Prácticas sostenibles en el desarrollo de software

M. en C. Miguel Santiago Villafuerte Ramírez

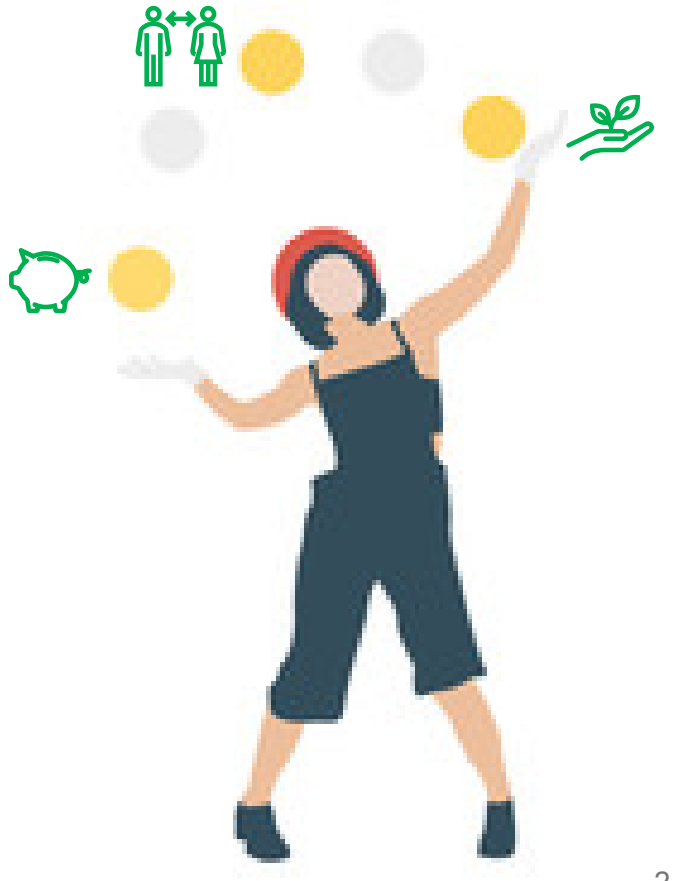
Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

¿De qué se encarga el desarrollo sostenible?

Se encarga de satisfacer las necesidades actuales de la sociedad y el ambiente sin comprometer las capacidades de las generaciones futuras.

Busca equilibrar el crecimiento económico, la inclusión social y la protección ambiental.

De acuerdo a un ejemplo de las Naciones Unidas (García, 2023), el desarrollo sostenible es parecido a un acto de malabarismo. Hay que mantener en el aire tres bolas al mismo tiempo: crecimiento económico, inclusión social y protección al medio ambiente.



Dicho en otras palabras:

- ✓ Todo humano debe tener un trabajo digno, buena atención de la salud y una educación de calidad.
- ✓ Se deben usar los recursos naturales sin generar contaminantes.
- ✓ No debe existir la inferioridad ni la discriminación.

Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

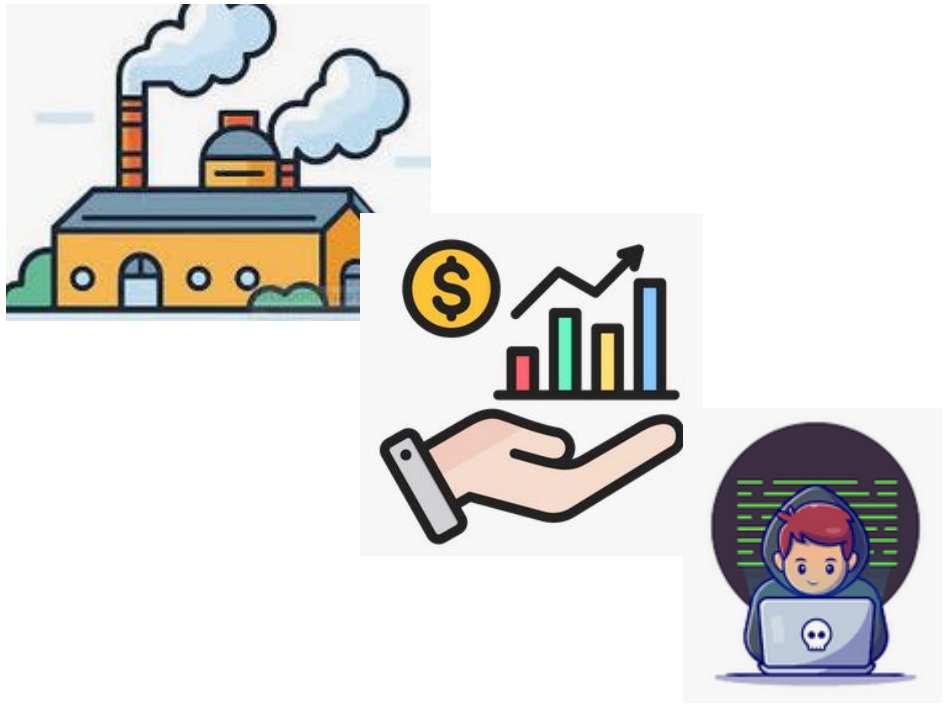
Software verde (*Green Software*)

El software verde es una práctica informática sostenible desde el punto de vista ambiental que busca minimizar la energía necesaria para procesar instrucciones de cómputo y para ayudar a las organizaciones a reducir el consumo energético total de sus operaciones.

