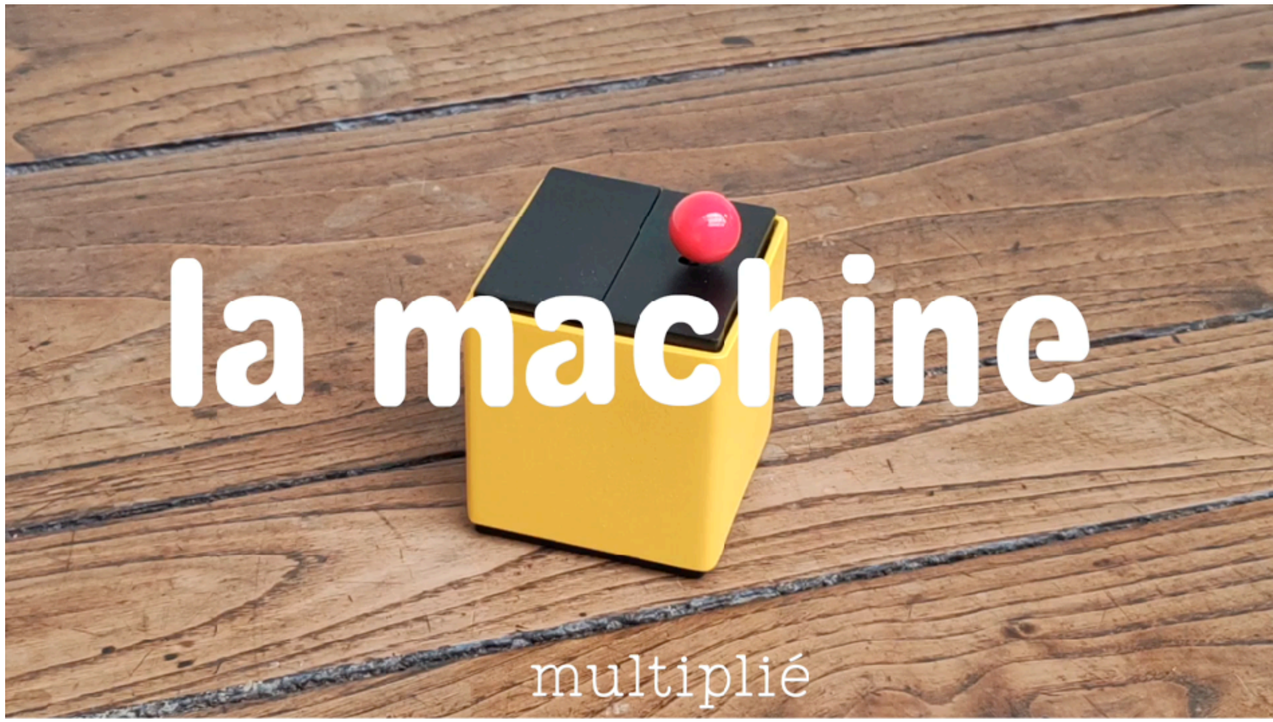
la machine

The Useless Box reloaded with Erlang and AtomVM

Paul Guyot (pguyot@kallisys.net) at Code BEAM Europe 2024

la machine

(september 2024 prototype)



Today's agenda

A brief history of la machine

A glimpse at hardware

AtomVM: a virtual machine and an API for microcontrollers

Software engineering Erlang code for microcontrollers

Going further: extending AtomVM with native code

Next steps & credits

A brief history of la machine

1952

Marvin Minksy invents
the ultimate machine



Sethwoodworth CC BY 3.0

2005

Olivier Mével wants to build a Useless Box.

