BeeHive: A flexible open electronics platform for controlling research equipment and teaching electronics principles

Ihor Sobianin, Mikkel Roald-Arbøl, Solomon Gitau Ngotho, Lydia Ellison, Sina Dominiak, Shahd Al Balushi, Alejandra Carriero, Marcus Burnell Spector, Eglantine Vignal, Carla Lemos Perinetti, Cansu Demirbatir, Moira Eley, Maria Cozan, Estelle Moubarak, Fillip Janiak, George Kemenes, Leon Lagnado, Sylvia Schroeder, Tomas Nowotny, Sarah King, Tom Baden, Miguel Maravall, André Maia Chagas

https://doi.org/10.1590/SciELOPreprints.15694

Submitted on: 2026-04-16
Posted on: 2026-04-23 (version 1)
(YYYY-MM-DD)

BeeHive: A flexible open electronics platform for controlling research equipment and teaching electronics principles

Authors: Ihor Sobianin*, Mikkel Roald-Arbøl, Solomon Gitau Ngotho, Lydia Ellison, Shahd Al Balushi, Alejandra Carriero, Marcus Burnell Spector, Eglantine Vignal, Carla Lemos Perinetti, Moira Eley, Maria Cozan, Cansu Demirbatir, Sina E Dominiak, Estelle Moubarak, Filip Janiak, George Kemenes, Leon Lagnado, Sylvia Schröder, Thomas Nowotny, Sarah King, Tom Baden, Miguel Maravall, Andre Maia Chagas*

*To whom correspondence should be addressed

IS: i.sobianin@soton.ac.uk AMC: andre.maia@puc-campinas.edu.br

Affiliations

Ihor Sobianin: University of Southampton, faculty of medicine, school of enterprise and innovation, Southampton, UK, ORCID https://orcid.org/0000-0002-5444-602X

Mikkel Roald-Arbøl: ORCID https://orcid.org/0000-0002-9998-0058. BIOB, Section 2, Animal Diversity, University of Bonn, Bonn, Germany

Solomon Gitau Ngotho: The Science and Technology Facilities Council, Oxford, UK ORCID https://orcid.org/0009-0002-2206-8121

Lydia Ellison: Sussex Neuroscience, School of Life Sciences, University of Sussex, ORCID https://orcid.org/0009-0004-8198-9398

Sina Dominiak: Sussex Neuroscience, School of Life Sciences, University of Sussex, ORCID https://orcid.org/0009-0002-6344-5683

Shahd Al Balushi: Sussex Neuroscience, School of Life Sciences, University of Sussex, ORCID https://orcid.org/0009-0009-5351-9707

Alejandra Carriero: Sussex Neuroscience, School of Life Sciences, University of Sussex, ORCID https://orcid.org/0009-0003-0897-861X

Marcus Burnell Spector: Sussex Neuroscience, School of Life Sciences, University of Sussex, ORCID https://orcid.org/0009-0000-8905-3034

Eglantine Vignal: Université Paris-Saclay, CEA, SRMA , 91191 Gif-sur-Yvette, France ; ORCID https://orcid.org/0009-0009-6080-1612

Carla Lemos Perinetti: University of Amsterdam ORCID https://orcid.org/0009-0007-9636-0666

Moira Eley: Sussex Neuroscience, School of Life Sciences, University of Sussex, ORCID https://orcid.org/0009-0004-0998-9051

Maria Cozan: Sussex Neuroscience, School of Life Sciences, University of Sussex, ORCID https://orcid.org/0009-0002-0095-005X

Cansu Demirbatir: Department of Pharmacology, Near East University ORCID https://orcid.org/0000-0003-2644-8666

Estelle Moubarak: Marine Biological Section, Department of Biology, University of Copenhagen, Copenhagen, Denmark ORCID: https://orcid.org/0000-0002-4561-989X

Fillip Janiak: Sussex Neuroscience, School of Life Sciences, University of Sussex, Brighton, UK, Department of Metrology and Optoelectronics, Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology, 11/12 Narutowicza Street, 80-233, Gdańsk, Poland, ORCID https://orcid.org/0000-0002-9295-2740

George Kemenes: Sussex Neuroscience, School of Life Sciences, University of Sussex, ORCID https://orcid.org/0000-0003-2004-8725

Leon Lagnado: Sussex Neuroscience, School of Life Sciences, University of Sussex, ORCID https://orcid.org/0000-0002-1098-8839

Sylvia Schoeder: Sussex Neuroscience, School of Life Sciences, University of Sussex, ORCID https://orcid.org/0000-0002-9938-3931

Tom Baden: Sussex Neuroscience, School of Life Sciences, University of Sussex; TReND in Africa, ORCID https://orcid.org/0000-0003-2808-4210

Sarah King: Sussex Neuroscience, School of Psychology, University of Sussex, ORCID https://orcid.org/0000-0002-7412-9558

Tomas Nowotny: Sussex Neuroscience, School of Engineering and Informatics, University of Sussex, ORCID https://orcid.org/0000-0002-4451-915X

Miguel Maravall: Sussex Neuroscience, School of Life Sciences, University of Sussex; ORCID https://orcid.org/0000-0002-8869-7206

Andre Maia Chagas: Espaço Manacás, Pontifícia Universidade Católica de Campinas, Campinas Brazil; Sussex Neuroscience, School of Life Sciences, University of Sussex, Brighton, UK; TReND

in Africa, Brighton UK; BioRTC, Yobe State University, Damaturu Nigeria; Open Neuroscience, Campinas, Brazil - ORCID: https://orcid.org/0000-0003-2609-3017

Conflict of interest

The authors declare no conflict of interest.

Ethics Committee approval

All experiments that involved the use of animals have been conducted following the UK Animals (Scientific Procedures) Act 1986 with the approval of the University of Sussex Animal Welfare and Ethical Review Board.

Data availability statement

All data supporting this research is freely available on GitHub and can be accessed on https://github.com/beehive-org/beehive and in the repositories of each described device, in the “Neuroscience applications” section.

AI use

No AI tools or scripts were used in the preparation of this manuscript.

Abstract

In this manuscript we present BeeHive, a standardized, open-source hardware electronics ecosystem designed to address challenges faced by researchers when developing custom scientific equipment. Beehive is flexible and interoperable, since it is organised around minimal design rules. It consists of a microcontroller-based mainboard and function-specific Daughter Boards (DBs) for controlling actuators or reading sensors, allowing researchers to rapidly combine and repurpose components to build complex, specialized systems. This modular approach significantly reduces the time and cost associated with instrumentation development, promoting effective collaboration through shared, standardized solutions. The platform's versatility is demonstrated across several neuroscience applications, including a reward delivery system for head-fixed mice, a modular mouse maze, and an odour stimulator, alongside a dedicated Training DB and curriculum designed to teach electronics and MicroPython programming in an integrated, project-based manner. By embracing open-source principles for hardware and software, BeeHive lowers the barrier to entry for researchers in developing their own setups, fostering greater accessibility, reproducibility, and innovation. The system is also discipline agnostic, and could be used to create tools in different fields.

Keywords

Open Hardware, Neuroscience, Open Science, Behaviour, Open Source

Introduction

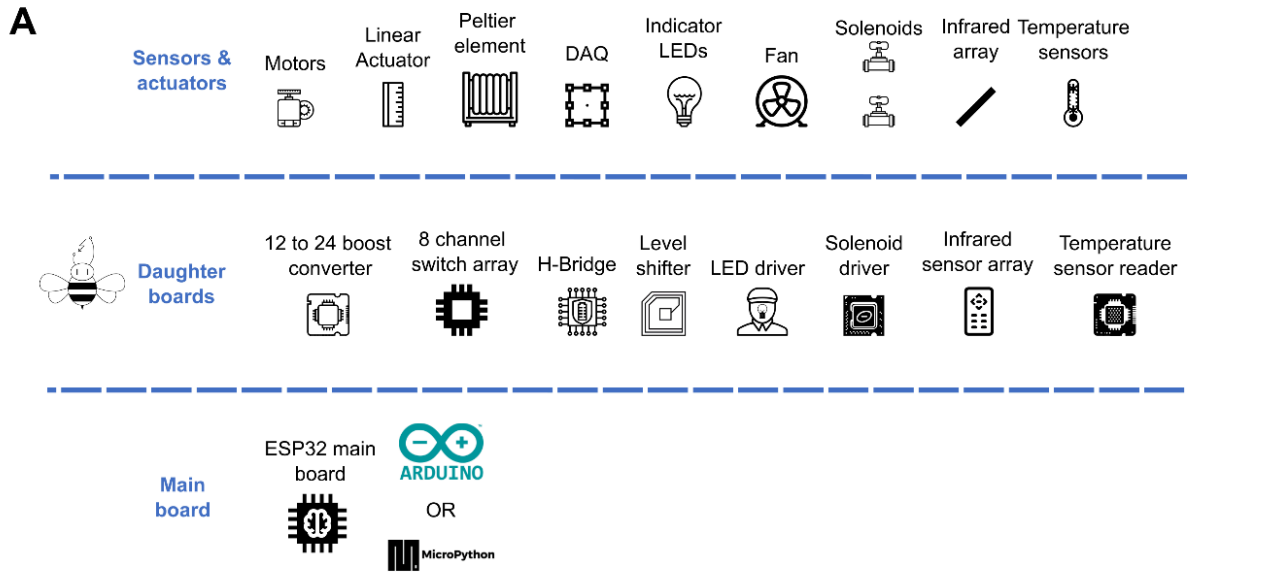
Research requires hardware. From general “consumer level” hardware such as personal computers to specialised scientific equipment. However, there is a divide between equipment production and use, where researchers often lack the knowledge to build or modify the tools they are using, hindering the pace of research as they are unable to repair, calibrate, modify existing tools or create new ones [1]. When the need for new types of experiments arises, researchers have to either request changes of existing tools to companies (which might not see commercial interest in the researchers’ goals) or they have to adjust their scientific experiments and research agenda to the tools that are available at their labs and/or institutions.

