Systems & Control Transactions

Original Research Article
Peer Reviewed Conference Proceeding
ESCAPE 36 - European Symposium on Computer Aided Process Engineering
Sheffield, UK. 21-24 June 2026

PSE PRESS

Enhancing Robotics and Automation Education Through the Development of Simulation Tool for Material Synthesis

Hsuan Chang[a], Adedayo Ogunnoiki[a], and Solomon Gajere Bawa[a*]

[a] University College London, Department of Chemical Engineering, London, United Kingdom
* Corresponding Author: solomon.bawa@ucl.ac.uk

ABSTRACT

As high-throughput experimentation (HTE) becomes a cornerstone of modern materials research, undergraduate and postgraduate curricula increasingly require students to possess Python programming skills to operate automated liquid-handling robots, such as the Opentrons OT-2 and Flex. However, the high cost of this hardware often necessitates shared equipment use during hands-on lab sessions, creating a significant pedagogical barrier: students lack the individual time required to iteratively test and debug their protocols on physical robotic platforms for automated material synthesis. Furthermore, we observe that the scarcity of robotic platforms creates an imbalance in group dynamics, where students with more coding experience often lead protocol development, while those with less experience remain disengaged. To address these challenges, we developed an interactive simulator that translates Python protocols into 2D animations on personal laptops. Using gold nanoparticle (AuNP) synthesis as a case study, we demonstrated how this platform enabled visualisation, captured common programming mistakes, and refined the efficiency of the code before implementation using the physical robot. By shifting the 'trial-and-error' phase from the lab to the laptop, the simulator showed potential to democratise students' access to automation education and ensure equitable skill application across the cohort.

Keywords: automation, simulation, material synthesis, visualisation, robotic, nanoparticles

1.0 INTRODUCTION

Industry 5.0 emphasises collaboration between humans and machines to develop innovative and sustainable solutions [1]. In chemical engineering, this paradigm is reflected in the use of robotic platforms to automate high-throughput chemical synthesis, curate large datasets for process optimisation, and support sustainability initiatives [2, 3]. The integration of these technologies into undergraduate and postgraduate curricula is becoming increasingly common, with students programming liquid-handling robots using Python-based protocols to automate experiments for high-throughput material synthesis. Data-Driven Materials Manufacturing is one of the core modules taught at UCL's MSc Digital Manufacturing of Advanced Materials (DMAM) programme. This module aims to teach students how to program liquid-handling robots, such as the Opentrons OT-2 and Flex, using Python.

However, several challenges were identified while teaching with the liquid handling robots. The first challenge arises from the limited infrastructure of the lab, where the necessity of sharing a single robot among five or six students results in a fragmented learning experience. Coding for automation is a skill that demands constant experimentation and immediate feedback; however, the current setup forces long wait times between code execution and physical observation. This lack of consistent access prevents students from developing coding skills through iterative testing. Furthermore, students with strong programming backgrounds often dominate the code development stages, thereby marginalising peers with limited or no prior coding experience. This imbalance raises a significant pedagogical concern and calls for more inclusive teaching strategies to ensure equitable participation and learning outcomes for all students [4]. Additionally, because students have limited opportunities to build basic computational skills, it is difficult to incorporate sustainability and optimisation principles into laboratory education. As a result, we observed

excessive use of pipette tips and other consumables due to inefficient protocol design by students. These challenges highlight the need for accessible digital solutions to support coding for liquid-handling robots and to create inclusive learning environments.

In this study, a 2D desktop-based simulation platform was proposed to overcome the pedagogical barrier of limited access to robots. By allowing students to run code and receive individual assessments virtually, the simulator reduces reliance on physical hardware and fosters independent programming. This extra time enables students to move beyond simply writing functional code, allowing them to design efficient liquid transfer steps that minimise pipette tips waste and reinforce principles of sustainability.

The aim of this paper is to share the development and perceived use of the simulator in the classroom environment. The simulator can be used to simulate both the Opentrons OT-2 and Flex, but this paper focuses on its application with the Opentrons Flex. Here, we explain how the simulator was used to visualise a Python protocol for gold nanoparticle (AuNP) synthesis, a material of growing importance in renewable energy and healthcare applications [5].

