ROYAL SOCIETY OPEN SCIENCE

royalsocietypublishing.org/journal/rsos

Research

Cite this article: McDermott S, Kotar J, Collins J, Mancini L, Bowman R, Cicuta P. 2024 Using old laboratory equipment with modern Web-of-Things standards: a smart laboratory with LabThings Retro. *R. Soc. Open Sci.* **11**: 240634.
https://doi.org/10.1098/rsos.240634

Received: 16 April 2024
Accepted: 27 June 2024

Subject Category:
Physics and biophysics

Subject Areas:
biophysics

Keywords:
laboratory automation, analytical instrumentation, smart switches

Author for correspondence:
Samuel McDermott
e-mail: sjm263@cam.ac.uk

Using old laboratory equipment with modern Web-of-Things standards: a smart laboratory with LabThings Retro

Samuel McDermott[1], Jurij Kotar[1], Joel Collins[2], Leonardo Mancini[1], Richard Bowman[2] and Pietro Cicuta[1]

[1]Cavendish Laboratory, University of Cambridge, Cambridge, UK
[2]Department of Physics, University of Bath, Bath, UK

SMD, 0000-0003-2736-5467; JK, 0000-0001-6246-5178; JC, 0000-0002-9382-7511; LM, 0000-0001-8906-1667; RB, 0000-0002-1531-8199; PC, 0000-0002-9193-8496

There has been an increasing, and welcome, open hardware trend towards science teams building and sharing their designs for new instruments. These devices, often built upon low-cost microprocessors and microcontrollers, can be readily connected to enable complex, automated and smart experiments. When designed to use open communication web standards, devices from different laboratories and manufacturers can be controlled using a single protocol and even communicate with each other. However, science labs still have a majority of old, perfectly functional equipment which tends to use older, and sometimes proprietary, standards for communications. In order to encourage the continued and integrated use of this equipment in modern automated experiments, we develop and demonstrate LabThings Retro. This allows us to retrofit old instruments to use modern Web-of-Things standards, which we demonstrate with closed-loop feedback involving an optical microscope, digital imaging and fluid pumping.

1. Introduction

Future discoveries in a wide context of biological experiments will depend on increasingly 'smart' laboratories. Much biological research relies on experiments aiming to understand causal relationships in systems with very large parameter spaces to

THE ROYAL SOCIETY PUBLISHING

explore and sources of complication that require high throughput and robust data. Exploring large parameter spaces while maintaining reproducibility often becomes a challenge for the human operator.

As the tasks of such an experiment are repeated over long periods of time, they can be tedious, prone to human error and not straightforward to standardize and replicate across labs. Currently, human operators are often necessary in experimental pipelines to identify events of interest and then trigger corresponding actions on equipment. This tends to discourage explorations of systems that do not fit well with human timescales, e.g. it is difficult for humans to intervene in processes that are either too fast or too slow, or where the trigger signal needs to be filtered from noise or from multiple inputs. The only feasible way to obtain data for these complex and 'unfriendly' systems is through experimental pipelines that exploit automation and allow feedback loops [1].

Many new laboratory devices and software, in particular open-source hardware approaches, are reflecting this paradigm shift. Equipment such as the liquid handling robot Opentrons [2] and measurement devices such as OpenFlexure [3,4] and UC2 [5] can now perform reproducible automation protocols [6]. Large datasets can now be analysed much more efficiently using software such as Napari [7], CellProfiler [8] and ImJoy [9]. By combining these technologies, it is possible to develop custom and affordable 'smart' laboratories where feedback loops can change the experiment based on previously acquired data.

However, many laboratories have old equipment. Although mechanically and electrically working, these do not have the modern communication interfaces required for any sort of smart experiment. Some contemporary equipment also does not come with a communication interface. We can think of three categories for these devices:

(i) *No external communications at all*. Adding feedback and automation for devices that do not normally have advanced controls. For example, units that might just be turned on or off with a switch, e.g. hotplates, rocking tables, lamps and ultrasonic baths.
(ii) *Controlled externally, using open (documented) communication protocols*. These more advanced devices typically use serial protocols and connectors such as USB or RS-232.
(iii) *Controlled externally, using proprietary and closed protocols*. These advanced devices use closed-source drivers and undocumented software meaning that it is difficult to extract them from the manufacturer's ecosystem.

We show how devices that are in categories (i) and (ii), and, depending on hardware constraints, some in category (iii) can be retrofitted so that they can be integrated into smart experiments. Our demonstration uses low-cost, open-source hardware and software.

In the case of category (iii), tools like PyAutoGUI [10] have been shown as useful, if crude, workarounds by simulating interactions with proprietary graphical user interfaces (GUIs) [11]. It may be possible for companies to 'unlock' access to their devices for a fee, but this creates uneven access to automation. Otherwise, one has to 'reverse engineer' the communication protocol, which is not something we describe here.

