Digital Discovery

Accepted Manuscript

This article can be cited before page numbers have been issued, to do this please use: K. Wang, D. J. Ward, M. Ord, B. Liu and A. P. Jardine, *Digital Discovery*, 2026, DOI: 10.1039/D6DD00177G.

ROYAL SOCIETY OF CHEMISTRY

rsc.li/digitaldiscovery

View Article Online
DOI: 10.1039/D6DD00177G

AI-designed and AI-implemented Control Systems for Bespoke Scientific Instrumentation: Application to Scanning Microscopy

Ke Wang*† David Ward* Matthew Ord* Boyao Liu* Andrew Jardine*

*Department of Physics, University of Cambridge, Cambridge , CB3 0HE, United Kingdom.

†Corresponding author. Email: kw570@cam.ac.uk

The pace of innovation in custom scientific instrumentation is frequently bottlenecked by the complexity of software engineering. While hardware designs evolve rapidly, developing robust and integrated control systems remains resource- intensive and often exceeds the software expertise available in experimental labo- ratories. Here, we present an AI-assisted workflow for constructing and validating an integrated control system for a bespoke scientific instrument, demonstrated on the Scanning Helium Microscope (SHeM). We emphasise that this work does not develop or train new AI models; instead, we use publicly accessible, general- purpose Large Language Models (LLMs) as practical engineering tools. Using these models, we co-develop a modular software stack spanning hardware inter- faces, communication middleware, scan control, and user interfaces. To reduce the risk of deploying AI-generated code to fragile hardware, we introduced a dig- ital sandbox that emulates instrument behaviour and supports pre-deployment verification, together with cross-validation using a second LLM and mandatory human review. We demonstrate successful deployment on a physical SHeM for 2D imaging and diffraction measurements, with behaviour consistent with a manually developed control system. This work provides a reproducible and safety-oriented

View Article Online
DOI: 10.1039/D6DD00177G

template for AI-assisted software engineering in bespoke scientific instrumenta- tion.

1 Introduction

Progress in experimental physics is often driven by new instruments. From the scanning tunnelling microscope (*1*) to modern quantum sensors (*2–4*), custom hardware enables measurements beyond the reach of commercial tools (*5*). However, a persistent gap has emerged between rapid advances in hardware design and the creation of integrated control systems that operate the whole instrument with an efficient user experience. Modern instruments depend on tightly synchronised control of multiple subsystems (*6,7*), but building such infrastructure demands software engineering skills that are uncommon in university science laboratories (*8–10*). As a result, many promising instruments remain limited by fragile or incomplete control stacks.

Large Language Models (LLMs) (*11–13*) offer a potential route to reduce this software bottleneck. They can generate functional code from natural language (*13–15*), lowering the barrier to instrument programming and enabling demonstrations of self-driving laboratory workflows (*16–18*). Recent studies have already demonstrated the power of LLMs in controlling diverse materials-science instrumentation (*14*) and achieving joint control of commercial hardware alongside non-commercial, bespoke components (*15*). Beyond simple script generation, these systems have enabled natural-language interaction with microscopy and user-facility tools (*14, 19, 20*), and more recently, autonomous microscopy platforms have integrated language models with computer vision to enable content-driven measurement strategies (*21*). These developments show substantial promise, but practical challenges remain for bespoke instruments: most reports focus on a single device, isolated scripts, or operation through mature APIs, whereas customised systems often require a control system spanning hardware interfaces, communication, scan logic, and user-facing tools. Safety concerns further arise when deploying stochastic AI outputs directly to hardware control (*22, 23*),particularly when synchronisation or scan-logic errors can damage hardware or silently corrupt datasets.

In this work, we present an AI-assisted workflow for developing and deploying an integrated control stack for a real multi-subsystem scientific instrument, the Scanning Helium Microscope

View Article Online
DOI: 10.1039/D6DD00177G

(SHeM) (*24–28*). Our contribution is not in demonstrating new capabilities of LLMs themselves, but in showing how existing, general-purpose models can be used as practical engineering tools to rapidly construct complete control systems for bespoke instruments. We do not claim the first use of LLMs in instrument control; rather, our contribution is a safety-oriented and reproducible engineering approach for translating experimental intent into a working control system for bespoke hardware. For clarity, we define the "integrated control system" in this context as the combination of: hardware control for motion and detection subsystems, a communication layer for commands and data streaming, scan control logic (trajectory generation, position verification, and dwell handling), and user interfaces for operation and monitoring. To mitigate risk, we develop a digital sandbox that emulates instrument-like responses and streams synthetic data through the same interface, enabling validation of AI-generated code prior to deployment. We further employ cross-validation using an independent LLM as a review channel with final human-verified acceptance, and we validate the resulting system on a physical SHeM with experimental data showing performance comparable to a manually engineered control system.

2 Methods

2.1 Instrumentation

The SHeM is a novel type of microscope that utilises charge-neutral thermal helium atoms as probing particles. A typical SHeM is composed of three main parts: beam generation, sample manipulation, and signal detection. Beam generation is based on a free jet expansion (*27, 29*) followed by micro-collimation and in current SHeM implementations, no software control is required. The helium microprobe is incident on the sample, which is scanned in order to create images. Sample manipulation is performed by stacking commercially available piezoelectric motors to achieve motions in X, Y, Z, and azimuthal rotation. Signal detection, used to give the intensity in each pixel, is achieved by collecting the helium beam scattered in a given direction, then ionising it in a home-made detector and reading the current using a picoammeter. SHeM is capable of generating images over a large lateral scale with sub-micrometre resolution (*30, 31*) and performing spot-profile diffraction (*32, 33*); it is designed for performing high precision imaging without any

View Article Online
DOI: 10.1039/D6DD00177G

beam damage or invasive sample preparation, while collecting local surface information through spot-profile diffraction. These functions require a complex, tightly synchronised control system.

2.2 AI-Assisited Development Route

To facilitate the development of the SHeM control system, we utilised two frontier Large Language Models (LLMs): ChatGPT-5 (OpenAI) and Claude Opus 4.1 (Anthropic). Both models were accessed via their respective official web interfaces. While a formal quantitative evaluation of LLM performance is beyond the scope of this study, our development experience revealed complementary strengths: ChatGPT-5 was found to be particularly effective during the initial conceptual brainstorming and architectural design phases, providing structured and insightful high-level frameworks. In contrast, Claude Opus 4.1 demonstrated higher proficiency in generating functional, low-level source code and implementing specific hardware communication logic. By leveraging these distinct capabilities, we established a robust workflow from conceptualisation to deployment.

