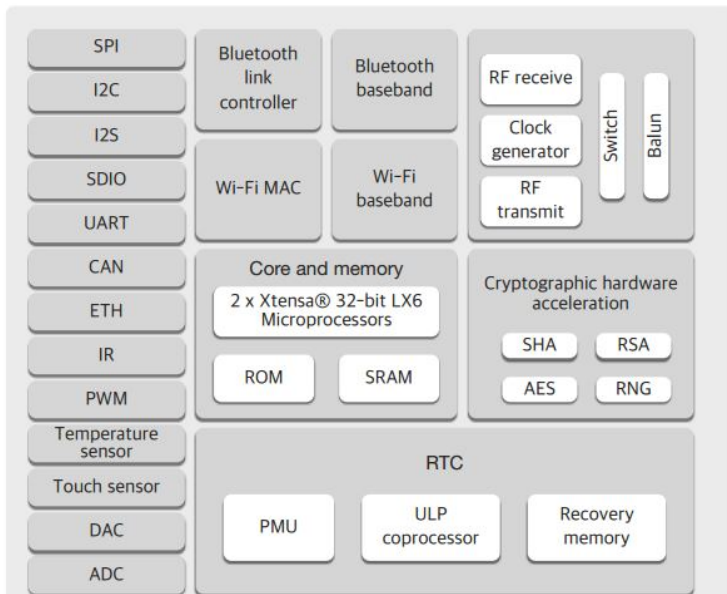
- A language: OCaml
- A platform: ESP32
- An application library: Mirage



ESP32 microcontrollers



ESP32 microcontrollers – hardware

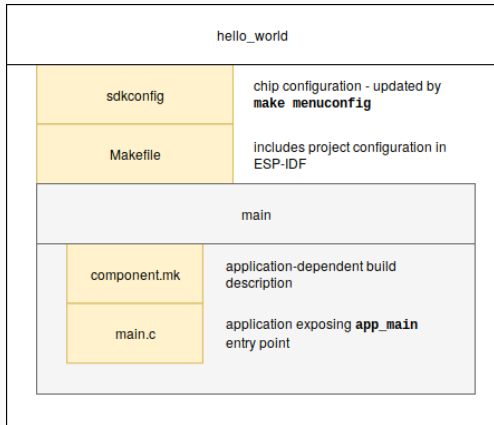


ESP32 microcontrollers – software

- Espressif IoT Development Framework (ESP-IDF)
- FreeRTOS (Real-Time Operating System)
- Written in C – Xtensa backend for GCC
- MicroPython port available

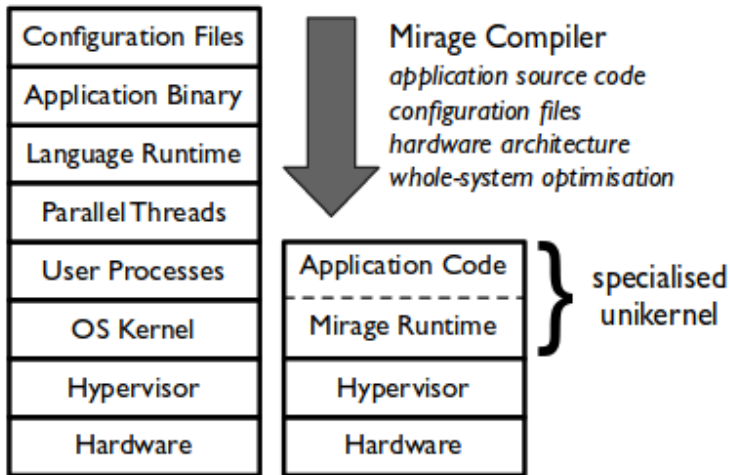
ESP32 microcontrollers – software

- Espressif IoT Development Framework (ESP-IDF)
- FreeRTOS (Real-Time Operating System)
- Written in C – Xtensa backend for GCC
- MicroPython port available