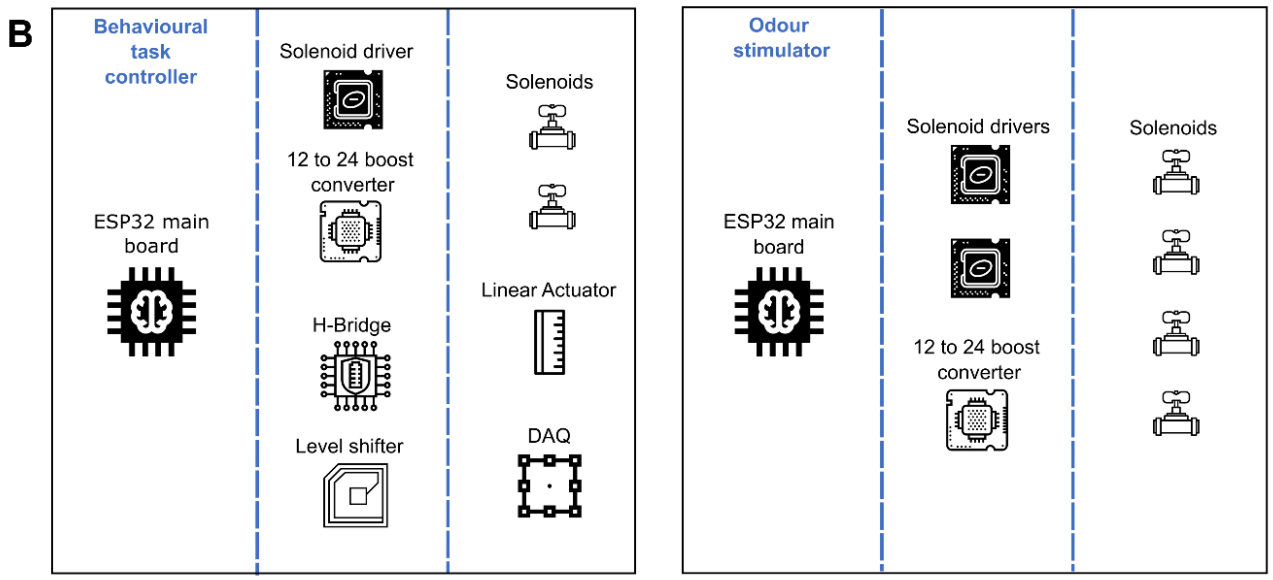
Fortunately, recent technological developments are making it easier for researchers to build or modify scientific equipment: the past decade has seen an increase in the number of papers sharing detailed descriptions on how to build and operate scientific equipment [2,3]. One of these technological developments are affordable electronic modules dedicated to specific functions, and their accompanying documentation explaining how they should be connected and programmed for different applications. Examples of such modules include “hats” for Raspberry Pi’s, “shields” for Arduino boards, and many modules developed by a number of companies like Adafruit, Sparkfun and SeeedStudio [4–6]. However, after extensive search in preprints, peer-reviewed articles and online repositories (e.g Open-Neuroscience [7]), we found most available tools are designed to solve an immediate problem/demand (e.g. a system to measure behaviour in rodents in a specific experiment) [8–15]. While the move towards creating and sharing open tools and applications is enormously positive, it is also clear that there is no pattern or standard that would allow individual designs (or parts thereof) to be easily reutilised in different applications.

Here, we present BeeHive, a standardized hardware ecosystem that has been used in different applications, from the creation of neuroscience experiments and laboratory equipment to teaching electronics and coding to academic audiences. What sets BeeHive apart from the aforementioned modules and shields are its minimal design rules. They enable sufficient flexibility for different applications in different domains while keeping it compatible with existing systems, making it a good choice for extending existing builds towards new capabilities. With this in mind, all presented modules were specifically created to fill the gap where no commercial alternatives were available. Beehive, therefore, does not seek to reinvent the wheel, but to fill gaps on what is currently available, while being interoperable with other existing hardware and software environments. We also describe how we have used this system for teaching electronics and programming. All designs are publicly available in Beehive’s github repository [16], and are released

under Open Source licences, ensuring that everyone can freely copy, learn, modify, improve and use the system as they see fit.

BeeHive's organisation is normally comprised of one mainboard that carries one microcontroller, as well as daughter boards (DBs) which control one actuator (e.g. motor, peltier element, LED, fan), or receive data from one sensor (e.g. temperature, humidity, infrared. Figure 1A). Accordingly, the system has two simple principles for development: 1-Each DB is dedicated to one single function (e.g. one controls a solenoid valve, another controls high powered LEDs) and 2-Their connection ports use a specific pattern of two data lines, one power line and one ground line (more details on Design rules section). This implementation with minimal requirements/rules maintains flexibility for future development, while remaining interoperable with existing systems, thus lowering the barrier for external contributors. Moreover, the many DBs can be combined in different ways for different applications (Figure 1B), and the same combinations can be used for applications that are similar, but used in different contexts (e.g. the same solenoid controller DB could be used for water delivery to a rodent or primate, the only change being the solenoid valves and water spouts) and/or research fields (e.g. the same DB could alternatively be used for controlling water irrigation in a greenhouse) (Figure 1C).





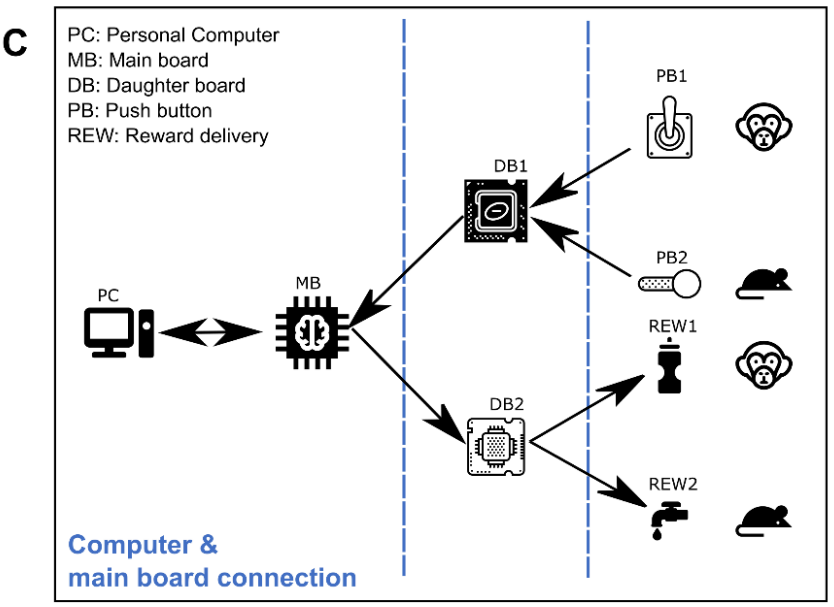


Figure 1 | Schematic representation of BeeHive and its governing principles *A) System overview. The bottom row represents a Mainboard carrying a ESP32 microcontroller, which can be controlled by MicroPython or Arduino (C++) programming languages. The middle row shows examples of different DBs. Top row shows the different sensors and actuators that can be connected to the Mainboard, via the DBs. B) Representation on how different applications can be constructed by combining the same Mainboard with different DBs. The left panel shows a behavioural task using four different DBs (Solenoid valve controllers, voltage booster, h-bridge and linear actuator controller) connected to four sensors/actuators. The right panel shows an odour stimulator using one DB type (solenoid valve, the same used in the behavioural task), controlling four actuators. These two tasks highlight how DBs can be used in different applications. C) An example showing how applications that are similar, but prepared for different animal models can be achieved with the same set boards, requiring only changes in the objects that will be in contact with the animals.*

Another consideration during development was providing a learning platform for researchers with different levels of programming skills. By leveraging MicroPython (Python 3 implementation for microcontrollers), and a training DB (more details below), our system has a very shallow learning curve, with most effectiveness, as researchers can use the same programming language to create experimental designs, collect data and analyse it. Moreover, since MicroPython is a “derivative” of Python, with identical syntax and structure, we take advantage of Python’s popularity among researchers [17] and the fact that it is one of the world’s most used programming languages [18]. For the remainder of this paper, we will detail BeeHive, showing implemented applications in neurosciences (our current field of study), how it can be used as a teaching tool for electronics and programming, as well as potential applications for other fields.

BeeHive in detail

Hardware

Currently BeeHive’s main board hosts a microcontroller, power management circuitry and connectors that make it easy to access all of the microcontroller’s digital, analog and communication ports (each of these parts will be described in detail in their respective sections below). To complement the mainboard we provide designs for boards that allow users to rapidly connect BeeHive to diverse components, such as LEDs, solenoid valves, speakers, motors, and different types of sensors. They follow the same connection pattern as the main hub and allow a wide array

of components to be integrated in the system, as each DB is responsible for one single function and can be easily replicated. BeeHive runs on 12V DC, which can be supplied using common AC/DC converters as well as through batteries, making the system portable, electrophysiology friendly, and most importantly, amenable to use in regions where the power grid infrastructure is not reliable.

Design rules

BeeHive has very few design rules, which gives it enough flexibility to be applicable in many different fields and applications, while keeping a minimal structure needed for continuous development and stability:

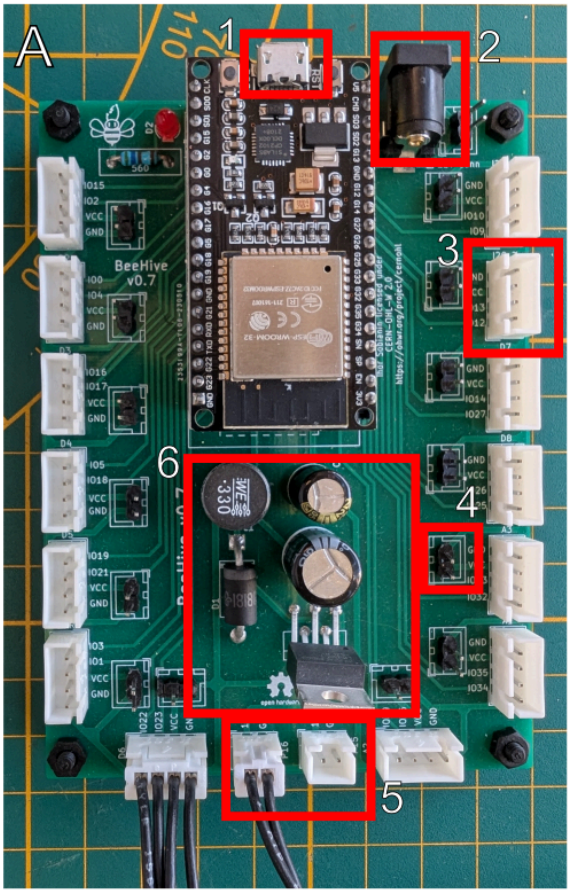
Strict rules:

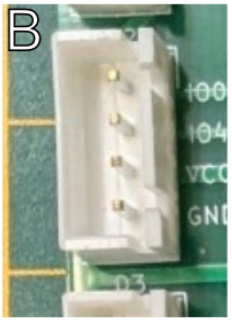
1. There are two types of boards. Mainboards, which carry a microcontroller, and DBs, each performing one function (e.g. Control a solenoid valve).
2. Connections between main and DBs are made via a 4 pin connector containing a power line, a ground line, and two general purpose input/output pins (Fig. 2).