2.0 DESIGN OF GOLD NANOPARTICLE SYNTHESIS PYTHON PROTOCOL

The Python protocol for synthesising gold nanoparticles was designed according to the Turkevich method [6]. The experimental steps of this method involved stirring and heating gold (III) chloride trihydrate ($HAuCl_4$) and trisodium citrate (Na_3Ct) solutions within a 96-well plate (reaction plate). The experimental design explored three significant factors of the Turkevich method that influence AuNPs size: initial $HAuCl_4$ concentrations, the molar ratio of $HAuCl_4$ to Na_3Ct, and temperature. The full factorial design of the experiment is shown in Table 1, with levels chosen based on the literature [6].

Table 1: Full factorial design of experiment for gold nanoparticles synthesis.

Factor	Levels
C (mM)	0.20, 0.25, 0.50, 0.75
n_{ratio}	1.00, 1.25, 1.50, 2.00, 2.40, 2.80, 3.20
T (°C)	60, 70, 80, 100

C is initial concentration, n_{ratio} is molar ratio, and T is temperature.

To perform experiments with the Opentrons Flex liquid handling robot, the Python protocol specified a sequence of liquid transfer steps. First, $HAuCl_4$ and diluent were transferred from the solvent reservoirs to the reaction plate using an eight-channel pipette. After filling the reaction plate with diluted $HAuCl_4$, the Python protocol instructed the robot to move the plate onto the Heater-Shaker Module with a gripper. The protocol specified the temperature and shaking speed. After heating, the reaction plate was moved back to the initial position. Then, the heated $HAuCl_4$ mixture was supplemented with different volumes of Na_3Ct and heated again with the Heater-Shaker Module. Note that each run of the protocol tested only one temperature, so to assess all four temperatures specified in Table 1, three additional runs of the protocol with different temperature values were conducted. Finally, the product was transferred from the reaction plate to a 360 uL Corning 96-well plate.

By transferring precise volumes of various reagents, the reaction plate was filled with reagents in the configuration shown in Figure 1. In this setup, each well in columns 1 through 4 represents a unique combination of $HAuCl_4$ concentration and the $HAuCl_4$ to Na_3Ct molar ratio; these combinations were repeated in triplicate across columns 5-8 and 9-12. Wells in row H were left empty, except for wells H1 through H4, which were filled with water to serve as blanks for UV-Vis analysis.

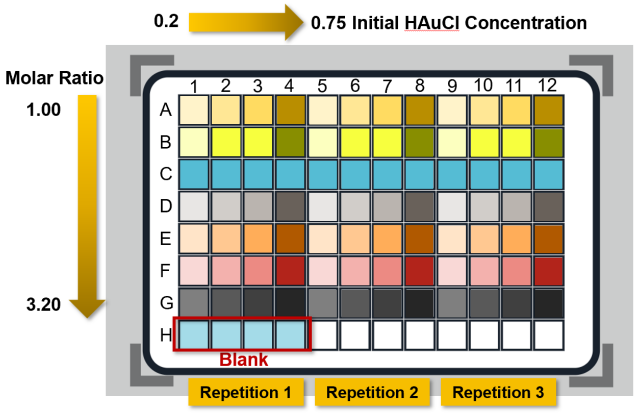

Figure 1. Layout of the conditions in AuNPs synthesis reaction plate.

3.0 SIMULATING THE PYTHON PROTOCOL

3.1 Basic Simulator Functions

Before rendering animation, the simulator first converts the Python protocol into a text file using the Opentrons-simulate package. The Opentrons-simulate package allows users to preview the steps taken by the robot in text form, as demonstrated below:

```
Latching labware on Heater-Shaker
Picking up tip from G1 of Opentrons Flex 96
Filter Tip Rack 1000 µL on slot B1
Aspirating 40.0 uL from A1 of NEST 12 Well
Reservoir 15 mL on slot D3 at 716.0 uL/sec
```

```
Dispensing 40.0 uL into G1 of NEST 96 Deep
Well Plate 2mL on slot D2 at 716.0 uL/sec
```

The simulator then works with this text to generate the animation. Essentially, the simulator employs web technologies for the graphical interface. JavaScript was used to drive the real-time animation of the robotic process, handling the dynamic updates to the deck layout. Cascading Style Sheets were used to style both the static and dynamic elements of the simulator's user interface, ensuring that the visual representation of the Opentrons Flex deck, labware (plates, tips), and pipette modules was accurately sized, positioned, and clearly visible, contributing to a realistic and easily understood user experience.