Unfortunately, in some cases, the limitations of proprietary systems are truly encrypted and/or linked to proprietary hardware acquisition cards, and this means that it is not in general straightforward to integrate such devices into smart laboratory ecosystems.

In this article, we describe the *LabThings Retro* controller, our solution to integrating and retrofitting devices in categories (i) and (ii) above, to allow automated smart experiments. A possible future implementation of a smart lab using this design is shown in figure 1: one computer (which could be in a separate building to the experiments), using a client written in the language of the user's choosing, is able to communicate with both new (top) and old devices (bottom) using the established Web-of-Things (WoT) standard. Communication with older devices is made possible using the LabThings Retro controller.

This work demonstrates how it is possible to integrate older devices into modern smart lab experiments, enabling laboratories globally to re-use equipment, save money, prevent e-waste, standardize protocols and empower researchers to tackle previously daunting experimental challenges, potentially unlocking knowledge that is only accessible via automated or high-throughput approaches.

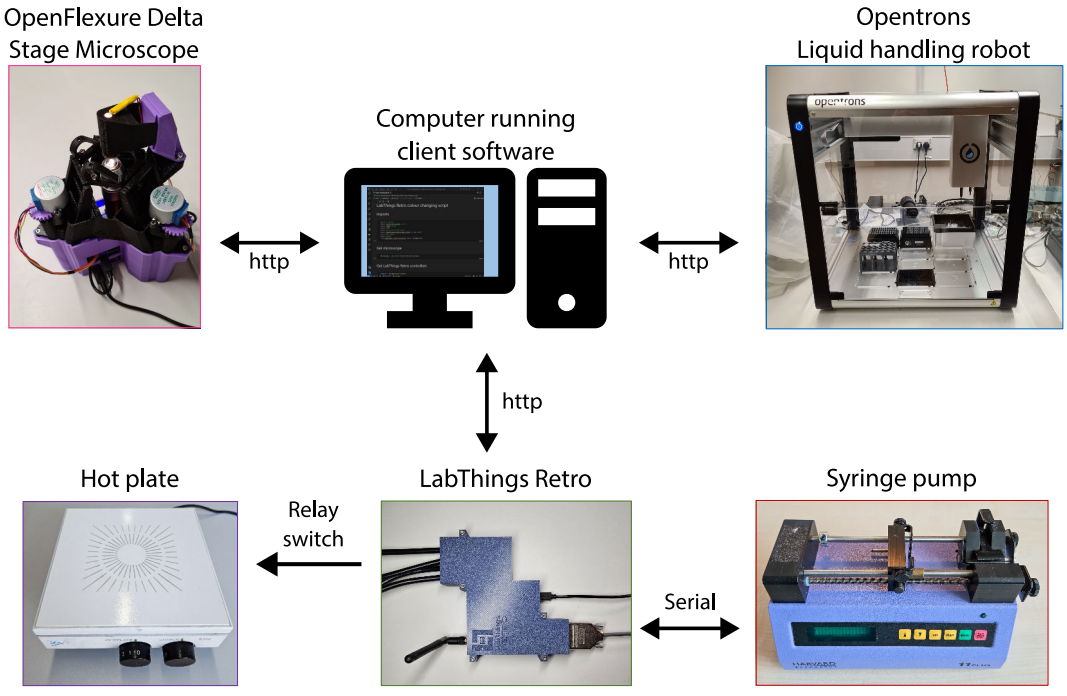

Figure 1. An example of our future ambition of a smart lab for automated experiments enabled with LabThings Retro. A computer running client software is able to communicate with a range of scientific instruments with modern communications standards. In addition to issuing commands, it can receive data from the instruments and make decisions about how to progress the experiment. LabThings Retro enables older, but still functioning, equipment to be integrated into this modern experimental set-up.

2. Design requirements

The purpose of the LabThings Retro controller is to enable the use of older laboratory devices with new web communication standards. Our objectives are the following:

- Develop a simple controller, which can be easily replicated. It should be possible to manufacture using three-dimensional (3D) printers and to be assembled using hand tools, following an easy-to-use manual.
- The controller, including electronics, should be available to manufacture at a low cost and be open source.
- The controller should be able to connect to a network for remote control.
- The controller should be able to control devices that can be turned on and off with a switch, e.g. light sources, stirrers or heaters.
- The controller should be able to send and receive more complex commands and data over a serial connection to e.g. syringe pumps, incubators or pH meters.

3. Hardware design

The hardware was designed to be easy to replicate and modify, and is shown in figure 2. The example we provide works with any category (i) device as defined in the introduction and can be adapted to work with any category (ii) device that is controlled using serial communication protocols.