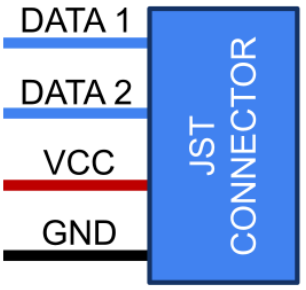
Recommended rules:

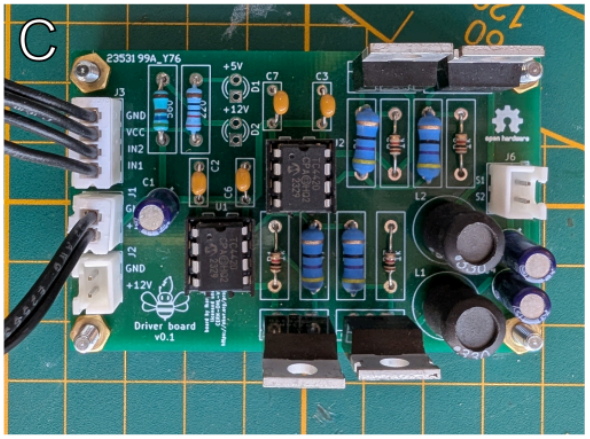
3. When designing printed circuit boards (PCB), designers are encouraged to layout the connections in their boards using one or two layers (top and bottom PCB layers), so that boards can be replicated locally, using perf boards, PCB chemical etching or CNC machining. All boards can also be ordered at affordable rates from many different companies (e.g. JLCPCB, PCBway, OSHPark, Aisler [19–22]).
4. Electronics components should be through-hole components, so that they can be hand soldered with minimal equipment by people without previous experience.
5. Populate only one side of a board.
6. Label your boards abiding the existing rules: put connector pins identifiers, DB name and version number.
7. Size of a DB should preferably be selected from the two predominantly used:
    a. Big - 71 x 45.5 mm
    b. Small - 48.5 x 45.5 mm
8. Size up if the board is getting too cramped

9. There are typically 4 mounting holes for M3 size screws. Each hole has its centre placed 3.5mm away from the board edges.
10. If in doubt - reach out!









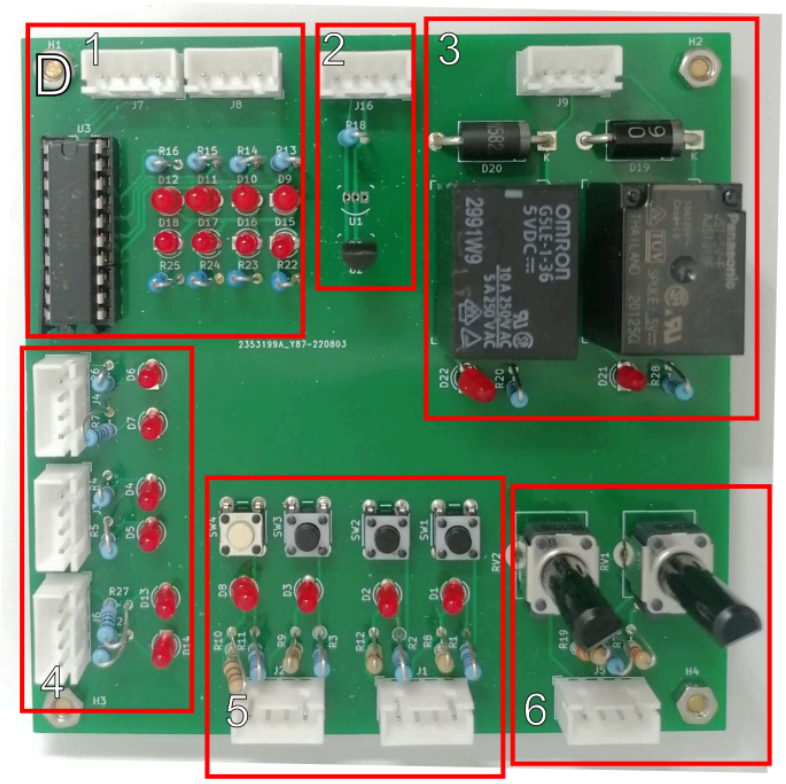


Figure 2 | BeeHive's board examples and connector pinout*. A) Main board, with the ESP32 microcontroller. Clockwise from the top red squares highlight: 1 - ESP32 USB micro connector, 2 - Power in barrel jack, 3 - pin out connector, detailed in panel B, 4 - jumper connector to control power line in each individual pin out connector, 5 - 12V connectors, 6 - 12 to 5V power converter. B) Pin out connector in detail, depicting BeeHive's "strict rule" 2, two IO pins, one power line (VCC) and one ground line (GND). C) Daughter board example, an H-Bridge, used to control DC motors or Peltier elements (yellow grid on the background has 1x1cm squares). D) Training board, a board containing different modules representing different types of electronic communication and control, which can be used for teaching electronic concepts, and for designing the logic behind experimental setups and research devices. 1 - 8 bit shift register and led matrix, 2 - digital temperature sensor module, 3 - Relay module, 4 - Digital output and analog output module, 5 - Digital input module, 6 - analog input module.*

Mainboard

Mainboards carry a microcontroller, and expose all its ports and pins to users and developers via standard plugs. Currently, two mainboards are available, using the ESP32 microcontroller. One of them is the original BeeHive mainboard (incorporating only the breakout connectors for the ESP32 and power lines), the other has the same features, plus added compatibility with MikroBUS™ ecosystem [23] (details below). Most applications will require only one mainboard.

Original Mainboard

Our first mainboard was developed using an ESP32 microcontroller. This choice was guided by the ESP32's accessibility (boards can be bought over Amazon, Aliexpress [24,25] and delivered worldwide), low cost (each board ranges from 2-6 British Pounds depending on supplier), number of ports (32 general purpose input and output (GPIO) ports, currently configured as 8 digital lines, 3 analog input lines, 2 I2C communication channels, 1 UART), diversity of communication protocols (serial, I2C, SPI, bluetooth and WiFi) and the fact that it can be programmed either with MicroPython [26] or C++, allowing users to leverage the Arduino environment and its many libraries [27] (more details on programming language are listed on the Software section).

MikroBUS Mainboard

This board was developed to make BeeHive compatible with the MikroBus open standard, and allow the integration with Click Boards™ [28] ecosystem (developed by MikroEletronika), adopted by over 100 companies, and comprising over 500 boards. This allows users to have a much wider selection of ready made boards and applications, while keeping the ability to use MicroPython for control. This also serves as an integration example of the BeeHive system with other already established platforms.

Daughter boards

In BeeHive, each DB is responsible for one single function. At the moment we have created 17 boards, responsible for 15 different functions (Table 1. For regular updates on available boards and functions, please visit https://BeeHive-org.github.io/ingredients/summary/). The reason for more DBs than functions, is that in some cases, the same function needs to be implemented in specific ways. Eg. For opening a solenoid valve in a hundreds of microseconds window, we have replicated a “Spike and Hold” circuit described by the LeeCompany [29]. We have also created a “traditional” solenoid driver for applications that do not require fast solenoid opening speeds, since the “Spike and Hold” circuit overdrives the solenoids, reducing their lifetime.

Table 1 *- overview of BeeHive main and daughterboards*

Board name	Function
Main boards	
ESP32 BeeHive board	The main part of the BeeHive system. The board is based around the ESP32 microcontroller. DBs can be connected to the Main board via JST connectors and are controlled through MicroPython
Mikrobus compatible BeeHive board	MikroBus compatible version of the main board, similar to the ESP32 board above, but integrating connectors for the Mikrobus ecosystem [23]
Daughter boards	

12V5A breakout	A 12V5A breakout board feeds 12V5A to other devices. It consists of a barrel jack connector, an LED power indicator and 9 connectors
5V3A breakout	A power supply converts 12V 5A to 5V 3A using LM2596 power converter. The main board has two connectors that supply 12V5A which can be connected to this board. The 5V3A breakout board can also be connected to the 5V 3A breakout board. The output of this board is 5V 3A maximum.
12V/24V boost converter	A boost converter board is based on an 555 IC and works in the “step up” mode. It is powered by 12V5A and can provide up to 24V1A. 24V1A is used to power up solenoids in "Spike and Hold" mode or in conjunction with any other device that needs 24V. The board can be powered by either the main board or the 12V5A breakout board.
Gas sensor	The board is used to detect any gas-related environmental variables. The central component of this module is an MQ-6 gas sensor (but swapping it by any other sensor from the MQ series allows for different gases to be detected).
H-bridge driver	An H-bridge driver module consists of an H-bridge and a connector for a ds18b20 temperature sensor. The H-bridge is based on the full-bridge configuration and consists of 2 n-channel and 2 p-channel MOSFETs. Transistors are controlled through a TC4420 driver. Output of the module is 12V5A maximum. The low pass filter ensures compatibility with Peltier elements.
Switch array	The board utilises a 74HC595 shift register, allowing it to connect more components with less number of pins. The power fed to a component being switched can vary from 5 to 12 volts.
High power switch array	The board is based on TC4427 driver and n-channel MOSFETs and can be used to switch power-demanding components. The high power switch array works with 12V5A, which can be accessed from the main hub or the 12V5A breakout.

Humidity and temperature sensor	The board monitors humidity and temperature. It is compatible with both DHT11 and DHT22 sensors.
IR sensor array	An infrared red (IR) sensor array is designed to accommodate 6 pairs of IR LEDs and the corresponding phototransistors.
Level shifter	A bi-directional level shifter board allows communication between 5V and 3.3V devices, i.e. different logic levels. The design is heavily influenced by Ada Fruit level shifters and is based on BSS138 MOSFET.
Solenoid control board	The board is mainly used for controlling solenoids.
Spike & Hold board	The purpose-built board which is designed to work with solenoids in the so-called “Spike & hold” mode, enabling faster opening times of the solenoids.