Instead he creates a Wifi Rabbit with rafi Haladjian

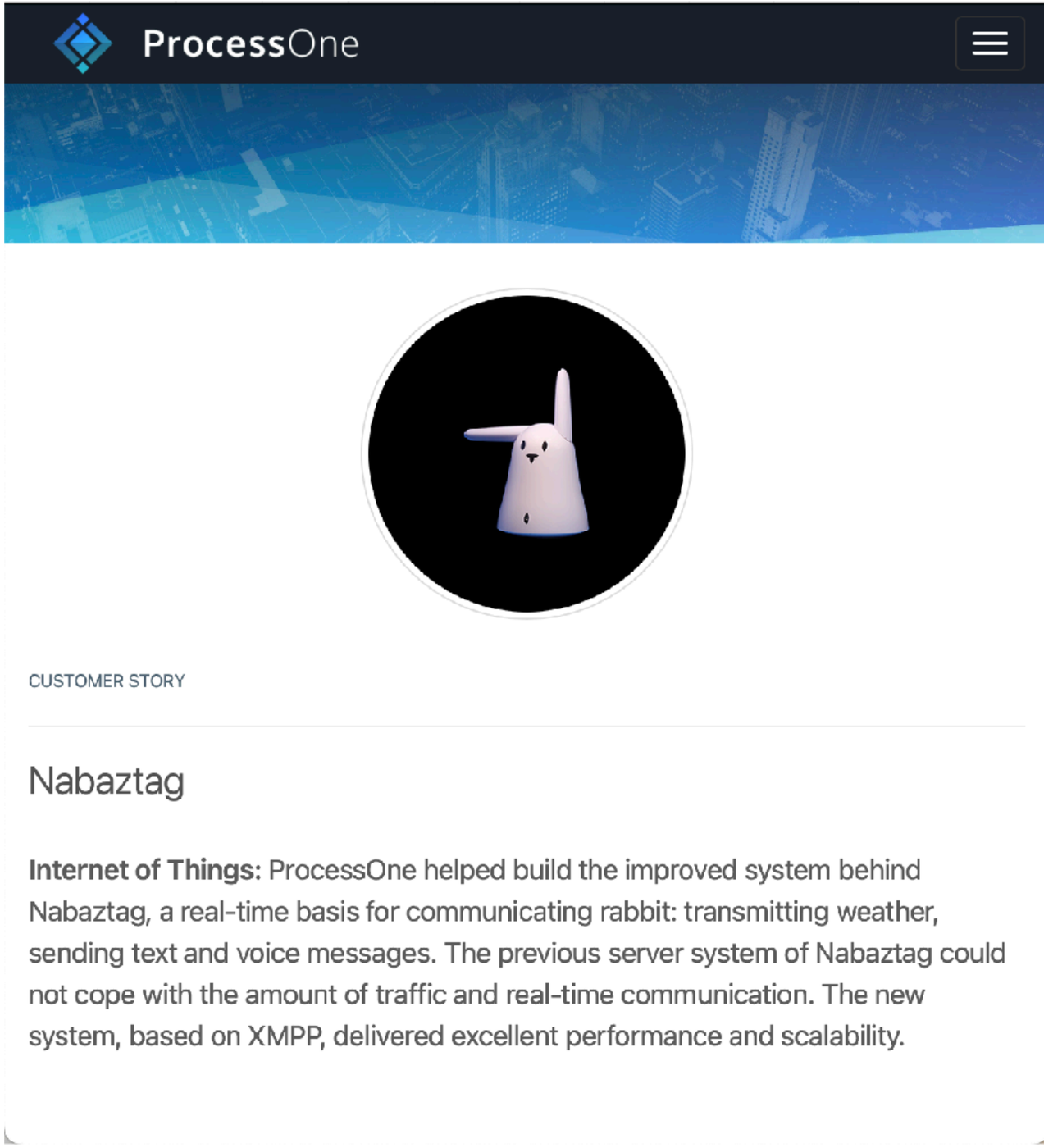


Rama CC BY-SA 2.0 FR

2007

I join Violet as CTO and drive the adoption of ejabberd to connect wifi rabbits.

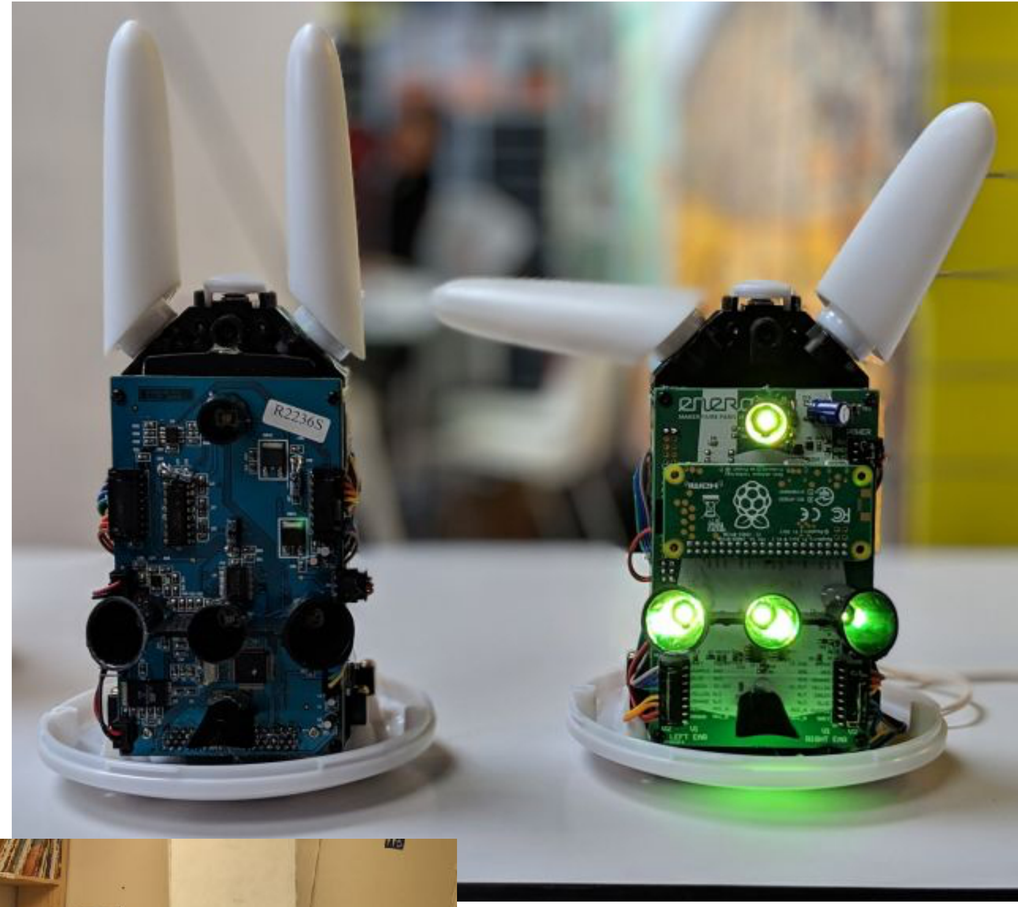
Eventually I learn and come to admire Erlang

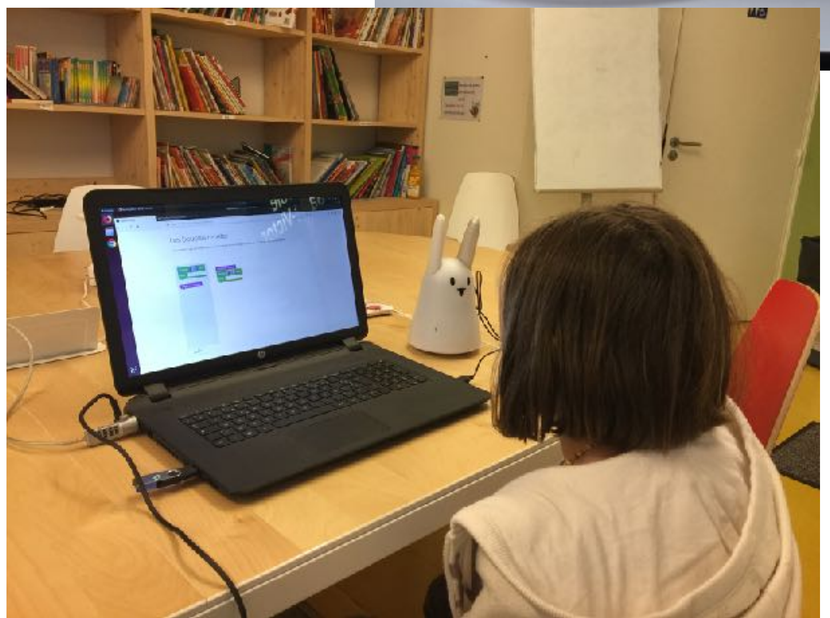


2019

Nabaztag returns with no server but a Raspberry Pi Zero

It runs Python but I write an Erlang app to teach kids how to code





2024

Olivier eventually
prototypes la machine.

