MCP SERVER | NO CODE | CLOUD HOSTED

ESPHome MCP

Control your physical hardware using natural conversation.

ESPHome MCP connects any AI agent to your ESPHome microcontrollers, letting you manage and monitor physical IoT devices using natural conversation. Get real-time sensor readings or change the color of a light without opening a dashboard. It lets you control switches, set blinds, arm alarms, and more, all through simple chat commands.

F Quality Score 3.6/100

esp32 | esp8266 | home-automation | smart-home | microcontrollers | sensors

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/esphome

SCAN TO VIEW

The infrastructure that powers AI agents in the real world.

Vinkius connects AI to the world's software through secure, enterprise-grade infrastructure — enabling real-world execution at scale, built on the Model Context Protocol (MCP).

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the cloud infrastructure where AI agents connect to the software your business already runs. We handle the hosting, the security, the credentials, the uptime — you get agents that actually do things.

We operate the world's largest managed MCP catalog. Major SaaS platforms, CRMs, databases, and cloud providers — running, monitored, production-ready. This MCP server is hosted and maintained by the Vinkius Cloud for AI Agents.

> *The agent doesn't manage credentials, doesn't manage uptime, doesn't manage security. Vinkius does.*
>
> — Architecture principle

Four Pillars of the Vinkius Runtime

01 — Security by design

Credentials stay encrypted at rest via AES-256. The AI agent never touches raw keys — they're injected into a sandboxed V8 isolate at runtime. Actions are logged, and connections have an emergency kill switch.

02 — Built on MCP Fusion

This MCP server was built with **MCP Fusion**, the open-source framework (Apache 2.0) that powers the entire Vinkius catalog. Schema-as-firewall strips undeclared fields, compiled PII redaction runs at zero overhead, and cryptographic lockfiles produce git-diffable audit trails.

03 — Deterministic observability

Eight immutable metrics per endpoint: request volume, p95 latency, error rate, active connections, cost attribution. A live payload feed logs every tool call with mutation detection.

04 — Autonomous operations

Servers are deployed, monitored, and patched autonomously. New capabilities and security patches ship weekly. Zero-downtime deployments ensure continuous availability across all managed MCP servers.

AES-256	Ed25519	24h TTL	V8 Isolate
Encryption at rest	PKI vault signatures	Ephemeral session keys	Sandboxed execution

One Token. Instant Access.

Every MCP server on Vinkius is accessed through a **Connection Token**. Tokens are generated in the cloud dashboard and produce a unique MCP endpoint URL. Paste this URL into any MCP-compatible client — no SDK required.

A single token can serve **multiple AI clients simultaneously**, or you can issue separate tokens per client for granular access control. Each token tracks its own request count, last activity timestamp, and can be individually enabled or revoked.

MCP ENDPOINT

```
https://edge.vinkius.com/{token}/mcp
```

Claude · Cursor · VS Code · Windsurf · Grok · Gemini

Security Is the Architecture

Security in Vinkius is not a feature — it's the foundation of the runtime. The gateway enforces multiple independent protection layers between AI agents and third-party APIs.

01 — Ed25519 PKI Vault

Every workspace has an Ed25519 Master Key. Session keys are generated ephemerally (24h TTL) and signed by the Master Key. Credentials never leave the vault boundary.

02 — V8 Isolate Sandboxing

Tool code runs inside isolated-vm V8 isolates with 64 MB memory caps and per-request timeouts. No filesystem access, no network access except through the SSRF-guarded fetch bridge.

03 — SSRF Guard

All outbound HTTP requests are DNS-resolved and validated before execution. Private IP ranges (10.x, 172.16-31.x, 192.168.x, AWS metadata 169.254.x) are blocked at the network layer.

04 — DLP & PII Redaction

A ResponseGuard pipeline intercepts every tool response. Configurable redaction patterns strip sensitive fields (emails, SSNs, card numbers) before data reaches the AI agent.

