ACES Asian Chemical Editorial Society

doi.org/10.1002/asia.202300380

CHEMISTRY
AN ASIAN JOURNAL
Research Article

www.chemasianj.org

Special Collection

# LABS: Laboratory Automation and Batch Scheduling – A Modular Open Source Python Program for the Control of Automated Electrochemical Synthesis with a Web Interface

Maximilian M. Hielscher+,[a] Maurice Dörr+,[a] Johannes Schneider,[a] and Siegfried R. Waldvogel*[a, b]

*Dedicated to Prof. Dr. Dennis Curran on the occasion of his 70th birthday.*

**Abstract:** With LABS, an open source Python-based lab software is established that enables users to orchestrate automated synthesis setups. The software consists of a user-friendly interface for data input and system monitoring. A flexible backend architecture enables the integration of multiple lab devices. The software allows users to easily modify experimental parameters or routines and switch between different lab devices. Compared to previously published projects, we aim to provide a more widely applicable and easily customizable automation software for any experimental setup. The usefulness of this tool was demonstrated in the oxidative coupling of 2,4-dimethylphenol to the corresponding 2,2'-biphenol. In this context, the suitable electrolysis parameters for flow electrolysis were optimized by way of design of experiments.

## Introduction

In recent years, electro-organic chemistry has developed into a cost-friendly synthesis alternative compatible with green chemistry principles[1] for the preparation of small molecules, the liberation of value-added products from waste streams, and the sustainable production of platform oxidizers.[2] The use of electrons as a reagent often facilitates the generation of radical intermediates, which can offer shortcuts from multi-step reaction sequences to a single direct step.[3,4] The direct (or mediated) electron transfer to or from the electrode often allows a metal-free catalytic reaction to be carried out.[4,5] Typical setups for electrochemical reactions are either batch-type mode,[6] or electrochemical flow cells through which the electrolyte is pumped, either in a single pass, or pumped multiple times through the flow cell using a reservoir.[7] Electrochemical reactions in flow setups often exhibit better scalability and higher space-time yields, and the (typically) shorter interelectrode distances allow more sustainable syntheses using significantly smaller amounts of supporting electrolyte.[8,9] However, electrochemical reactions in flow require more peripheral equipment such as pumps, valves, and fraction collectors, and are therefore often operated with only a single electrolyzer, whereas for screening in batch a variety of systems exist that allow parallel operation of several reactions. A review article for high-throughput screening in electro-organic synthesis was compiled by J. C. Vantourout and co-workers.[10] With regards to electrochemical flow cells, methods like 3D printing allow prototyping and miniaturization to speed up development.[11] Additional reaction parameters for electrolysis in flow, such as flow rate, can complicate the optimization. This challenge can be overcome systematically by methodological approaches such as design of experiments or Bayesian optimization.[12]

Automation of the setup for electrolysis in flow can help to increase the efficiency of screening runs, while at the same time observables such as the cell voltage can be continuously monitored. Analysis of this data can provide information on reproducibility and status of the electrodes, e.g., fouling or corrosion events.[17] Due to the growing importance of electronic laboratory notebooks, flexible control loops capable of automated processing of measurement data are required.

Automated systems are increasingly being established not only for commercial applications, but also as free and open source software.[18] In particular, if the functional scope of the control software is to be changed, open source projects offer great advantages over closed source. One such advantage is that changes can be implemented, tested, and made generally available to the community by the user. Since 2016 several groups have reported projects dealing with laboratory automation (Figure 1). These are commonly built using LabView or

[a] M. M. Hielscher,+ M. Dörr,+ J. Schneider, Prof. Dr. S. R. Waldvogel
Department of Chemistry
Johannes Gutenberg University
Duesbergweg 10–14, 55128 Mainz (Germany)
E-mail: waldvogel@uni-mainz.de

[b] Prof. Dr. S. R. Waldvogel
Institute of Biological and Chemical Systems – Functional Molecular Systems (IBCS-FMS)
Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen (Germany)

[+] These authors contributed equally to this work.

Supporting information for this article is available on the WWW under https://doi.org/10.1002/asia.202300380

This manuscript is part of an Indo-German Workshop (IGW) special collection.