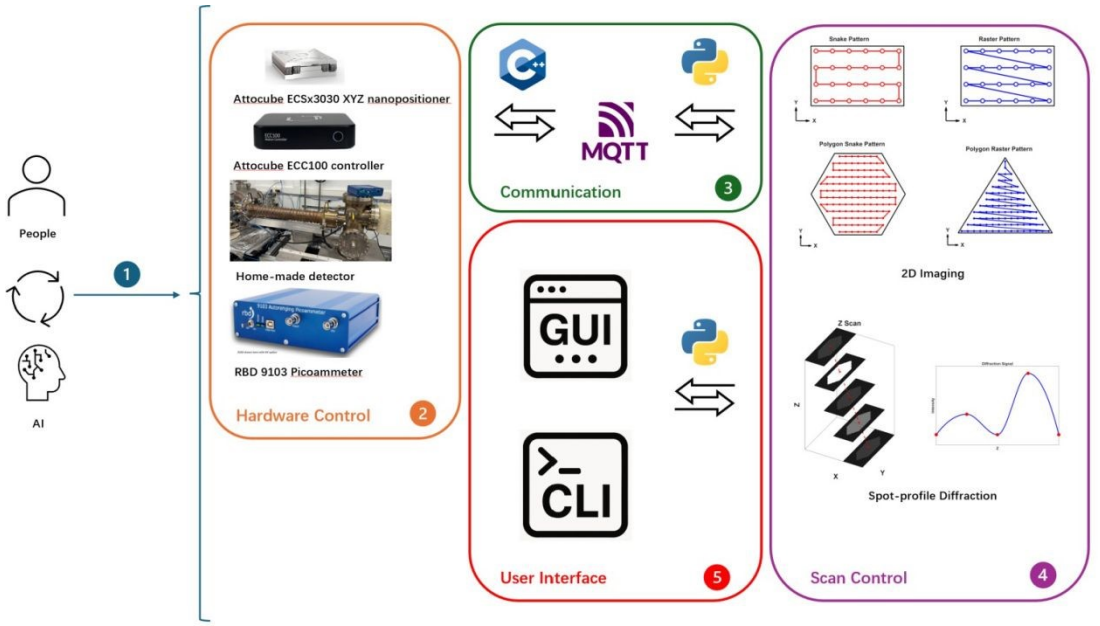
2.2.1 Conceptual System Architecture Co-designed with AI

The workflow for utilising AI in the development of the control system for the Scanning Helium Microscope (SHeM) is shown in Figure 1 as a demonstration of the methodology. At the beginning of the project, the overall concept of modularising the control system and separating the development process into distinct steps was established through iterative interactions with the AI. The first prompt illustrates how we engage the AI for conceptual design. We provide the essential background information and the overall aim of the project, followed by an open request for ideas. The outcome of this conceptual design from the AI is a division of the control system into four major modules: the hardware control layer, communication layer, scan control layer, and user interface layer. These modules form a logical progression from low-level hardware interaction through to the front-end user interface.

2.2.2 Hardware Control Layer Generation

In the second prompt, with the conceptual design, we ask the AI to develop code for sample manipulation (via the Attocube nanopositioners ECC100 controller) and for signal detection (with

View Article Online
DOI: 10.1039/D6DD00177G



1. I am designing a controlling system for a customized scanning microscope. It has stacked piezoelectric sample manipulators for XYZ and rotation motion. It has a detector where the intensity is read by a picoammeter. I want to have a modularized controlling program that can communicate with hardware, send commands based on specified scan patterns, save datapoints and generate plots and support future development. How will you struct the whole program? **Give me your idea first.**

2. For the hardware control module, I can provide you the API library from the ECC100 controller and RBD 9103 picoammeter. For controlling the sample manipulator, I need the code be able to move the piezoelectric sample manipulator and report its encoder position with device time. For the picoammeter, I need the code be able to report its reading with device time based on request. **Write the code in C++ and keep it simple, leave the synchronous function to future codes.**

3. In the communication layer, **based on the hardware control layer code you have written**, I need a MQTT client that can stream the manipulator positions and signals with their timestamps in high frequency as well as receive commands from upper level codes to move the manipulator to designated position. Write the code in Python.

4. **I need the scan control layer to be separated into four parts.** The first part should the functions that interact with the MQTT client to send commands based on the code you written for the communication layer; the second part is pure math calculation functions that generate scan patterns like raster and snake path in 2D imaging; the third part is utility functions like checking whether the manipulator is on the designated position, controlling the dwelling time and filter the signal based on the detector lag, and generating data points; the final part is the general controller that call previous three parts and conduct the whole scan process. Write the code in Python.

5. **I need to run the controlling program in multiple systems like Windows, Linux, and MacOS both locally and remotely.** So, I need the command line interface to support customized scan sequences for experienced users and graphic user interface for new students to understand the scan process much easier.

Figure 1: A schematic figure illustrating the workflow of human–AI interactions in developing the control system for the Scanning Helium Microscope. The modular structure of the system and the separation of its functional components were refined iteratively through discussions with the AI. The initial prompt demonstrates the strategy used to engage the AI for conceptual development. The full control system is separated into four main modules, where the specific prompts used to generate each module are also shown. All prompts are summarised from many iterations with the AI at that stage to demonstrate the idea.

View Article Online
DOI: 10.1039/D6DD00177G

the RBD 9103 picoammeter reading currents from the home-made helium detector) by stating the function we want to achieve. The manufacturer-provided API libraries are supplied to the AI during this stage. We also specify that synchronous functions should not be implemented at this point, as AI tends to build up future functions based on the conceptual design. The outcome is two separate pieces of code for controlling the nanopositioner and the picoammeter.

2.2.3 Communication Layer Generation

The third prompt focuses on the development of the communication layer. We give the AI the code for the hardware control it generated and the conceptual design, which serves as a base for the communication layer. In this design, the code generated in the hardware control layer is modified so that both the picoammeter and the nanopositioner encoders publish their signals and position data with timestamps under separate topics on a shared MQTT (Message Queuing Telemetry Transport, a lightweight, publish/subscribe network protocol designed for machine-to-machine communication) server. The server also accepts commands issued from the subsequent scan control layer.

2.2.4 Scan Control Layer Generation

The fourth prompt describes the interaction with the AI used to develop the most central module of the entire system: the scan control layer. Similar to the conceptual design stage in Prompt 1, this module required extensive iteration with the AI to create a framework suitable for future instrument development. The outcome of the conceptual design in this specific layer is having four components: MQTT command functions, which issue movement instructions to the hardware through an MQTT communication layer; scan pattern generators, which compute coordinates for each pixel in the scan; position verification and dwell control functions, which check whether the nanopositioner has reached the target position, manage dwell times, and extract a data point from the continuously published picoammeter signal while accounting for detector lag; a high level scan controller, which integrates all previous components and executes the complete scanning sequence. Figure 1 demonstrates several scan patterns, including “snake” and “raster” trajectories in both rectangular and polygonal geometries. It also shows the spot-profile diffraction mode, in which the sample is moved in X and Z to maintain a constant beam incidence, which allows the SHeM to vary the detection angle (*32*). The outcome of this stage is a group of codes that fulfil

View Article Online
DOI: 10.1039/D6DD00177G

our requirement for controlling the scan. However, since this layer contains complex geometric transforms and synchronisation logic, where even small computational inaccuracies risk hardware collisions or measurement artefacts, which in our experience are not capability strengths for LLM AI models (*13, 34, 35*). Thus, further verification is required, which is presented below.

