AtomVM

Where we are now & where we're going

Davide Bettio

Nagoya | piyopiyo.ex

About me (Davide Bettio)

https://github.com/bettio/ | davide@uninstall.it | https://uninstall.it/

- I’ve been tinkering with hardware and embedded systems since 2004.
- Long-time open-source dev (since ~2005, contributing to KDE Plasma and others).
- Fell in love with Elixir in 2017, while creating Astarte Platform.
- Started the AtomVM project in 2017.
- I love hiking!

Nice to meet you!

AtomVM

About Today

- This is a talk (no hands-on session today)
- We'll cover "Where are we now?" and "Where are we going?" for AtomVM
- Want a hands-on? On March 21st there'll be an in-person "AtomVM beginner workshop" (AtomVM入門～大須散策を添えて～)
    - piyopiyo.ex event page: https://piyopiyoex.connpass.com/event/373137/

AtomVM

Agenda

Today's topics:

- **Where are we now?**
  - Highlights of recent changes
- **Where are we going?**
  - v0.7
  - v0.8
    - Zephyr (and NuttX)
    - AtomVM and Nerves
  - AtomGL
  - Towards v1.0

AtomVM

AtomVM in 30 seconds

- **Write Elixir/Erlang and deploy it to microcontrollers (ESP32 / RP2040 / STM32…) or other constrained environments (such as WASM)**
- You still get the Erlang programming model: lightweight processes, supervision, message passing
- The goal: make embedded + distributed + fault-tolerant systems practical on constrained hardware

AtomVM

2025 has been **amazing for AtomVM!**

AtomVM

2025

- In 2025, our main focus was the VM core.
- A lot of improvements have landed in 2025: Erlang Distribution, JIT, Big Integers, increased compatibility, OTP-28 support etc.
- A lot of work has been done in order to support Popcorn use cases
- What we did in 2025 will be released as AtomVM v0.7

AtomVM

2025 =

Distributed, Faster, More Compatible

AtomVM

Erlang Distribution

- AtomVM nodes can take part in clusters with both AtomVM and BEAM nodes (running Erlang/OTP 24 and higher)
- What is supported?
  - serialization of all types
  - epmd protocol (client and server)
  - message passing
  - monitoring processes
  - I/O distribution (“group leader”)
  - RPC from Erlang/OTP to AtomVM
- Shell works (but additional modules need to be provided)

AtomVM

Why Erlang Distribution

- Short answer: lots of possibilities
- Ideas:
  - Serial lines / I²C / SPI can be used instead of TCP
  - Hybrid systems with Nerves running on a powerful CPU
  - An MCU running AtomVM can act as a smart I/O expansion board
  - An MCU running AtomVM can be used as an always-on low-power domain
  - Robotics?

AtomVM

JIT

- Since November AtomVM has support for both Just-in-Time (JIT) and Ahead-of-Time (AOT) compile
- Code is translated to native code for optimal execution speed
    - Drawback: it requires more flash or RAM
- Supported targets are arm6m (the Cortex-M cores), aarch64 (64-bit ARM CPUs), RISC-V (32-bit) and x86_64

AtomVM

AoT / JIT

- Ahead-of-Time can be done on your development machine or CI
    - Faster startup / module loading speed
    - Works better for devices with less RAM than JIT
    - Shipping the JIT engine is optional
    - Code will be generated for a specific target (architecture)
- Just-in-Time can be executed on the device itself
    - No need for any additional step in advance
    - Modules can be loaded at runtime

AtomVM

AtomVM in numbers

- 267 core native functions (196 NIFs and 71 BIFs) builtin in the VM
    - 167 functions are from erlang module (OTP erlang module has 350 functions)
- 51 Erlang stdlib modules available
    - erlang, gen_server, gen_event, gen_statem, gen_tcp, gen_udp, lists, queue, etc.
- 5 platforms supported
    - generic_unix (Linux, macOS, FreeBSD), rp2 (RaspberryPi Pico), ESP32, STM32 and emscripten
- contributions from 11 new developers

AtomVM

Is 47% of `:erlang` module functions enough?