3.1. Electronics

3.1.1. Microcontroller board

For the microcontroller board, we evaluated ESP32-based boards. This is because the ESP32 is a well-documented and versatile microcontroller with a good developer ecosystem around it. It also has built-in WiFi and sufficient CPU and memory. In addition, a LabThings library already exists for this architecture [12], based on the WebThingsIO webthing-arduino library [13]. Alternative

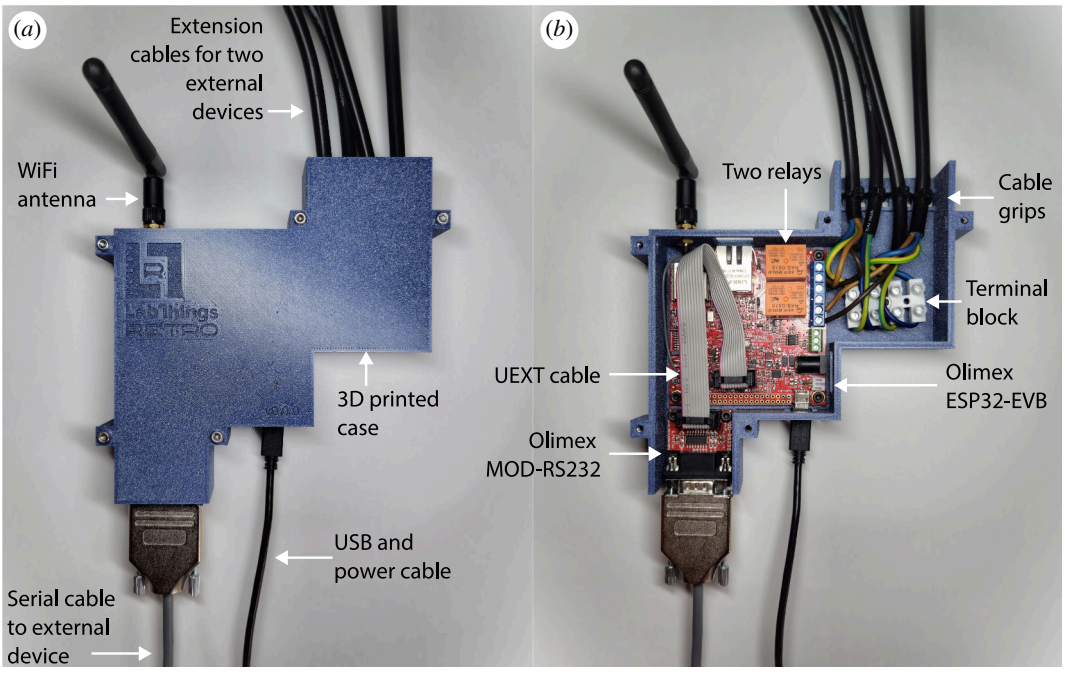

Figure 2. LabThings Retro hardware design. (*a*) The controller with the lid on, showing the external connections. (*b*) The controller with the lid off, showing the internal components.

microcontroller options we considered include Adafruit FeatherWings, Arduino, Phidgets, Tinkerforge, Seeed Studio's Grove and Raspberry Pi Pico.

There are several boards built for the ESP32. We wanted a board, which had relays to turn laboratory equipment on and off. In addition, the board should be able to communicate with common serial protocols, such as RS232. Finally, we looked for a board that was open source, to enable better documentation, community support and longevity of the controller. Therefore, we decided on the Olimex ESP32-EVB [14]. This ESP32-based board has a number of connection options including WiFi, Bluetooth Low Energy, and an ethernet port. It has two 10 A/250 VAC relays to switch externally powered devices, can connect to Olimex's range of UEXT modules (such as serial interfaces) via a UEXT connector and is certified as OSHWA Open Source Hardware with UID BG000011 [15].

As all this functionality is on one board, it provides many future opportunities for development without changing the hardware. In addition, with the high-quality documentation and helpful community of support, it is possible to design firmware for the board using the C implementation of LabThings on the ESP32.

3.1.2. Serial communication interface

To control external serial devices, we used the Olimex MOD-RS232 [16]. This module connects to the ESP32-EVB microcontroller board with UEXT. It has an RS232 level-shifter to enable serial communications from the ESP32 to serial devices with a DB-9 connector operating at the standard 5 V level. It is also possible to use the Olimex MOD-RS485 for communication with RS485 and RS422 serial devices [17].

3.1.3. Electronic relays

Many devices in the lab are category (i) devices (as defined in the introduction) and just need to be turned on or off, for example, lamps, heaters or shakers. The Olimex ESP32-EVB has two 10 A/250 VAC relays, which can power on or off two connected devices. We recommend that rewiring mains power for use with the relays should be carried out by a certified electrician in discussion with their laboratory's safety department.

