

Anthropic previews the Model Hardware Standard for AI agents that operate lab and factory equipment

Source copy kept by wheresthe.ai for the story “Anthropic previews the Model Hardware Standard for AI agents that operate lab and factory equipment” (<https://wheresthe.ai/d/us-anthropic-previews-model-hardware-standard-for-agents-running-lab-equipment-2026-08/>).

Source: <https://www.anthropic.com/news/model-hardware-standard-research-preview>

Text saved 24 Oct 2026 (the page as published).

SHA-256 of the saved text: b99c59982695b7d221a3b834c39ae7d95aa65c60f06e374e060ffd9909df116b

Chapter 01 : How MHS works

We’re opening a research preview of the Model Hardware Standard (MHS), a shared specification for AI agents to safely operate physical devices, to a first group of scientific research labs and advanced manufacturers. MHS enables AI agents to operate multiple lab and manufacturing instruments, such as microscopes, liquid handlers, and robotic arms, in parallel, and perform intricate tasks ranging from routine drug discovery experiments to laser calibration on a quantum computer. The development of MHS began as a collaboration between Anthropic and HHMI Janelia Research Campus . It typically takes a lab or manufacturing facility weeks, if not months, to set up and integrate their hardware. Most devices don’t communicate with each other, instead requiring specialists to build bespoke integrations. MHS reduces this integration work to hours or minutes. And by incorporating AI into these tools, MHS also helps researchers and engineers more readily orchestrate autonomous, round-the-clock experiments and workflows, with agents able to reason through each step in an experiment, update parameters in real time, and, in some cases, recover from hardware errors without intervention. We’re sharing an early version of MHS with partners across science, robotics, electronics, and manufacturing so we can collaborate to build safety evaluations and develop best practices for AI systems operating physical equipment, ahead of making the standard open source. MHS works with any device that has a programmable interface. It is also model-agnostic, and any agent harness can access it using standard protocols, such as the Model Context Protocol . To apply for access to the research preview, head here .

How MHS works Before and after the Model Hardware Standard (MHS).

Getting multiple devices in a lab or on a factory floor to communicate with one another can be challenging, even setting aside the added difficulty of integrating AI into the setup. Each device tends to have its own programming interface, and so far there has been no standardized way to integrate them. And once the devices are connected, there is no common way for them to share data with an AI agent, nor to let the agent operate them safely. MHS addresses these challenges by introducing a standardized driver: software that translates between a computer’s operating system and a hardware device. The MHS driver uses a simple set of primitives—commands like “read” (for example, “get temperature”) or “write” (for example, “set temperature”)—that any hardware device can understand and act on. And it makes each device discoverable in a standard format, so that devices and agents can find each other and communicate across networks without needing a bespoke “translator” program in between. The MHS driver also helps an AI agent understand how to use a device it has never seen before, giving it information about machine characteristics that may not be discernable from code alone (for example, the weight of a robot arm, which is important for knowing how to manipulate it safely). To date, much of this information has been stored in paper manuals, on a user’s computer, or as tacit knowledge. But the MHS driver contains tags that let the user write this information directly in natural language (users can either do this themselves, or by chatting to an agent that interviews them about their hardware setup). With the information from these tags, the MHS driver then automatically produces a reference file with information about a device’s general characteristics, such as what it can measure, what can be adjusted, and what safety limits will be enforced. This file gives the agent everything it needs to know to operate the device. After the devices are connected and the agent knows how to use each one, the agent needs a way to control the hardware. For MHS, there are three such mechanisms: MCP, the command line interface, and code files (APIs). These work together to enable orchestration across multiple devices via a single line of code. Once the agent can control the devices, it’s able to receive operating data from each one and supervise and direct the work at a high level. The agent can sequence steps across instruments, monitor results, and adjust parameters as conditions change in real time. When the agent needs to execute long-running tasks or operate devices faster than its online reasoning would allow, it can chain together driver commands from one or more devices in code files. This allows the devices to carry out operations themselves,

without the agent needing to reason at every step. As we've tested MHS, we've found that Claude interacts with experiments and hardware in an exploratory manner, much as a scientist would. For example, we observed Claude make an adjustment to a laser, observe the results through a camera to assess how its adjustment moved the laser beam, and repeat the process, seeking to understand the sequence of events. Claude then packaged what it learned into code files, writing a deterministic script that let it align the laser without having to reason at each step, so the whole process could run as a single command.

Early examples from MHS We are only just beginning to see what people can do with frontier models and MHS, but our hope is that the standard can be of use to researchers, engineers, and other practitioners in speeding up the process of discovery and experimentation in any domain that uses devices with a programmable interface. As we developed MHS, we shared it with a handful of labs and hardware manufacturers in biotech, robotics, quantum computing, and other fields. Across these early projects, we saw MHS reduce the time it took to integrate devices, make it possible to iterate faster in a variety of experimental settings, and assist with the live operation of machines and real-time fault detection. Below, our partners share the details of some of their early projects involving MHS.

Genentech: Implementing MHS for lab automation Researchers at Genentech implemented and tested MHS as a proof-of-concept for automating the BCA protein assay, a standard procedure to measure total protein concentration in a sample, which requires coordinating across a liquid handler, a robotic arm, and a plate reader.