- The 183 missing functions from erlang module are not strictly required
  - A number of them are undocumented, internal, etc…
  - Some other are very specific to certain scenarios unlikely on microcontrollers
  - Just a few of them is likely useful (and a welcome addition)

Some examples:

- erlang:set_cpu_topology/1 : undocumented, your code is likely not using it
- erlang:phash/2: deprecated

We'll likely happily live without them

AtomVM

I’m not convinced, show me proof

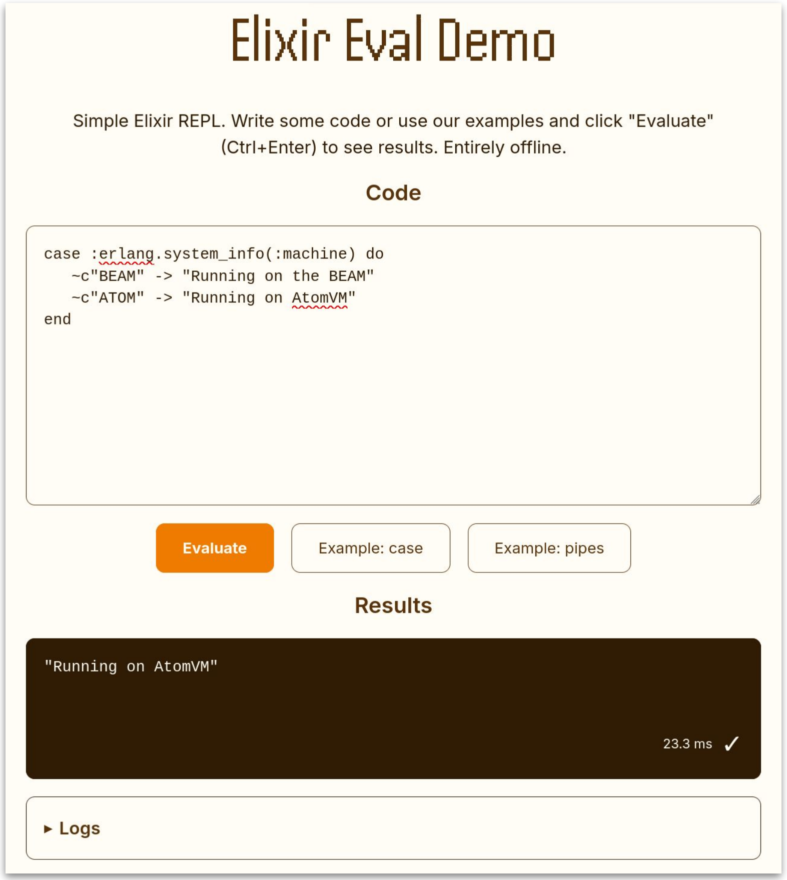
AtomVM

The Proof

Fact: Elixir compiler is quite complex.

The AtomVM main branch is complete enough to allow running the Elixir compiler on AtomVM

Self-hosting is an amazing target, and it means that more complex applications can happily live with the OTP subset we have


Elixir Eval Demo

Simple Elixir REPL. Write some code or use our examples and click "Evaluate" (Ctrl+Enter) to see results. Entirely offline.

Code

```
case :erlang.system_info(:machine) do
   ~c"BEAM" -> "Running on the BEAM"
   ~c"ATOM" -> "Running on AtomVM"
end
```

