The Journal of Open Source Software

ARES OS 2.0: An Orchestration Software Suite for Autonomous Experimentation Systems and Self-Driving Labs

Arthur W. N. Sloan [1,2], **Robert W. Waelder** [1,3], **Morgen L. Smith** [1,3,4], **Nicholas Kleiner**[5], **Arnas Babeckis**[5], **Jason Wheeler**[5], **Daylond Hooper** [5], **and Benji Maruyama** [1]

1 Air Force Research Laboratory, Foundational Technologies Directorate, United States of America **2** The National Research Council, United States of America **3** AV Inc., United States of America **4** Kansas State University, Tim Taylor Department of Chemical Engineering, United States of America **5** DCS Corp., United States of America

DOI: 10.21105/joss.10461

Software

- Review
- Repository
- Archive

Editor: Evan Spotte-Smith

Reviewers:

- @RyanTheRobothead
- @SissiFeng

Submitted: 09 February 2026
Published: 07 November 2026

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

ARES OS 2.0 (hereinafter ARES OS) is an open-source software suite to enable laboratory automation and closed-loop autonomous experimentation. Its function is to orchestrate experimental actions and data handoff between lab equipment, analysis routines, and experimental planning modules through a service-oriented architecture. ARES OS is abstracted to apply to general experimental flows common in materials science, chemistry, biology, and related disciplines. The core of ARES OS provides central control over all modules, along with the heavy lifting of user interface (UI) creation, data management, and experimental design tools. ARES OS modules communicate with the core software over protobuf (Google LLC., 2026) and gRPC (gRPC Authors, 2026), allowing them to be language-agnostic and user-creatable. This allows users to easily implement modules that control experimental hardware, process collected data, or plan experiments to meet their specific research needs. ARES OS lowers the barrier to entry for researchers to build their own self-driving labs, allowing them to focus on scientific programming for their use case and reducing the effort and time needed to bring an autonomous experimentation system online.

Statement of Need

Research and technology development in the physical sciences has historically been a slow, expensive, and labor-intensive process. To overcome these issues and accelerate the pace of discovery, researchers across a variety of fields have started a revolution in how science is done: autonomous experimentation (AE). AE, also called self-driving labs (SDLs), combines robotic high-throughput experimentation (HTE) techniques with in situ and in-line analysis methods, and artificial intelligence/machine learning (AI/ML) planning routines to autonomously plan, execute, and analyze experiments in pursuit of a user-defined goal (Abolhasani & Kumacheva, 2023; Stach et al., 2021), with the objective of making scientific research faster, better, and cheaper. This process flow is shown in Figure 1. AE systems have been demonstrated to provide faster research progress, lower experimental variability, and a reduced number of experiments to reach a goal compared to traditional manual planning and experimentation (Stach et al., 2021). Our group published the first autonomous experimentation system, ARES, for materials in 2016, and ARES OS has been in development since (Nikolaev et al., 2016).

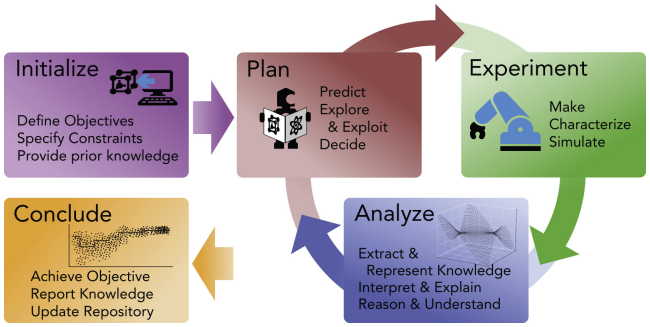


Figure 1: A closed loop, research autonomy process flow. Used with permission from Stach et al. (2021). Copyright Elsevier 2021.

Implementing a new SDL traditionally has a high barrier to entry, with software being a major contributor (Lo et al., 2024). Today there is a growing number of SDL orchestration software offerings, and while some are low-cost and/or open-source, they tend to be specific to a research domain. Many SDLs rely either on expensive offerings from commercial vendors, or are bespoke, researcher-built systems, with long implementation timelines due to the complexity and array of technical disciplines required to successfully develop and integrate all aspects of an SDL (Lo et al., 2024) (e.g., software architecture, mechatronics, AI/ML, domain-specific scientific knowledge). These factors pose a high barrier to entry and constrain SDL development to well-funded research organizations, slowing the application of SDLs to new research problems.

From a researcher's standpoint, the largest hurdle in developing an SDL is the integration of separate elements into a functioning autonomous system (Seifrid et al., 2022). Thanks to the wealth of data analysis, ML, and other scientific libraries available, many researchers have sufficient competence with Python to create the individual modules of an autonomous system but may lack the software engineering expertise to integrate them in a robust and flexible manner. `ARES OS` was developed to address this core issue by providing researchers with a modular framework for coordinating hardware, software, and data management. This framework is combined with an easy-to-use, self-populating UI and companion Python library, `PyAres` (AFRL-ARES, 2026b), which allows users to rapidly develop, test, and integrate system components.