Mirage unikernel framework

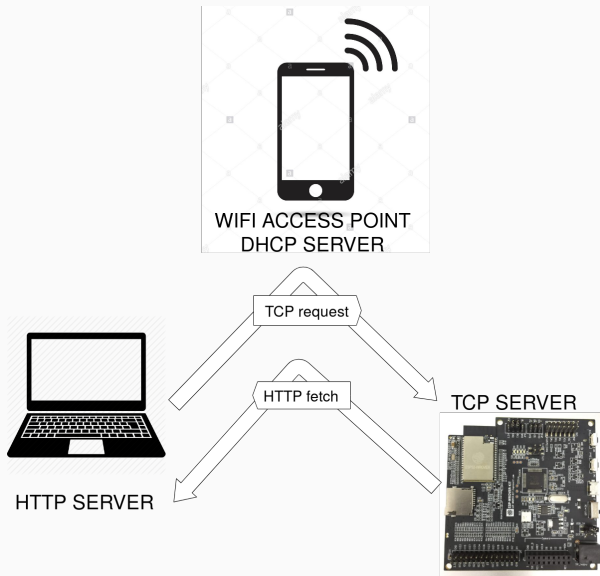
What is an unikernel ?



Picture from *Unikernels: Library Operating Systems for the Cloud*

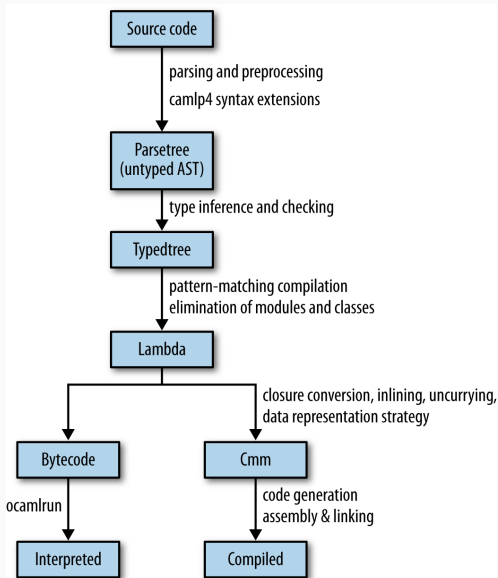
Time for a demonstration

Time for a demonstration

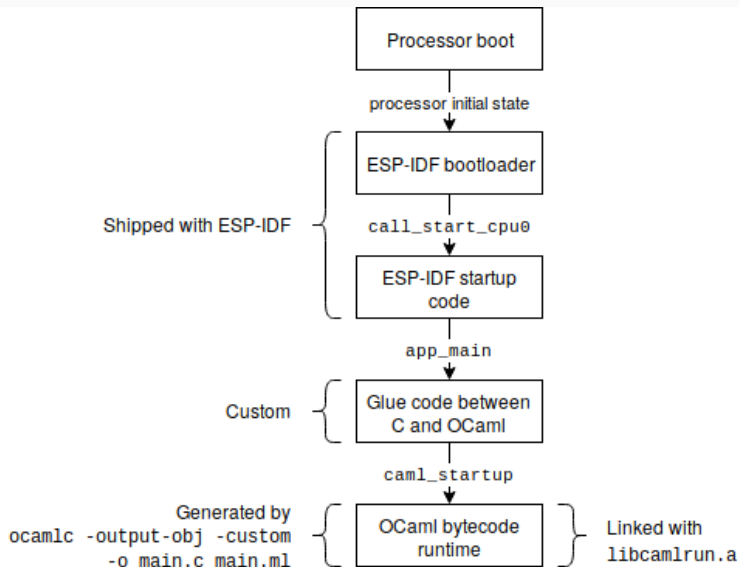


Compiling OCaml for ESP32

Compilation paths



Bytecode execution path on ESP32

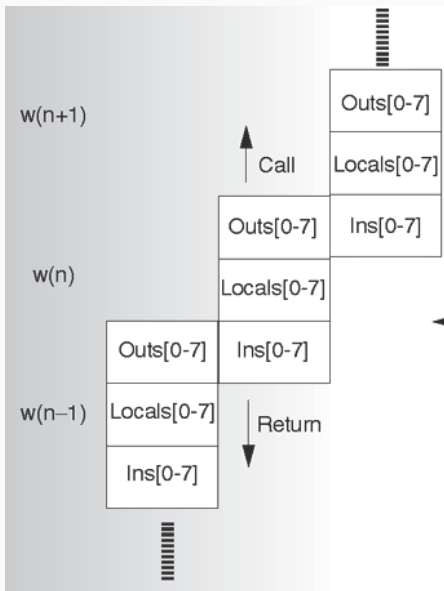


OCaml compiler backend

- `asmcomp/xtensa/`
 - `proc.ml`: processor and calling conventions
 - `arch.ml`: architecture
 - `emit.ml`: assembly emission
- `asmrun/xtensa.S` runtime interface between OCaml and C