La programación ecológica busca limitar el impacto ambiental de la tecnología, reduciendo la huella de carbono en operaciones de cómputo intenso, tales como los centros de datos que están emergiendo velozmente en la actualidad (IBM Cloud Education, 2023).

Los beneficios de la programación verde son como sigue:

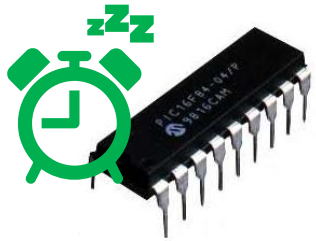
- ✓ Reducción de consumo de energía y generación de CO_2 con avance acelerado hacia metas de cero emisiones.
- ✓ Mayores beneficios económicos que reflejan la salud financiera de la empresa.
- ✓ Mejor disciplina de los desarrolladores de código al estar siempre conscientes de que su código va de la mano del consumo de energía.



Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

Prácticas sostenibles en el desarrollo de software

Buenas prácticas para software a seguir (Bernardino, 2024):



1. Eficiencia energética. Es necesario minimizar el uso del CPU, optimizar el código y reducir la cantidad de información enviada a través de la redpara reducir la generación de CO_2.

2. Uso inteligente de la nube. Si bien la nube o los centros de datos consumen grandes cantidades de energía, también ofrecen opciones para ejecutar software sostenible. Las técnicas de escalado de recursos gobernadas por las necesidades dinámicas de cómputo evitan el desperdicio de recursos energéticos y de hardware.

3. Ciclo de vida del producto. Un software flexible y portable promueve el reuso de hardware y reduce el tiempo de desarrollo del mismo (usando lenguajes multiplataforma).

4. Reducción de desechos electrónicos. Evitar la obsolescencia programada del software ayuda a que el hardware permanezca útil por más tiempo. La compatibilidad con versiones anteriores de hardware reduce el desperdicio electrónico.

Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

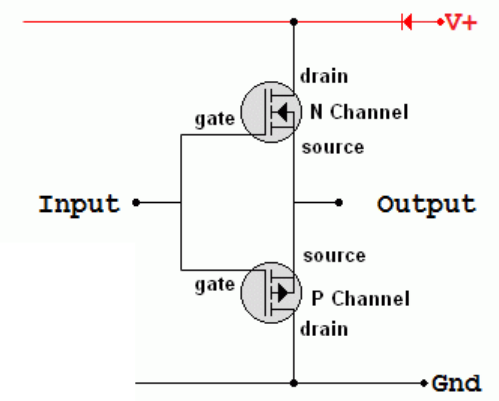
¿Por qué un CPU genera calor?

Los CPUs tienden a generar calor (conversión de energía eléctrica en calor) debido a los siguientes patrones (Wikipedia, 2025):

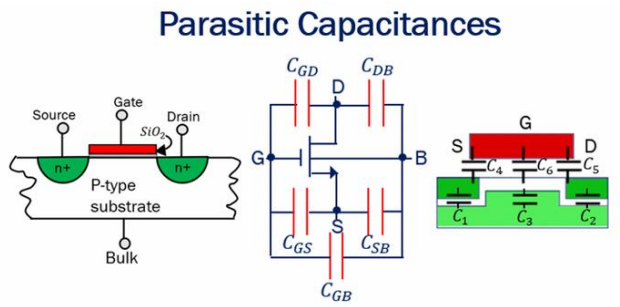
- Para un núcleo dado, el uso de energía va de la mano de su frecuencia de reloj. La frecuencia de reloj de un CPU es a veces dinámica y el software decide elevarla cuando las necesidades de cómputo son continuas y mayores.

- Algunos procesadores requieren de mayor cantidad de compuertas lógicas (and, or, not) y circuitos secuenciales (flip-flops) para mejorar su capacidad de cómputo en tiempo y anchura de datos. Al tener más circuitos electrónicos conectados se incrementa el consumo de energía.

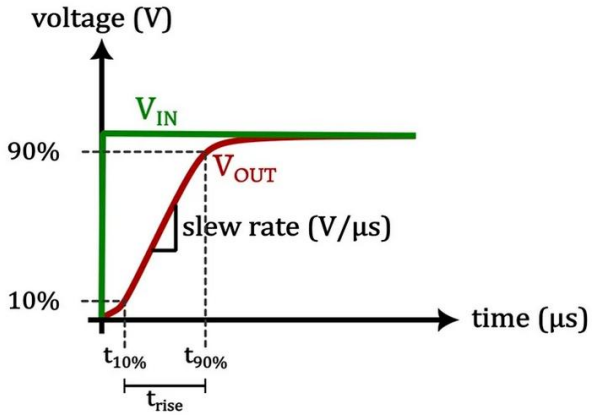
- Los CPUs están construidos con MOSFETs. Estos transistores cambian de estados (0 a 1 y viceversa) muy velozmente. Existe una capacitancia inherente en los MOSFETs que suaviza los cambios de estado y es por ello que se les debe aplicar mayor voltaje para que no exista el efecto *slew rate* en la salida del transistor. Este mayor voltaje conlleva un mayor consumo de energía.