3.2 Visualising Liquid Aspiration and Dispense

The simulator simulates liquid being aspirated from or dispensed into a well by changing the height of the rectangle that represents the liquid level. At startup, the simulator initialises each well's current volume as either empty (0 uL) or as the maximum liquid volume a well can hold. When rendering the animation, every aspirate or dispense command triggers an update routine that changes the current volume of the affected well by subtracting or adding the transferred volume, respectively. The visual representation of liquid filled is derived from a level fraction (L_f) defined as:

$$L_f = \frac{V_{current}}{V_{capacity}} \quad (1)$$

where $V_{current}$ is the current well volume and $V_{capacity}$ is the total liquid volume each labware well could hold. For rectangular reservoirs, this fraction determines the height h_{fill} of a bottom-anchored filled rectangle, computed as:

$$h_{fill} = L_f \bullet H_{well} \quad (2)$$

$$Y_{top} = Y_{well} + (1 - L_f) \bullet H_{well} \quad (3)$$

with H_{well} being the constant pixel height of the well. The top Y-coordinate of the filled region is therefore given by ensuring proportional visual depletion or filling as volume changes. A sequence of frames where the coloured rectangle height decreases as $HAuCl_4$ solution was aspirated from the reservoir is shown in Figure 2.

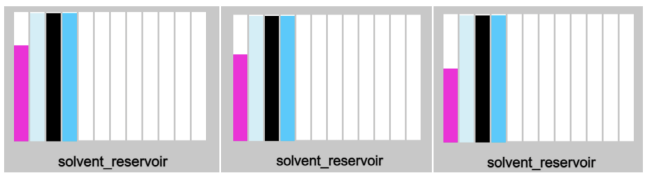

Figure 2. Animation of reduction in liquid height as 120 uL was being aspirated from solvent reservoir well A1.

3.3 Visualising Dilution and Liquid Mix

Colour-coding is used to indicate the presence of liquid within a well. For example, the well plates were initialised as empty and therefore appeared uncoloured. Conversely, the reservoirs were initialised as full and displayed in blue, reflecting the standard practice of pre-filling reservoirs with solvents or solutions at the start of a protocol. The simulator also initialised all liquid colours as blue, as shown in Figure 3(a).

Although the simulator has a default setting for labware, it also allowed this initial state of the labware to be customised. As shown on the right-hand side of Figure 3(b), for the AuNP protocol, the $HAuCl_4$ solution was set as bright pink, the diluent was set as dark blue, Na_3Ct was set as black, and water remained blue. Other wells were selected as "empty", so the wells were uncoloured. While the diluent in solvent reservoir A2 was set as dark blue in Figure 3(b), there was a difference in colour between the colour set in the right panel and the colour being shown in the animation on the left. This was because solvent reservoir A2 was set to be a "diluent", so the simulator replaced the diluent with light blue instead.

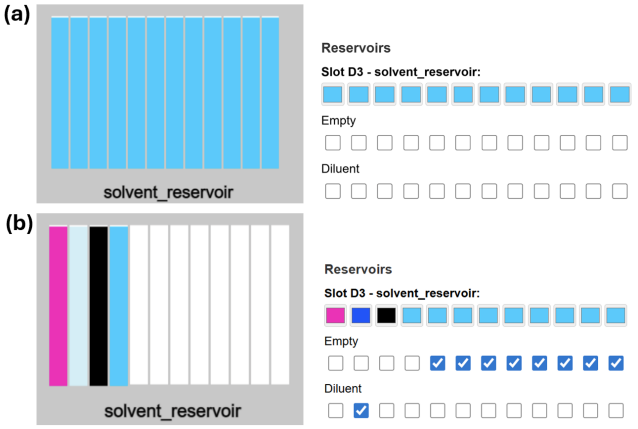

Figure 3. (a) Default setting of reservoir and (b) user setting of reservoir.

When $HAuCl_4$ was transferred to the reaction plate, the wells in the reaction plate were coloured bright pink to indicate the presence of $HAuCl_4$. Upon the addition of a diluent, the simulator calculated the ratio of non-diluent to current liquid volume in the well using Equation 4 and updated the opacity of the well's colour accordingly:

$$\kappa_{final} \propto \frac{V_{non-diluent}}{V_{total}} \quad (4)$$

where V_{total} is the total liquid volume in the well, $V_{non-diluent}$ is the volume of non-diluent liquid in the well, and K_{final} is the final opacity.