05 — Cryptographic Audit Trail

Every request is signed into a SHA-256 hash chain with Ed25519 signatures. Events form a tamper-proof, SIEM-exportable forensic record.

06 — Honeytoken Trap System

Phantom credentials are injected into isolated environments. If a honeytoken is used outside Vinkius infrastructure, the server is quarantined instantly.

Emergency Kill Switch

EU AI Act Art. 14(1) Compliant

The kill switch is an **emergency halt** mechanism — not a simple toggle. When triggered, it executes three actions atomically:

01 — Server deactivated

The MCP server is immediately taken offline across the entire cluster.

02 — All tokens revoked

Every connection token is invalidated. Total lockout — reconnection blocked until new tokens are issued.

03 — WebSocket connections killed

Active connections terminated via Redis pubsub broadcast. Propagates to every runtime node in the cluster.

Full Visibility. Zero Guesswork.

The Vinkius cloud dashboard includes a full MCP Governance suite — real-time analytics and security controls for production AI operations.

Control Plane

KPI dashboard with request volume, latency, success rate, token consumption, and AI-generated operational briefings.

Agent Activity

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

FinOps

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

Tool Health

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

Firewall & DLP

PII redaction activity, sensitive data protection counters, and security event timeline.

Incident Log

Error trends, failure rates, status-code breakdowns, and forensic audit trail access.

> Get started at **cloud.vinkius.com** — connect your AI agent in under 60 seconds.

ESPHome MCP

10 tools available
Cloud-hosted on Vinkius

Managing smart home hardware usually means jumping between apps and complex dashboards. This MCP changes that. It connects your ESPHome-powered devices to any AI agent, giving your natural language prompts direct access to the physical world. You can ask your agent for the current temperature reading or tell it to change the living room light to a specific color. Need to automate something? Your agent handles complex routines—like turning off lights and setting the alarm after you leave. This capability is hosted on Vinkius, giving your AI client access to thousands of other hardware controllers, making it the central hub for all your smart devices.

Core Capabilities

01 — Monitor sensor data

Retrieve the current status and value from any connected sensor or digital switch.

02 — Control lighting systems

Adjust lights by turning them on, off, or setting precise colors and brightness levels.

03 — Manage mechanical components

Operate blinds, garage doors, fans, and standard switches using conversational commands.

04 — Handle security protocols

Arm or disarm alarm panels directly from your chat interface.

05 — Set numerical states

Adjust specific values on devices, like temperature set points or dimmer levels.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/esphome** — connect your AI agent in three steps.

01 First, enable the `web_server` component in your ESPHome YAML configuration file.

02 Next, subscribe to this MCP and provide your device's local URL and any required password within Vinkius.

03 Finally, tell your AI agent to perform an action, like 'Set the kitchen light to blue,' and it handles the rest.

The bottom line is that your physical hardware speaks directly to your AI client over a simple API connection.

Built For

This MCP serves developers who build custom IoT setups and engineers tired of managing dozens of siloed dashboard interfaces. If you run smart systems on ESP32 or ESP8266, this is for you.

IoT Developer

Tests device behaviors and debugs entity states using natural language prompts instead of writing specific API calls.

Home Automation Enthusiast

Creates complex, multi-step routines—like 'goodnight' mode—by simply talking to the system through an AI agent.

Hardware Engineer

Monitors sensor metrics and triggers physical actions during prototyping phases without needing a dedicated dashboard setup.

What Changes When You Connect

01 Stop clicking through menus. You can now monitor any sensor reading or change a light's color just by asking your agent, which uses the `get_entity_state` and `light_action` tools.

02 Run complex routines like 'movie night' in one go. Your agent coordinates multiple actions, such as setting a specific number using `number_set`, dimming lights with `light_action`, and closing blinds via `cover_action`.

03 Get performance data without logs. Need to check the device health? The `get_metrics` tool pulls Prometheus data directly into your conversation context for immediate debugging.