SOFTWARE

Currently, most of the code developed for BeeHive has been written in MicroPython [30], a Python 3 implementation dedicated for microcontrollers and resource limited environments. We chose this over C++, because Python 3 is widespread and widely used by academics, with a number of online tutorials, courses and other resources guiding people on the first steps into programming and (often) data analysis. Given that the syntaxes for MicroPython and Python 3 are the same, and some libraries are present in both, Python users will have no problem developing their own applications in BeeHive to control hardware. For users that are not programmers, or that do not know Python; MicroPython and BeeHive can be used as a teaching tool (more details on section “BeeHive as a programming and electronics learning platform”) for both programming and electronics, making an introductory system for a programming language that will be used from experimental design all the way to data analysis and presentation.

Software modules

Example software modules to control each DB are provided to support users. Coupled with the code available in the documentation of specific equipment that uses each DB, users will be able to adjust the system to any given conditions in order to run their desired experiments.

DOCUMENTATION

BeeHive's documentation is hosted at https://BeeHive-org.github.io/ [31]. There, users will find information about each board, their specific function, the skills and components necessary to assemble each board, and example code to control each of them.

BeeHive as a programming and electronics learning platform

One of BeeHive's features is the Training DB and the publicly available guide/exercises [32] to support users in learning programming and how to control electronic devices. In the current iteration, the board contains circuitry to control LEDs, push buttons, potentiometers, shift registers, rotary encoders, relays, and temperature sensors. These actuators and sensors were selected because they cover a wide range of electronic principles and communication protocols (digital input and output, analog input, SPI communication) that are widely used in practical applications. We complement the usefulness of the board with related exercises (e.g. designing a reaction time task using a LED and a push button) to guide learners in a project based manner, and decoupling coding from electronic setup/wiring. By using the Training DB, people who are getting introduced to building their own equipment and need to learn coding and electronics can focus on one of these at a time, making sure concepts are mastered before moving on to the next domain. Another benefit of the board, is that it can be used to create code that will later be used in experimental setups and devices, without the need to block an experimental setup or device to update code/trial new routines or updates, (in fact the footprint of the main board and the learning DB is small enough that users can easily pack them in a bag and take them for out-of-the-lab demos and training). A practical example to this would be to create the logic for a behavioural task for neurosciences using the Training DB, mimicking behavioural responses with the push buttons. Once complete, the code can then be uploaded to the experimental setup.

To make sure the Training DB would properly cater to the desired audience, we ran a workshop at the University of Sussex, in the department of Neurosciences, where early career researchers, bachelor students and post-graduate candidates were guided through electronics and programming theory, combined with hands-on exercises and a short session on soldering, where participants were given a brief explanation on soldering practices and got to solder DBs on their

own. With a total duration of 20 hours, participants were able to create their own mock up experimental paradigms with code and the Training DB.

Neuroscience applications

Currently, applications developed using BeeHive fall into the remit of Neurosciences, mostly because this is the study field of most authors rather than a system requirement. In this section we will show some of these applications and try to make parallels to applications in other fields, hoping that readers outside Neuroscience will see potential benefits in using BeeHive as well. At the moment we have developed 07 different setups using BeeHive, namely: a behavioural system for head fixed animals under 2-photon microscopes, a mouse maze, a controller for setting the running speed on a mouse wheel, an odour stimulator for Drosophila, a multiplexer for metabolic measurements in multiple small invertebrates (based on a Licor 850 [33]), a 5-choice serial reaction time system, a controller for an OpenFlexure delta stage [34]. In the next section, we briefly describe each application. All experiments that involved the use of animals have been conducted following the UK Animals (Scientific Procedures) Act 1986 with the approval of the University of Sussex Animal Welfare and Ethical Review Board.

Current implementations

Head fixed setup (documentation available here)

Boards used: main board (1x), solenoid control board (1x)

We used the BeeHive system to incorporate a reward delivery system and a lick sensor into an existing set up that was designed to deliver visual stimuli to head fixed mice running on a treadmill under a two-photon microscope. The original set up included two LED monitors (BenQ XL2410T, isoluminant at 25cd/m^2, 120 Hz refresh rate) that were positioned on each side of the animals presenting the visual stimuli (Figure 03). Stimuli consisted of 10 s long drifting gratings that were generated by the Python library PsychoPy [35]. The information about the stimulus timing was sent to a LabJack Data Acquisition board (DAQ) to be paired with an LED driver for optogenetic manipulation and then directly fed into the imaging software (ScanImage5; Wavemetrics) via an analogue channel of a National Instruments DAQ for automatic alignment with two-photon imaging. Mice were up to this point viewing the gratings passively, meaning they were simply exposed to the gratings without giving the stimulus any behavioural significance.

To change the relevance of the stimulus we integrated a reward delivery system using BeeHive, to condition animals to associate the gratings with a reward. The reward system consisted of a metal spout used to deliver a liquid reward and a piezo sensor embedded in a custom-made, 3D-printed frame and connected to an adjustable amplifier (developed by the CWW Research Workshop, Charité Berlin). To prevent the animal from licking during stimulus presentations, the lick spout was out of the animal's reach and was moved towards the animal by a servo motor (Geekservo) only for the time of the reward delivery. A solenoid pinch valve (WZ-12021-23, Spex® VapLock™, normally closed) was used to control the amount of the liquid reward. The timings of the motor movement and opening of the Solenoid pinch valve were thereby controlled by the BeeHive system: The motor began to move the spout 0.5 s before the end of the 10 s stimulus so that the lick spout reached the animal right at the offset of the stimulus. The reward was then delivered with a delay of 0.25 s, via the opening of the Solenoid for 150 ms. Subsequently, mice were given 1 s to lick for the reward before the lick spout was retracted. The information about the timings of the stimulus, solenoid activation and licks was sent from the BeeHive system as an analogue voltage signal, where the duration of the signal meant how long each event was, and the voltage level identified which event was being transmitted. This signal was fed into the analogue input channel of the NI BNC DAQ to be automatically aligned with the two-photon image acquisition (Figure 03).

Additionally, BeeHive's flexibility in controlling stimulus and reward timings facilitated the development of different behavioural tasks using MicroPython. For example, animals could be trained to lick the spout only in response to a specific stimulus to obtain a reward while suppressing licks to other stimuli. Furthermore, a second lick spout and solenoid could be added to the system giving animals two different response options. Consequently, animals could be trained to lick a spout positioned on the left side of their snout when a stimulus is shown on the left monitor while licking a spout positioned on the right side of their snout when a stimulus is shown on the right monitor.

Overall, the BeeHive system provided a straightforward and flexible solution to gain timely control over reward delivery and lick detection, enabling the incorporation of various behavioural paradigms into the set up.

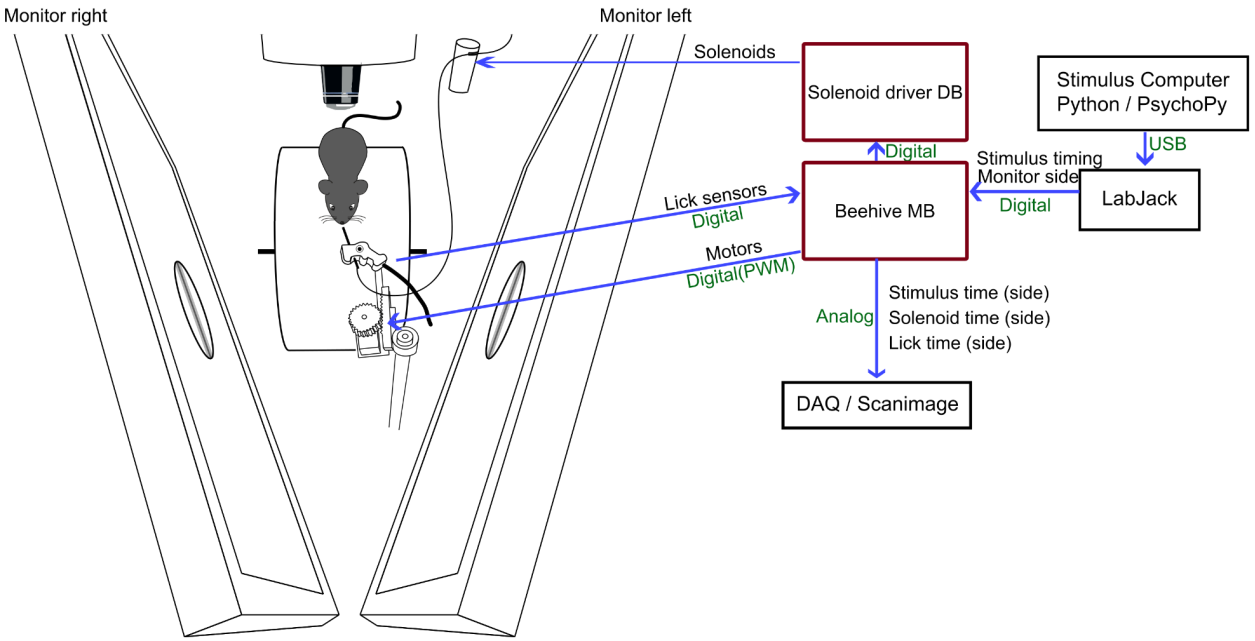


Figure 3 | Behavioural setup with integrated reward delivery system. *Animals were head-fixed on top of a treadmill in between two screens under a two-photon microscope. Stimuli were generated by PsychoPy and presented on either one or both screens. The information about the stimulus timing and the presenting monitor was sent from the stimulus computer into the BeeHive controller board (red) via a LabJack. The BeeHive controller then timed the movement of the lick spout towards the animal and the opening of the solenoid valve to deliver a reward based on the stimulus timing and the behavioural paradigm. Lick times were detected by the lick sensor and fed into the BeeHive system. For automatic alignment with the two-photon imaging acquisition information about the stimulus, solenoid opening and lick times (and side whenever two lick sensors were used) was output from the BeeHive system to the imaging software ScanImage.*

Mouse Maze (documentation available here)

Boards used: main board (1x), IR sensor array (1x), Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface (1x)

Using BeeHive, we have developed a modular maze which aims to interrogate sensing and decision-making in freely moving mice, providing high levels of experimental control while ensuring simplicity and flexibility of behavioural task design.