As a result, when diluent was added to $HAuCl_4$, the colour changed across the column and repeated every four columns. When Na_3Ct was added to the wells, the volume of Na_3Ct added to each row was the same, resulting in uniform colour across each row. However, the

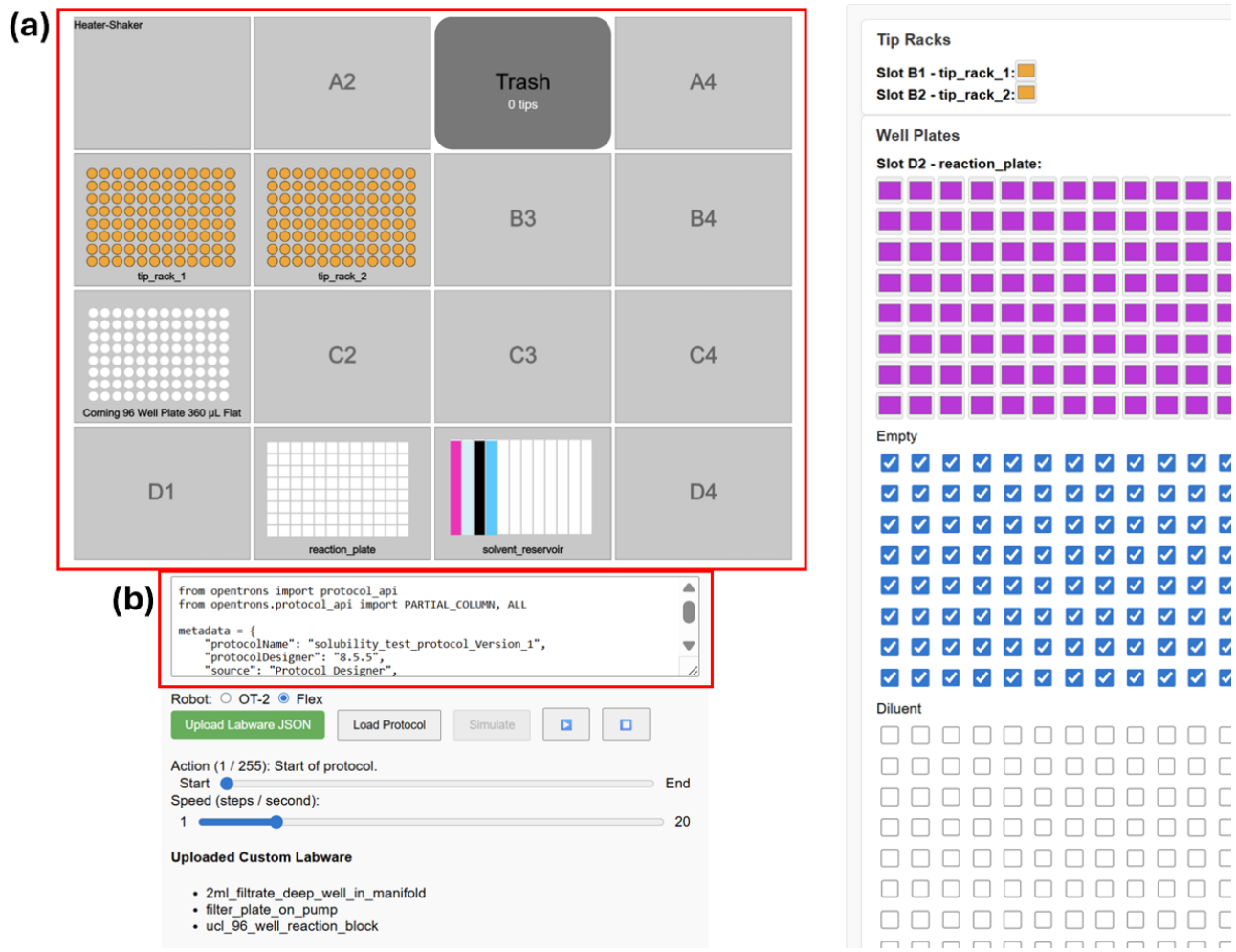

Figure 4. The simulator's user interface consists of several key sections: (a) the animation area and (b) the code editor where the Python protocol for AuNP synthesis is uploaded. Below the code editor, users can select the robot model (OT-2 or Flex) and use a dedicated button to upload JSON files for custom labware. Clicking 'Load Protocol' triggers the system to either display an error message or signal that the 'simulation is ready'. In the latter case, the 'Start Simulation' button becomes active. Adjacent to this are the pause and restart controls for the animation. The execution timeline is displayed under the 'Action' panel, which includes a slider to adjust the animation speed. Finally, the right-hand panel specifies the initial state of the labware based on three parameters: colour, fill status (empty), and diluent type.