2.2.5 User Interface layer Generation

Finally, the fifth prompt concerns cross-platform system compatibility. Since the underlying modules are written in C++ and Python, which are widely supported across operating systems, both a graphical user interface (GUI) and a command line interface (CLI) can be implemented as frontend clients that access the same underlying functionality. Most laboratory scientists do not have the ability to develop a good user interface (*10*) as it is usually not part of their expertise. However, AI is good at building such interfaces based on its knowledge from the front-end coding community (*13, 36, 37*). For the SHeM system, the AI helped transition our operational workflow from a series of fragmented and command-line-based scripts to a single dashboard, as shown in Figure 2. This integrated interface organises the complex control stack into four logical regions: connectivity and system metadata, motion control gallery, live display, and dynamic data visualisation.

2.3 Sandbox Simulation Environment

Given the high cost and fragility of the nanopositioners and custom detector, direct deployment of unverified AI-generated code onto physical hardware is risky. As a mitigation, we employed a "sandbox" approach. We prompted the AI to develop a digital sandbox, as shown in Figure 3, a simulation environment designed to mimic the physical instrument's behaviour. The simulator connects to the same MQTT server as the real hardware, subscribing to movement commands and publishing positions and signals based on synthetic data. It is important to clarify that the sandbox was not intended to function as a high-fidelity physical simulator. Instead, its primary purpose is to isolate and verify key aspects of the control system, including MQTT communication integrity, scan sequencing, coordinate handling, and user interface responsiveness. As such, the scope of the sandbox is intentionally limited to logical validation under nominal operating conditions. This method allowed us to validate the integrity of the MQTT communication, the accuracy of the

View Article Online
DOI: 10.1039/D6DD00177G

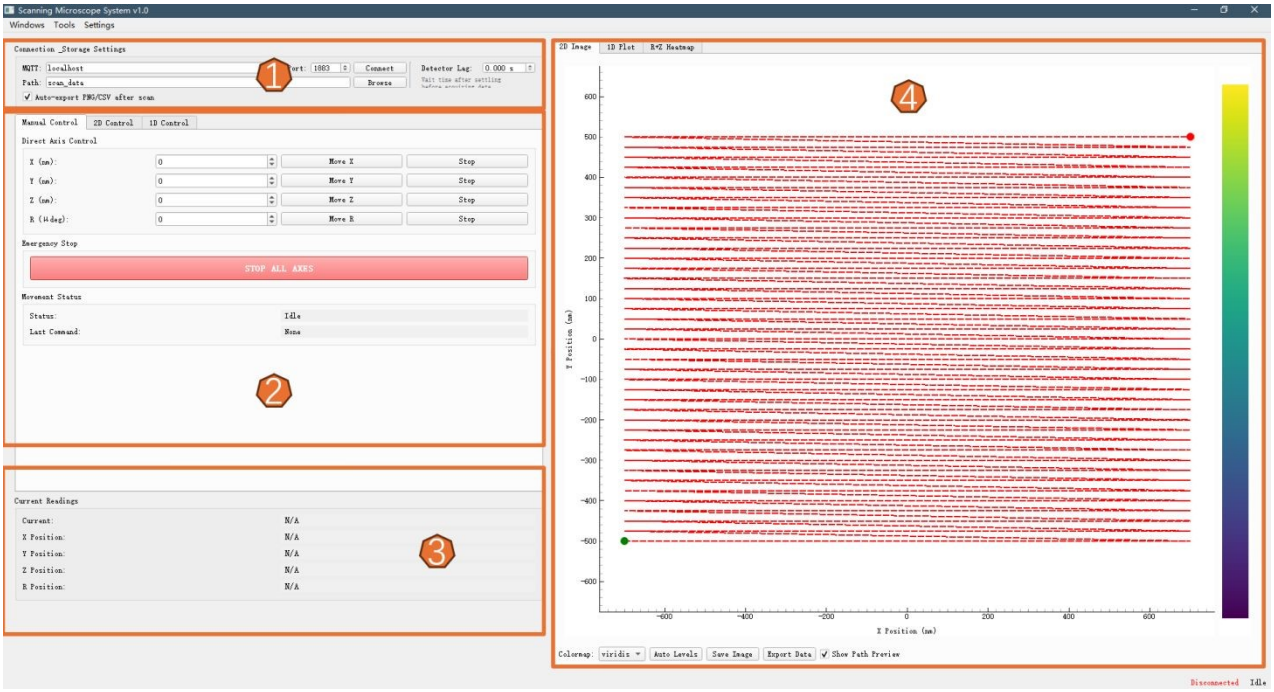


Figure 2: A screenshot of the GUI. Region 1: Connectivity and Metadata Settings. Manages MQTT client protocols, file storage paths, and detector lag compensation parameters. Region 2: Integrated Scan Control. configures manual axis positioning, 1D/2D scan geometries, and automated multi-dimensional series logic, including a global emergency stop mechanism. Region 3: Live Telemetry Display. Provides real-time instrument status, including axis coordinates and signal intensities. Region 4: Live Display. Features live rendering of 2D images, 1D line plots, and scan path previews to monitor experiment progress. Details of the GUI can be found in the Data and materials availability section.

View Article Online
DOI: 10.1039/D6DD00177G

scan control, and the responsiveness of the user interface in a virtual environment before physical implementation. To reduce the risk of correlated errors between the control code and the simulation environment, the fundamental rules governing the sandbox were explicitly defined by the authors based on the known geometry and operation of the SHeM instrument. These include coordinate transformations within a global reference frame, the definition of a centre of rotation (COR) for azimuthal motion, and the geometric coupling between Z translation and lateral displacement. The role of the AI was restricted to implementing these predefined constraints, rather than generating them autonomously. In addition, the implemented transformations were cross-checked against analytical expectations and prior experimental observations.

As illustrated in Figure 3, the sandbox operates on a digital image that serves as the virtual sample surface. The image is assigned an initial position within a global reference frame, and all pixels outside this region are treated as background, producing zero intensity for simplicity. The sample can then be translated, rotated, and interpolated in three dimensions, where each axis has a speed setting for testing the on-position check logic in the scan control layer. After constructing the sample plane, the sandbox defines a probe point that reads both the positions and the corresponding intensity value at that location. The intensity will be multiplied by a gain value with an offset, which can be randomly generated for each reading to mimic the Poisson background noise in a real detector. These data are streaming using MQTT in the same format used by the physical hardware. The current implementation does not include higher-order physical effects such as mechanical hysteresis, thermal drift, or real-time communication latency. While these factors are important for quantitative performance prediction and robustness testing, they are not required for the present objective of verifying logical correctness.