The basic maze is constructed from acrylic panels opaque under visible illumination but transparent in the IR range [36], allowing tracking of animals with an IR camera as they move. Wall panels of size 50 mm x 50 mm slot between 'Makerbeam XL' posts (Figure 04): standard panels at any location can be replaced by mechanical device panels incorporating static textures, moveable gratings or 3D-printed shapes, or reward ports (Figure 04). This allows flexible reconfiguration of the maze. Each of the devices is controlled and activated depending on the animal's behaviour in real time: for example, a given stimulus or food treat may be delivered as soon as the animal enters a chamber in the maze. This was implemented by tracking the animal's behaviour using a camera input via OpenCV [37] and a state machine written in Python. This state machine sends serial commands to a BeeHive MB to control reward delivery. The mechanical part of the delivery systems was redesigned from [14] so that it works with a servo motor, instead of a stepper motor, making its design easier to 3D print (editable design files are available here [38]). The MB controls the servo motor through an Adafruit PCA9685 16-Channel Servo Driver (which can be daisy-chained to control up to 992 motors using only two data lines from the microcontroller).

We benefitted from the system's flexible design in this application by writing the MB firmware in C++, instead of MicroPython, so that we could leverage existing C++ libraries for the Adafruit servo driver (Adafruit PWM Servo Driver library [39]) and for parsing serial commands (serial command parsing library [40]) between the PC and the microcontroller and the MB. .

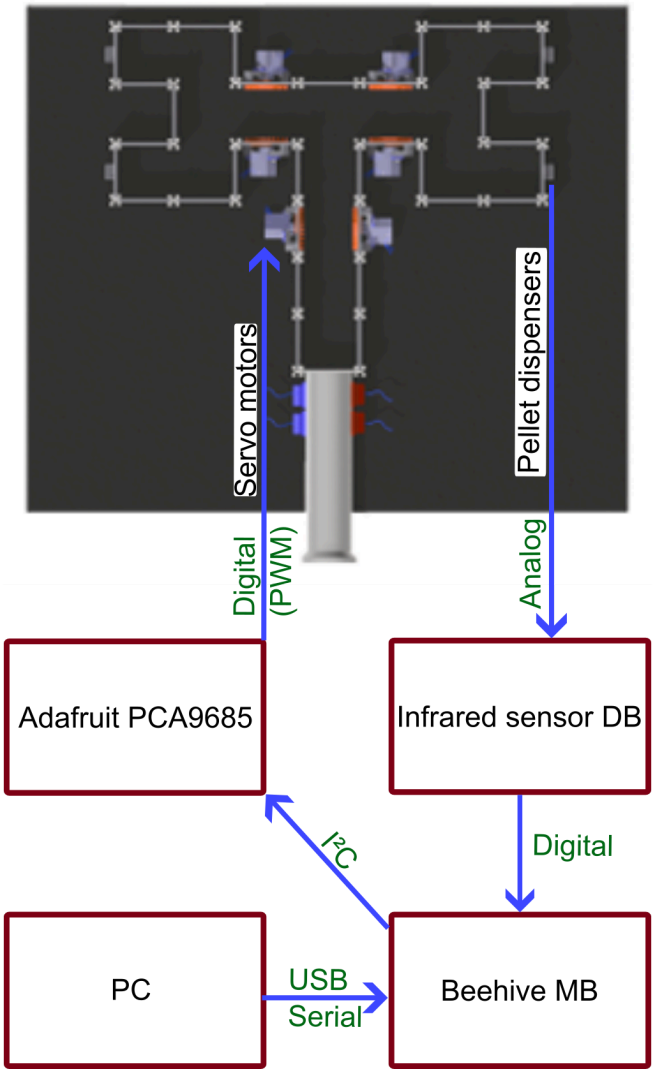


Figure 4 | Mouse maze with closed loop control. *Schematic showing maze configuration and how BeeHive is used to control reward delivery.*

Mouse wheel speed controller (documentation available here)

Boards used: main board (1x), stepper motor driver (1x)

Using BeeHive, we enhanced a commonly used, open-source mouse running wheel, KineMouse [41,42], with a motor and a clutch. This combination provides the experimenter with control over the time and minimum running speed of the animal.

The original KineMouse wheel is used in setups where the mouse needs to be head-fixed to measure neural activity while the animal is free to run at any speed. If the goal of the experiment is

to correlate running behaviour with neural activity, the ideal mouse would run approximately half the time during the experiment at regular intervals. However, mice rarely exhibit this ideal behaviour, and experimenters often extend experiments to gather data across various running states or exclude sessions where the mouse's behaviour deviated significantly from the ideal.

Our motorised running wheel features a stepper motor attached to the wheel axle. The rotation speed of the wheel is controlled by an analog input from a potentiometer. By incorporating a clutch between the motor and the wheel axle, we designed a motorised running wheel that allows the experimenter to gently nudge the mouse to initiate locomotion at a minimum speed while the animal can choose to run at a higher speed. The mouse can also continue running even when the motor is disabled.

This design enhances the experimenter's control over the animal's behaviour while minimising constraints on the animal. By reducing the number of necessary experimental sessions and the duration of each session, the motorised wheel aligns with the 3Rs of animal research [43].

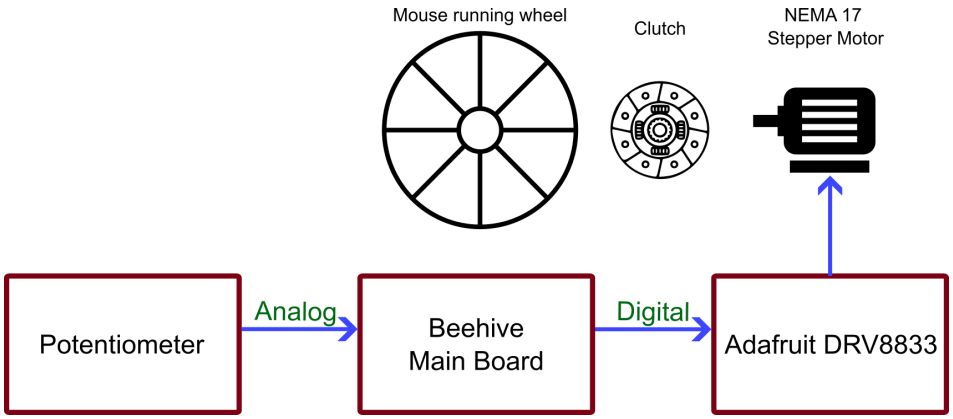


Figure 5 | Schematic of motorized mouse wheel. *Leveraging an existing open source wheel design, we added a clutch+motor system that allows experimenters to control animal movement. The clutch allows animals to move faster than the motor movement.*

Odour stimulator (documentation available here)

Boards used: main board (1x), Spike and Hold solenoid driver DB (2x)

We have reimplemented an odour stimulator described in [44] (Fig 6A-D). Briefly, it enables the delivery of two precisely controlled odour streams, balanced by two additional clean air streams to maintain stable air flow during odour fluctuations with millisecond precision. Air streams are switched by three-way Teflon solenoid valves (LHDA1233415H, Lee Company, Westbrook, CT),

controlled by separate “Spike and Hold” solenoid driver DBs and a single main board. Valved air streams pass through a custom 3D printed mixing block (printed in polypropilene), with the potential for further channels to be added with relative ease.

The two-channel odour stimulator delivers odour pulses with a high degree of precision, enabling the observation of kinetic features of different odours as reported previously (Fig 6E) [44,45] and attributed to different volatilities and interactions with the stimulator’s internal surfaces. The two odour channels also operate equally and independently (Fig 6E,F). Flexibility/reproducibility/low cost of the system enables comparable odour stimuli to be delivered across multiple experimental systems. Current implementations include odour delivery for single sensillum recordings, flight and walking behaviour in an insect wind tunnel and single fly behavioural chambers.

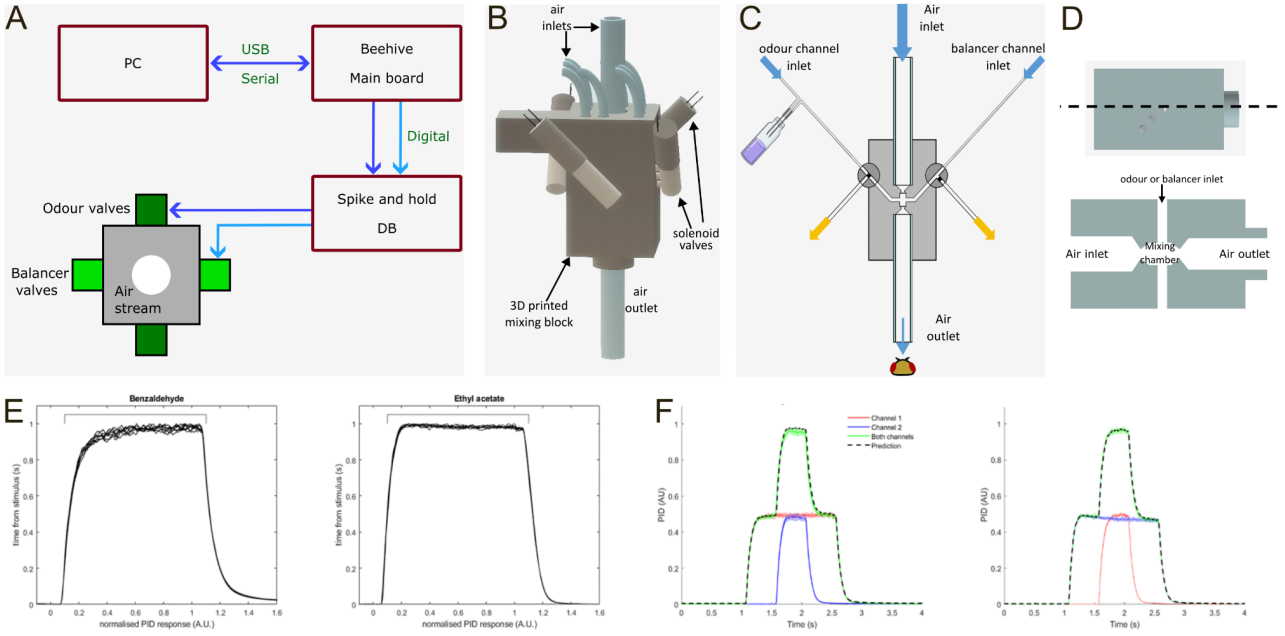


Figure 6 | Two channel odour stimulator. *A) Electronic designs schematic (here showing control for only one channel - the second channel works in the same way). B) 3D rendering of the mixing block, and solenoid valves responsible for odour stream control. C) 2D view on how one odour channel mixes the clean air stream, and how a balancer channel guarantees that the amount of air reaching the animal remains the same once an odour is added. D)*