volume varied between rows, producing differences in colour down the plate. This difference was captured by the simulator through colour blending. When dispensing liquid that is not labelled as a diluent, the simulator took the red, green, and blue channel intensity (RGB) of the Na_3Ct and the RGB of the previously diluted $HAuCl_4$ and averaged them based on their respective volume using Equation 5:

$$R_{final} = \frac{R_{source} V_{source} + R_{dest} V_{dest}}{V_{total}} \quad (5)$$

where R_{final} is the resulting RGB of the new mixture, R_{source} is the RGB of the liquid being poured in, V_{source} is the volume of the liquid being poured in, R_{dest} is the RGB of the liquid already present in the destination container, and V_{dest} is the volume of the liquid already present in the destination container.

Figure 4 displays the entire simulator's user interface. When a user uploads a Python protocol for AuNP synthesis into the code editor, the simulation deck (Figure 4(a)) will update to show each piece of labware at the position specified in the Python protocol.

3.4 Simulating Gripper

The function of the gripper was to move the reaction plate from its starting slot D2 onto the Heater-Shaker

Module when heating was needed. The simulator simulated this movement by detecting command lines that instructed the robot to move labware, such as:

```
Moving nest_96_wellplate_2ml_deep to Heater-
ShakerContext at Heater-Shaker Module GEN1
on A1 lw None
```

Once a movement command was detected, the system converted this movement of labware into a step of reassigning the labware from its current slot to the new slot. When this step was parsed to the function that ran the animation. Because the function rendered labware based on their assigned slot positions in the dictionary, the labware appeared to move instantly to the new location on the screen as the slot position in the dictionary changed as illustrated in Figure 5. This allowed visualisation of the reaction plate's movement to any position on the deck, including on top of the Heater-Shaker Module.

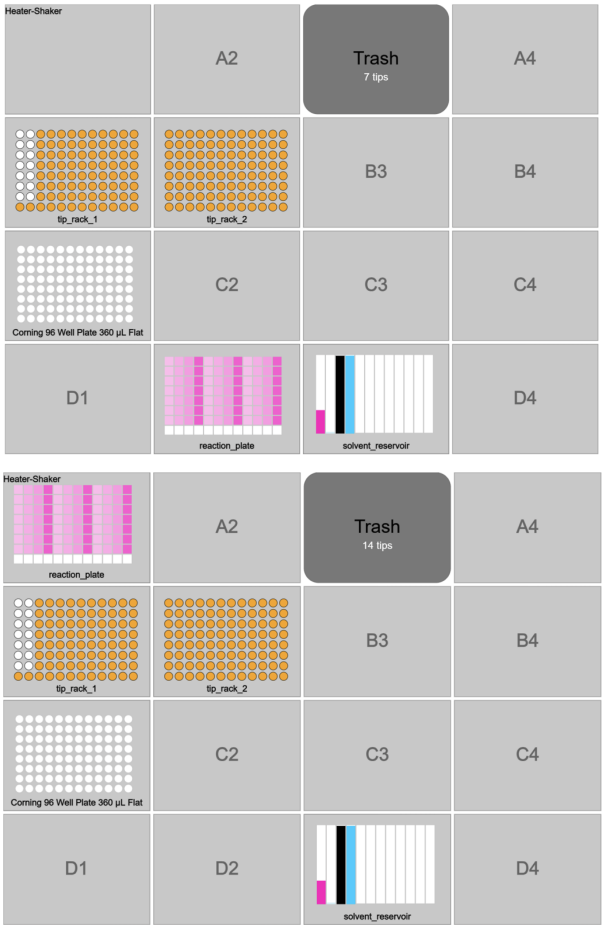

Figure 5. Gripper moving reaction plate from D2 (top) to the Heater-Shaker Module in A1 (bottom).