Human supervision remains essential throughout development. Logical inconsistencies in coordinate transformations, timing control, and data handling are identified through iterative inspection and refinement. Within these constraints, the sandbox provides a controlled and safe environment for validating the control architecture before deployment on physical hardware.

View Article Online
DOI: 10.1039/D6DD00177G

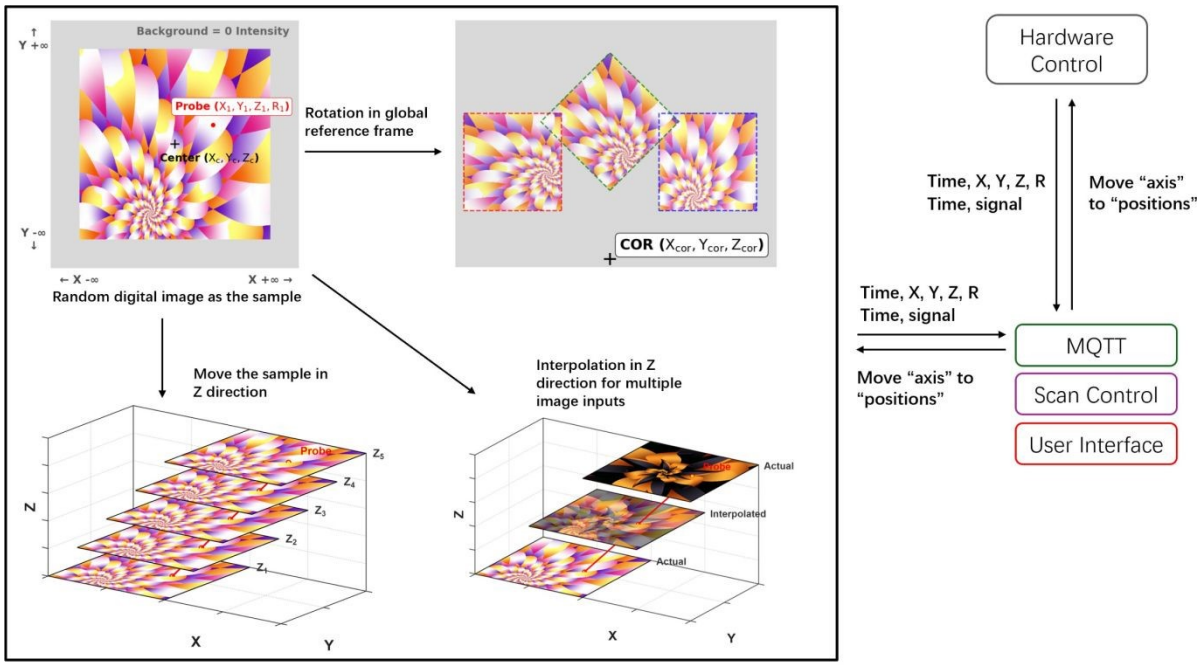


Image Configuration

- `--images IMAGE [IMAGE ...]` - Image file paths (PNG, JPEG, etc.)
- `--z-positions Z [Z ...]` - Z position for each image

To position the digital image as the sample in Z slice

- `--fov-x FOV_X` - Field of view width
- `--fov-y FOV_Y` - Field of view height

Image can be interpolated in X and Y compress or extend

- `--sample-center-x X` - Sample X position at Z=0
- `--sample-center-y Y` - Sample Y position at Z=0

Image can be positioned outside the origin to test rotation

- `--x-per-z-nm RATIO` - X shift per Z change ratio

Set the X drift per Z to test different incident beam angle

Stage Configuration

- `--speed-xy SPEED` - X/Y stage speed
- `--speed-z SPEED` - Z stage speed
- `--speed-r SPEED` - Rotation speed

Set speed for motion to test the on-position check

- `--cor-x X` - Center of rotation X
- `--cor-y Y` - Center of rotation Y
- `--cor-z Z` - Center of rotation Z

Center of Rotation can be set to test rotation and calibration

- `--limit-x-min` , `--limit-x-max` - X axis limits
- `--limit-y-min` , `--limit-y-max` - Y axis limits
- `--limit-z-min` , `--limit-z-max` - Z axis limits
- `--limit-r-min` , `--limit-r-max` - R axis limits

Hard limit for axis movement to mimic real situation

Signal Configuration

- `--gain-pa GAIN` - Signal gain
- `--offset-pa OFFSET` - Signal offset

Gain as the multiplier in signal, and offset as constant or random noise

Communication

- `--broker HOST` - MQTT broker address
- `--port PORT` - MQTT broker port

MQTT setting to test the communication

- `--pos-rate HZ` - Position broadcast rate in Hz
- `--sig-rate HZ` - Signal broadcast rate in Hz

To mimic the real streaming format

Figure 3: A schematic figure of the sandbox used to simulate the behaviour of hardware in the real SHeM. A synthetic digital image is used as the sample, which can be positioned at any location, translated, rotated, and interpolated by settings to mimic real sample behaviour. The output is in the same format as the real hardware to test the robustness of other parts of the control system.

View Article Online
DOI: 10.1039/D6DD00177G

2.4 Cross-validation with Multiple AI Models

Alongside the "sandbox" approach, we employed a cross-validation workflow using two independent LLMs mentioned above (ChatGPT-5 and Claude Opus 4.1) since different training datasets reduce correlated hallucination errors (*38, 39*). This process is illustrated in Figure 4. For each module of the control system, initial code or design suggestions were generated by one model and subsequently reviewed by the second model. This comparison was useful, as these two models frequently produced different interpretations of the same prompt, enabling the detection of hallucinations, errors, or bugs generated by a single model. Discrepancies between models were resolved by a human reviewer: we adopted the smallest modification that satisfied the stated interface constraints and passed sandbox validation. In practice, many cross-model disagreements were traceable to missing assumptions in the prompt; therefore, resolution frequently involved tightening the prompt with explicit constraints and rerunning a scoped generation step. No change affecting scan logic was permitted to proceed to supervised hardware operation unless it passed sandbox verification.