04 Simplify security management. Use `alarm_action` to arm or disarm panels, eliminating the need to physically interact with a separate key switch just to run an automation script.

05 Handle all inputs: From simple on/off toggles using `switch_action` to pressing virtual buttons via `button_press`, this MCP covers every common interaction point.

Real-World Applications

The 'Leaving Home' routine

A homeowner asks their agent, 'I'm leaving.' The agent triggers the sequence: it uses `light_action` to turn off all non-essential lights, executes `cover_action` to close the garage door, and finally calls `alarm_action` to arm the system. All done with one command.

Troubleshooting a sensor reading

A developer asks their agent about a temperature fluctuation. The agent immediately uses `get_entity_state` to pull the live data, and then runs `get_metrics` to check if the device itself is reporting unusual performance metrics.

Setting a complex scene

An engineer needs to set up a climate control test. They ask the agent to 'Set the target temperature to 24 degrees and ensure the fan is running.' The MCP uses `number_set` for the temperature and `fan_action` to activate the cooling.

Remote access testing

A hardware team member needs to test a prototype blind. They simply tell their agent, 'Close the main blinds.' The MCP executes the action using `cover_action`, allowing them to verify functionality without needing local physical access.

Patterns to Avoid

Assuming state changes are automatic

✗ AVOID

The user asks, 'What is the status of the attic sensor?' and expects a response, but the agent fails because it didn't specify which tool to use.

✓ INSTEAD

Always start by asking your agent to `get_entity_state` for the specific entity ID (e.g., 'attic_temperature'). This guarantees you get the current metric reading.

Trying to control a value that needs selection

✗ AVOID

A user asks, 'Set the thermostat to Eco Mode,' but the agent fails because it only recognizes simple number inputs.

✓ INSTEAD

If the device has predefined states or options (like modes), use `select_option` and specify the exact mode name ('Eco').

Ignoring necessary actions

✗ AVOID

The user says, 'Turn on the bathroom lights,' but the system requires a specific state change that must be explicitly commanded.

✓ INSTEAD

Use `light_action` to ensure the light is powered up. If it's dimmer-controlled, use `number_set` alongside it.

The Right Fit

Use this MCP if your goal is physical control and real-time monitoring of connected hardware. You need an agent to act as a natural language interface for a system built on microcontrollers (like ESP32/ESP8266). If you only need to read data from external databases, or process documents, this isn't the right tool; stick with generic API connectors. However, if your workflow involves triggering physical actions—turning things on or off, changing colors, moving blinds, checking a live sensor reading—then this MCP is exactly what you need. It connects the digital intelligence of your AI client to tangible electricity and sensors.

The dashboard maze

Today, if you want to know why the garage door is open or turn off a specific set of lights, you're stuck clicking through three different manufacturer apps. You check the weather widget, then open the lighting app for dimmer control, and finally jump into the alarm panel interface just to arm it. It takes five different screens and seven clicks.

With this MCP, your agent handles all those steps automatically. Instead of opening dashboards, you simply ask: 'Check if I need to lock up.' The agent checks sensors, runs `cover_action` on the door, and then triggers `alarm_action`. You get the outcome instantly.

ESPHome MCP Gives You Total Control

You don't have to manually write code or build complex automations in YAML files just to check if a sensor is working or adjust the ambiance. The agent handles the API calls for you.

Now, every interaction—from reading temperature using `get_entity_state` to adjusting blinds with `cover_action`—is handled through conversational language. It's hands-free device management.

ESPHome: 10 Device Control Tools

These tools let your agent perform specific physical actions, from checking a sensor reading using get_entity_state to toggling complex components like fans and alarms.