I manage to convince him
to power it with Erlang



A glimpse at hardware

la machine v1.1

Button

Servo for the arm

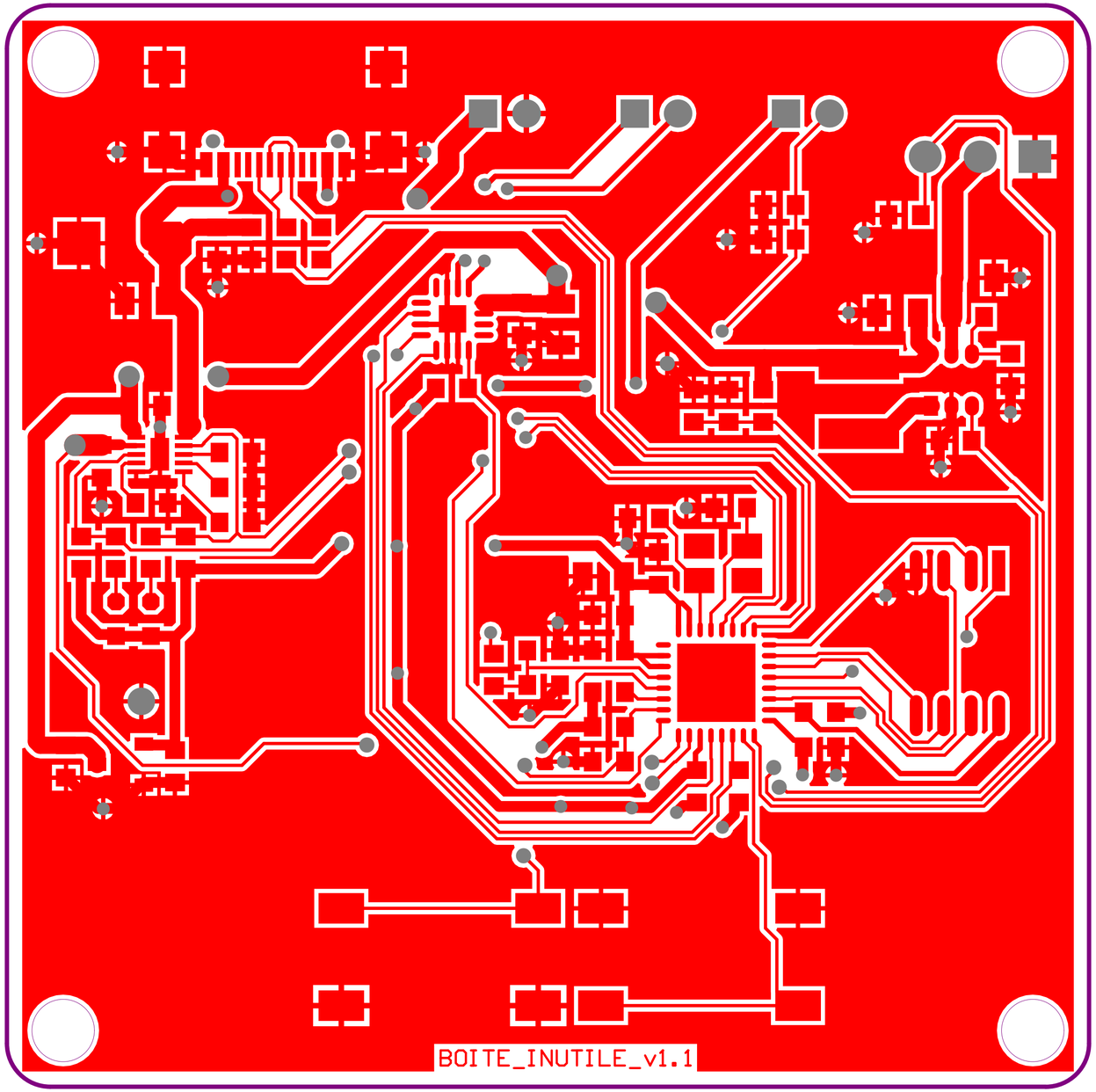
Amplifier and speaker

ESP32-C3 microcontroller

16MB external flash

820mAh battery

USB charging logic



ESP32-C3

32-bit RISC V single core @ 160 MHz
400 KB of SRAM
Ultra-low power (5μA in deep sleep)