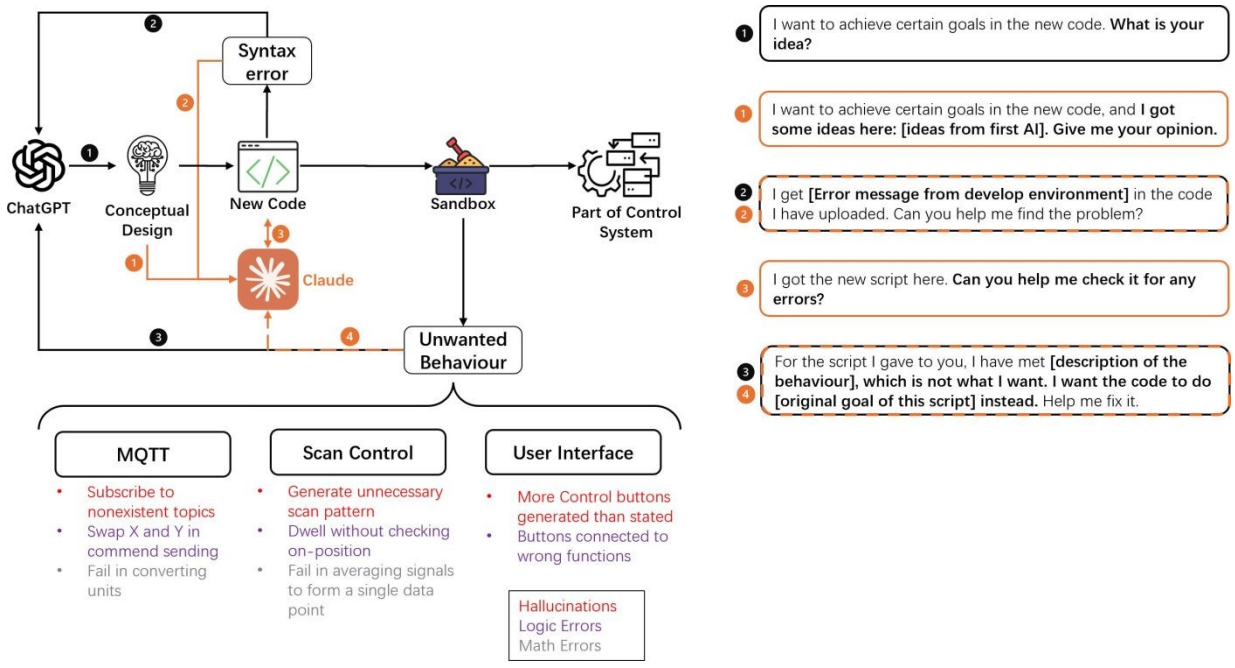


Figure 4: The schematic illustrates the workflow for cross-validating outputs from two AI models. Examples of unwanted behaviours encountered during the development of the SHeM control system are shown, along with a summary of the prompts used throughout the cross-validation process.

View Article Online
DOI: 10.1039/D6DD00177G

During development, in addition to the syntax errors that are easily detected by the coding environment, many other issues were identified through a combination of cross-validation between AI models and human review of sandbox behaviour. The sandbox played a critical role in revealing errors introduced by AI during code generation, including hallucinations, logic errors, and mathematical errors, as illustrated in Figure 4. Hallucinations such as generating additional entities not requested or referencing nonexistent functions were typically detected through standard syntax checks, whereas most logical and mathematical errors in the scan control layer emerged only when the code was exercised in the sandbox and evaluated by a human reviewer.

Although cross-validation and sandbox testing mitigated the majority of risks, AI-generated code remains unvalidated under certain conditions. In particular, direct interaction with hardware without an intermediate verification layer is hazardous, and human inspection of every script remains essential. Silent logical inconsistencies, such as misaligned device timestamps or incorrect synchronisation between independent hardware components, may still occur and cannot be reliably detected by AI cross-validation. For these reasons, all AI-generated code was subjected to manual inspection and offline testing. No model was permitted to issue direct hardware commands without first passing sandbox verification.

2.5 LLM Interaction Protocol and Prompt Engineering

We first summarise the practical protocol used during AI-assisted development. The control system was developed in an iterative prompt–review–test cycle. Each prompt included: the objective and required behaviours, the relevant API constraints and available commands, the required input/output signatures, and explicit exclusions (e.g. not generate placeholder code for future development). Generated code was first subjected to static checks (syntax, imports, functions consistency), then executed in the digital sandbox where applicable, and finally reviewed by a human for logic and safety before any supervised hardware deployment.

During extended interactions, we observed recurring failure modes: hallucinated functions or entities not present in the provided API, logical errors in coordinate transforms and scan sequencing, mathematical inconsistencies in scan geometry or dwell timing, and regression in code quality, such as duplicated implementations and overcomplicated abstractions. As an estimate, these failure

View Article Online
DOI: 10.1039/D6DD00177G

modes were encountered routinely during early iterations (e.g., frequently in initial generations for complex scan logic), but were effectively eliminated through the structured workflow described. We mitigated these issues by enforcing a stable module interface, keeping prompts and code changes scoped to a single module per iteration, and using sandbox execution as a verification for scan-logic changes. To concretely illustrate these limitations and provide a transition to our proposed workflow, Table 1 demonstrates how a common logic error was identified and resolved. Additional representative prompt–response excerpts and correction cycles are provided in the Supplementary Information.

2.6 Maintainability and Extendability

The control system may be required to change over time in order to accommodate future developments of the instrument. Beyond operational reliability, it is therefore important that the control system remains easy for a human or another AI model to understand and modify. It was found that the high-level design led to a clear conceptual separation of concerns between low-level communication, control logic, and the user interface. This logical modularity meant that new changes often only required modification of a single component's logic, and individual code segments were generally very clear and accompanied by comprehensive comments. However, to prioritise immediate operational capability and rapidly produce scientific results, the final deployment was consolidated into a single Python script.

While this single-file approach successfully drove the experimental hardware and facilitated data acquisition, it significantly increased the cost and difficulty of long-term maintenance. Housing the entire codebase in one monolithic file makes navigation, version control, and isolated testing cumbersome for a human developer. Furthermore, the model had a tendency to rely on unstructured data types to pass information between the internal layers of the program. This meant future development would rely heavily on comments to interpret data flow, making it almost impossible for automated tooling, such as type checkers, to identify simple bugs. When tasked with implementing additional features within this single-file structure, the model frequently prioritised the shortest path to functional correctness, often at the expense of engineering best practices. This approach led to the gradual accumulation of technical debt, specifically through code duplication and the creation of

View Article Online
DOI: 10.1039/D6DD00177G

Table 1: Representative correction cycle: improving position verification logic using sandbox validation.

Interaction Stage	Content / Code Snippet
Initial User Prompt	”I need a position check function to ensure the stage has reached the target before acquiring data.”

LLM Response	`while current_position != target_position:`
	`pass`
	Analysis: The equality check assumes exact numerical matching. In practice, due to noise and finite resolution, the stage position rarely matches the target exactly, causing the loop to stall indefinitely.

Sandbox Check and Human Feedback	In the sandbox (with artificial noise added to the position readings), the stage approaches the target but never reaches an exact match, causing the scan to stall. The AI is asked to fix this issue by adding tolerance to the position verification.

LLM Response	`tolerance = 5e-9 # 5 nm`
	`while abs(current_position - target_position) > tolerance:`
	`pass`
	Outcome: The revised condition allows reliable position verification under realistic noise and resolution limits. Sandbox validation confirms stable scan progression without stalling.