(reviseomatic.org, 2026)



(rahsoft.com, 2021)



(allaboutcircuits.com, 2026)

Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

¿Cómo se puede medir el CO_2 que un programa generaría?

Existen diversas librerías de software que nos ayudan a medir el consumo de potencia y energía de nuestro software en ejecución. Funcionan como cronómetros a los que uno les indica cuándo iniciar la medición y cuándo detenerse.

La librería "energyusage" desarrollada por (Lottick, 2019) presenta una manera simple de medir este consumo de energía en programas de Python. En su paper publicado, los autores detallan cómo es que calculan el consumo de energía y cómo lo traducen a un consumo equivalente en diversas fuentes de energías o en otros dispositivos que consumen energía (televisiones, residencias, etc.).

```python
import energyusage

# user function to be evaluated
def recursive_fib(n):
    if (n <= 2): return 1
    else: return recursive_fib(n-1) + recursive_fib(n-2)

energyusage.evaluate(recursive_fib, 40, pdf=True)
# returns 102,334,155
```

Energy Usage Report

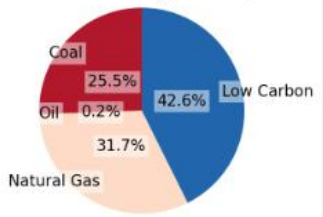
Energy usage and CO_2 emissions for the function exp with input 10.

Energy Usage Readings

Average baseline wattage:	2.35 watts
Average total wattage:	15.53 watts
Average process wattage:	13.18 watts
Process duration:	0:16:40

Energy Mix Data

Location: Pennsylvania, USA



Total kilowatt hours used:	0.00367 kWh
Effective emissions:	1.78e-03 kg CO_2

Assumed Carbon Equivalencies

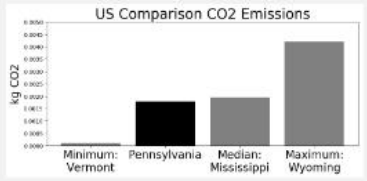
Coal:	996 kg CO_2/MWh
Oil:	817 kg CO_2/MWh
Natural gas:	744 kg CO_2/MWh
Low carbon:	0 kg CO_2/MWh

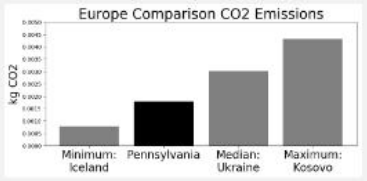
CO_2 Emissions Equivalents

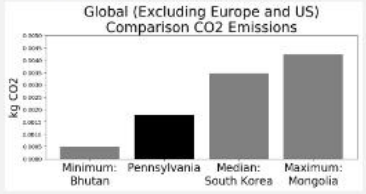
Miles driven:	7.26 e-10 mi
Min. of 32-in. LCD TV:	1.10 min.
% of CO_2 per US house/day:	5.84 e-10%

Emission Comparisons

CO_2 emissions for the function if the computation had been performed elsewhere.







(Lottick, 2019)

Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

¿Cómo optimizar el uso de CPU de un programa en C o C++?

C y C++ son lenguajes que convierten las instrucciones en lenguaje humano a instrucciones en ensamblador o códigos de operación de un CPU de manera muy eficiente (Pereira et al., 2021).

Existen maneras de convertir el código que uno escribe en instrucciones de CPU más eficientes mediante el uso de banderas de compilación (gcc Optimize Functions, 2026). La bandera -O3 fuerza al compilador de C/C++ a generar código tan grande como él necesite pero que obligadamente corra en el CPU lo más rápido posible.

Algunas de las optimizaciones para velocidad que aplica son:

- ✓ Ejecutar instrucciones de múltiples cálculos en una sola instrucción.
- ✓ inline de funciones.
- ✓ Desenrollamiento de ciclos en ejecución lineal y optimización de acceso de RAM en caché.
- ✓ Elimina ramas de código que nunca correrán y evalúa si una función debe reducirse.

Table 4: Normalized global results for Energy, Time, and Memory

Total					
	Energy (J)		Time (ms)		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

¿Es eficiente Python respecto al uso de CPU?

Python es un lenguaje de propósito general que permite programar de manera rápida y fácil programas complejos y de mucha utilidad. Sin embargo, al ser un lenguaje interpretado, presenta las siguientes desventajas:

- Tiene que correr en una máquina virtual, lo cual ralentiza su ejecución.
- Las variables que uno define en el código son sujetas a validación de tipos en tiempo de ejecución (no sabe si una variable es un entero, un flotante, una instancia de una clase, etc. y debe checarlo al vuelo).
- La memoria es administrada por Python y no por el usuario, lo cual genera tiempos de ejecución no controlables en los que se corre el *garbage collector*, interrumpiendo la ejecución de nuestro programa principal.

Python se usa mucho en IA debido a que las librerías de cálculos matemáticos (optimizaciones, redes neuronales, cálculo de matrices, etc.) están implementadas en C y Python invoca esas librerías directamente. Esto reduce los cuellos de botella relacionados a librerías puramente implementadas en Python.

Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

Estadísticas de energía, tiempo, y memoria de Python

Table 4: Normalized global results for Energy, Time, and Memory

Total					
	Energy (J)		Time (ms)		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

¿Conviene usar Python para nuestros programas?