3.5 Simulating Heater-Shaker Module

In the Python protocol for AuNPs, the Opentrons Heater-Shaker Module was used for heating and mixing the diluted $HAuCl_4$ and $HAuCl_4/Na_3Ct$ mixture. The Heater-Shaker has two parameters: temperature and shaking speed. In the Python protocol, the parameters were set using the following code:

```
hs_mod.set_and_wait_for_shake_speed(300)
hs_mod.set_and_wait_for_temperature(60)
protocol.delay(minutes=10)
```

The simulator recognised this instruction from the Python protocol to extract the Heater-Shaker temperature and speed settings. It then broke them into a sequence of smaller, increasing values so that the animation showed the temperature and shaking speed rising to the target values (Figure 6). The animation started at a low baseline (23 °C for temperature or 0 RPM for the shaker), pushed intermediate values that stepped up slowly at first and faster later (small increments under a threshold, larger ones above), and then finally reached the exact target value.

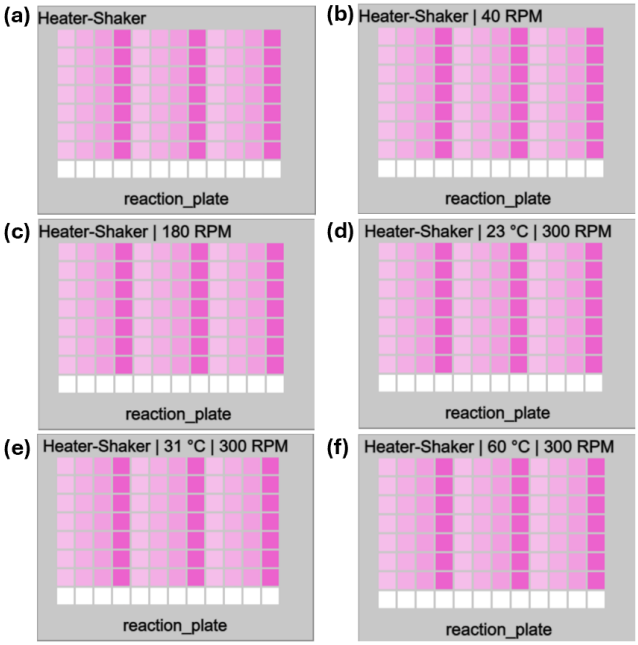

Figure 6. Chronological demonstration of heating and shaking process for the heater-shaker module during animation from (a) to (f).

After both temperature and shaking speed reached their target values, according to the Python protocol, the protocol would pause for 10 minutes. The simulation does not pause for 10 minutes. Instead, a pop-up window informs the user that the protocol was delaying for 10 minutes and 0.0 seconds. Upon clicking the "Ok" button, the simulator skips the waiting period and proceeds with execution. Overall, these features enable the simulation of the Heater-Shaker module.

4.0 CAPTURING COMMON MISTAKES WITH SIMULATOR

The simulator captured several mistakes made in the Python protocol for AuNP synthesis without running it on a physical robot. The first mistake was transferring liquid to the wrong well. A typo in the Python protocol, such as “A1” instead of “A2”, was hard to capture by human eyes reading through each line of code, especially in nanoparticle material synthesis where the code and simulated text could be complex and long. Although this mistake could be captured with a dry run, a single dry run took around 15 minutes, and one would have to perform several dry runs if there were multiple typos. The simulator speeds up this trial-and-error process by capturing unintended liquid transfers on the screen. In the classroom, students would be able to capture multiple errors in one simulation and change their code accordingly. This is much faster than testing the code on the robot.

The second mistake was transferring more liquid from the solvent reservoir to the reaction plate than the maximum amount of liquid a reservoir well could hold. Keeping track of how much liquid was aspirated from the reservoir by reading through lines of code was challenging; thus, one common mistake was designing a Python protocol that tried to aspirate more liquid than the maximum volume. During AuNPs synthesis, drying up the reservoir well could cause inaccurate transfer of liquid into the reaction plate and inevitably mess up the molar ratio or concentration of reactants. Note that this mistake could not be captured with a physical dry run or the Opentrons-simulate package, as both do not keep track of how much liquid was aspirated from a reservoir well.

The simulator kept track of the volume in each well and showed changes in liquid level by calculating how much liquid was left in the well. Building on this basic function, we added a feature to stop the simulation when the liquid level falls below 0 uL after aspiration. This would allow students to notice that more diluent than the maximum well volume was drawn by the Python protocol and fix their code accordingly before deploying it on the robot.