View Article Online
DOI: 10.1039/D6DD00177G

overly complex functions with excessive arguments. Future maintenance, such as fixing a bug in the scanning logic, would require tracking changes across multiple repeated sections of the large file. Consequently, while the current architecture was sufficient for proof-of-concept experimental work, adopting a strictly multi-file architecture is highly recommended for future iterations to physically enforce modularity and reduce maintenance costs.

Despite these architectural trade-offs, we found that many of the AI's structural issues were possible to fix with additional prompting, and they were often detectable through the application of standard code linters. All AI-generated code operates entirely offline in the instrument environment, ensuring that no experimental data or hardware access is transmitted to external servers. Manual inspection and sandbox-based verification mitigate the risk of unintended vulnerabilities or hidden dependencies.

3 Results

With the full control system validated in the sandbox, we deployed the AI-generated system on the physical SHeM to assess its performance during real measurements. In Figure 5, to provide a clear benchmark, we compared datasets across three scenarios: in Figure 5(a), a reference 2D image and diffraction measurement acquired from a synthetic digital sample in the sandbox, followed by experimental MoS_2 and biofilm datasets obtained using the AI-generated control system; in Figure 5(b), corresponding experimental datasets acquired using the original manually written control software. These comparisons allow us to evaluate the control system made by the AI in a real experimental environment. In the left panel of Figure 5(a), the sandbox results confirm the expected behaviour of the scan controller. The image reproduces the input digital sample, and the diffraction measurement yields the expected straight line, demonstrating that the system successfully keeps the probe fixed on the same pixel throughout the full Z range.

When the AI-generated control system is deployed on the physical SHeM, the resulting spatial image and diffraction measurement from a MoS_2 flake on a SiO_2 substrate (middle panel of Figure 5(a)) exhibit the same behaviour observed in data acquired using the manually written control system on a separate sample of MoS_2 flakes on hBN/SiO_2 (left panel of Figure 5(b), Ref (*40*)). This agreement demonstrates that the AI-generated scan logic functions correctly on real hardware.

View Article Online
DOI: 10.1039/D6DD00177G

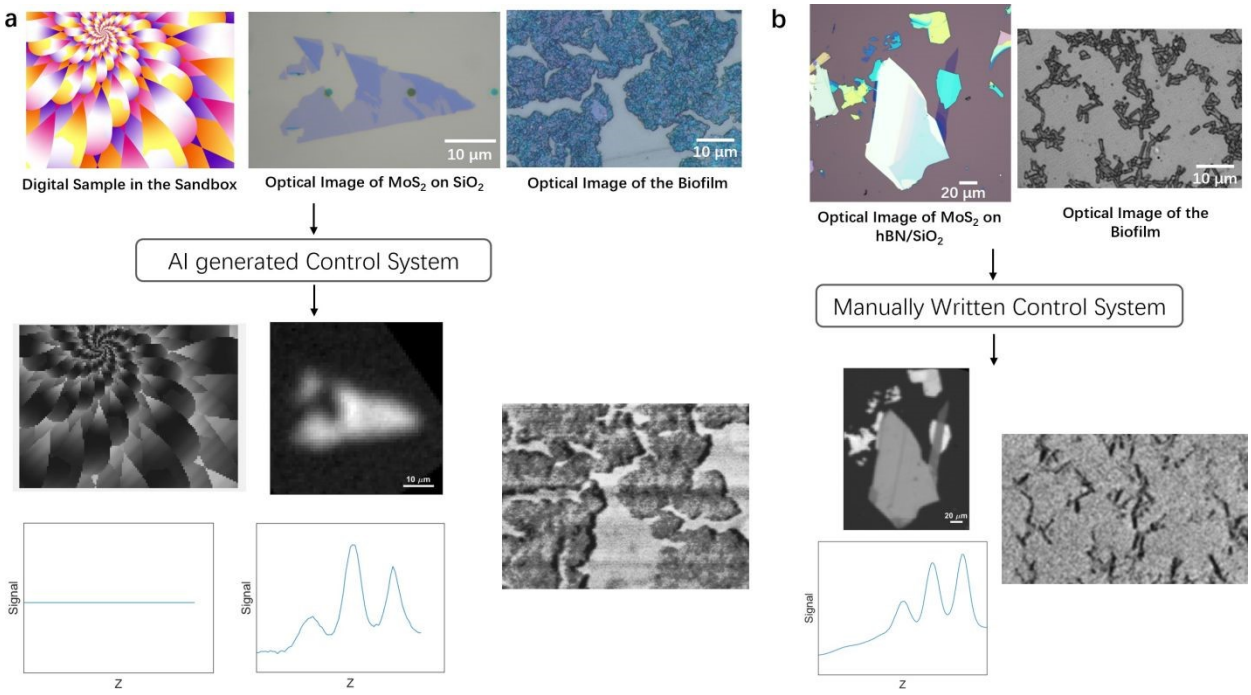


Figure 5: A schematic comparison of results obtained using the AI-generated control system and a manually developed control system is shown. (a) From left to right, a digital image is used as the input sample in the sandbox, where both the 2D scan and the Z-dependent diffraction scan exhibit the expected behaviour. The straight diffraction line confirms that the scan controller maintains the probe on the same pixel across the full Z range, as required for diffraction measurements. In the middle panel, a real MoS_2 sample is imaged using the AI-generated control system, producing both a spatial image and a diffraction profile that closely match those obtained using the manually developed control code. The right panel shows an image of a biofilm sample acquired using the SHeM at a resolution of 250 nm, representing operation under optimal instrument conditions. (b) For comparison, the left panel shows MoS_2 data from Ref (*40*), which exhibits similar contrast and diffraction characteristics to those obtained using the AI-generated control system. The right panel shows biofilm data from Ref (*41*) (reproduced with permission), also acquired at a resolution of 250 nm. There is no degradation in performance relative to manual control that was observed within experimental uncertainty.

View Article Online
DOI: 10.1039/D6DD00177G

Both MoS_2 spatial images were acquired with identical settings (2.5 µm per pixel resolution and a 1 s dwell time), while both diffraction measurements used a 3 s dwell per point and a 50 µm step size.

To further evaluate the system under optimal instrument conditions, a biofilm sample was imaged at a higher resolution of 250 nm using the AI-generated control system (right panel of Figure 5(a)). Comparing this to biofilm data previously acquired with the manual control system at the same resolution (right panel of Figure 5(b), Ref (*41*)), no degradation in performance relative to the manual control was observed within experimental uncertainty. Across all tested samples, the overall image quality and diffraction signatures remain consistent between the two control systems. Minor variations in noise levels or fine-scale contrast are attributed to detector fluctuations and sample differences rather than to the control logic itself.