State of the Field

Several other open-source SDL orchestration software packages are available, primarily due to the novelty of the field. Notable examples include MadSci (Self-Driving-Laboratories-at-Argonne, 2026), ChemOS2.0 (Sim et al., 2024), Minerva-OS (Zaki et al., 2025), and Ivory OS (Zhang et al., 2025).

One way `ARES OS` differentiates itself from other software packages is its autonomy-first approach to lab orchestration. While capable of operating without them, ARES OS expects closed-loop experimental analysis and planning to be part of every experimental campaign, placing these features directly in front of the user. This is also reflected in our PyAres companion library, with analyzer and planner service modules distinct from devices, allowing researchers to supply their own. Of the four other packages highlighted, ChemOS2.0 and IvoryOS have integrated support for closed-loop experimental planning, and MinervaOS lists this feature in beta. As far as could be discerned, these packages also only support semi-baked in optimization algorithms and do not have a clear path for users to supply their own experimental planning logic.

This degree of granular modularity is another way `ARES OS` differentiates itself: breaking out modules for device interaction (software or hardware), data analysis, and experimental planning.

IvoryOS, for example, bakes analysis into the hardware device driver, where the device driver returns processed data with one method call. While it is possible to use this approach within ARES OS, the default approach is to split the data collection and data analysis into different processes, allowing both detailed raw data and processed results to be automatically catalogued in the experiment database entry.

Additionally, ARES OS offers the most straightforward installation process of any of the available orchestrators. Simply downloading and running the ARES Launcher will handle installation, updates, database management, version management, and PyAres service management in a friendly, graphical interface. The other software packages highlighted here include detailed installation instructions but rely on software workflows that an experimental scientist may not be familiar with. The preferred installation method for MADSci is Docker (though installation though PyPI is also an option), ChemOS2.0 utilizes the Nix package manager, MinervaOS follows a manual python package-style installation with nested repositories, and IvoryOS installs via PyPI.

Software Design

ARES OS uses a service-oriented architecture with a C# and ASP.NET core, written to follow SOLID principles for understandability, flexibility, and maintainability (Martin, 2003). The core application handles the backend logic necessary for automation and autonomy, such as experimental routines and database interactions (ARES OS supports SQL Server, SQLite, and Postgres) and provides frameworks for interacting with system modules, such as custom GUIs, laboratory hardware, experimental planners, and data analyzers.

Communication between the core and system module services is facilitated by Google's protobuf and gRPC. The use of protobuf allows for easy data transmission over the network, facilitating the use of both local and remote experimental or computing resources. Protobuf also allows ARES OS to be language-agnostic, enabling the creation or re-use of modules written in any supported language (e.g., C#, Python, JavaScript, R).

By default, ARES OS includes a Blazor UI, designed as an intuitive hub for customizing and using an AE system, allowing for both centralized computer control of experimental hardware and the execution of user-defined campaigns for automated or autonomous experimentation. The `PyAres` library is available via PyPI and provides an easy-to-use interface to create and configure ARES OS-compatible devices, planners, and analyzers with only a few lines of Python code (AFRL-ARES, 2026b). For ease of use, we have also created the `ARES Launcher` application, which streamlines the installation and configuration of ARES OS and the necessary databases and certificates (AFRL-ARES, 2026a). The `ARES Launcher` also supports installation from specific forks of ARES OS to enable users to develop modified versions that fit their specific use cases.

Research Impact Statement

ARES OS was designed primarily to be used by experimental researchers in the physical sciences for the implementation of AE/SDL systems. Additionally, ARES OS is suitable for use by students in a classroom setting to study ML and AE principles. As part of its development, ARES OS has been used in experimental systems to study a variety of materials science problems such as carbon nanotube synthesis (Bulmer et al., 2023; Waelder et al., 2024) and fused deposition modeling 3D printing (Deneault et al., 2021). ARES OS will also be used in a new curriculum under development by the University at Buffalo's Department of Materials Design and Innovation.

Availability

The ARES OS Core source code is available from the public GitHub repository (https://github.com/AFRL-ARES/ARES/releases). The ARES Launcher source code is available from the public GitHub repository (https://github.com/AFRL-ARES/ARES-Launcher/releases) with downloadable binaries for Linux, Windows and macOS. The PyAres companion library is available on PyPI (https://pypi.org/project/PyAres) for installation with `pip` or from the public GitHub repository (https://github.com/AFRL-ARES/PyAres/releases).

AI Usage Disclosure

Multiple versions of Google Gemini and OpenAI ChatGPT were used during the development of ARES OS to generate templates, test new concepts, review code, and write documentation. All AI output was reviewed, modified and validated by human team members. No generative AI was used in the preparation of this manuscript.

Acknowledgements

The authors gratefully recognize funding from the Air Force Office of Scientific Research under LRIR 25COR019, R. Doug Riecken, PO.