3.2. Three-dimensional printing

To contain the electronics and make the relay circuit safe, the controller is contained within a 3D-printed enclosure. The two-part case can be printed with simple filament deposition 3D printers using standard materials. If using the relay circuits, we recommend using a UL94V-0 flame retardant ABS. The enclosure is designed using OpenSCAD and can be easily adapted to other configurations. The files are available on the GitLab repository [18].

3.3. Assembly

An important aspect of the design is the ease of assembly. A thorough set of assembly instructions, powered by gitbuilding [19], contains photos and a bill of materials. The controller can be assembled using simple hand tools and common hardware. The assembly instructions are available on the GitLab website [20].

4. Software design

We have developed example software for using the controller to control the relays and send commands over the serial interface. For an exemplar serial device, we chose a Harvard Apparatus *11 Plus syringe pump*. This is a category (ii) device, which can be controlled using the RS-232 serial protocol. Syringe pumps are used for many biological experiments, for example, to perfuse liquids in cell culture or to control flow through microfluidics. They are long-lasting and typically reliable pieces of laboratory equipment, so older syringe pumps typically remain functional for use in experiments. However, as they use older serial communication protocols, it is difficult to control them with modern computers. For example, most modern computers do not have the required serial ports or drivers, and the software for the devices is old and may not work with current operating systems.

4.1. LabThings

In order to present a useful and intuitive network interface to hardware devices, we have built on several widely used Internet standards for application programming interfaces (APIs). Building on existing standards reduces duplication of effort and enables us to take advantage of a wealth of well-maintained and tested code with much larger user bases than most scientific software. The highest-level standard is the WoT standard managed by W3C [21]. This standard defines a 'Thing Description' document, which describes a 'Thing' in terms of actions, properties and events—each of which maps to defined network commands. Network communication uses the ubiquitous HyperText Transport Protocol (HTTP) to format commands and requests for information. The list of possible HTTP requests is documented in a standardized way using the OpenAPI standard [22]. OpenAPI provides a human- and machine-readable document that can be easily rendered into interactive documentation that not only describes each possible command but also allows the user to try them out in a web browser and see the results. This is a valuable tool for developing new code or debugging instruments. Code generation allows client libraries to be created automatically for most modern languages, based on the OpenAPI description. Automating the generation of client code makes the instrument easier to maintain and enables any language to be used to control it in a consistent manner. Code generation from Thing Description documents would be ideal for hardware control, as it gives a higher-level description of the device than the often numerous HTTP commands described in OpenAPI; however, fewer languages are currently supported.

The LabThings project aims to make use of both Thing Description and OpenAPI to improve the documentation and interoperability of laboratory hardware [23]. It was originally created for the OpenFlexure Microscope [24] but lends itself well to generalization. Currently, it provides a Python library that simplifies the task of exposing an HTTP API along with a Thing Description and an OpenAPI description, together with interactive documentation. As the API description is generated automatically based on documentation in the code, it is guaranteed to be consistent with the implementation—an approach usually termed 'self-documenting'. A Python client library is also provided, making it intuitive to control instruments complying with these standards in Python. It is important to note that the client code should be compatible with any HTTP-based hardware exposing

a Thing Description not just hardware using the `LabThings` Python library. There is also a LabThings implementation for ESP32 [12], written in C, and we have based LabThings Retro on this codebase.

4.2. Serial communication

When two devices communicate using serial communication protocols, each data bit is sent sequentially over a single data line. In contrast, in parallel communication, multiple bits are sent simultaneously over several wires. Serial links are more commonly used for device communications due to their simplicity and lower cost of implementation. Modern serial interfaces such as USB are well known due to their use in everyday electronic products. Older serial interfaces such as RS-232 and RS-485 are commonplace in industrial and scientific instruments due to their wide adoption and ease of implementation. However, such interfaces are rarely present on modern computers, and low-level adapters providing serial ports over USB suffer from difficulties around device discovery and consistent addressing.

Our exemplar device, the syringe pump, can be controlled remotely using serial communication. It uses the RS-232 interface and the manual describes the commands, queries and responses. These are sent as ASCII codes over the RS-232 connection. Each command consists of a string of ASCII characters. The first two characters are the address of the pump. This is included when you are using multiple pumps in a daisy chain, and the address of the pump is set on the pump itself. In our example, it is `00`. The next characters identify the command. For example, to start running the pump, we send `RUN`. The message is terminated with a carriage return character `\r`. The full char array `00RUN\r` is then sent to the pump. A prompt is returned from the pump, indicating its status, such as running or stopped. It is also possible to send commands with numbers, for example, the flow rate or the target infusion volume. To get information back from the pump, such as the current infused volume, it is possible to send queries.