Results

```
"Running on AtomVM"
```

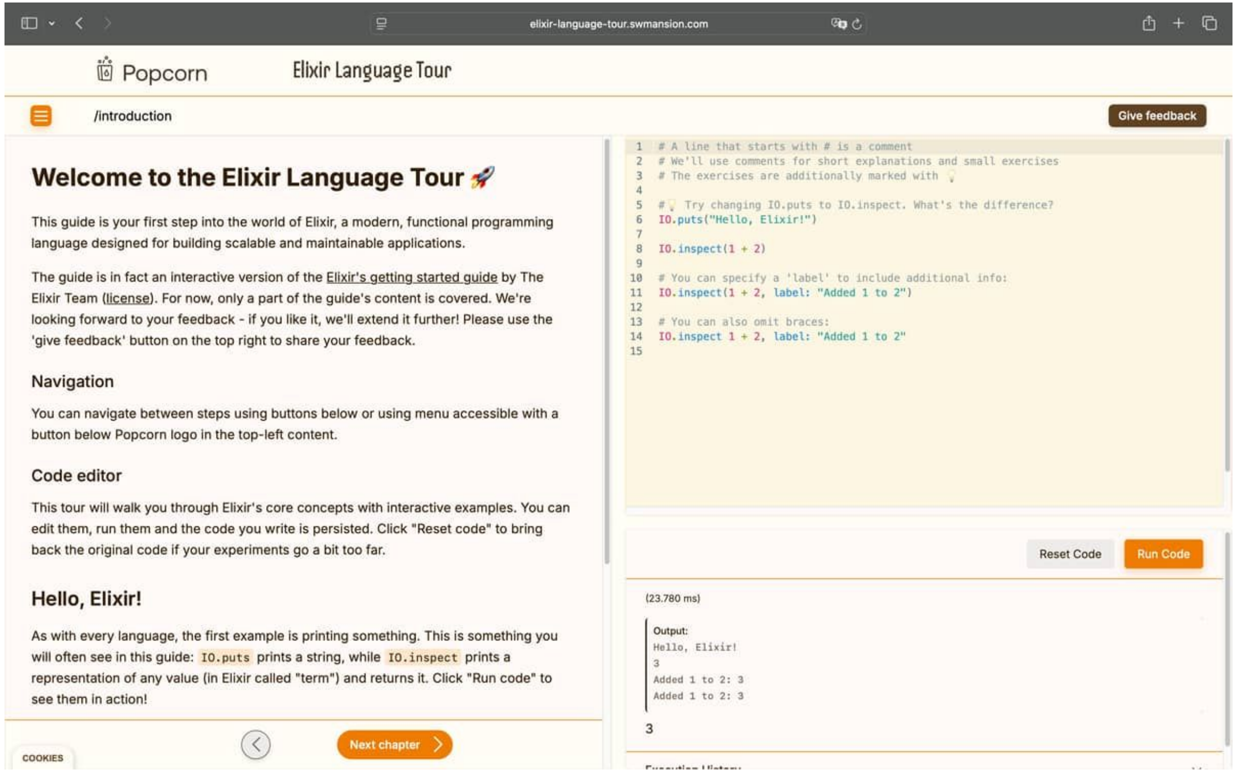
23.3 ms ✓

Popcorn

- Elixir in the browser
- It relies on AtomVM compiled to WebAssembly (WASM)
  - Elixir code is compiled in the usual way, you don't need local Elixir tooling
- Popcorn is a library and also provides the tooling (mix tasks) for building your Elixir frontend web app
- To make Popcorn possible, a lot of improvements have been merged into AtomVM, that are useful for any kind of scenario

AtomVM

This new Elixir Language Tour lets you learn about Elixir, using just your browser (no local install required!)



Welcome to the Elixir Language Tour 🚀

This guide is your first step into the world of Elixir, a modern, functional programming language designed for building scalable and maintainable applications.

The guide is in fact an interactive version of the Elixir's getting started guide by The Elixir Team (license). For now, only a part of the guide's content is covered. We're looking forward to your feedback - if you like it, we'll extend it further! Please use the 'give feedback' button on the top right to share your feedback.

Navigation

You can navigate between steps using buttons below or using menu accessible with a button below Popcorn logo in the top-left content.

Code editor

This tour will walk you through Elixir's core concepts with interactive examples. You can edit them, run them and the code you write is persisted. Click "Reset code" to bring back the original code if your experiments go a bit too far.

Hello, Elixir!

As with every language, the first example is printing something. This is something you will often see in this guide: `IO.puts` prints a string, while `IO.inspect` prints a representation of any value (in Elixir called "term") and returns it. Click "Run code" to see them in action!

```elixir
# A line that starts with # is a comment
# We'll use comments for short explanations and small exercises
# The exercises are additionally marked with 💡

#💡 Try changing IO.puts to IO.inspect. What's the difference?
IO.puts("Hello, Elixir!")

IO.inspect(1 + 2)

# You can specify a 'label' to include additional info:
IO.inspect(1 + 2, label: "Added 1 to 2")

# You can also omit braces:
IO.inspect 1 + 2, label: "Added 1 to 2"
```