#	TOOL	DESCRIPTION
01	`alarm_action`	Triggers an action on an alarm control panel, such as arming or disarming the system.
02	`button_press`	Simulates a physical button press event on an entity.
03	`cover_action`	Controls motorized covers, including adjusting blinds or garage doors to specific positions.
04	`fan_action`	Changes the operational state of a fan entity (e.g., on/off or speed).
05	`get_entity_state`	Retrieves the current status, value, or reading from any specified sensor or switch.
06	`get_metrics`	Pulls detailed Prometheus performance metrics directly from the connected device.
07	`light_action`	Manages lights, allowing you to turn them on or off and set specific color values (RGB).
08	`number_set`	Sets a precise numeric value for an entity that accepts continuous data.
09	`select_option`	Forces an entity to adopt one specific option from a predefined list of choices.
10	`switch_action`	Toggles the power state (on/off) for a standard electrical switch.

See It in Action

Real prompts you can use once this MCP is connected to your AI agent through Vinkius Cloud.

U What is the current reading of the 'living_room_temperature' sensor?

Checking the sensor... The current state of 'living_room_temperature' is 22.5°C.

U Turn on the kitchen light and set it to a warm orange color.

I've sent the command to the kitchen light. It is now on with the RGB values set to (255, 165, 0).

U Close the garage door cover.

Executing the close action on the garage door cover. The command has been sent successfully.

Frequently Asked Questions

01 **How does the ESPHome MCP handle multiple devices?**

The MCP uses entity IDs to manage every single device, whether it is a light or a fan. You simply reference the specific ID when you ask your agent for an action.

02 **Can I use the ESPHome MCP to read sensor data?**

Yes. Use `get_entity_state` to pull real-time readings from any connected temperature, humidity, or binary sensor.

03 **Is the ESPHome MCP only for simple on/off actions?**

No. You can do far more. For example, you can use `light_action` to set specific RGB colors, not just turn them on or off.

04 **What if I need to check the device health metrics with ESPHome MCP?**

You call the `get_metrics` tool. This fetches detailed performance data from your device using Prometheus standards for advanced debugging.

05 **How do I make the blinds move using the ESPHome MCP?**

Use the `cover_action`. You can tell your agent to 'Close the garage door' or specify a position and tilt angle within that command.

Go Live in 60 Seconds

Get your connection token from **cloud.vinkius.com**, then paste the endpoint URL into any MCP-compatible client.

YOUR MCP ENDPOINT

```
https://edge.vinkius.com/[TOKEN]/mcp
```

CLIENT	WHERE TO CONFIGURE
Claude AI	Profile → Customize → Connectors → "+" → Add custom connector → Paste endpoint
Cursor	Settings → Features → MCP Servers → "+ Add New MCP Server" → Type: SSE → Paste endpoint
VS Code	Ctrl/Cmd+Shift+P → "MCP: Add Server" → add `"esphome": { "url": "…" }`
Windsurf	MCP Settings → `mcp_settings.json` → Add endpoint URL
ChatGPT	Settings → Tools & plugins → Add MCP server → Paste endpoint
Gemini	Extensions → Add MCP Server → Paste endpoint URL

ASK AN AI ABOUT THIS

Let your preferred AI explain this MCP server

- Ask ChatGPT
- Ask Claude
- Ask Perplexity
- Ask Gemini
- Ask Grok

READY TO CONNECT

ESPHome is live on Vinkius Cloud.

Get your connection token, paste it into your AI agent, and start building. No SDK. No deployment. Just results.

Start at cloud.vinkius.com →

vinkius.com · support@vinkius.com

INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by ESPHome. All third-party trademarks, logos, and brand names are the property of their respective owners. Their use in this document is strictly for informational purposes to identify service compatibility and interoperability.

DOCUMENT INFORMATION

Generated	June 2026
MCP Server	ESPHome MCP
Server ID	`019e3892-4f83-718f-9c79-7a5490a66a72`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE

This document is generated automatically by the Vinkius PDF Engine. Content reflects the MCP server configuration at the time of generation and may change as updates are deployed. For the most current information, visit vinkius.com/mcp/esphome.