Sí, para tareas simples donde el tiempo de ejecución es despreciable pero la complejidad de la tarea fue resuelta más rápidamente con Python que con otro lenguaje más rápido. También para tareas donde se hace uso paralelo de librerías en C o Fortran donde el cómputo intenso no presentará cuellos de botella.

No, porque la mayoría de aplicaciones comerciales deben correr lo más rápido posible si es que queremos ser más competitivos con respecto a otros desarrolladores.

Los requisitos computacionales de tu aplicación dictan el lenguaje de programación a usar... además de tu experiencia personal con el lenguaje.

Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

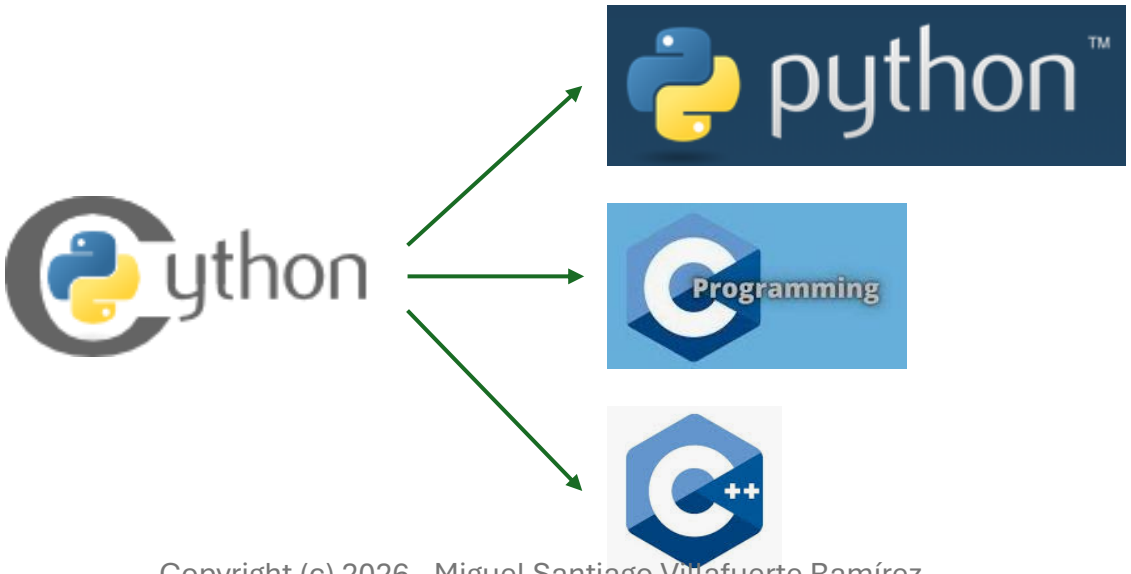
¿Se puede convertir código en Python a código en C?

Se puede usar Cython (otro lenguaje de programación muy similar a Python) para convertir el código a C o C++. Estos lenguajes correrán más eficientemente el código (cython.org, 2026). Este código puede interactuar con C, C++ o incluso Python de manera directa.

El código mantiene su alto nivel, orientación a objetos y programación dinámica.

Su ventaja principal es el soporte de declaraciones estáticas de tipos como parte del lenguaje. Ya no presenta la validación de tipos al vuelo que Python normalmente hace.

Mantiene la alta productividad de programación que hace a Python muy popular.



Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

Modos de bajo consumo en microcontroladores

Algunos microcontroladores poseen la capacidad de variar su consumo energético dependiendo de lo que el software (conocido como *firmware*) les ordene. Los desarrolladores conocen bajo qué condiciones de uso deben variar el consumo de los dispositivos y habilitan los diversos modos de consumo de energía con los que cuentan los microcontroladores.

La cantidad de modos disponibles depende de la marca, modelo y complejidad de cada microcontrolador.

En sistemas embebidos, un microcontrolador debe hacer uso eficiente de su energía disponible ya que a veces son equipos que están funcionando durante muchas horas al día, y a veces se alimentan desde una batería o con energía solar.

En la tabla publicada por (Tollefson et al., 2026) se muestran nombres genéricos de los modos de consumo más comunes en diversos microcontroladores.

Cuando uno diseña un producto embebido, es de vital importancia habilitar los modos de consumo bajo para evitar la presencia de *vampiros electrónicos (consumo fantasma)*.

Mode	Core	FLASH	Peripherals	Voltage Regulator	RAM	Savings over RUN Mode	Of Note
Run	ON	ON	ON	ON	ON	0%	Full current consumption.
Doze	Slower Clock Rate	Slower Clock Rate	ON	ON	ON	25-65%	CPU is slowed down to as much as 1/128 of its Run mode speed.
Idle	OFF	OFF	ON	ON	ON	70%	Use in moments when code execution is not needed
Sleep	OFF	OFF	OFF - Except for peripherals with their own clock	ON	ON	70% - 85%	RAM and registers maintained. Latencies of up to tens of microseconds are expected.
Low Voltage Sleep	OFF	OFF	OFF - Except for peripherals with their own clock	Low Voltage Enabled	ON	90%	For longer-term sleep than Sleep, when longer latencies are acceptable.
Deep Sleep	OFF	OFF	OFF	OFF	OFF	97%+	Most of the MCU is OFF except for maybe a timer, interrupt handling, and a Real Time Clock and Calendar.