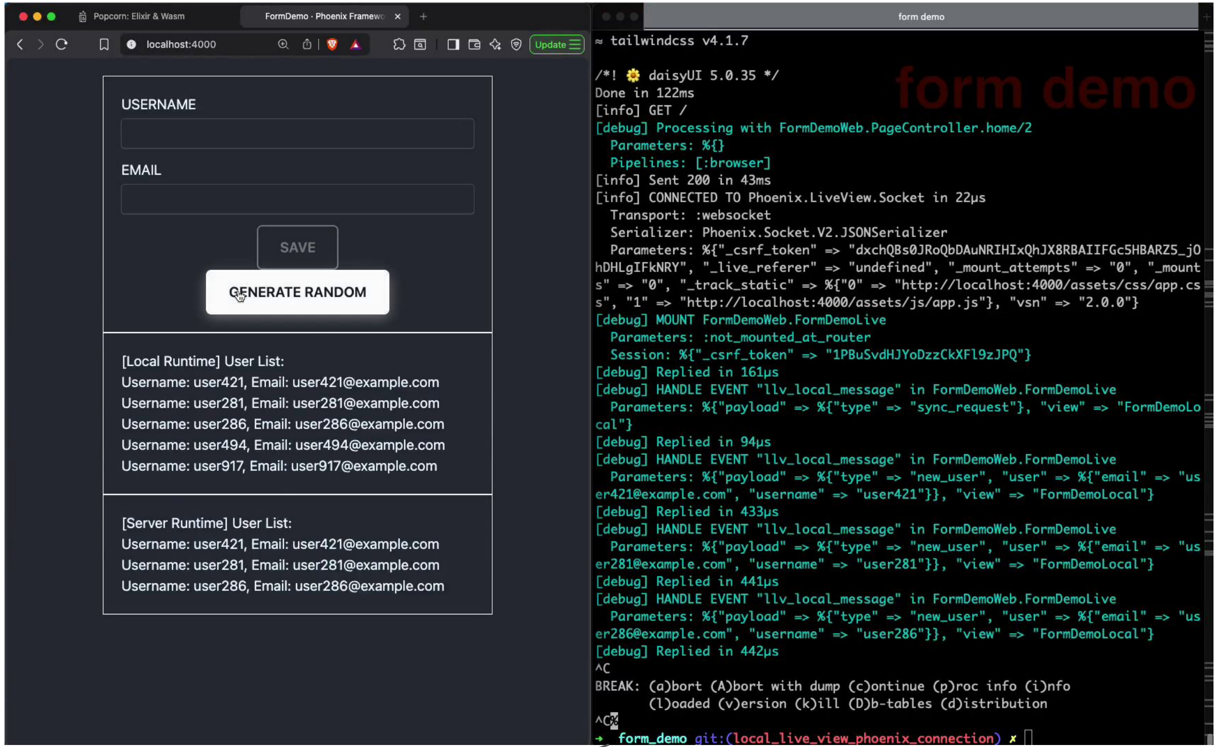
(23.780 ms)

```
Output:
Hello, Elixir!
3
Added 1 to 2: 3
Added 1 to 2: 3
```

3

AtomVM

What about running Phoenix Live View in the browser? The Software Mansion team did that.



[Local Runtime] User List:
Username: user421, Email: user421@example.com
Username: user281, Email: user281@example.com
Username: user286, Email: user286@example.com
Username: user494, Email: user494@example.com
Username: user917, Email: user917@example.com

[Server Runtime] User List:
Username: user421, Email: user421@example.com
Username: user281, Email: user281@example.com
Username: user286, Email: user286@example.com

```
≈ tailwindcss v4.1.7

/*! 🌼 daisyUI 5.0.35 */
Done in 122ms
[info] GET /
[debug] Processing with FormDemoWeb.PageController.home/2
  Parameters: %{}
  Pipelines: [:browser]
[info] Sent 200 in 43ms
[info] CONNECTED TO Phoenix.LiveView.Socket in 22µs
  Transport: :websocket
  Serializer: Phoenix.Socket.V2.JSONSerializer
  Parameters: %{"_csrf_token" => "dxchQBs0JRoQbDAuNRIHIxQhJX8RBAIIFGc5HBARZ5_jO
hDHLgIFkNRY", "_live_referer" => "undefined", "_mount_attempts" => "0", "_mount
s" => "0", "_track_static" => %{"0" => "http://localhost:4000/assets/css/app.cs
s", "1" => "http://localhost:4000/assets/js/app.js"}, "vsn" => "2.0.0"}
[debug] MOUNT FormDemoWeb.FormDemoLive
  Parameters: :not_mounted_at_router
  Session: %{"_csrf_token" => "1PBuSvdHJYoDzzCkXFl9zJPQ"}
[debug] Replied in 161µs
[debug] HANDLE EVENT "llv_local_message" in FormDemoWeb.FormDemoLive
  Parameters: %{"payload" => %{"type" => "sync_request"}, "view" => "FormDemoLo
cal"}
[debug] Replied in 94µs
[debug] HANDLE EVENT "llv_local_message" in FormDemoWeb.FormDemoLive
  Parameters: %{"payload" => %{"type" => "new_user", "user" => %{"email" => "us
er421@example.com", "username" => "user421"}}, "view" => "FormDemoLocal"}
[debug] Replied in 433µs
[debug] HANDLE EVENT "llv_local_message" in FormDemoWeb.FormDemoLive
  Parameters: %{"payload" => %{"type" => "new_user", "user" => %{"email" => "us
er281@example.com", "username" => "user281"}}, "view" => "FormDemoLocal"}
[debug] Replied in 441µs
[debug] HANDLE EVENT "llv_local_message" in FormDemoWeb.FormDemoLive
  Parameters: %{"payload" => %{"type" => "new_user", "user" => %{"email" => "us
er286@example.com", "username" => "user286"}}, "view" => "FormDemoLocal"}
[debug] Replied in 442µs
^C
BREAK: (a)bort (A)bort with dump (c)ontinue (p)roc info (i)nfo
       (l)oaded (v)ersion (k)ill (D)b-tables (d)istribution
^C%
➜  form_demo git:(local_live_view_phoenix_connection) ✗
```