LI-850 multiplexer (documentation available here)

Boards used: main board (1x), solenoid control board (6x)

To increase experimental throughput in ethological projects involving measurement of metabolic rate, we have used BeeHive to “multiplex” a Licor LI-850 CO2/H20 gas analyser [46], and parallelise data collection. Briefly, by placing small invertebrates in gas tight chambers, this system measures the animals’ CO2 excretion and water loss over a specific amount of time. To do so, ambient air is scrubbed of all CO2 and water content, before being pumped through the chamber (Fig.7A, showing normal setup schematic). From there the air is pumped into the gas analyser which estimates the CO2 and water excreted by the animal. For every new experiment, the chamber needs to be opened to place a new animal for testing, during which the chamber fills with ambient un-scrubbed air. Due to the high sensitivity of such recordings, a period of time required, depending on the chamber size, is required to clear out any ambient air from the chamber and stabilise at baseline. These stabilisation periods greatly prolong the time spent on each individual experiment, particularly when recordings are short (e.g. 20 mins) as does the manual work an experimenter has to do in frequently changing the animals. To solve this, we split the air pathway of the LI-850, so up to 6 chambers are connected to the system, but each chamber has a solenoid valve on its air inflow side and one solenoid valve on its air outflow side which we can control independently (Fig.7B, showing multiplexed setup schematic with 3 chambers instead of 6). This enables us to clear chambers faster and with less time spent by the experimenter, before recording from all 6 chambers in a series of user-defined intervals. We use BeeHive’s solenoid driver DBs for controlling solenoid valves and MicroPython code to open the air pathway for a specific chamber, then ping the LI-850 (using serial commands provided on the device’s documentation [47]) to register and send over data to be logged as csv file. Our initial tests (Fig. 7C-E) show that the stabilisation phases of the multiplexed system behave in the same way as the non-multiplexed system.

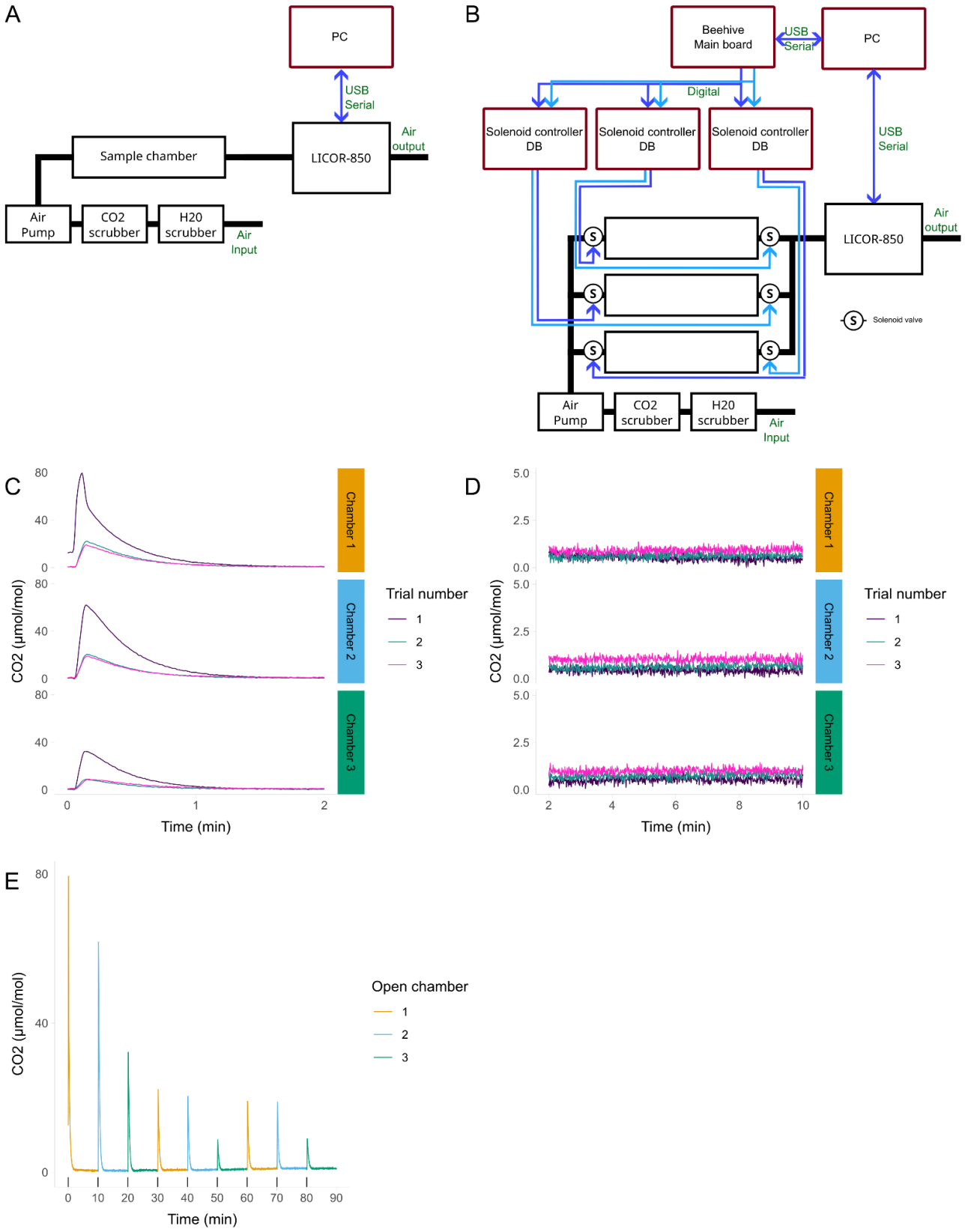


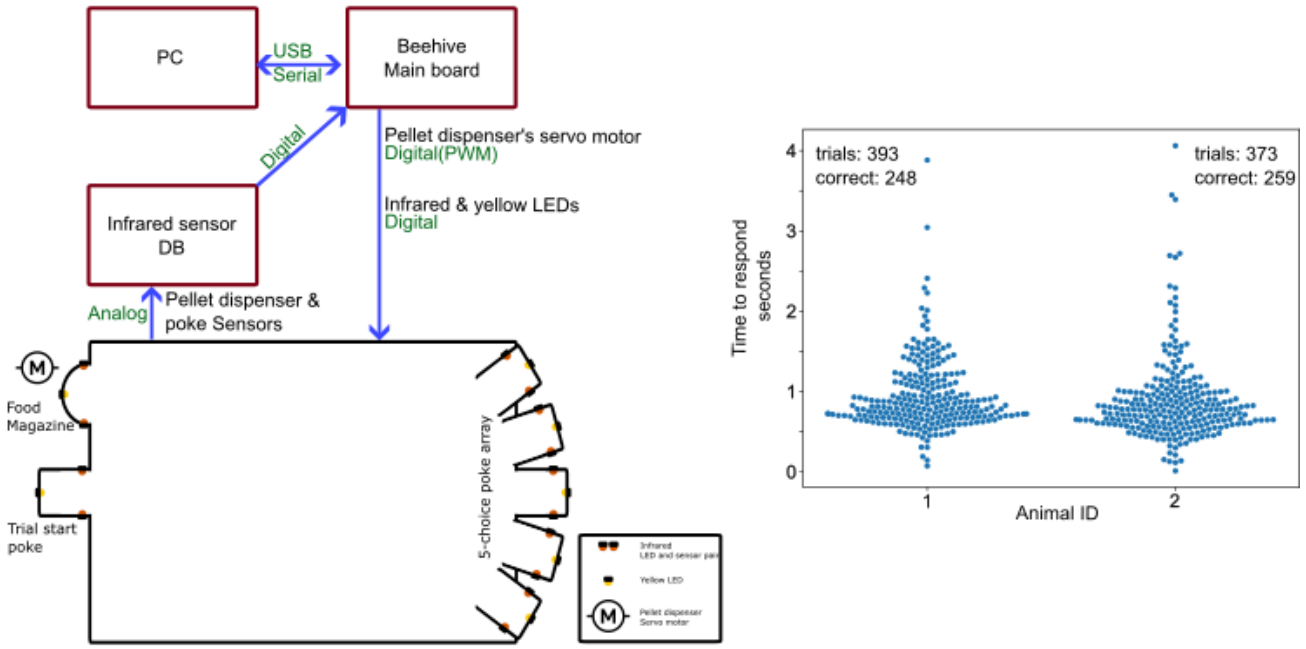
Figure 7 | LI850 Multiplexor. *A) Schematic showing the LI850 being used in its single chamber configuration B) Schematics depicting how the LI850 pathway is multiplexed using solenoid valves connected in parallel and synchronised using BeeHive. C) Control experiment results showing CO2 concentration levels over time in three empty independent chambers. Each chamber was opened for 2 minutes, while the others remained closed, with 3 repetitions in total. CO2 concentration levels peak in the first 20 seconds, then decrease to baseline levels. D) Similar experiment in C, except each chamber is opened for 10 mins each. At each trial, CO2 levels stabilize and remain at baseline from minute 2. E) Data from experiment shown in D, depicting the peaks in CO2 at each chamber opening over time.*

5 choice serial reaction time task (documentation available here)

Boards used: main board (1x), IR sensor array (1x)

We replicated a five choice serial reaction task (5-CSRTT) behavioural paradigm for mice [48], using BeeHive. Our implementation follows training protocols previously published [49], which allows mice to self-direct their learning, avoiding long periods of food deprivation and reaching stable performance of the 5CSRTT within 7 to 10 days (as opposed to 3-5 months in traditional paradigms). Our interest in reimplementing the paradigm stems from the fact that the training schedule is a great advancement for behaviour measurements, but it was built on top of proprietary hardware and a custom programming language. This makes the system inaccessible to many, as well as hard to modify/improve. By using open source hardware and Python we make it easier to adapt this paradigm and to scale the number of behavioural boxes running in parallel, leading to increased data collection rates and decreasing the amount of time dedicated to setting up and keeping experiments running.