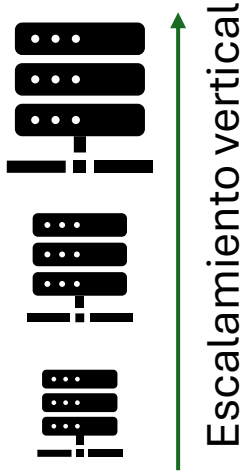
Servidores escalables bajo demanda

El escalamiento de servidores se refiere a expandir la capacidad de un sistema para manejar cargas de trabajo mayores en tiempo real. Asegura que las aplicaciones que corre el servidor corran correctamente incluso cuando hay una alta demanda de solicitudes de clientes (Shaw, 2026).

Las soluciones de escalamiento ayudan a los negocios a manejar cargas de trabajo integradas mediante la optimización de los recursos del servidor. Cuando la demanda de solicitudes incrementa, las compañías pueden elegir entre escalamiento horizontal o vertical. Esto le permite a las empresas incrementar sus ganancias sin elevar gastos administrativos y sin tiempos muertos para reiniciar servicios.

Escalamiento vertical. Se refiere a incrementar los recursos y capacidades de cómputo de un solo servidor. Puede incrementarse la RAM, almacenamiento, CPUs, capacidades de red. Se necesita desconectar el servidor para actualizarlo.

Escalamiento horizontal. Expande la capacidad del sistema total mediante la adición de múltiples servidores para distribuir las cargas de trabajo. El software debe estar listo para saber cómo distribuir esas cargas al vuelo.



Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

Inteligencia artificial y su consumo

La inteligencia artificial está emergiendo muy rápidamente desde los últimos años y esto ha sido gracias a que la capacidad de cómputo ha mejorado drásticamente, además de que los campos de investigación de este rubro han recibido mucha atención.

Los modelos extensos de lenguaje (*Large Language Models (LLM)*, como Google Gemini, ChatGPT de OpenAI, Copilot de Microsoft) fueron entrenados con equipos de capacidades inmensas de cómputo.

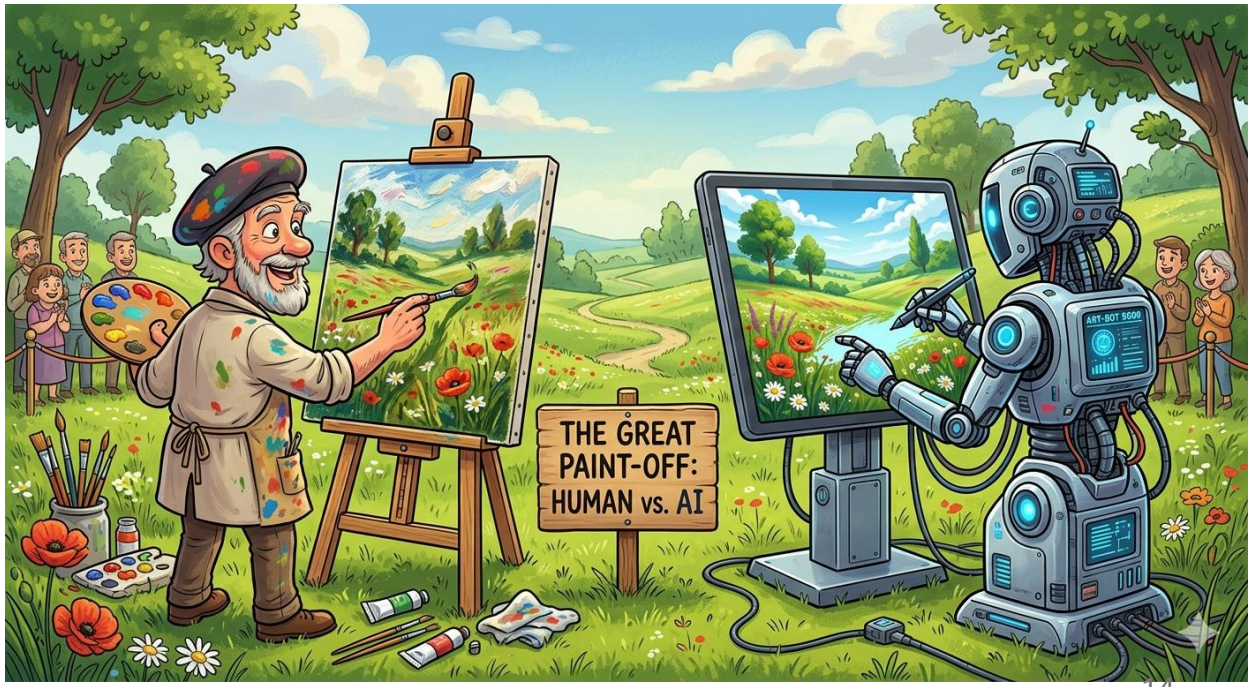
Entrenar un LLM puede tomar semanas o incluso meses de cómputo intenso e ininterrumpido. La versión 3 de ChatGPT requirió 1,287MWh de energía y produjo un equivalente a 552 toneladas métricas de CO_2. La versión 4 generó emisiones tan altas como 21,660 toneladas métricas de CO_2 (Tomlinson et al., 2024).