University of Washington Baker and Pinglay labs: Bringing AI agents to the bench Zihao Song, a PhD student in the University of Washington Baker and Pinglay labs, used MHS to build a dashboard to remotely monitor his instruments; an AI agent-supervised qPCR (which copies a target DNA sequence through repeated cycles of heating and cooling) that watches amplification curves and halts the procedure at the right moment; and an integration between a robotic arm and a liquid handler for collision-free plate handoffs.

Carnegie Mellon University: Determining dose-response curves through rapid automation Researchers at Carnegie Mellon University used MHS to run serial dilution dose-response experiments about three times faster than before, with an AI agent orchestrating a liquid handler, a plate reader, a robotic arm, and monitoring cameras spread across three computers with fundamentally incompatible interfaces.

HHMI Janelia: Using MHS to accelerate microscopy research At HHMI Janelia Research Campus, researchers are using MHS to speed up a range of microscopy-related projects. Here, Virginie Ruetten, a scientist in the Ahrens lab who studies how sleep helps the body recover from stress, shares how she used MHS to unify and orchestrate a rig that previously involved seven different vendor programs without a shared interface.

QuEra Computing: Using MHS in quantum laser stabilization QuEra, a company that builds quantum computers using neutral atoms, used MHS to give an AI agent control over parts of the laser system inside its quantum machines. The agent developed a controller that recovers the laser's "lock"—the ultra-precise frequency the lasers must hold to interact with the atoms—99.3% of the time without human intervention.

Tetsuwan Scientific: Using MHS to run qPCRs to profile local pollution Researchers at Tetsuwan integrated MHS with its automated biology lab platform, ResearchOS. MHS helped orchestrate a qPCR workflow to contribute to citizen science efforts to characterize pollution in California's San Pedro Creek.

01 / 06

Hardware vendors and the software companies that support them are also building MHS support into their equipment so AI agents can more easily discover and operate their devices. For example: Amazon Web Services will support MHS through Strands Robots, the library for connecting AI agents to physical devices. AWS will provide participants a private, pre-release version of the Strands Robots package for the duration of the MHS research preview. Automata is adding MHS support to LINQ, their lab automation platform, to perform intelligent error handling of instruments in autonomous labs. Danaher and Anthropic are actively exploring how MHS-supported capabilities could enable its smart instruments and autonomous laboratories to scale biomedical research and development. Doosan Robotics is testing MHS with their robotic arms, including to perform automated quality assurance and coordinate tasks across multiple robots. MBF Bioscience is building an MHS driver for ScanImage, the software that runs laser-scanning microscopes in hundreds of neuroscience labs worldwide, to integrate AI agents into real-time data analysis and experiments. QIAGEN is experimenting with MHS through a working proof-of-concept on its nucleic acid purification platform, QIASymphony Connect, showing how AI agents could help laboratories troubleshoot instrument issues faster, guide operators through recovery, and improve instrument uptime while reducing risk to biological samples. Tecan is adding MHS support for their

Fluent liquid handling platforms so AI agents can discover and operate them directly. Universal Robots has had early access to MHS and plans to add support to its robotics platform. Joining the research preview These early results from our partners are encouraging, but we have more work to do on the standard before we open-source it. As a large language model, Claude learns about the physical world through text and images, meaning its spatial and physical reasoning have limitations that still require expert oversight. When working with protein samples, for example, Genentech researchers had to guide Claude to recognize that errors caused by foaming in samples were physical failures, not software bugs, that could only be mitigated through the appropriate physical corrections. MHS also doesn't yet work with hardware that lacks a programming interface, so we're working with the manufacturers of such devices to build in MHS drivers. Many developers already use Claude Code to work with individual pieces of physical equipment; for the next phase of MHS, we hope to expand the standard to cover more of the devices developers build on. Early adopters include Hugging Face, who are adding MHS support in LeRobot, their robotics library, and Raspberry Pi, who are enabling MHS integration across a number of their products following successful tests using their Camera MHS Driver. We will also use the research preview to build additional safety evaluations with our launch partners and strengthen protections for the use of AI in the physical world. We are developing a physical safety roadmap to further bolster our safeguards policy and enforcement coverage against the risk of misuse. When we open-source MHS, we will release findings from the research preview as part of our guidance for deploying the standard safely. We're inviting stakeholders across industries to join the waitlist for our research preview of MHS. If you'd like to participate, submit your interest here .

Authors & acknowledgements

Acknowledgements MHS began as a collaboration between Alek Kemeny on Anthropic's Beneficial Deployments team and Arco Bast, a postdoctoral scientist at HHMI Janelia Research Campus. Bast was running complex brain-imaging experiments on a rig that combined lasers, motorized focusers, and specialized cameras from different vendors with no common interface. To speed up his experiments, he developed a shared memory dictionary that enabled the instruments to communicate with one another at memory speed. Kemeny and Bast worked together to integrate AI models into that interface. We thank everyone who has contributed to this work so far, including, but not limited to, Aaron Boswell, Ben Arthur, Boaz Mohar, Gagan Bhat, Mark Kittisopikul, Nadine Yasser, Nick Purcell, Takashi Kawase, and Virginie Ruetten. We look forward to moving MHS forward with our industry partners and, soon, with the open-source community.