Upgrading a serial device with LabThings to the WoT standard can be achieved in two ways:

- A network-to-serial converter, in which serial commands are sent to the LabThings Retro controller over HTTP and are then relayed to the device's communications port. Replies would then be returned to the client in the HTTP response. Operating in this mode has the advantage of immediately exposing all the device's capabilities to the client and using one firmware image to control a variety of different devices. However, it would push more complexity into the client code: the device's capabilities are not exposed in the HTTP API or the Thing Description, so the user would still need to look up command codes in the device manual. The low-level interface also means the device could behave oddly if multiple clients connect, and so many of the limitations of a serial connection still apply.
- A firmware image that exposes the device's commands as HTTP endpoints is documented as WoT Actions and Properties. This would require more effort to develop, but it is much easier to use: all capabilities of the device are documented in the API, so client code can be generated automatically and the user need not refer to the device's manual. Using a higher-level interface over the network also means that the LabThings Retro controller can ensure communications do not become confused—for example, ensuring that each command receives the correct response and preventing commands from being sent when the instrument is not ready to receive them. This confers a level of robustness to concurrent use by multiple clients, which can be helpful when monitoring an ongoing experiment.

Our exemplar syringe pump implementation uses the second of these approaches. As such, the LabThings Retro controller documents all the capabilities of the syringe pump in the API. When the LabThings Retro controller receives a WoT Action or Property request, it converts that request into the relevant serial command and sends it to the device. It then returns the result to the client.

4.3. Electronic relays

The state of each electronic relay is a Boolean property. When the server receives a WoT Property set request from the client, it updates the relay switch accordingly. The client can also get the current value of the property to determine whether the relay is on or off.