5.0 USING THE SIMULATOR IN CLASSROOM AND WIDER EDUCATIONAL IMPACT

The simulator enables a structured, multi-stage pedagogical workflow that integrates digital laboratory automation into teaching and training activities. In the initial phase, students receive instruction on the manual synthesis of AuNPs, including discussion of key experimental variables such as reaction temperature, initial $HAuCl_4$ concentration, and the molar ratio between $HAuCl_4$ and Na_3Ct. Building on this foundational knowledge, each student then independently develops a Python protocol on their personal laptop and evaluates its performance using the simulator. This allows students to iteratively test, debug, and refine their protocols in a low-risk virtual environment prior to working with the physical robotic system.

The iterative cycle—develop, simulate, evaluate, and improve—not only strengthens computational thinking skills but also creates an accessible and scalable approach to teaching laboratory automation. The simulator additionally provides quantitative feedback not available in existing tools (e.g., tracking pipette tip usage), enabling assessment based on efficiency, sustainability, and resource optimisation.

Once students complete the simulation stage, they implement their protocols on the Opentrons Flex robot. Small-group review and peer critique of protocols foster collaborative problem-solving and mirror engineering practices. This supports teamwork while preserving individual accountability. In settings without access to robotic hardware, the simulator supports fully remote learning, with students submitting both their protocols and screen-recorded simulations for assessment.

5.1 Extended Educational Impact Across Programmes

The impact of the simulation tool extends beyond the immediate classroom environment. Postgraduate researchers within the School of Pharmacy and chemistry have adopted the simulator to upskill new starters whose research involves the use of liquid-handling robots with similar operational principles. By engaging with the simulator prior to accessing physical instruments, these students report increased confidence, reduced anxiety, and improved preparedness for hands-on experimentation. This has proven particularly valuable for students transitioning into laboratory automation with limited prior programming experience. The simulator has been used to validate developed code deployed for applications in combinatorial chemistry, solvent screening, and the assessment of solubility for various chemicals in selected solvents.

Furthermore, the simulator has now been fully integrated into the Digital Manufacturing of Advanced Materials (DMAM) MSc programme, where it supports training in automation-ready digital workflows. Its inclusion strengthens the programme’s emphasis on computational methods, digital manufacturing, and reproducible laboratory practices. Early feedback indicates that students benefit substantially from the ability to repeatedly practise and refine automated protocols outside of scheduled laboratory hours.

There is also ongoing discussion with the School of Pharmacy to incorporate the simulator more formally into postgraduate-taught training. This integration would extend its impact to a broader cohort of students, embedding digital laboratory automation into curricula where traditionally wet-lab skills dominate. The simulator’s

alignment with digital skills agendas and its accessibility on standard laptops make it an attractive teaching tool for programmes seeking to modernise laboratory education.

5.2 Industrial Upskilling and Professional Training

Beyond academic environments, the simulator has also been identified as a valuable tool for upskilling industrial workers. In a recent funding bid submitted by the Institute of Robotics, the simulator was highlighted as a resource capable of supporting workforce development in automated laboratory processes. Its ability to train users remotely, safely, and without access to expensive hardware makes it suitable for industrial training pipelines, particularly in sectors where laboratory automation is expanding rapidly.

The tool thus supports a continuum of training—from taught postgraduate students to incoming researchers, to professionals requiring automation-related reskilling. Its scalability positions it as an enabling technology for modern laboratory education and industrial digital transformation.

5.3 Advancing Research Reproducibility and Digital-First Laboratory Practices

The simulator contributes to ongoing efforts to improve research reproducibility. Challenges in experimental reproducibility increasingly arise from undocumented implicit knowledge, variability in manual procedures, and inconsistent reporting of protocol parameters. By requiring users to encode all steps in a Python-based, machine-readable protocol, the simulator encourages a level of procedural precision difficult to achieve through traditional textual descriptions alone. The ability to directly execute these digital protocols on automated liquid-handling platforms further ensures that methods can be reproduced with high fidelity across laboratories, institutions, and time.

The simulator also supports the adoption of digital-first laboratory practices that are central to emerging frameworks and experimental workflows. In environments where liquid-handling robotics are increasingly integrated with electronic lab notebooks and laboratory information management systems, the simulator can act as a bridge between conceptual training and real-world digital infrastructure.