Briefly, the box uses a simple system with a pair of infrared light emitting diode (LED) and infrared sensor (IRS) pointing at each other, placed within “nose poke” ports. When the animals place their nostrils in those ports, the IRS stops detecting IR light and this gets registered by the system, with microsecond precision. The back of each port contains a yellow LED that signals to the animal where it should poke. For food reward, we redesigned an open source pellet dispenser described in [50]. It is simpler to 3D print, as all parts were made to be laid flat on the printing bed and assembled later. We also replaced the stepper motor from the original design for a servo motor, simplifying the electronics needed to control it. Our dispenser is freely available [51] and has also been used in the mouse maze system described below. This combination of IR leds and sensors,



plus an indicative light in each nose poke port and a pellet dispenser allows for the design of different behavioural paradigms by just changing Python code.

Figure 8 | 5 Choice serial reaction time test behavioural box. *A) Electronics schematic, depicting how BeeHive controls motors, IR sensors and different coloured LEDs for behavioural control. B) Data showing correct responses of two animals during one of the 5-CSRTT training phases (in this phase animals had to wait for a specific amount of time, until one of the yellow LEDs in the poke array turned on. Then they had to insert their nose into a hole containing the lit LED.*

Open Flexure stage controller (documentation available here)

Boards used: main board (1x), rotary encoders DB (3X), 28byj48 motor driver (3X), LCD panel (1x).

The Open Flexure stage controller module was developed as part of a larger project building an open source 2-photon microscope [52]. The module is a standalone system designed to get user manual input through rotary encoders to control the movement of an OpenFlexure (OF) Delta Stage [34]. The MB converts the input from each encoder into the delta movement coordinates, and provides feedback to the users via an I2C connection to an off-the-shelf Liquid Crystal Display (HD44780 LCD driver). The controller can be simplified for use with other non-delta (cartesian) stages, as long as there are compatible motor drivers and control libraries available. This system differs from the OpenFlexure original design as the electronics are greatly simplified, as the system does not use an onboard camera, and can operate entirely independently from other devices. Here,

BeeHive has added one more way to control OF devices, as well as real-time control and LCD readout, making the use of a computer to control OF devices optional.

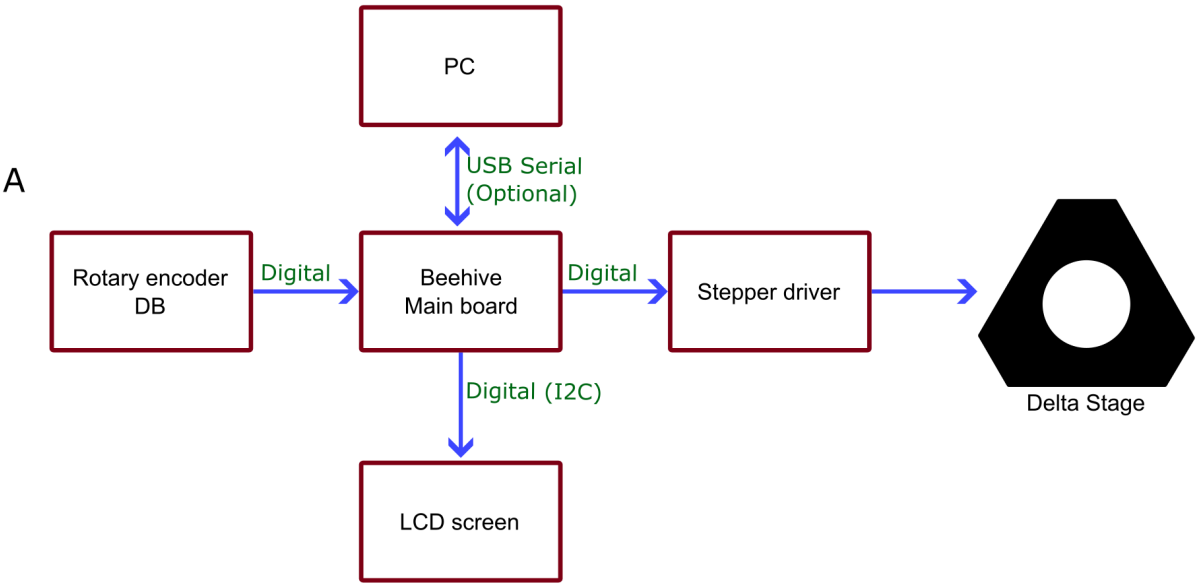


Figure 9 | A standalone controller for OpenFlexure's Delta Stage. Schematics showing how BeeHive is used to gather user manual input and convert it into delta stage movement.

Discussion

Our goal with BeeHive is not to reinvent the wheel, but rather fill existing gaps on existing electronic modules and systems available to scientists and developers. BeeHive is a modular electronics toolkit to both create research equipment, and to initiate researchers and graduate students in coding and electronics. We demonstrate that having a minimal set of rules to guide the BeeHive's development leads to a system that is sufficiently flexible for diverse applications (e.g. microscope stage control, behavioural measurements, different types of stimulators) while being a cohesive ecosystem for structured development and integration with other existing systems. Having a single, standardised platform allows multiple labs to collaborate more effectively, where researchers from different labs can contribute to shared hardware and software solutions, reducing the time and cost of developing new equipment. For example, a DB designed for controlling solenoids in one lab could be repurposed for an entirely different application in another lab, without

the need to develop new hardware from scratch (see “head-fixed setup” and “multiplexor” for one such example). As a result, labs can help each other overcome technical challenges, speeding up the development process and enhancing the overall research output. The common platform also makes it easier to standardise training for new lab members, as they can quickly get up to speed with the same hardware used across the department.

Another key aspect of BeeHive is the inclusion of a dedicated DB designed for teaching electronics and programming. This board serves a dual purpose: it not only provides a platform for learning fundamental electronics and coding concepts, but also acts as a testbed for validating code logic before applying it to more complex experimental setups. Researchers in training can use this DB to gain hands-on experience in a simplified environment, which helps to lower the barriers to entry for scientists without a strong engineering background. To our knowledge, there are no recent examples in the life sciences of hardware platforms fully dedicated to teaching both electronics and programming in an integrated way. The BeeHive system fills this gap by providing a unified platform that bridges the divide between scientific experimentation and the development of the underlying hardware used to conduct those experiments. By learning to build and test their own equipment, researchers are empowered to become more independent and creative in their experimental designs.

Furthermore, the open-source nature of BeeHive means that contributions are not confined to a single institution; external collaborators can also contribute to the development and refinement of the system, accelerating innovation. By providing a platform that integrates seamlessly with commercial systems, while offering the flexibility to develop custom solutions, BeeHive represents a significant advancement in the field of modular research hardware. It not only meets the immediate needs of individual labs but also fosters a collaborative, interdisciplinary research environment where resources are shared, and innovations are quickly disseminated.

Conclusion

In this manuscript, we have introduced BeeHive, a flexible, open-source electronics platform designed to assist researchers in creating custom scientific equipment while promoting electronics and coding education. While the focus of this paper has been on demonstrating neuroscience applications, such as behavioral setups, stimulus control, and data acquisition, the modular nature of BeeHive makes it adaptable for a wide range of scientific fields. For example, it could be applied to plant sciences, environmental monitoring, and other research that requires precise control over experimental hardware. The versatility of the system, coupled with its minimal design rules, allows

researchers to tailor it to their specific needs beyond neuroscience. By leveraging open-source principles, we aim to address key challenges in accessibility and reproducibility within scientific research. The system's ease of use, affordability, and flexibility position it as a valuable tool for fostering innovation and advancing open science across diverse disciplines. Future development efforts will focus on expanding the range of compatible DBs and enhancing integration with other open-source platforms, enabling more researchers to design, build, and share custom research tools globally.

References

1. Fantner GE, Oates AC. Instruments of change for academic tool development. Nat Phys. 2021;17: 421–424. doi:10.1038/s41567-021-01221-3

2. Pearce JM. Economic savings for scientific free and open source technology: A review. HardwareX. 2020;8: e00139. doi:10.1016/j.ohx.2020.e00139

3. Oellermann M, Jolles JW, Ortiz D, Seabra R, Wenzel T, Wilson H, et al. Open Hardware in Science: The Benefits of Open Electronics. Integr Comp Biol. 2022;62: 1061–1075.

4. Adafruit Industries, Unique & fun DIY electronics and kits. [cited 3 Feb 2026]. Available: https://www.adafruit.com/

5. SparkFun Electronics - SparkFun Electronics. [cited 3 Feb 2026]. Available: https://www.sparkfun.com/

6. Seeed Studio Bazaar, The IoT Hardware enabler. [cited 3 Feb 2026]. Available: https://www.seeedstudio.com/

7. Open Neuroscience. [cited 11 Feb 2026]. Available: https://open-neuroscience.com/en/

8. Sanders JI, Kepecs A. A low-cost programmable pulse generator for physiology and behavior. Front Neuroengineering. 2014;7. doi:10.3389/fneng.2014.00043

9. Chen X, Li H. ArControl: An Arduino-Based Comprehensive Behavioral Platform with Real-Time Performance. Front Behav Neurosci. 2017;11. doi:10.3389/fnbeh.2017.00244

10. Saunders JL, Wehr M. Autopilot: Automating behavioral experiments with lots of Raspberry Pis. bioRxiv. 2019; 807693. doi:10.1101/807693

11. Akam T, Lustig A, Rowland JM, Kapanaiah SK, Esteve-Agraz J, Panniello M, et al. Open-source, Python-based, hardware and software for controlling behavioural neuroscience experiments. Kemere C, Wassum KM, Kemere C, Siegle J, editors. eLife. 2022;11: e67846. doi:10.7554/eLife.67846

12. Bpod Wiki. [cited 12 Nov 2025]. Available: https://sanworks.github.io/Bpod_Wiki/

13. Longley M, Willis EL, Tay CX, Chen H. An open source device for operant licking in rats. PeerJ. 2017;5: e2981. doi:10.7717/peerj.2981

14. Wong CC, Ramanathan DS, Gulati T, Won SJ, Ganguly K. An automated behavioral box to assess forelimb function in rats. J Neurosci Methods. 2015;246: 30–37. doi:10.1016/j.jneumeth.2015.03.008

15. Steenbergen PJ. Response of zebrafish larvae to mild electrical stimuli: A 96-well setup for behavioural screening. J Neurosci Methods. 2018;301: 52–61. doi:10.1016/j.jneumeth.2018.03.002

16. BeeHive-org/BeeHive. BeeHive-org; 2025. Available: https://github.com/BeeHive-org/BeeHive

17. Muller E, Bednar JA, Diesmann M, Gewaltig M-O, Hines M, Davison AP. Python in neuroscience. Front Neuroinformatics. 2015;9. doi:10.3389/fninf.2015.00011