References

Abolhasani, M., & Kumacheva, E. (2023). The rise of self-driving labs in chemical and materials sciences. *Nature Synthesis*, *2*(6), 483–492. https://doi.org/10.1038/s44160-022-00231-0

AFRL-ARES. (2026a). ARES Launcher. In *GitHub repository*. GitHub. https://github.com/AFRL-ARES/ARES-Launcher

AFRL-ARES. (2026b). PyAres. In *GitHub repository*. GitHub. https://github.com/AFRL-ARES/PyAres

Bulmer, J. S., Sloan, A. W. N., Glerum, M., Carpena-Núñez, J., Waelder, R., Humes, J., Boies, A. M., Pasquali, M., Rao, R., & Maruyama, B. (2023). Forecasting carbon nanotube diameter in floating catalyst chemical vapor deposition. *Carbon*, *201*, 719–733. https://doi.org/10.1016/j.carbon.2022.08.001

Deneault, J. R., Chang, J., Myung, J., Hooper, D., Armstrong, A., Pitt, M., & Maruyama, B. (2021). Toward autonomous additive manufacturing: Bayesian optimization on a 3D printer. *MRS Bulletin*, *46*(7), 566–575. https://doi.org/10.1557/s43577-021-00051-1

Google LLC. (2026). Protocol Buffers. In *GitHub repository*. GitHub. https://github.com/protocolbuffers/protobuf

gRPC Authors. (2026). gRPC. In *GitHub repository*. GitHub. https://github.com/grpc/grpc

Lo, S., Baird, S. G., Schrier, J., Blaiszik, B., Carson, N., Foster, I., Aguilar-Granda, A., Kalinin, S. V., Maruyama, B., Politi, M., Tran, H., Sparks, T. D., & Aspuru-Guzik, A. (2024). Review of low-cost self-driving laboratories in chemistry and materials science: The "frugal twin" concept. *Digital Discovery*, *3*(5), 842–868. https://doi.org/10.1039/D3DD00223C

Martin, R. C. (2003). *Agile software development: Principles, patterns, and practices*. Pearson Education. ISBN: 9780135974445

Nikolaev, P., Hooper, D., Webber, F., Rao, R., Decker, K., Krein, M., Poleski, J., Barto, R., & Maruyama, B. (2016). Autonomy in Materials Research: A Case Study in Carbon Nanotube

Growth. *npj Computational Materials*, *2*, 16031. https://doi.org/10.1038/npjcompumats.2016.31

Seifrid, M., Pollice, R., Aguilar-Granda, A., Morgan Chan, Z., Hotta, K., Ser, C. T., Vestfrid, J., Wu, T. C., & Aspuru-Guzik, A. (2022). Autonomous Chemical Experiments: Challenges and Perspectives on Establishing a Self-Driving Lab. *Accounts of Chemical Research*, *55*(17), 2454–2466. https://doi.org/10.1021/acs.accounts.2c00220

Self-Driving-Laboratories-at-Argonne. (2026). MADSci. In *GitHub repository*. GitHub. https://github.com/AD-SDL/MADSci

Sim, M., Vakili, M. G., Strieth-Kalthoff, F., Hao, H., Hickman, R. J., Miret, S., Pablo-García, S., & Aspuru-Guzik, A. (2024). ChemOS 2.0: An orchestration architecture for chemical self-driving laboratories. *Matter*, *7*(9), 2959–2977. https://doi.org/10.1016/j.matt.2024.04.022

Stach, E., DeCost, B., Kusne, A. G., Hattrick-Simpers, J., Brown, K. A., Reyes, K. G., Schrier, J., Billinge, S., Buonassisi, T., Foster, I., Gomes, C. P., Gregoire, J. M., Mehta, A., Montoya, J., Olivetti, E., Park, C., Rotenberg, E., Saikin, S. K., Smullin, S., … Maruyama, B. (2021). Autonomous experimentation systems for materials development: A community perspective. *Matter*, *4*(9), 2702–2726. https://doi.org/10.1016/j.matt.2021.06.036

Waelder, R., Park, C., Sloan, A., Carpena-Núñez, J., Yoho, J., Gorsse, S., Rao, R., & Maruyama, B. (2024). Improved understanding of carbon nanotube growth via autonomous jump regression targeting of catalyst activity. *Carbon*, 119356. https://doi.org/10.1016/j.carbon.2024.119356

Zaki, M., Prinz, C., & Ruehle, B. (2025). A Self-Driving Lab for Nano- and Advanced Materials Synthesis. *ACS Nano*, *19*(9), 9029–9041. https://doi.org/10.1021/acsnano.4c17504

Zhang, W., Hao, L., Lai, V., Corkery, R., Jessiman, J., Zhang, J., Liu, J., Sato, Y., Politi, M., Reish, M. E., Greenwood, R., Depner, N., Min, J., El-khawaldeh, R., Prieto, P., Trushina, E., & Hein, J. E. (2025). IvoryOS: An interoperable web interface for orchestrating Python-based self-driving laboratories. *Nature Communications*, *16*(1), 5182. https://doi.org/10.1038/s41467-025-60514-w