The web server has been stopped, but everything still works.

AtomVM

What's coming in 2026?

AtomVM

2026

- In 2026, our focus will be hardware support.
- We’ll do our best to ship v0.7 and v0.8 releases during 2026, in order to allow anyone to benefit from expanded hardware support
- Let’s see this in more detail

AtomVM

v0.7

- A stable release including all the features I just mentioned
- We’re pretty close
- A few changes are still missing:
  - Improved error handling (function arguments will be visible on the stack trace, etc…)
  - New module loader (it should allow us to further reduce flash usage)
  - Miscellaneous smaller PRs and fixes
  - Remove code for supporting OTP < 26 and older Elixir releases
  - Bitstrings (e.g. `<<42::size(7)>>`, not yet sure if they will be in v0.7 or v0.8)

AtomVM

v0.8

- Will be about platforms, hardware support, and tooling
- I don’t expect major changes to the core VM

AtomVM

What about hardware and platforms?

Zephyr

- The problem: we have a number of ports that target very specific platforms, such as generic_unix, esp32, and rp2040
- There are plenty of other silicon platforms (such as Nordic Semiconductor, NXP, etc...) that are not yet supported
- Writing platform support for each of them can be a lot of work
- Zephyr gives us a Hardware Abstraction Layer, that lets us write platform support once

AtomVM

Zephyr: What have we done so far?

- In 2023, there was some initial work
- There were some initial issues:
    - How to compile AtomVM core inside a Zephyr project?
    - How do we integrate the two different build systems?
    - Time and priorities.

AtomVM

Zephyr: What are we doing next?

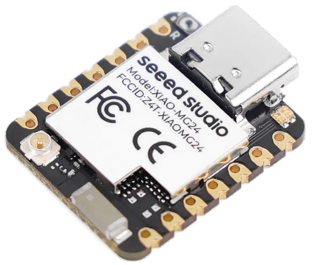
- This year, we're going to work on this platform again
- It will be at the top of my to-do list
- If anyone has experience with Zephyr, I'd really appreciate contributions of any kind (also PR reviews are super valuable)

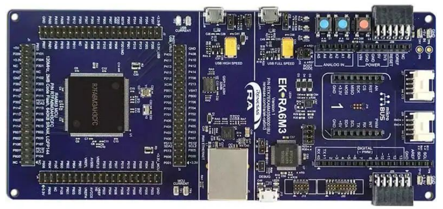
AtomVM

Zephyr: What to expect

Support for SoCs from: Microchip, Nordic Semiconductor, NXP, Silicon Labs, STMicroelectronics, Renesas, etc...

What's supported will depend on the specific Zephyr port (and the amount of available RAM)





AtomVM

Apache NuttX?

- Mostly an alternative to Zephyr
- It will be transparent to developers on top of AtomVM
- It gives us more options.