AtomVM

BEAM apps within 400 KB or less

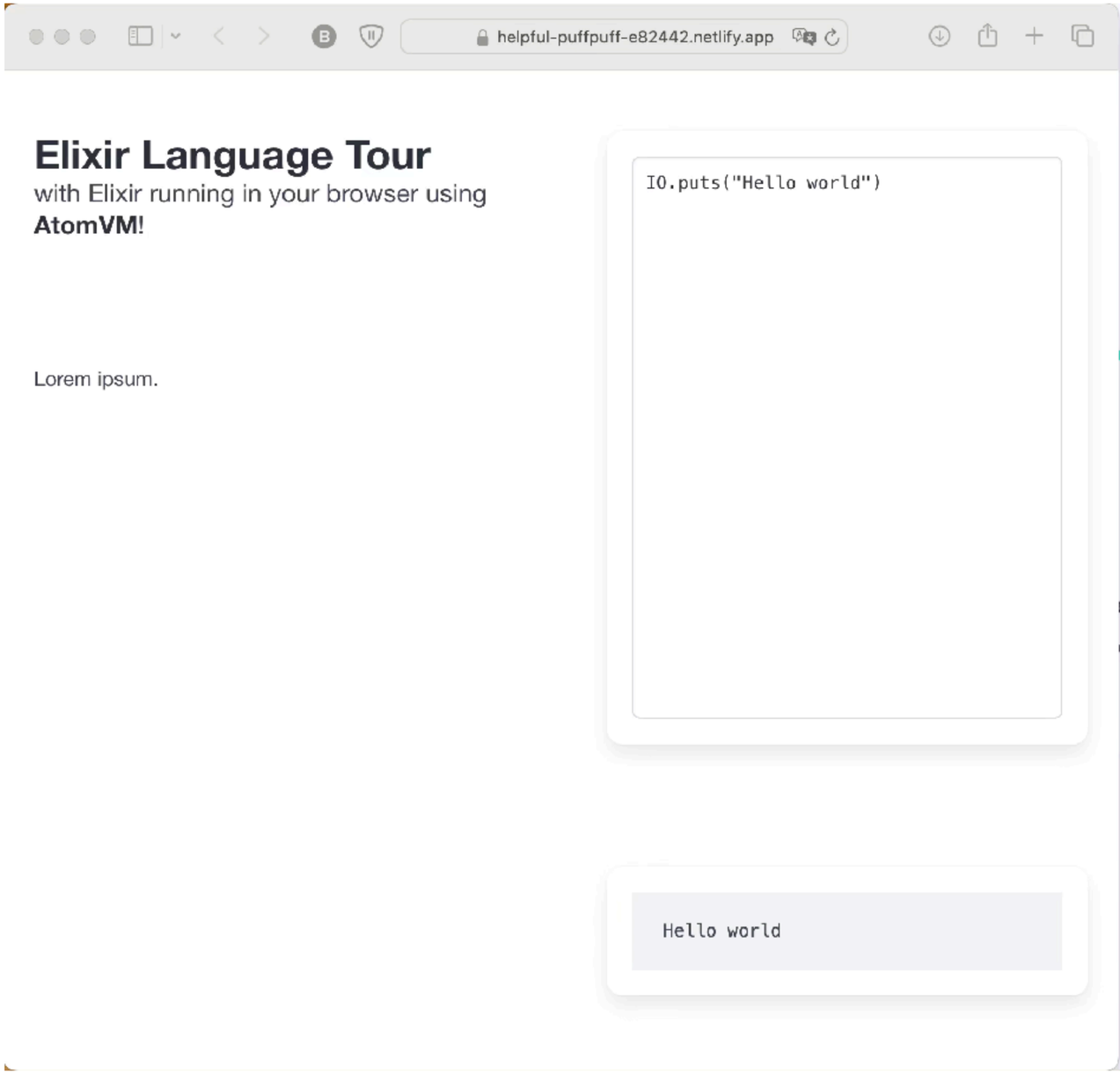
AtomVM

The Erlang virtual machine for IoT devices

Open source project created by Davide Bettio

Initially targetting ESP32, now runs on all ESP32s, STM32, Pico & Pico 2 and even in the browser using WebAssembly.

Subset of **Erlang/OTP** standard libs
Subset of **Elixir** libs
Gleam POC still WIP
Luerl could be fun



GPIOs

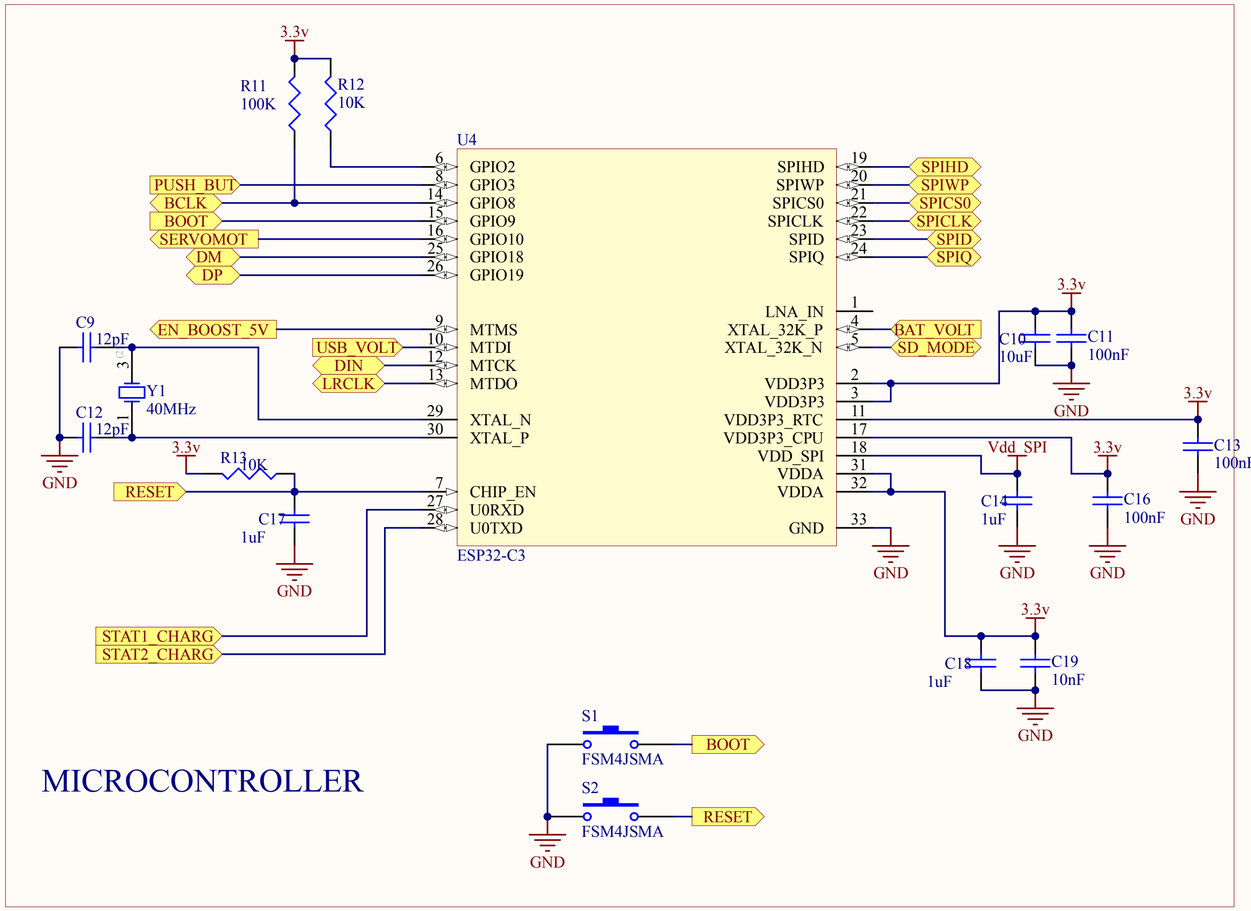
General Purpose Input Output

ESP32-C3 has 22

On **la machine**, GPIO are used for:

- button (read)
- enabling servo (written)
- audio (through driver)
- enabling audio (written)
- detecting charge (read)

Pins are also used for SPI (flash) and ADC (battery voltage)