De manera contrastante, una vez que los modelos han sido entrenados, el uso de la IA representa un consumo mucho menor de energía:

- Una LLM emite entre 130 y 1500 veces menos CO_2e por página de texto generada que un humano.

- Un generador de imágenes como Dall-E emite entre 310 y 2900 veces menos CO2e por imagen que un humano.

Generar una imagen con IA equivale a cargar un celular. Crear texto con LLM 1,000 veces consume 16% de lo que consumiría generar una imagen (Heikkilä, 2023).



¿Cómo se puede medir el CO_2 que un GPU consume?

Los GPUs Nvidia tienen una aplicación de consola (nvidia-smi) que reporta el uso de los GPUs instalados en un equipo.

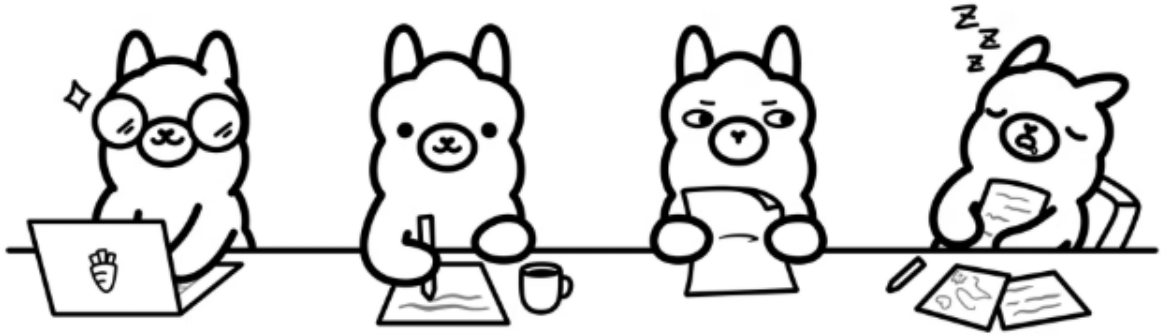
```
Mon Apr 20 17:23:41 2026
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 555.42.06              Driver Version: 555.42.06      CUDA Version: 12.5     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA RTX A6000               Off |   00000000:16:00.0 Off |                  Off |
| 30%   54C    P5             30W /  300W |       2MiB /  49140MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   1  NVIDIA RTX A6000               Off |   00000000:34:00.0 Off |                  Off |
| 58%   81C    P0            142W /  300W |   47950MiB /  49140MiB |    100%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   2  NVIDIA RTX A6000               Off |   00000000:52:00.0 Off |                  Off |
| 35%   64C    P5             24W /  300W |       2MiB /  49140MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   3  NVIDIA RTX A6000               Off |   00000000:70:00.0 Off |                  Off |
| 30%   55C    P5             24W /  300W |       2MiB /  49140MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    1   N/A  N/A   3428784      C   python                                     47916MiB |
+-----------------------------------------------------------------------------------------+
```

¿Podemos tener una LLM sin conexión a un servidor?

Los modelos extensos de lenguaje (LLM) de última generación normalmente corren en servidores remotos (ChatGPT, Copilot, Gemini, Claude) y al ser equipos de acceso masivo su consumo de energía es extremo.

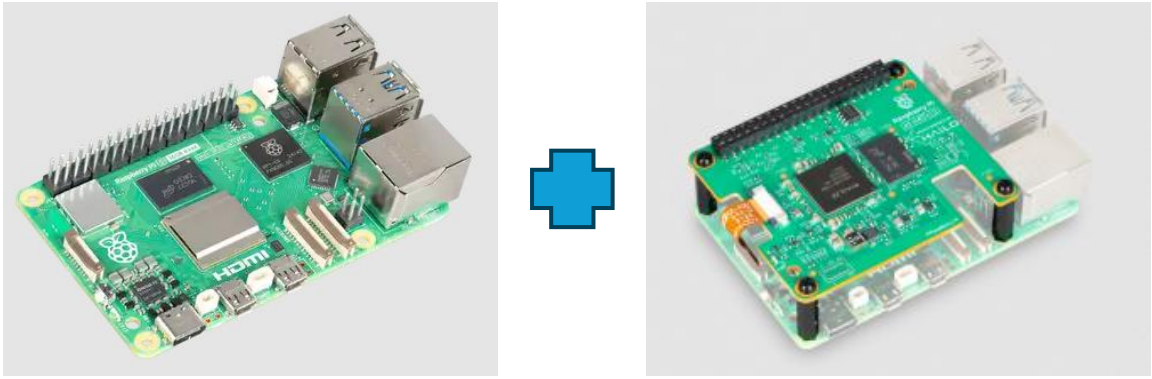
Existen soluciones alternativas de LLMs como Ollama, el cual permite correr una LLM localmente. Esto ayuda a reducir la dependencia de servicios web y a resguardar de mejor manera la privacidad de nuestras consultas hechas con LLMs.



También existen versiones portátiles sin conexión de LLMs que podemos usar: Raspberry Pi 5 con AI Hat+2 (Hailo). La Raspberry Pi es una computadora capaz de correr Linux con dimensiones similares a una tarjeta de crédito (raspberrypi.com, 2026).