The successful deployment of this system highlights a shift in experimental engineering: This exemplifies a transition from manual coding to AI-assisted development and validation. While the image quality is equivalent to the manual implementation, the disparity in development velocity is profound. The complete development cycle, from conceptual design to experimental deployment, was compressed from a timeline of many months for the manually written control system to approximately three weeks. Crucially, this acceleration did not come at the cost of reliability. The sandbox ensured that errors, AI hallucinations, and bugs were resolved within the sandbox environment, resulting in fault-free operation during physical deployment. This confirms that the combination of generative AI for code production and sandbox verification constitutes a pipeline for overcoming the software bottleneck in custom instrumentation.

4 Conclusion and Outlook

We have presented an end-to-end, AI-generated control system for the bespoke instrument, using the Scanning Helium Microscope as a demonstration of the methodology. Through careful direction of Large Language Models, we successfully integrated different hardware subsystems into a single control system. The sandbox simulation environment bridged the gap between the stochastic nature of generative AI and the operational concerns of using AI-generated code in real hardware. Because the methodology is not tied to any specific hardware or probe physics, this work provides a pathway

View Article Online
DOI: 10.1039/D6DD00177G

for rapidly transforming bespoke laboratory concepts into operational instruments across many scientific domains.

Looking forward, this modular control system enables extension toward autonomous instrument operation, as the control stack is software-defined and API-driven, which is suitable for machine learning. Future work will focus on integrating reinforcement learning algorithms into the scan control layer to achieve autonomous calibration and image recognition to select regions for diffraction measurements. Overall, the AI-assisted software engineering significantly reduces the bottleneck in experimental science, freeing researchers to focus on exploration and innovation in their scientific field rather than concentrating on software development and maintenance.

View Article Online
DOI: 10.1039/D6DD00177G

References and Notes

1. G. Binnig, H. Rohrer, C. Gerber, E. Weibel, Surface Studies by Scanning Tunneling Microscopy. *Phys. Rev. Lett.* **49**, 57–61 (1982), doi:10.1103/PhysRevLett.49.57, https://link.aps.org/doi/10.1103/PhysRevLett.49.57.

2. M. J. Molaei, A review on nanostructured carbon quantum dots and their applications in biotechnology, sensors, and chemiluminescence. *TALANTA* **196**, 456–478 (2019), doi:10.1016/j.talanta.2018.12.042.

3. C. T. Matea, *et al.*, Quantum dots in imaging, drug delivery and sensor applications. *INTERNATIONAL JOURNAL OF NANOMEDICINE* **12**, 5421–5431 (2017), doi:10.2147/IJN.S138624.

4. N. P. de Leon, *et al.*, Materials challenges and opportunities for quantum computing hardware. *SCIENCE* **372** (6539), 253+ (2021), doi:10.1126/science.abb2823.

5. J. M. Pearce, Building Research Equipment with Free, Open-Source Hardware. *SCIENCE* **337** (6100), 1303–1304 (2012), doi:10.1126/science.1228183.

6. P. T. Starkey, *et al.*, A scripted control system for autonomous hardware-timed experiments. *REVIEW OF SCIENTIFIC INSTRUMENTS* **84** (8) (2013), doi:10.1063/1.4817213.

7. V. Chandrasekhar, M. M. Mehta, RTSPM: Real-time Linux control software for scanning probe microscopy. *Review of Scientific Instruments* **84** (1), 013705 (2013), doi:10.1063/1.4775717, https://doi.org/10.1063/1.4775717.

8. N. A. Ernst, J. Klein, M. Bartolini, J. Coles, N. Rees, Architecting complex, long-lived scientific software. *JOURNAL OF SYSTEMS AND SOFTWARE* **204** (2023), doi:10.1016/j.jss.2023.111732.

9. D. S. Katz, *et al.*, *Software Sustainability amp; High Energy Physics*, Tech. rep. (2020), doi:10.5281/ZENODO.4082137, https://zenodo.org/record/4082137.

10. G. Wilson, *et al.*, Best Practices for Scientific Computing. *PLOS Biology* **12** (1), 1–7 (2014), doi:10.1371/journal.pbio.1001745, https://doi.org/10.1371/journal.pbio.1001745.

View Article Online
DOI: 10.1039/D6DD00177G

11. K. G. Barman, *et al.*, Large physics models: towards a collaborative approach with large language models and foundation models. *EUROPEAN PHYSICAL JOURNAL C* **85** (9) (2025), doi:10.1140/epjc/s10052-025-14707-8.

12. H. Pan, *et al.*, Quantum many-body physics calculations with large language models. *COMMUNICATIONS PHYSICS* **8** (1) (2025), doi:10.1038/s42005-025-01956-y.

13. I. M. M. M. Stefano Bistarelli, Marco Fiore, Usage of Large Language Model for Code Generation Tasks: A Review. *SN Computer Science* **6** (6), 673 (2025), doi:10.1007/s42979-025-04241-5, https://doi.org/10.1007/s42979-025-04241-5.

14. D. Febba, K. Egbo, W. A. Callahan, A. Zakutayev, From text to test: AI-generated control software for materials science instruments. *DIGITAL DISCOVERY* **4** (1), 35–45 (2025), doi:10.1039/d4dd00143e.

15. Y. Xie, K. He, A. Castellanos-Gomez, Toward Full Autonomous Laboratory Instrumentation Control with Large Language Models. *SMALL STRUCTURES* **6** (8) (2025), doi:10.1002/sstr.202500173.

16. F. Hase, L. M. Roch, A. Aspuru-Guzik, Next-Generation Experimentation with Self-Driving Laboratories. *TRENDS IN CHEMISTRY* **1** (3), 282–291 (2019), doi:10.1016/j.trechm.2019.02.007.

17. J. R. Kitchin, The evolving role of programming and LLMs in the development of self-driving laboratories. *APL MACHINE LEARNING* **3** (2) (2025), doi:10.1063/5.0266757.

18. Y. Liu, *et al.*, Building an affordable self-driving lab: Practical machine learning experiments for physics education using Internet-of-Things. *APL MACHINE LEARNING* **3** (4) (2025), doi:10.1063/5.0283529.

19. S. Mathur, N. van der Vleuten, K. G. Yager, E. H. R. Tsai, VISION: a modular AI assistant for natural human-instrument interaction at scientific user facilities. *MACHINE LEARNING-SCIENCE AND TECHNOLOGY* **6** (2) (2025), doi:10.1088/2632-2153/add9e4.

View Article Online
DOI: 10.1039/D6DD00177G

20. Y. Liu, M. Checa, R. K. Vasudevan, Synergizing human expertise and AI efficiency with language model for microscopy operation and automated experiment design. *MACHINE LEARNING-SCIENCE AND TECHNOLOGY* **5** (2) (2024), doi:10.1088/2632-2153/ad52e9.

21. J. Yang, *et al.*, Zero-Shot Autonomous Microscopy for Scalable and Intelligent Characterization of 2D Materials. *ACS NANO* **19** (40), 35493–35502 (2025), doi:10.1021/acsnano.5c09057.