PWM

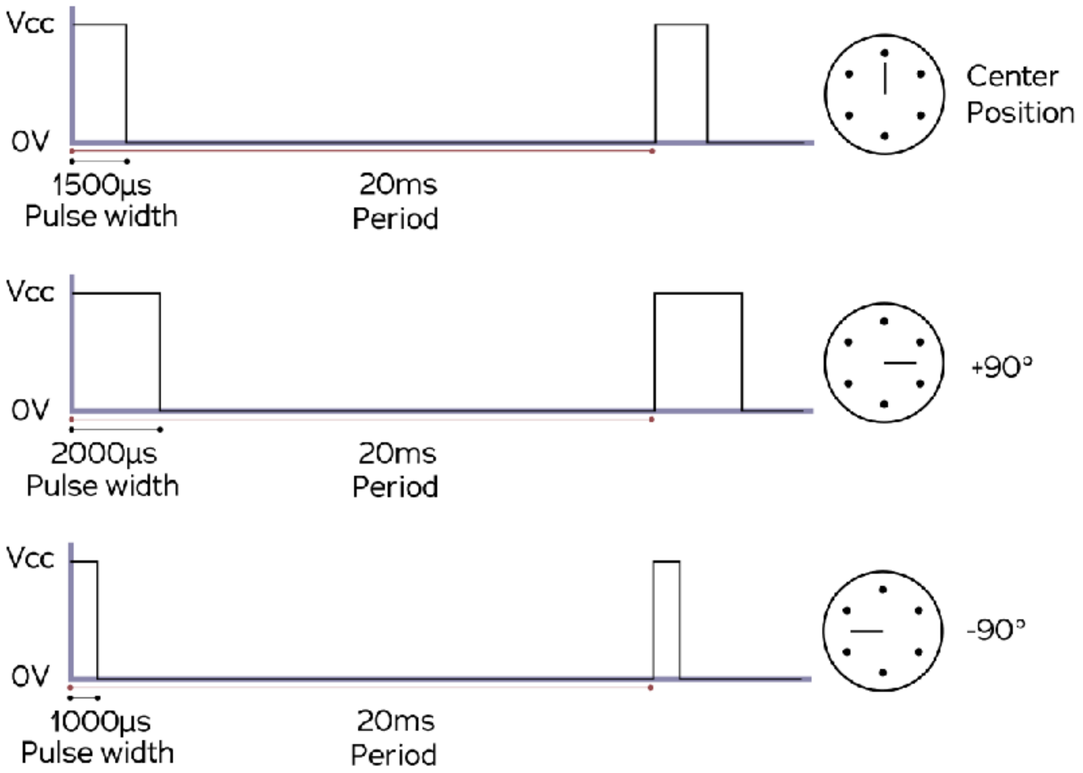
Pulse-width Modulation

Built in most microcontrollers

Often used to dim LEDs, hence "LED Control" in ESP32 API

AtomVM provides this API through Erlang ledc and Elixir LEDC modules

Pico's hardware_pwm not ported yet



Hforesti CC BY-SA 4.0

Deep sleep

Microcontroller is halted and can be woken up:

- **by a GPIO**
- **after a delay**
- by a program running on ULP

Few KB of RAM persist

la machine (v1.1) draws 18μA when sleeping

```
start() ->
  ok = gpio:init(?BUTTON_GPIO),
  ok = gpio:set_pin_mode(?BUTTON_GPIO, input),
  ok = gpio:set_pin_pull(?BUTTON_GPIO, down),
  ok = esp:deep_sleep_enable_gpio_wakeup(
    1 bsl ?BUTTON_GPIO, 1),

  ...

  WakeupCause = esp:sleep_get_wakeup_cause(),
  ...
  esp:deep_sleep(SleepMS).
```

Deep sleep

Microcontroller is halted and can be woken up:

- by a GPIO
- after a delay
- by a program running on ULP

Few KB of RAM persist

la machine (v1.1) draws 18μA when sleeping

```
start() ->
   ...
   State = case esp:reset_reason() of
     esp_rst_deepsleep -> esp:rtc_slow_get_binary();
     _ -> ...
   end,
   ...


   esp:rtc_slow_set_binary(NewState),
   ...
```

Debugging, testing and typing

Hello World

1. Install esptool.py
2. Download AtomVM release image for ESP32-C3
3. Flash AtomVM with esptool.py
4. Use rebar3 with **atomvm_rebar3_plugin**
5. **Update code and go to 4**

atomvm_examples / erlang / hello_world / rebar.config



fadushin Update copyright and license files

Code | Blame | 32 lines (32 loc) · 888 Bytes

```
1   %
2   % This file is part of AtomVM.
3   %
4   % Copyright 2023 Fred Dushin <fred@dushin.net>
5   %
6   % Licensed under the Apache License, Version 2.0 (the "License");
7   % you may not use this file except in compliance with the License.
8   % You may obtain a copy of the License at
9   %
10  %    http://www.apache.org/licenses/LICENSE-2.0
11  %
12  % Unless required by applicable law or agreed to in writing, software
13  % distributed under the License is distributed on an "AS IS" BASIS,
14  % WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15  % See the License for the specific language governing permissions and
16  % limitations under the License.
17  %
18  % SPDX-License-Identifier: Apache-2.0 OR LGPL-2.1-or-later
19  %
20  {erl_opts, [debug_info]}.
21  {deps, []}.
22  {plugins, [
23      atomvm_rebar3_plugin
24  ]}.
25  {atomvm_rebar3_plugin, [
26      {packbeam, [prune]}
27  ]}.
28  {profiles, [
29      {check, [
30          {plugins, [erlfmt]}
31      ]}
32  ]}.
```

Debugging

Output can be read using a serial port emulator (minicom)

On ESP32, it's better to use:

idf.py monitor

from esp-idf which starts program and displays output

On Pico, program can be started when serial is connected

```
I (368) cpu_start: Max chip rev:     v1.99
I (372) cpu_start: Chip rev:         v0.4
I (377) heap_init: Initializing. RAM available for dynamic allocation:
I (384) heap_init: At 3FC94C80 len 00028380 (172 KiB): RAM
I (391) heap_init: At 3FCC0000 len 0001C710 (113 KiB): Retention RAM
I (398) heap_init: At 3FCDC710 len 00002950 (10 KiB): Retention RAM
I (404) heap_init: At 50001028 len 00000FC0 (3 KiB): RTCRAM
I (412) spi_flash: detected chip: winbond
I (415) spi_flash: flash io: dio
W (419) spi_flash: Detected size(16384k) larger than the size in the binary image header(4096k). Using the size in the binary image header.
W (433) i2c: This driver is an old driver, please migrate your application code to adapt `driver/i2c_master.h`
I (443) sleep: Configure to isolate all GPIO pins in sleep state
I (450) sleep: Enable automatic switching of GPIO sleep configuration
I (457) main_task: Started on CPU0
I (457) main_task: Calling app_main()

    ########################################################

      ###    ########  #######  ##     ## ##     ## ##     ##
     ## ##      ##    ##     ## ### ### ## ##     ## ###   ###
    ##   ##     ##    ##     ## #### #### ##     ## #### ####
   ##     ##    ##    ##     ## ## ### ## ##     ## ## ### ##
   #########    ##    ##     ## ##     ##  ##   ##  ##     ##
   ##     ##    ##    ##     ## ##     ##   ## ##   ##     ##
   ##     ##    ##     #######  ##     ##    ###    ##     ##

    ########################################################

I (507) AtomVM: Starting AtomVM revision 0.6.5+git.0e3c9e24
I (517) sys: Loaded BEAM partition boot.avm at address 0x1d0000 (size=262144 bytes)
I (547) AtomVM: Found startup beam esp32init.beam
I (547) AtomVM: Starting esp32init.beam...
---
AtomVM init.
I (557) sys: Loaded BEAM partition main.avm at address 0x210000 (size=1048576 bytes)
Starting application...
I (577) gpio: GPIO[3]| InputEn: 0| OutputEn: 0| OpenDrain: 0| Pullup: 0| Pulldown: 0| Intr:0
I (587) gpio: GPIO[1]| InputEn: 0| OutputEn: 0| OpenDrain: 0| Pullup: 0| Pulldown: 0| Intr:0
I (587) gpio: GPIO[4]| InputEn: 0| OutputEn: 0| OpenDrain: 0| Pullup: 0| Pulldown: 0| Intr:0
Warning: function erlang:nif_error/1 cannot be resolved.
CRASH
======
pid: <0.1.0>

Stacktrace:
[{crypto,strong_rand_bytes,1,[{file,"/__w/atomvm_esp_adf/atomvm_esp_adf/AtomVM/libs/estdlib/src/crypto.erl"},{line,114}]},{la_machine,run,0,[{file
,"/Users/paul/Projets/AtomVM/la_machine/src/la_machine.erl"},{line,141}]}]

cp: #CP<module: 4, label: 40, offset: 34>

x[0]: error
x[1]: undef
x[2]: {2,2,125,2,[{4,374},{11,72}],error}

Stack
------

[]
{ok,<0.2.0>}
#CP<module: 4, label: 21, offset: 23>
[]
{state,139545,0,0,0,<<"">>}
<<224,201,201,63,84,133,201,63>>
#CP<module: 0, label: 35, offset: 0>


Mailbox
--------

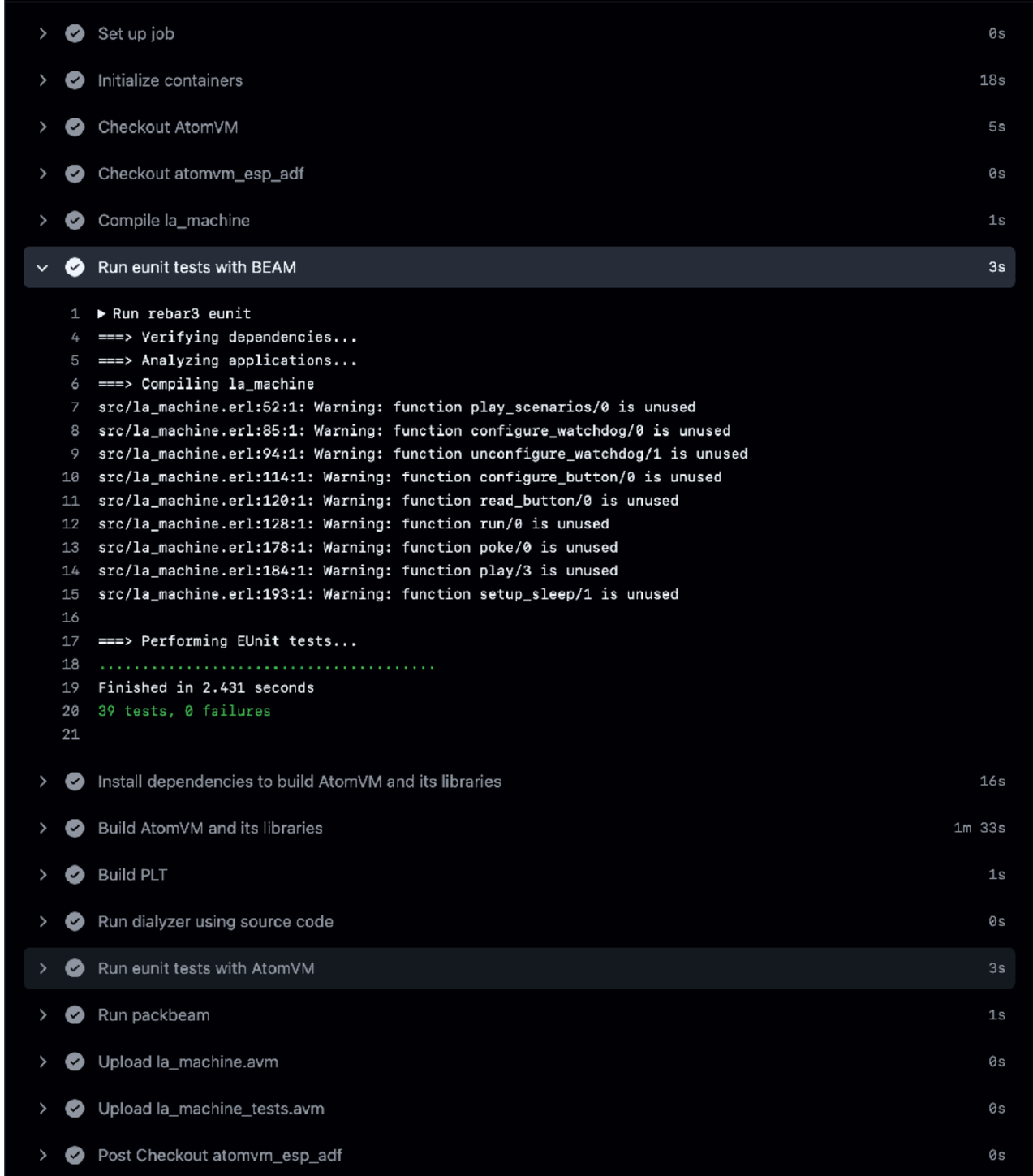

Monitors
--------
```