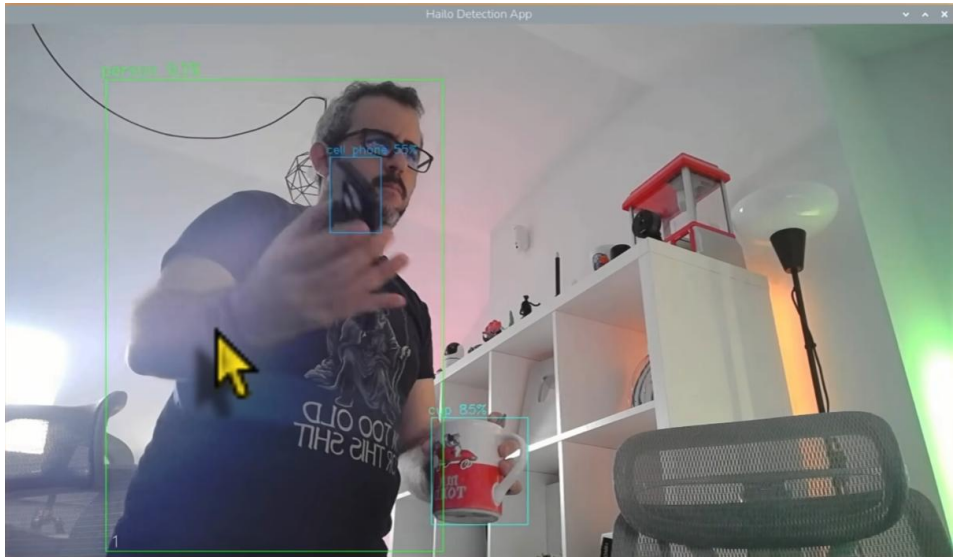
A la Raspberry Pi 5 se le puede montar un módulo neuronal que agrega 8GB de memoria dedicada a cálculos de IA con los que uno puede implementar aplicaciones de visión y LLMs de manera portátil, entre muchas otras (PyTorch, Tensorflow).

Puede correr modelos como Qwen (incluso coder) y Deepseek cuyos tiempos de respuesta con contextos complejos rondan los 30s a 60s (youtube.com, Raspberry Pi AI Hat, 2026). Es más lento el rendimiento que el de un servidor dedicado en la nube, sin embargo nos da la opción de poder hacer cómputo sin conexión.

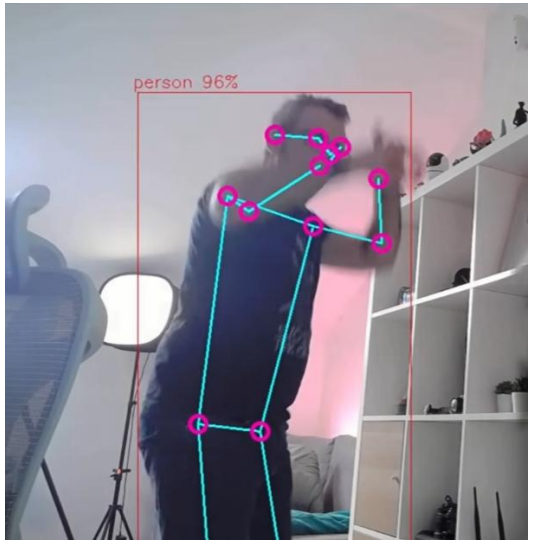


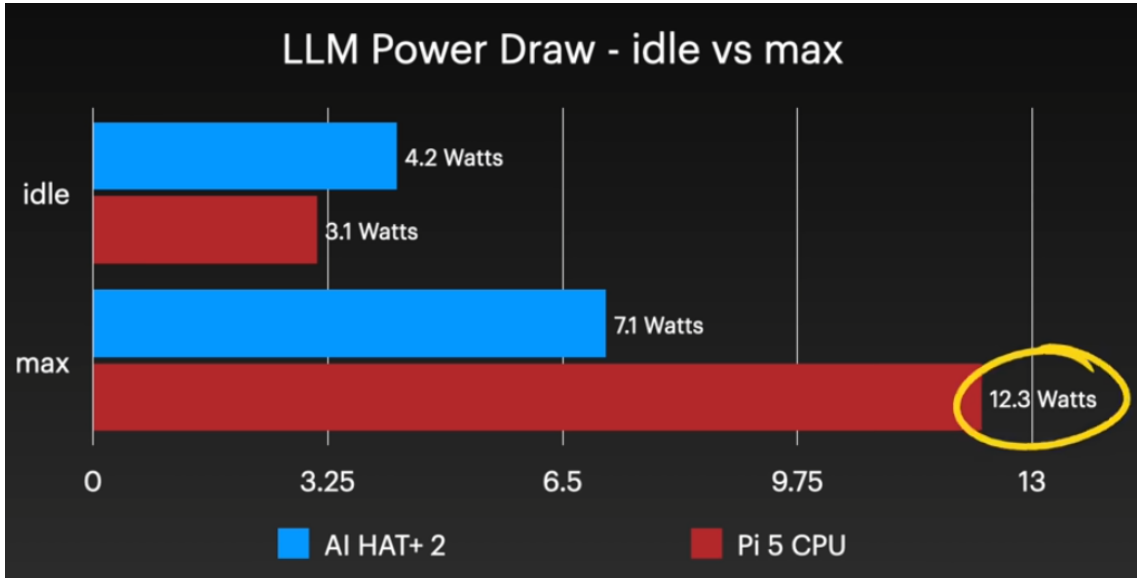
Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