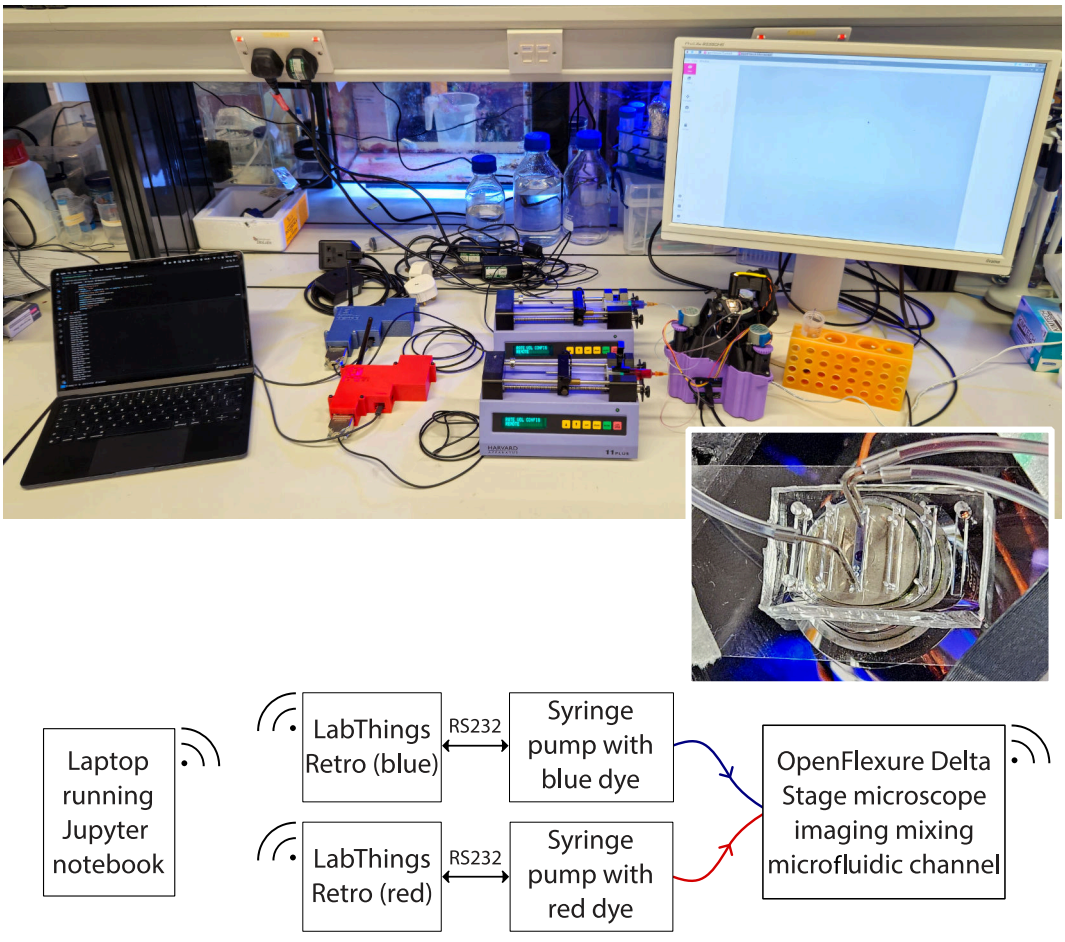

Figure 3. Experimental set-up for demonstrating a closed feedback control across various devices using LabThings Retro. A laptop running a Jupyter notebook is connected to two LabThings Retro controllers over WiFi. The laptop is also connected to an OpenFlexure Delta Stage microscope over WiFi. Each LabThings Retro controller is connected with a serial cable to a syringe pump, one with a syringe containing blue dye and the other holding a syringe with red dye. The insert shows the microfluidic channel where the dyes pass along two tubes and are mixed in a microfluidic channel. The microfluidic channel is imaged using the OpenFlexure Delta Stage microscope.

4.4. Installation

The full instructions for installation can be found in the assembly instructions [20]. To install and configure the software, we use PlatformIO [25]. The example server code can be downloaded from the GitLab repository [26]. Once downloaded, the LabThings library can also be installed. The user will need to add their WiFi hotspot service set identifier (SSID) and password to the code so that it can be controlled from other devices on the same network. For security and speed reasons, it is recommended that this WiFi hotspot be isolated from other devices and not connected to the outside Internet. The code can then be flashed to the Olimex ESP32-EVB, and the controller is ready to use.

4.5. Client

As discussed in §4.1, the advantage of controlling LabThings Retro controllers using the LabThings library is that it extends the WebThings Library. As this open standard is self-documenting, it is not necessary to code clients to work with individual devices. There are, therefore, a range of client libraries that have been designed to work on a variety of devices. It is also possible to create a library for any language that has HTTP support or to automatically generate one based on the OpenAPI description provided by the LabThings Retro server. This makes it easy for users to build scripts in their preferred language on the Internet-enabled device of their choice. For example, LabThings Retro controller scripts can be included within existing MATLAB or Python scripts.

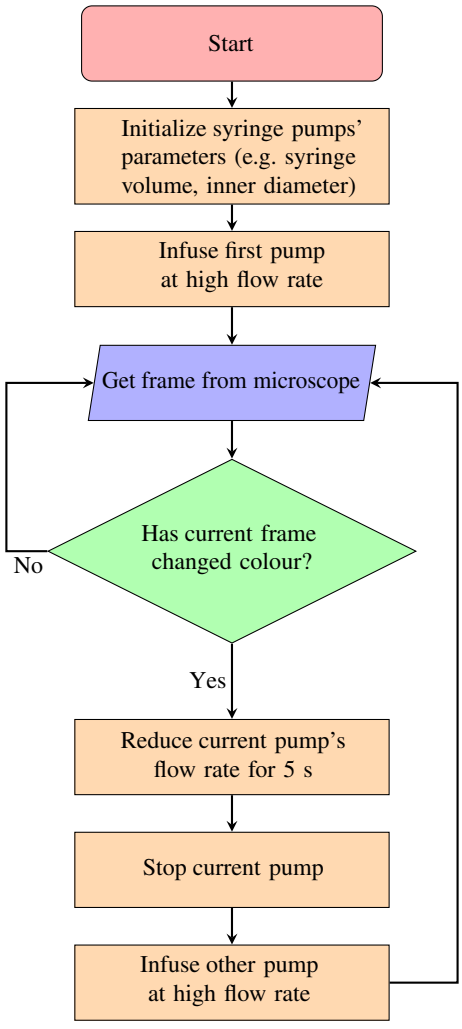

Figure 4. The algorithm flow for the automation case study. The result of this algorithm is that the microscope's field of view oscillates between red and blue. Once the algorithm detects that one pump has changed the colour in the microscope's field of view, it reduces the pump's flow rate for 5 s before turning off that pump and starting the other.

Scripts can be easily developed to control several controllers and devices simultaneously. Scripts are useful for automating complex experiments, such as parameter sweeps, where manual control of equipment would be overly time-consuming, and they are essential in more complex automation that aims to deploy feedback and respond to events in the experiment itself. Examples of client scripts and installation instructions are available in the GitLab repository [27].

It is also possible to control LabThings Retro controllers using GUIs. For example, WebThings Gateway [28] creates a web interface that can automatically control WebThings devices. As LabThings extends the WebThings Library, WebThings Gateway can control LabThings Retro controllers, as well as other open and commercial communication protocols.