EUnit

Tests can be run on both BEAM and AtomVM for Unix

Suitable for functions not using ESP32 API

AtomVM Eunit support is in main branch only (planned for 0.7)



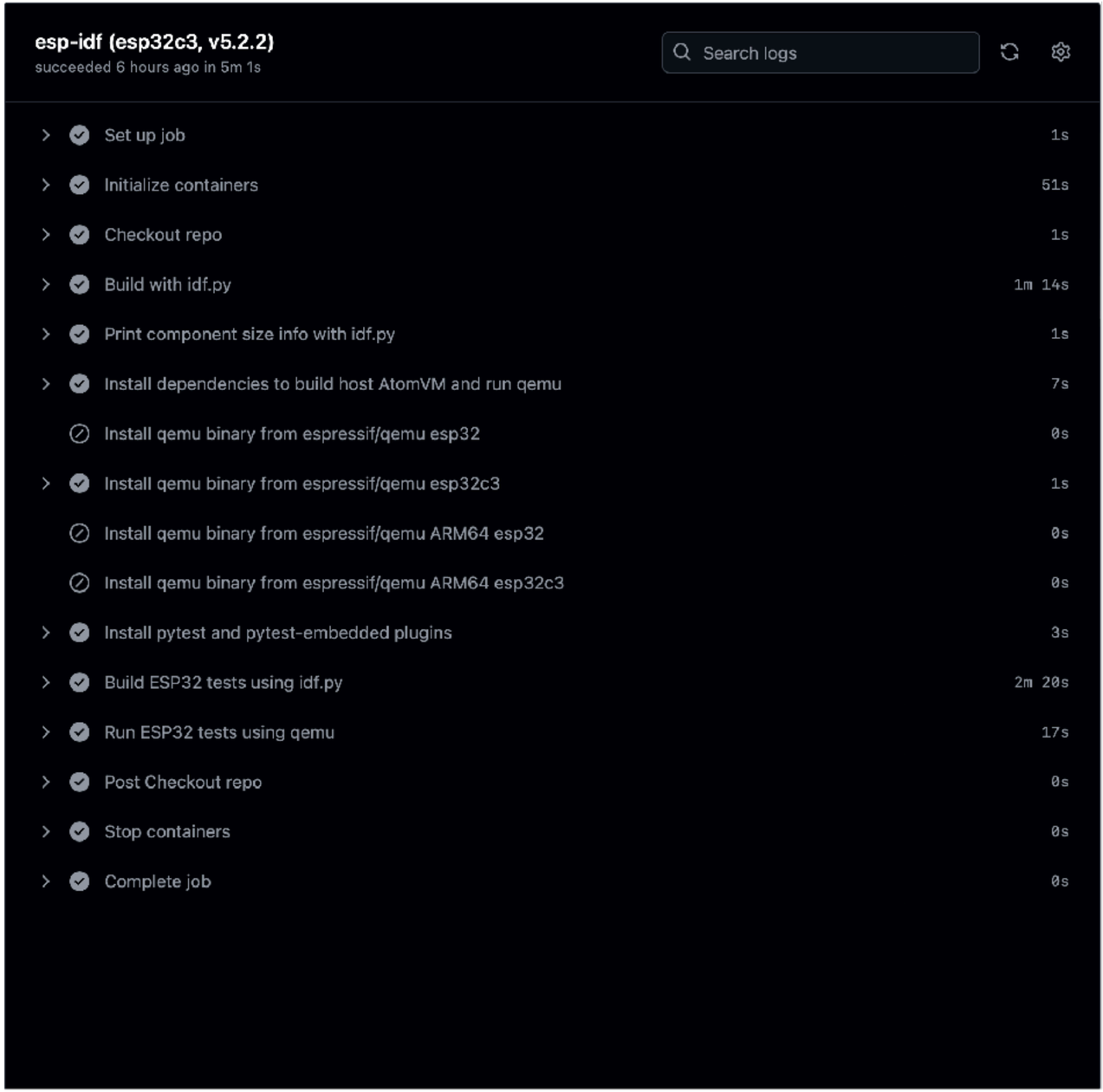
qemu

AtomVM EUnit output is compatible with Unity

Tests can be run with esp-idf qemu (ESP32 & ESP32-C3 only)

qemu is used for AtomVM to test SMP, networking and SD cards

rp2040.js is used for Pico and the team is considering Wokwi CI



dialyzer

AtomVM and Erlang/OTP standard modules API differ

dialyzer can be used to detect discrepancies before flashing

Incompatible with rebar3 dialyzer target

mix ExAtomVM plugin detects calls to unimplemented functions

```
> rebar3 compile
> dialyzer --build_plt \
      --output_plt la_machine_deps.plt \
      ${AtomVM}/estdlib/src/beams \
      ${AtomVM}/eavmlib/src/beams \
      _build/default/lib/atomvm_esp_adf/ebin/
> dialyzer --plt la_machine_deps.plt --src src/
```

NIFs & ports

Sound

la machine uses esp-adf framework for AAC decoding

This was ported as NIFs with resources.

esp-adf uses FreeRTOS xQueue for messaging.

Ports

Scheduled like processes

Run when messages are received

Send messages

Receive messages from **listeners**

NIFs

Synchronous

Easier way to port an API

Can rely on resources

Resources can be selected

Resources can monitor processes

AtomVM Listeners

IRQ

- Enqueue event (ESP32: xQueue)

FreeRTOS Task (ESP32)

- Enqueue event or directly send message

Scheduler thread

- Loop forever
  - Pick up a ready process
  - Run code until reduction counter reaches 0 or process waits
  - Check Queue until timer expires
  - Run listeners on Queue events

You shall not send a message from an interrupt handler

Interrupt handlers

Sending a message requires:

- Memory allocation (only possible from ISR on ESP32*)

- Locking the process table

Sending a message from a FreeRTOS task is possible with a specific API that tries to lock and enqueues if it cannot

Lock heap or table

Interrupt!

Lock heap or table

Deadlocks!

globalcontext_send_message : port, nif, listener

globalcontext_send_message_from_task : task

globalcontext_send_message_no_lock : ☠️

Memory

Each process has their own heap.

memory_ensure_free : allocates memory on the heap. May garbage collect. nif and ports only.

memory_ensure_free_opt
memory_ensure_free_with_roots

memory_init_heap : create a heap. Suitable for listeners.