Raspberry Pi & AI Hat+2



(youtube.com, Raspberry Pi AI Hat, 2026)





(Geerling, Raspberry Pi AI Hat, 2026)

Conclusiones

- ✓ El software verde es una práctica sostenible desde el punto de vista ambiental que busca minimizar la energía que consumen las aplicaciones computacionales.
- ✓ Algunas prácticas sostenibles del software verde son el uso de eficiencia energética, el uso inteligente de hardware y software en la nube, el aumento del ciclo de vida del software y la reducción de desechos electrónicos.
- ✓ Un CPU consume mayor energía entre más procesamiento se le demande debido a las pérdidas energéticas por las características electrónicas de los transistores que constituyen al CPU.
- ✓ Los desarrolladores pueden estar conscientes del consumo energético de sus aplicaciones mediante el uso de librerías que miden el consumo en Watts-hora y sus equivalentes en kg de CO_2.
- ✓ Los desarrolladores deben usar el lenguaje de programación que más se adecue a las cargas computacionales de su aplicación. C es muy eficiente en cuanto a consumo de energía, tiempo y memoria, pero Python es más eficiente en cuanto a tiempo de desarrollo de una aplicación.
- ✓ Si los desarrolladores trabajan en sistemas embebidos, deben utilizar de la mejor manera los modos de consumo de energía de sus microcontroladores.
- ✓ El uso de IA reduce tiempos y esfuerzos de trabajo de los humanos, pero incrementa el consumo de energía si no son bien aprovechados.

Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

Les agradezco su atención.

Copyright (c) 2026 - Miguel Santiago Villafuerte Ramírez - www.migsantiago.com

Referencias

- *AI HATs—Raspberry Pi Documentation*. (n.d.). Retrieved April 28, 2026, from https://www.raspberrypi.com/documentation/accessories/ai-hat-plus.html
- Bernardino, F. (2024, September 12). Desarrollo Sostenible en el Sector Software: Cómo Crear Soluciones Tecnológicas Ecológicas. *beecrowd*. https://beecrowd.com/es/blog-posts/desarrollo-sostenible/
- *Cython—An overview—Cython 3.3.0a0 documentation*. (n.d.). Retrieved April 21, 2026, from https://docs.cython.org/en/latest/src/quickstart/overview.html
- Garcia, M. B. (2023, August 8). ¿En qué consiste el desarrollo sostenible? *Desarrollo Sostenible*. https://www.un.org/sustainabledevelopment/es/2023/08/what-is-sustainable-development/
- Heikkilä, M. (n.d.). *Making an image with generative AI uses as much energy as charging your phone*. MIT Technology Review. Retrieved April 23, 2026, from https://www.technologyreview.com/2023/12/01/1084189/making-an-image-with-generative-ai-uses-as-much-energy-as-charging-your-phone/
- Jeff Geerling. (2026, January 15). *AI this, AI that, Here's Raspberry Pi's new AI HAT* [Video recording]. https://www.youtube.com/watch?v=jRQaur0LdLE
- Lottick, K., Susai, S., Friedler, S. A., & Wilson, J. P. (2019). *Energy Usage Reports: Environmental awareness as part of algorithmic accountability* (arXiv:1911.08354). arXiv. https://doi.org/10.48550/arXiv.1911.08354
- *MOSFET Push Pull Amplifier*. (n.d.). Retrieved April 21, 2026, from https://reviseomatic.org/help/s-push-pull/Push%20Pull%20MOSFET%20Amp.php
- *Op Amp Slew Rate and Rise Time Explained—Technical Articles*. (n.d.). Retrieved April 21, 2026, from https://www.allaboutcircuits.com/technical-articles/op-amp-slew-rate-and-rise-time-explained/
- *Optimize Options (Using the GNU Compiler Collection (GCC))*. (n.d.). Retrieved April 21, 2026, from https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html
- Processor power dissipation. (2025). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Processor_power_dissipation&oldid=1311562655
- Un loco y su tecnología. (2026, January 25). *Raspberry Pi AI HAT+ 2: What you can actually do* [Video recording]. https://www.youtube.com/watch?v=E-klGfbOf0s
- Shaw, D. (n.d.). *Vertical Scaling vs Horizontal Server Scaling Solutions & Strategies*. Retrieved April 22, 2026, from https://www.cloudpanel.io/blog/server-scaling-solutions/
- Team, I. C. E. (2023, January 9). *¿Qué es la programación verde y por qué importa? | IBM*. https://www.ibm.com/mx-es/think/topics/green-coding
- Tomlinson, B., Black, R. W., Patterson, D. J., & Torrance, A. W. (2024). The carbon emissions of writing and illustrating are lower for AI than for humans. *Scientific Reports*, *14*(1), 3732. https://doi.org/10.1038/s41598-024-54271-x
- *Understanding & Managing the Plethora of MCU Low-Power Modes*. (n.d.). Retrieved April 22, 2026, from https://www.mouser.mx/applications/low-power-ewc-low-power/?srsltid=AfmBOopKZXgnG29LBtu7ECRr8tlVwhchmv54O2LTf_eMPuBWiMG3zCty