18. Most used languages among software developers globally 2024. In: Statista [Internet]. [cited 30 May 2025]. Available: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/

19. PCB Prototype & PCB Fabrication Manufacturer - JLCPCB. [cited 3 Feb 2026]. Available: https://jlcpcb.com/

20. China PCB Prototype & Fabrication Manufacturer - PCB Prototype the Easy Way. [cited 3 Feb 2026]. Available: https://www.pcbway.com/

21. OSH Park ~. [cited 3 Feb 2026]. Available: https://oshpark.com/

22. AISLER - Quick and affordable manufacturing for your Electronic Project. In: AISLER - Quick and affordable manufacturing for your Electronic Project [Internet]. [cited 3 Feb 2026]. Available: https://aisler.net

23. mikroBUS. In: MIKROE [Internet]. [cited 3 Feb 2026]. Available: http://www.mikroe.com/mikrobus

24. AZDelivery 3 x ESP32-DevKitC NodeMCU WiFi WLAN CP2102 ESP32-WROOM-32D IoT 2-In-1 Microcontroller Bluetooth Module Development Board compatible with Arduino including E-Book! : Amazon.co.uk: Electronics & Photo. [cited 3 Feb 2026]. Available: https://www.amazon.co.uk/AZDelivery-NodeMcu-CP2102-Development-including/dp/B074RGW2VQ

25. 1PCS ESP32 Development Board WiFi+Bluetooth Ultra-Low Power Consumption Dual Core ESP-32S ESP32-WROOM-32D ESP32-WROOM-32U ESP 32 - AliExpress 502. [cited 3 Feb 2026]. Available: https://www.aliexpress.com/item/1005004879572949.html?gatewayAdapt=glo2bra

26. 1. Getting started with MicroPython on the ESP32 — MicroPython latest documentation. [cited 27 July 2023]. Available: https://docs.micropython.org/en/latest/esp32/tutorial/intro.html

27. Installing — Arduino-ESP32 2.0.6 documentation. [cited 27 July 2023]. Available: https://espressif-docs.readthedocs-hosted.com/projects/arduino-esp32/en/latest/installing.html

28. Click boards™ | Modular Add-On Boards for Rapid Prototyping - MIKROE. [cited 3 Feb 2026].

Available: https://www.mikroe.com/click-boards

29. Impulse D. Drive Circuits. In: The Lee Co [Internet]. [cited 3 Dec 2025]. Available: https://www.theleeco.com/support-resources/engineering-tools/electrical-engineering/drive-circuits/

30. MicroPython - Python for microcontrollers. [cited 29 Sept 2020]. Available: http://micropython.org/

31. Beehive - BeeHive. [cited 3 Feb 2026]. Available: https://beehive-org.github.io/

32. BeeHive/workshops/2022_autum at master · BeeHive-org/BeeHive. In: GitHub [Internet]. [cited 3 Feb 2026]. Available: https://github.com/BeeHive-org/BeeHive/tree/master/workshops/2022_autum

33. LI-COR Environmental. In: LI-COR Environmental [Internet]. [cited 6 Mar 2026]. Available: https://www.licor.com/products/gas-analysis/LI-830-LI-850

34. McDermott S, Ayazi F, Collins J, Knapper J, Stirling J, Bowman R, et al. Multi-modal microscopy imaging with the OpenFlexure Delta Stage. Opt Express. 2022;30: 26377–26395. doi:10.1364/OE.450211

35. Peirce JW. PsychoPy—Psychophysics software in Python. J Neurosci Methods. 2007;162: 8–13. doi:10.1016/j.jneumeth.2006.11.017

36. Rosenberg M, Zhang T, Perona P, Meister M. Mice in a labyrinth show rapid learning, sudden insight, and efficient exploration. Mathis MW, Dulac C, Berman GJ, editors. eLife. 2021;10: e66175. doi:10.7554/eLife.66175

37. OpenCV - Open Computer Vision Library. [cited 9 Apr 2024]. Available: https://opencv.org/

38. open-neuroscience/pellet_dispenser. Open Neuroscience; 2025. Available: https://github.com/open-neuroscience/pellet_dispenser

39. adafruit/Adafruit-PWM-Servo-Driver-Library. Adafruit Industries; 2026. Available: https://github.com/adafruit/Adafruit-PWM-Servo-Driver-Library

40. Rado S. kroimon/Arduino-SerialCommand. 2026. Available: https://github.com/kroimon/Arduino-SerialCommand

41. KineMouse Wheel. [cited 30 May 2025]. Available: https://hackaday.io/project/160744-kinemouse-wheel

42. Warren RA, Zhang Q, Hoffman JR, Li EY, Hong YK, Bruno RM, et al. A rapid whisker-based decision underlying skilled locomotion in mice. Goldberg JH, Calabrese RL, Goldberg JH, editors. eLife. 2021;10: e63596. doi:10.7554/eLife.63596

43. Barkus C, Bergmann C, Branco T, Carandini M, Chadderton PT, Galiñanes GL, et al. Refinements to rodent head fixation and fluid/food control for neuroscience. J Neurosci Methods. 2022;381: 109705. doi:10.1016/j.jneumeth.2022.109705

44. Raiser G, Galizia CG, Szyszka P. A High-Bandwidth Dual-Channel Olfactory Stimulator for Studying Temporal Sensitivity of Olfactory Processing. Chem Senses. 2017;42: 141–151.

doi:10.1093/chemse/bjw114

45. Martelli C, Carlson JR, Emonet T. Intensity Invariant Dynamics and Odor-Specific Latencies in Olfactory Receptor Neuron Response. J Neurosci. 2013;33: 6285–6297. doi:10.1523/JNEUROSCI.0426-12.2013

46. LI-COR Environmental. In: LI-COR Environmental [Internet]. [cited 11 Feb 2026]. Available: https://www.licor.com/products/gas-analysis/LI-830-LI-850

47. LI-850 | Configuration grammar. [cited 11 Feb 2026]. Available: https://www.licor.com/support/LI-850/topics/grammar-01.html#About

48. Peña-Oliver Y, Buchman VL, Dalley JW, Robbins TW, Schumann G, Ripley TL, et al. Deletion of alpha-synuclein decreases impulsivity in mice. Genes Brain Behav. 2012;11: 137–146. doi:10.1111/j.1601-183X.2011.00758.x

49. Remmelink E, Chau U, Smit AB, Verhage M, Loos M. A one-week 5-choice serial reaction time task to measure impulsivity and attention in adult and adolescent mice. Sci Rep. 2017;7: 42519. doi:10.1038/srep42519

50. jackpkenn. jackpkenn/PelletDispenser. 2025. Available: https://github.com/jackpkenn/PelletDispenser

51. open-neuroscience/pellet_dispenser. Open Neuroscience; 2025. Available: https://github.com/open-neuroscience/pellet_dispenser

52. Open-2-Photon-Microscope. In: GitHub [Internet]. [cited 12 Nov 2025]. Available: https://github.com/Open-2-Photon-Microscope

Author's contributions

Contributions follow Contributor Role Taxonomy (CRediT):

Ihor Sobianin: Conceptualization, Data Curation, Formal Analysis, Methodology, Software, Validation, Visualization, Writing - Original Draft Preparation
Mikkel Roald-Arbøl: Data Curation, Formal Analysis, Investigation, Methodology, Validation, Visualization, Writing - Original Draft Preparation
Solomon Gitau Ngotho: Data Curation, Methodology, Software, Validation
Lydia Ellison: Data Curation, Methodology, Validation, Visualization, Writing - Original Draft Preparation
Shahd Al Balushi: Data Curation, Methodology, Investigation, Validation, Visualization, Writing - Original Draft Preparation
Alejandra Carriero: Data Curation, Methodology, Investigation, Validation, Writing - Original Draft Preparation
Marcus Burnell Spector: Data Curation, Methodology, Investigation, Software, Validation, Writing - Original Draft Preparation
Eglantine Vignal: Investigation, Methodology, Software
Carla Lemos Perinetti: Data Curation, Investigation
Moira Eley: Supervision, Writing - Review and Editing
Maria Cozan: Investigation, Writing - Original Draft Preparation
Cansu Demirbatir: Investigation, Methodology, Data Curation
Sina E Dominiak: Investigation, Data Curation, Software, Validation, Visualization, Writing - Original Draft Preparation
Estelle Moubarak: Supervision, Investigation, Data Curation
Filip Janiak: Supervision
George Kemenes: Supervision, Resources
Leon Lagnado: Supervision, Resources
Sylvia Schröder: Supervision, Resources, Writing - Original Draft Preparation
Sarah King: Supervision, Resources, Writing - Original Draft Preparation
Thomas Nowotny: Supervision, Resources, Writing - Review & Editing
Tom Baden: Supervision, Resources, Writing - Review & Editing
Miguel Maravall: Supervision, Resources, Writing - Review & Editing
Andre Maia Chagas: Conceptualization, Data Curation, Formal Analysis, Methodology, Software, Validation, Visualization, Resources, Supervision, Writing - Original Draft Preparation

This preprint was submitted under the following conditions:

- The authors declare that the necessary Terms of Free and Informed Consent of participants or patients in the research were obtained and are described in the manuscript, when applicable.
- The authors declare that the preparation of the manuscript followed the ethical norms of scientific communication.
- The authors declare that they are aware that they are solely responsible for the content of the preprint and that the deposit in SciELO Preprints does not mean any commitment on the part of SciELO, except its preservation and dissemination.
- The authors declare that the data, applications, and other content underlying the manuscript are referenced.
- The deposited manuscript is in PDF format.
- The authors declare that the research that originated the manuscript followed good ethical practices and that the necessary approvals from research ethics committees, when applicable, are described in the manuscript.
- The authors declare that once a manuscript is posted on the SciELO Preprints server, it can only be taken down on request to the SciELO Preprints server Editorial Secretariat, who will post a retraction notice in its place.
- The authors agree that the approved manuscript will be made available under a Creative Commons CC-BY license.
- The submitting author declares that the contributions of all authors and conflict of interest statement are included explicitly and in specific sections of the manuscript.
- The authors declare that the manuscript was not deposited and/or previously made available on another preprint server or published by a journal.
- If the manuscript is being reviewed or being prepared for publishing but not yet published by a journal, the authors declare that they have received authorization from the journal to make this deposit.
- The submitting author declares that all authors of the manuscript agree with the submission to SciELO Preprints.