22. E. Ginzburg-Ganz, *et al.*, Statistical Foundations of Generative AI for Optimal Control Problems in Power Systems: Comprehensive Review and Future Directions. *ENERGIES* **18** (10) (2025), doi:10.3390/en18102461.

23. G. Recupito, *et al.*, Technical debt in AI-enabled systems: On the prevalence, severity, impact, and management strategies for code and architecture. *JOURNAL OF SYSTEMS AND SOFTWARE* **216** (2024), doi:10.1016/j.jss.2024.112151.

24. D. Maclaren, B. Holst, D. Riley, W. Allison, Focusing elements and design considerations for a scanning helium microscope (SHeM). *SURFACE REVIEW AND LETTERS* **10** (2-3), 249–255 (2003), 7th International Conference on the Structure of Surfaces, NEWCASTLE, AUSTRALIA, JUL 21-26, 2002, doi:10.1142/S0218625X03005062.

25. M. Barr, *et al.*, A design for a pinhole scanning helium microscope. *NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH SECTION B-BEAM INTERACTIONS WITH MATERIALS AND ATOMS* **340**, 76–80 (2014), 20th International Workshop on Inelastic Ion-Surface Collisions (IISC), AUSTRALIA, FEB 16-21, 2014, doi:10.1016/j.nimb.2014.06.028.

26. M. Bergin, D. J. Ward, J. Ellis, A. P. Jardine, A method for constrained optimisation of the design of a scanning helium microscope. *ULTRAMICROSCOPY* **207** (2019), doi:10.1016/j.ultramic.2019.112833.

27. C. Zhao, *et al.*, A multi-detector neutral helium atom microscope. *VACUUM* **234** (2025), doi:10.1016/j.vacuum.2024.114006.

28. M. Barr, *et al.*, Unlocking new contrast in a scanning helium microscope. *NATURE COMMUNICATIONS* **7** (2016), doi:10.1038/ncomms10189.

View Article Online
DOI: 10.1039/D6DD00177G

29. M. Barr, K. M. O'Donnell, A. Fahy, W. Allison, P. C. Dastoor, A desktop supersonic free-jet beam source for a scanning helium microscope (SHeM). *MEASUREMENT SCIENCE AND TECHNOLOGY* **23** (10) (2012), doi:10.1088/0957-0233/23/10/105901.

30. S. M. Lambrick, *et al.*, True-to-size surface mapping with neutral helium atoms. *PHYSICAL REVIEW A* **103** (5) (2021), doi:10.1103/PhysRevA.103.053315.

31. M. Bergin, *et al.*, Standardizing resolution definition in scanning helium microscopy. *ULTRAMICROSCOPY* **233** (2022), doi:10.1016/j.ultramic.2021.113453.

32. N. A. von Jeinsen, *et al.*, 2D Helium Atom Diffraction from a Microscopic Spot. *PHYSICAL REVIEW LETTERS* **131** (23) (2023), doi:10.1103/PhysRevLett.131.236202.

33. M. Bergin, *et al.*, Observation of diffraction contrast in scanning helium microscopy. *SCIENTIFIC REPORTS* **10** (1) (2020), doi:10.1038/s41598-020-58704-1.

34. S. Frieder, *et al.*, Mathematical Capabilities of ChatGPT, in *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 36 (NEURIPS 2023)*, A. Oh, *et al.*, Eds., Advances in Neural Information Processing Systems (2023), 37th Conference on Neural Information Processing Systems (NeurIPS), New Orleans, LA, DEC 10-16, 2023.

35. J. C. Dunlap, R. Sissons, R. Widenhorn, Descending an inclined plane with a large language model. *PHYSICAL REVIEW PHYSICS EDUCATION RESEARCH* **21** (1) (2025), doi:10.1103/PhysRevPhysEducRes.21.010153.

36. K. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, D. Poshyvanyk, Machine Learning-Based Prototyping of Graphical User Interfaces for Mobile Apps. *IEEE Transactions on Software Engineering* **46** (2), 196–221 (2020), doi:10.1109/TSE.2018.2844788.

37. P. Duan, J. Warner, Y. Li, B. Hartmann, Generating Automatic Feedback on UI Mockups with Large Language Models, in *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24 (Association for Computing Machinery, New York, NY, USA) (2024), doi:10.1145/3613904.3642782, https://doi.org/10.1145/3613904.3642782.

View Article Online
DOI: 10.1039/D6DD00177G

38. M. Chelli, *et al.*, Hallucination Rates and Reference Accuracy of ChatGPT and Bard for Systematic Reviews: Comparative Analysis. *J Med Internet Res* **26**, e53164 (2024), doi:10.2196/53164, https://www.jmir.org/2024/1/e53164.

39. I. D. Mienye, T. G. Swart, Ensemble Large Language Models: A Survey. *Information* **16** (8) (2025), doi:10.3390/info16080688, https://www.mdpi.com/2078-2489/16/8/688.

40. N. von Jeinsen, *et al.*, Helium atom micro-diffraction as a characterisation tool for 2D materials (2024), https://arxiv.org/abs/2409.20461.

41. N. A. von Jeinsen, *et al.*, Surface visualisation of bacterial biofilms using neutral atom microscopy. *Journal of Microscopy* **301** (1), 107–115 (2026), doi:https://doi.org/10.1111/jmi.70038.

Acknowledgments

Ke Wang acknowledges financial support from the CSC and the Cambridge Trust. This work was supported by EPSRC (Grant No. EP/R008272/1) and EPSRC (Award No. EP/T00634X/1). The work was performed in part at CORDE, the Collaborative R&D Environment established to provide access to physics-related facilities at the Cavendish Laboratory, University of Cambridge.

Author contributions: **Ke Wang**: Writing - Original Draft, Writing - Review & Editing, Conceptualisation, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data Curation, Visualisation, Project administration. **David Ward**: Writing - Review & Editing, Conceptualisation, Methodology, Software, Project administration. **Matthew Ord**: Writing - Review & Editing. **Boyao Liu**: Writing - Review & Editing. **Andrew Jardine**: Writing - Review & Editing, Conceptualization, Project administration.

Competing interests: The authors declare no competing interests.

Data and materials availability: The "sandbox" simulator and the scanning control system can be found on: https://github.com/0karl96/shem_gui and https://doi.org/10.17863/

View Article Online
DOI: 10.1039/D6DD00177G

CAM.129966 (Apollo—University of Cambridge Repository). A minimal working example, including example input data and configuration files, is provided in the repository to enable reproduction of the sandbox validation workflow.

View Article Online
DOI: 10.1039/D6DD00177G

The “sandbox” simulator and the scanning control system can be found on: https://github.com/Okarl96/shem_gui or https://doi.org/10.17863/CAM.129966 (Apollo—University of Cambridge Repository). A minimal working example, including example input data and configuration files, is provided in the repository to enable reproduction of the sandbox validation workflow.