Moreover, the quantitative performance data generated by the simulator—such as consumable usage, timing estimates, and workflow efficiencies—enable users to evaluate protocols using metrics that are rarely captured in traditional laboratory teaching. These data create opportunities for optimisation-driven teaching and research, where experimental design can be informed by resource efficiency, sustainability considerations, and throughput modelling. Incorporating such metrics into training better prepares learners to operate within modern research laboratories, where cost-aware and sustainability-oriented decision-making is increasingly prioritised.

6.0 CONCLUSION

This study presented the development of a 2D desktop-based simulation tool designed to enhance robotics and automation education in chemical engineering. By translating Python-based Opentrons protocols into real-time animations, the simulator effectively addresses the pedagogical barriers of limited hardware access and uneven group dynamics. Using gold nanoparticles (AuNPs) synthesis as a case study, we demonstrated the platform's ability to visualise complex liquid-handling steps, track consumable usage, and capture critical programming errors (such as volume deficits) that dry runs overlook.

This work serves as a blueprint for other UCL modules seeking to bridge the gap between theoretical coding and physical implementation. Programmes in Biochemical Engineering, material science or computer science can look to this model to provide equitable access to high-cost hardware. By decoupling code development from hardware availability, modules can ensure that students with less prior programming experience are not marginalised by "dominant" peers in a group setting. It demonstrates that digital competency is best fostered when students have the autonomy to fail and iterate in a low-stakes, individual digital environment before moving to the physical laboratory.

For UCL programmes aiming to embed the UN Sustainable Development Goals (SDGs) into their curricula, this manuscript illustrates how "Digital Twins" can directly reduce the carbon footprint of laboratory education in terms of minimising waste. Other modules involving high resource consumption, such as chemistry or synthetic biology, can adopt similar simulation-based assessments in which resource efficiency is used as a graded learning objective. By providing quantitative feedback on consumable usage (e.g., plastic tips and chemical reagents) during the simulation phase, departments can train a new generation of engineers who view optimisation and waste minimisation as core components of experimental design rather than afterthoughts.

Future research will focus on integrating more visualisation features to provide a more immersive experience and expanding the library of supported labware and modules to cover a broader range of high-throughput experimentation (HTE) workflows. Additionally, we aim to implement an automated feedback system that provides students with real-time suggestions for optimising reagent efficiency and protocol duration. In the context of

HTE, this simulator serves as a scalable digital twin, enabling researchers and students alike to validate complex, data-driven experimental designs before physical deployment, thereby reducing waste and accelerating material discovery.

ACKNOWLEDGEMENTS

The authors would like to thank University College London (UCL) and the Department of Chemical Engineering for providing the laboratory facilities, the robotic hardware (Opentrons Flex), and computational resources necessary for the development and testing of this simulation platform. We also acknowledge the MSc Digital Manufacturing of Advanced Materials (DMAM) cohort for their participation and feedback during the pilot implementation of this tool.

REFERENCES

1. Akundi A, Euresti D, Luna S, Ankobiah W, Lopes A, Edinbarough I. State of industry 5.0—analysis and identification of current research trends. ASI 5:27 (2022). https://doi.org/10.3390/asi5010027
2. Lu JM, Pan JZ, Mo YM, Fang Q. Automated intelligent platforms for high?throughput chemical synthesis. Artificial Intelligence Chemistry 2:100057 (2024). https://doi.org/10.1016/j.aichem.2024.100057
3. Vu H, Bui KH, Dang KDD, Duong-Tuan M, Le DD, Nguyen-Dang T. Finding environmental-friendly chemical synthesis with AI and high-throughput robotics. Journal of Science: Advanced Materials and Devices 10:100818 (2025). https://doi.org/10.1016/j.jsamd.2024.100818
4. Florian L, Linklater H. Preparing teachers for inclusive education: using inclusive pedagogy to enhance teaching and learning for all. Cambridge Journal of Education 40:369-386 (2010). https://doi.org/10.1080/0305764x.2010.526588
5. European Commission. (2024). Advanced Materials for Industrial Leadership. https://research-and-innovation.ec.europa.eu/research-area/industrial-research-and-innovation/chemicals-and-advanced-materials/advanced-materials-industrial-leadership_en
6. Dong J, Carpinone PL, Pyrgiotakis G, Demokritou P, Moudgil BM. Synthesis of precision gold nanoparticles using turkevich method. KONA 37:224-232 (2020). https://doi.org/10.14356/kona.2020011