You shall not allocate terms on another process heap

SMP

One scheduler per core

First ready scheduler polls events from IRQ

Requires synchronization

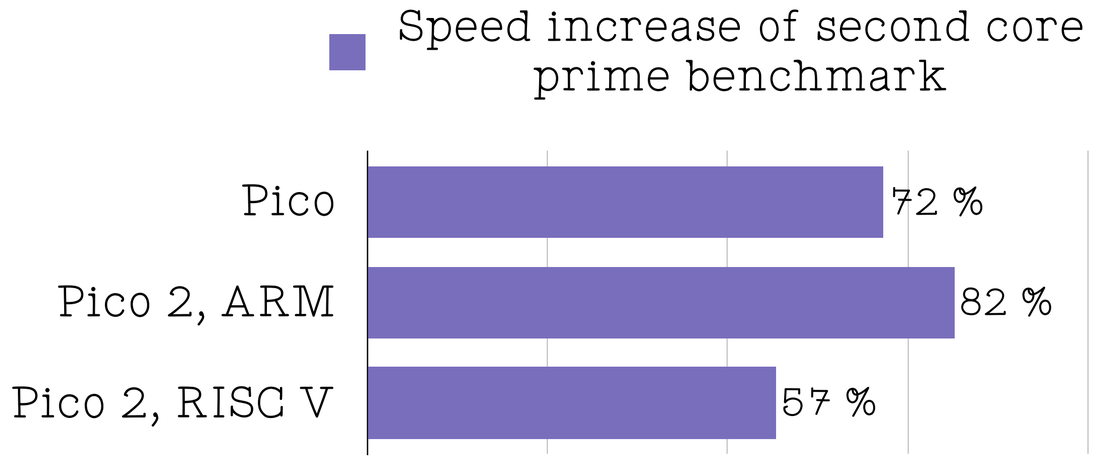
Platforms

ESP32 (ESP32, ESP32-S3, ESP32-P4*)

Pico & Pico 2

Generic Unix

Emscripten



Next steps & credits

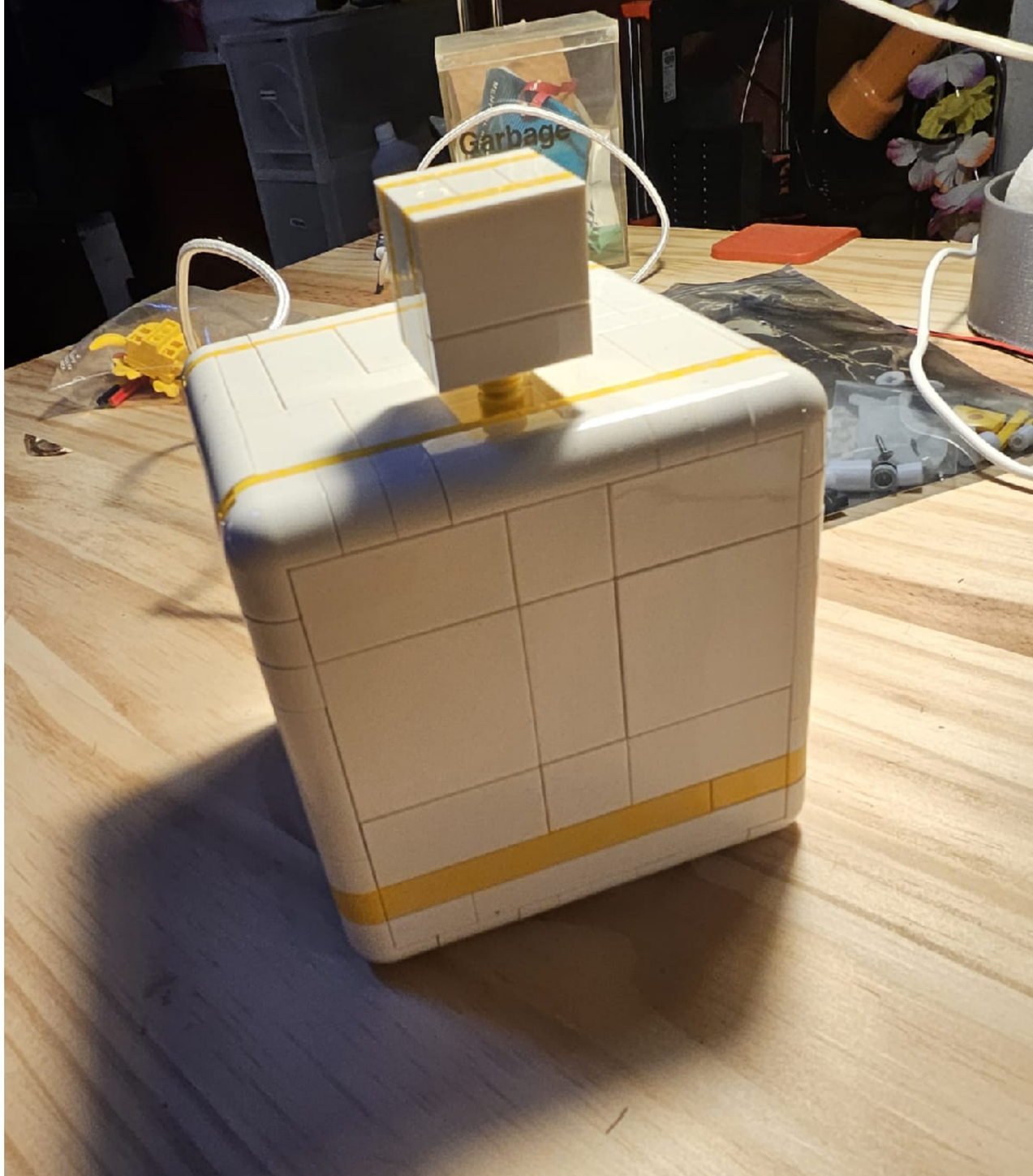
Roadmap

Hardware v1.2: few hardware fixes, also featuring an accelerometer

Software roadmap:
- accelerometer behavior & wakeup
- taking advantage of 16MB
- sequenced behaviors

la machine LEGO preorder launch

la machine Kickstarter campaign starting in 2025



Credits

Special thanks to:

AtomVM
atomvm.net

Davide Bettio
Fred Dushin†
Winford (Uncle Grumpy)
Riccardo Binetti
Peter Madsen-Mygdal
Hideki Takase
+16 other contributors

la machine
la-machine.fr

Olivier Mével
Philippe Debadier
Elium
Loïc Brun

low-carbon
Code BEAM Europe

DB
ÖBB
SNCF

Thank you