5. Automation case study

To demonstrate the potential use of LabThings Retro, we designed the experiment shown in figure 3. This experiment combines two LabThings Retro controllers controlling two syringe pumps, and an OpenFlexure Delta Stage microscope [3]. All three devices are controlled remotely using a Python script [27]. This experiment demonstrates a smart experiment made possible using LabThings Retro, which creates an autonomous feedback loop between the new microscope and the older syringe pumps.

Two coloured dyes (one red and one blue) were diluted 1 : 10 in water. They were drawn into two 5 ml syringes and inserted into the two syringe pumps. The two syringes were connected to a polydimethylsiloxane (PDMS) microfluidic mixing channel, which was positioned on the microscope.

Each LabThings Retro controller was connected to its corresponding syringe pump with an RS232 serial cable. Although this brand of syringe pump can be addressed and daisy-chained to one serial connection, we used two LabThings Retro controllers to demonstrate how a generalized smart experiment can scale. The two LabThings Retro controllers were connected to the same WiFi network as the laptop and OpenFlexure Delta Stage. A simple Python script was written to run the automation. The algorithm flow is shown in figure 4, and a video of the experiment is in the electronic supplementary material.

This completely autonomous algorithm takes input from the microscope, makes a decision based on the contents of the frame and controls the syringe pumps accordingly. LabThings Retro changes the operational mode and settings of the syringe pumps during the experiment, for example, reducing the flow rate or running both pumps simultaneously at different rates. It could therefore easily be scaled up to control more equipment or use advanced computer vision techniques in order to develop a smart biological experiment.

6. Conclusion

In this article, we have demonstrated how it is possible to integrate older equipment into modern smart experiments with LabThings Retro. This will enable more laboratories to develop automated experiments. Our solution is low cost, easily modifiable and can be extended to a range of devices and experiment types.

LabThings Retro has the advantage that it is built on an established protocol for communicating with devices. There are already a range of clients and GUIs available for controlling devices based on WoT protocols. This means that it is not necessary for the control scripts to use the same framework that is used in the instrument code.

In future work, we will extend the networking capabilities of the controller by developing a LabThings ethernet adapter to use the ESP32-EVB's ethernet port. Ethernet is not susceptible to wireless signal interference from multiple WiFi devices and can therefore scale better for larger experiments. It will also provide an alternative interface for scientists who work in environments with poor or no WiFi availability. To assist with the adoption of this smart lab approach, the sharing of code for controlling particular instruments with the LabThings Retro framework would make it quicker to create experiments.

Ethics. This work did not require ethical approval from a human subject or animal welfare committee.

Data accessibility. In addition to the GitLab repositories cited in the article, a snapshot of the code has been uploaded to Zenodo [29] along with a video of the experiment [30].

Declaration of AI use. We have not used AI-assisted technologies in creating this article.

Authors' contributions. S.M. led the design of the devices and software and wrote the draft of the paper and assembly instructions. With L.M., S.M. prepared and carried out the demonstration experiment. J.K. and J.C. contributed to the device design and software. R.B. and P.C. contributed to the design of the study. All authors contributed to the final manuscript.

All authors gave final approval for publication and agreed to be held accountable for the work performed therein.

Conflict of interest declaration. At the time of writing, P.C. is a Board Member of Royal Society Open Science but had no involvement in the review or assessment of the paper.

Funding. S.M. was funded by an EPSRC IAA grant. J.K. and P.C. were funded by an Alborada Trust grant in Cambridge University. L.M. acknowledges funding from the Herchel Smith Postdoctoral Fellowship. R.B. is supported by a Royal Society award URF\R1\1 80 153.

References

1. Szymanski NJ *et al*. 2023 An autonomous laboratory for the accelerated synthesis of novel materials. *Nature* **624**, 86–91. (doi:10.1038/s41586-023-06734-w)
2. Opentrons. 2023 Opentrons lab automation lab robots for life scientists. See https://opentrons.com/.
3. McDermott S, Ayazi F, Collins J, Knapper J, Stirling J, Bowman R, Cicuta P. 2022 Multi-modal microscopy imaging with the Openflexure Delta Stage. *Opt. Express* **30**, 26 377–26 395. (doi:10.1364/OE.450211)
4. Collins JT *et al*. 2020 Robotic microscopy for everyone: the OpenFlexure Microscope. *Biomed. Opt. Express* **11**, 2447–2460. (doi:10.1364/BOE.385729)