AtomVM

AtomVM and Nerves

- We aim to reach some level of compatibility with Nerves
- Ideally, we’d like to allow the same I2C/SPI/GPIO device driver (e.g., a temperature sensor) running on both Nerves and AtomVM with no changes
- The extent of compatibility and the outcome is not clear yet
    - We’ll do our best to provide the best compatibility

Tooling and DevX

- Tooling right now (such as exatomvm) allows to automate boring tasks, and provide some handy tasks such as **`mix atomvm.esp32.install`**
- Good tooling is not enough, we want excellent tooling and developer experience (DevX)
- More automation
- More pre-built images
    - Images for both stable releases and main branch
    - Images with native extensions such as AtomGL

AtomVM

AtomGL

AtomVM is not just the core VM, but the project includes a broader ecosystem of components such as AtomGL

- AtomGL provides an abstraction for displays and display controllers
- A growing number of supported displays is available/supported:
    - SPI color LCD: ILI9341 / ILI9342C, ILI9486 / ILI9488, ST7789 / ST7796
    - SPI monochrome LCD: Sharp Memory LCD
    - SPI color E-Paper: Waveshare 5.65-inch ACeP 7-Color E-Paper, E Ink® Spectra™ 6 (reTerm E1002)
    - I2C monochrome OLED: SSD1306 / SH1106
    - More can be added

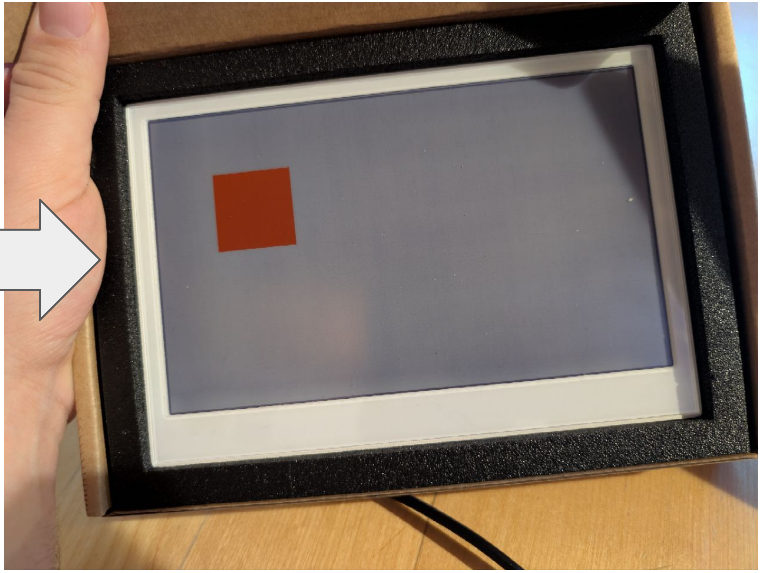
AtomVM

How does AtomGL work?

Declarative approach:
You tell AtomGL what to show (not how to draw it).

```
def handle_info(:show_hello, state) do
  items = [
    {:text, 10, 20, :default16px, 0x000000, @bg_color,
      "Hello"},
    {:rect, 100, 100, 128, 128, 0xFF0000},
    {:rect, 0, 0, 800, 480, @bg_color}
  ]

  {:noreply, state, [{:push, items}]}
end
```



AtomVM

How AtomGL works internally

- The item list is rendered starting from tail (lowest item) to head (topmost item)
- Adding an item on top basically means: `[new_top_item | items]`
- No need to use memory for a framebuffer, everything is rendered just-in-time, scan-line by scan-line
- This approach allows us to drive displays that would require more memory than what we have
- Also color conversion, dithering and everything else is done just-in-time

AtomVM

Toward v1.0

We are approaching v1.0, that means:

- API will be frozen and stable
- Core OTP functionality will be available
- Must-have features will be available on main platforms

In short: v0.x doesn’t mean you can’t use it in production.

We are already making bug-fix releases in our maintenance branch.

AtomVM

Sponsors

All the amazing work we did for Popcorn has been possible thanks to our sponsors, who have enabled people to work on the AtomVM ecosystem part-time and full-time.

Thank you!

Dashbit

software mansion™

AtomVM

New Year, New Sponsors

We are looking for new sponsors to speed up development.

The project has grown more complex, every day, new PRs and issues come in and developers are starting to use AtomVM in production.

Working on AtomVM in my free time is no longer sustainable, so starting from this year I will be working on AtomVM full-time.

New sponsors are welcome. Even small sponsorships help.

AtomVM

Join Us

https://atomvm.org/

Discord: https://discord.gg/QA7fNjm9Nw

Telegram: https://t.me/atomvm

Documentation: https://doc.atomvm.org/

AtomVM

Thank you!

Sponsor AtomVM

Let’s get in touch