No interference with the OCaml compiler code !

Register windowing and calling conventions

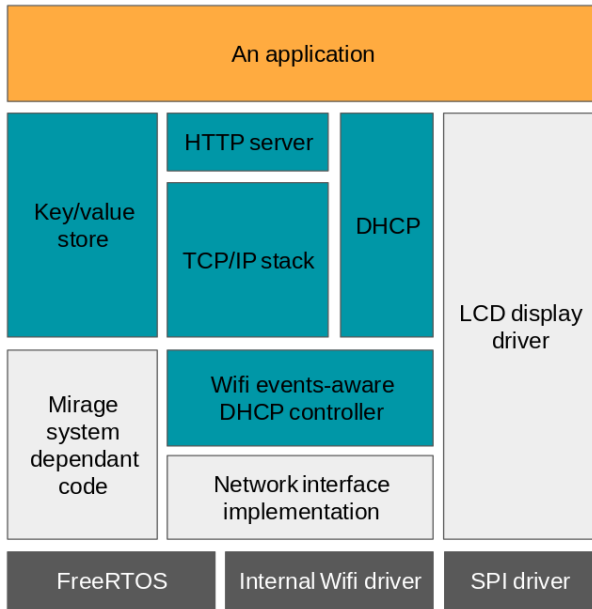


Cross-compiling for ESP32 microcontrollers

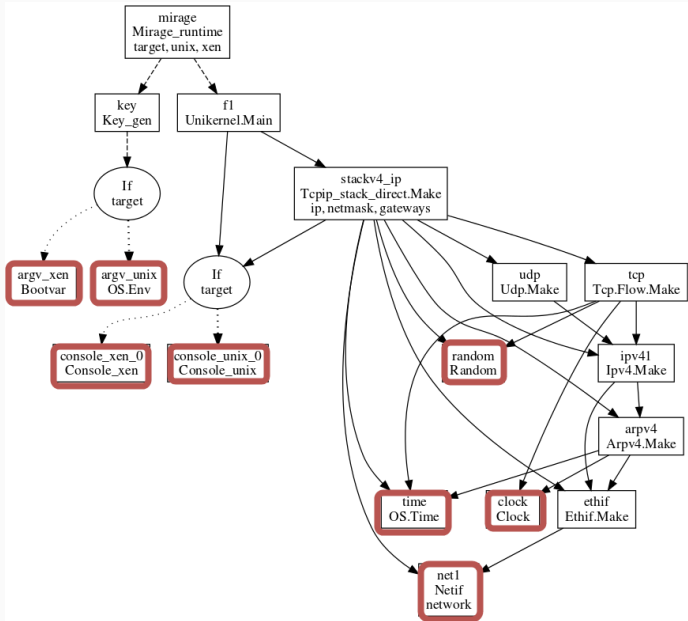
- Integration with build systems: from a single parameter to more extensive tweaking.
- Integration with opam:
 - OCaml 4.06.0+32bit switch
 - Cross-compiler in `[switch root]/esp32-sysroot`
 - This allows to access both host and target packages.
- `opam-cross-esp32`: 127 packages ported for cross-compilation.