5. Diederich B, Lachmann R, Carlstedt S, Marsikova B, Wang H, Uwurukundo X, Mosig AS, Heintzmann R. 2020 A versatile and customizable low-cost 3D-printed open standard for microscopic imaging. *Nat. Commun*. **11**, 5979. (doi:10.1038/s41467-020-19447-9)
6. Ouyang W, Bowman RW, Wang H, Bumke KE, Collins JT, Spjuth O, Carreras-Puigvert J, Diederich B. 2022 An open-source modular framework for automated pipetting and imaging applications. *Adv. Biol.* **6**, e2101063. (doi:10.1002/adbi.202101063)
7. Ahlers J *et al*. 2023 Napari: a multi-dimensional image viewer for python. (doi:10.5281/ZENODO.3555620)
8. Stirling DR, Swain-Bowden MJ, Lucas AM, Carpenter AE, Cimini BA, Goodman A. 2021 CellProfiler 4: improvements in speed, utility and usability. *BMC Bioinformatics* **22**, 433. (doi:10.1186/s12859-021-04344-9)
9. Ouyang W, Mueller F, Hjelmare M, Lundberg E, Zimmer C. 2019 ImJoy: an open-source computational platform for the deep learning era. *Nat. Methods* **16**, 1199–1200. (doi:10.1038/s41592-019-0627-0)
10. Sweigart A. 2023 *PyAutoGUI*. See https://github.com/asweigart/pyautogui.
11. Bertaux F, Sosa-Carrillo S, Gross V, Fraisse A, Aditya C, Furstenheim M, Batt G. 2022 Enhancing bioreactor arrays for automated measurements and reactive control with ReacSight. *Nat. Commun*. **13**, 3363. (doi:10.1038/s41467-022-31033-9)
12. GitHub - labthings/esp-thingserver. 2023 Simple server for the ESP8266 and ESP32 boards that implements the W3C web of things API. See https://github.com/labthings/esp-thingserver.
13. GitHub-WebThingsIO/webthing-arduino. 2023 Simple server for ESP8266, ESP32, ethernet, or wifi101-compatible boards compliant with mozilla's proposed wot API. See https://github.com/WebThingsIO/webthing-arduino.
14. Olimex. 2023 *ESP32-EVB - open source hardware board*. See https://www.olimex.com/Products/IoT/ ESP32/ESP32-EVB/open-source-hardware.
15. OSHWA. 2023 *OSHWA certified projects list*. See https://certification.oshwa.org/list.html.
16. Olimex. 2023 MOD-RS232 - open source hardware board. See https://www.olimex.com/Products/ Modules/Interface/MOD-RS232/open-source-hardware.
17. Olimex. 2023 MOD-RS485 - open source hardware board. See https://www.olimex.com/Products/ Modules/Interface/MOD-RS485/open-source-hardware.
18. McDermott S, Bowman R. 2023 LabThings retro / LabThings retro case. *GitLab*. See https://gitlab.com/labthings-retro/labthings-retro-case.
19. Stirling J, Bumke K, Bowman R. 2022 GitBuilding: A software package for clear and consistent documentation of instrument assembly. *TechRxiv*. (doi:10.36227/techrxiv.20060903).
20. McDermott S. 2023 *LabThings retro*. See https://labthings-retro.gitlab.io/.
21. W3C Web of Things. 2023 *Home- Web of Things (WoT)*. See https://www.w3.org/WoT/.
22. OpenAPI. 2024 *OpenAPI homepage*. See https://www.openapis.org/.
23. GitHub. 2023 *LabThings*. See https://github.com/labthings.
24. Collins JT, Knapper J, McDermott SJ, Ayazi F, Bumke KE, Stirling J, Bowman RW. 2021 Simplifying the OpenFlexure microscope software with the web of things. *R. Soc. Open Sci*. **8**, 211158. (doi:10.1098/rsos.211158)
25. PlatformIO. 2023 PlatformIO is a professional collaborative platform for embedded development. See https://platformio.org.
26. McDermott S, Kotar J. 2023 LabThings retro / LabThings retro server. *GitLab.* https://gitlab.com/labthings-retro/labthings-retro-server
27. McDermott S. 2023 LabThings retro / LabThings Retro client. *GitLab.* https://gitlab.com/ labthings-retro/labthings-retro-client
28. GitHub. 2023 GitHub- WebThingsIO/gateway: WebThings Gateway. See https://github.com/WebThingsIO/ gateway.
29. McDermott S, Kotar J, Collins J, Mancini L, Bowman R, Cicuta P. 2024 Using old laboratory equipment with modern Web-of-Things standards: a smart laboratory with LabThings Retro. Supplementary material - code. *Zenodo*. See https://doi.org/10.5281/zenodo.11926548.
30. McDermott S, Kotar J, Collins J, Mancini L, Bowman R, Cicuta P. 2023 Using old laboratory equipment with modern Web-of-Things standards: a smart laboratory with LabThings Retro. Supplementary material - video demonstration. *Zenodo*. See https://doi.org/10.5281/zenodo.10123735.