Unikernels for embedded applications

Unikernels and the Mirage project

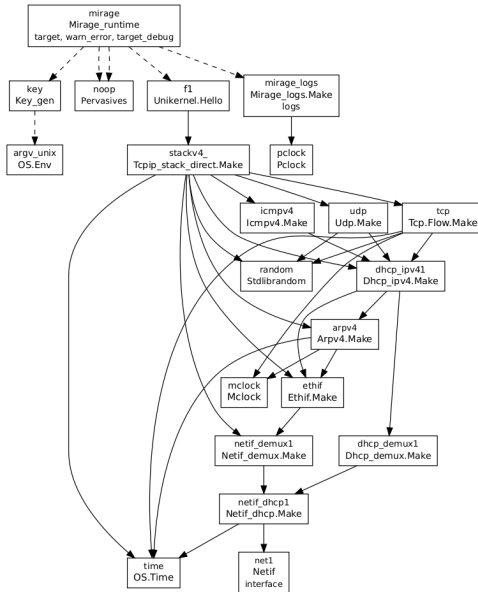


What to you need to build a standalone application ?



- Collaborative threading with Lwt library:
bind: 'a Lwt.t -> ('a -> 'b Lwt.t) -> 'b Lwt.t
return: 'a -> a Lwt.t
join: unit Lwt.t list -> unit Lwt.t
pick: 'a Lwt.t list -> 'a Lwt.t
- Timer feature:
Time.sleep_ns: int64 -> unit Lwt.t
- Event system:
Event.wait_for_event: int -> unit Lwt.t

Porting network features



Porting network features

- Netif:
 - `write: t -> buffer -> (unit, error) result Lwt.t`
 - `listen: t -> (buffer -> unit io) -> (unit, error) result Lwt.t`
 - `mac: t -> macaddr`
 - `get_stats_counters, reset_stats_counters`
- Netif_DHCP: input a Netif and outputs a Netif and a DHCP module. Acts as a multiplexer.

Results

Applications

- LCD screen control
- Wifi AP/Station mode/both
- HTTPS
- DHCP
- DNS

Applications

- LCD screen control
- Wifi AP/Station mode/both
- HTTPS
- DHCP
- DNS

Application	Code	Magic (LTO)	Rodata	Dynamic RAM
Hello world	764K	270K	151K	133K
AP - DHCP server	1058K	405K	256K	270K
STA - DHCP client	1217K	446K	289K	215K
HTTP fetch	2366K	1083K	622K	600K
HTTPS fetch	2364K	1224K	735K	700K
LCD canvas over HTTP	2368K	1038K	592K	700K

Applications

- LCD screen control
- Wifi AP/Station mode/both
- HTTPS
- DHCP
- DNS

Application	Code	Magic (LTO)	Rodata	Dynamic RAM
Hello world	764K	270K	151K	133K
AP - DHCP server	1058K	405K	256K	270K
STA - DHCP client	1217K	446K	289K	215K
HTTP fetch	2366K	1083K	622K	600K
HTTPS fetch	2364K	1224K	735K	700K
LCD canvas over HTTP	2368K	1038K	592K	700K

LTO is fantastic! See PR#608 in `ocaml/ocaml`

Main issues

- Memory usage

Main issues

- Memory usage: fixed by micro-optimizing assembly generation, taking care of where data is stored, and porting a dead-code elimination patch

Main issues

- Memory usage: fixed by micro-optimizing assembly generation, taking care of where data is stored, and porting a dead-code elimination patch
- Bad cross-compilation support

Conclusion

Main issues

- Memory usage: fixed by micro-optimizing assembly generation, taking care of where data is stored, and porting a dead-code elimination patch
- Bad cross-compilation support

Overview

- Lot of exploration that resulted in a great proof of concept
- Opportunity for further research in the field of unikernels for embedded devices
- Very pleasant team and lab!

Resources and conclusion

- `well-typed-lightbulbs` Github organization.
- <https://www.lortex.org/esp32/> blog posts.

Resources and conclusion

- well-typed-lightbulbs Github organization.
- <https://www.lortex.org/esp32/> blog posts.



Götz Salzmänn

@gtz42

Suivre



Wow, this makes me want to finally learn ocaml:

Justin Cormack @justincormack

Really excited by Mirage unikernels on ESP32 microcontrollers [lortex.org/posts/mirage/e...](https://www.lortex.org/posts/mirage/e...) #unikernelsarenotaboutvirtualmachines

Traduire le Tweet

19:00 - 30 août 2018

4 J'aime



4