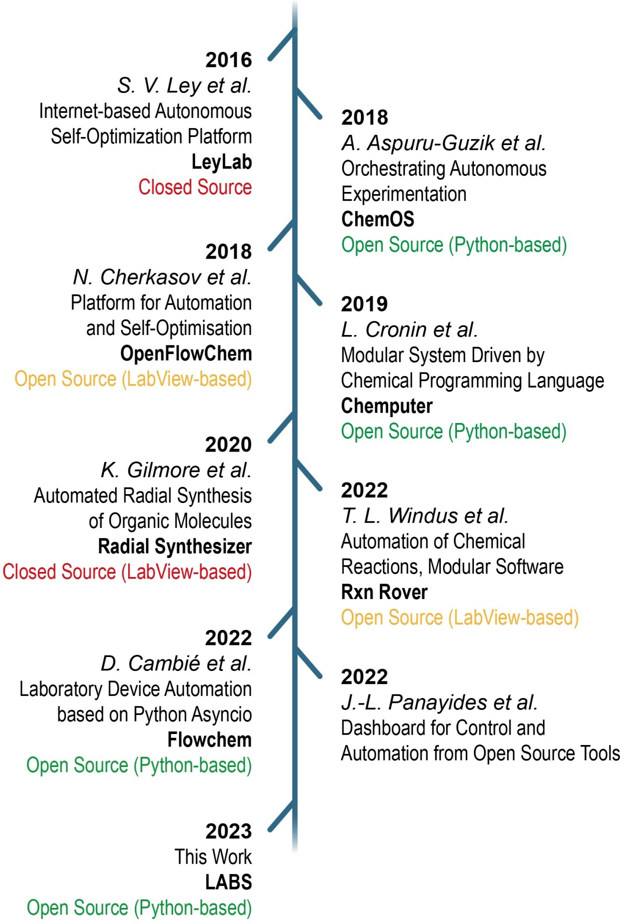

2016
S. V. Ley et al.
Internet-based Autonomous Self-Optimization Platform
LeyLab
Closed Source
2018
A. Aspuru-Guzik et al.
Orchestrating Autonomous Experimentation
ChemOS
Open Source (Python-based)
2018
N. Cherkasov et al.
Platform for Automation and Self-Optimisation
OpenFlowChem
Open Source (LabView-based)
2019
L. Cronin et al.
Modular System Driven by Chemical Programming Language
Chemputer
Open Source (Python-based)
2020
K. Gilmore et al.
Automated Radial Synthesis of Organic Molecules
Radial Synthesizer
Closed Source (LabView-based)
2022
T. L. Windus et al.
Automation of Chemical Reactions, Modular Software
Rxn Rover
Open Source (LabView-based)
2022
D. Cambié et al.
Laboratory Device Automation based on Python Asyncio
Flowchem
Open Source (Python-based)
2022
J.-L. Panayides et al.
Dashboard for Control and Automation from Open Source Tools
2023
This Work
LABS
Open Source (Python-based)

**Figure 1.** Recent projects towards automated synthesis platforms.[13,14–16]

Python and the majority were published in an open source fashion. While these projects are all very inspiring and impressive contributions, they unfortunately did not lead to the sought after communalization and widespread availability of automated reaction setups. Our initial contributions to this field were published in 2021; a semi-automated setup for electrolysis using a reservoir with variable electrolysis parameters in flow cells with an automated fraction collector.[19] However, no purging step was possible with this setup and the control was implemented in LabView as a closed source file. In contrast to single setup automation, Aspuru Guzik *et al.* emphasized the orchestration of multiple setups and the use of AI informed experimental planning. However, they did not report the actual code that controlled the experimental setup itself.[14,16] Cronin *et al.* elegantly showed the applicability of a formal and general description of chemical experiments in a machine-readable format.[20] As with the previously mentioned work, the focus was not on the communication to the physical devices.[15] More recently, Cambié *et al.* published their project Flowchem on GitHub; it is the only open source code among those discussed that is actively maintained in 2023.[21,22] Its aim was to expose an interface that makes the devices' functionality available to other software (a RESTful API) for every given device in a setup, wherein the flow of commands in the experiment has to be implemented by the user in a programming language of their choice.[22] This is undeniably valuable to users with programming skills, however, for the uninitiated it can be a roadblock to implementing these systems. While the *Chemputer* abstracts the setup on an even higher level and *Flowchem* on a similar level than LABS, many of the above-mentioned projects lack generalizability or abstraction layers in regard to the devices used in the setups. Their accordingly high degree of specialization is a disadvantage for broader application, and adaptation and subsequent application of the experimental setup is often difficult, especially if software components need to be refactored on a major scale. Furthermore, the use of LabView as basis for the software adds upfront cost and the need to familiarize oneself with LabView in order to run and adapt the programs. In our project, we aim to close this gap by providing a free and open source software base for sequence-based control of arbitrary experimental setups. At the same time, we want to provide the user with a modern, centralized user interface where experimental conditions can be planned and running experiments can be monitored. Furthermore, an administrative view allows for the adjustment of general parameters (e.g., a flush step between two different runs or within one run).

Although automated chemistry is of interest to the wider chemical community, it is outside of many traditional chemists' skillset to operate and adapt custom solutions to these software packages. Therefore, we aim to provide a package that enables the facile building of automated systems from the supported devices with negligible prior knowledge needed. Since only the configuration for the experimental setup has to be defined, little to no programming knowledge is required.

## Results and Discussion

Our approach to implement the system is based on two components (Figure 2): the web-based user interface provides an interface for the user while the backend deals with the logic for controlling the setups. The backend communicates with the connected devices based on the asynchronous network framework *Twisted*, and communicates with the user interface in a many-to-one relationship.[23] The use of *Twisted* for device communication has been pioneered by Ley *et al.* with their project octopus.[24] This framework is especially well suited, since it is designed to allow a simple implementation of networking protocols independent of the physical connection like RS232 and Ethernet. A setup class organizes the execution of the scheduled experiments and can stop the sequence if necessary. The experiment sequence is provided by the experiment class, which reads the commands to be executed from a configuration file in YAML format. This can be changed by the user, and individual sequences can be combined into larger sequences, so that any intermediate steps, such as purging routines, can be actualized in addition to the actual experiment to be carried out. A device driver class provides generic adapters for communication with the devices, such as pumps or power supply units, which provide device-specific functions.

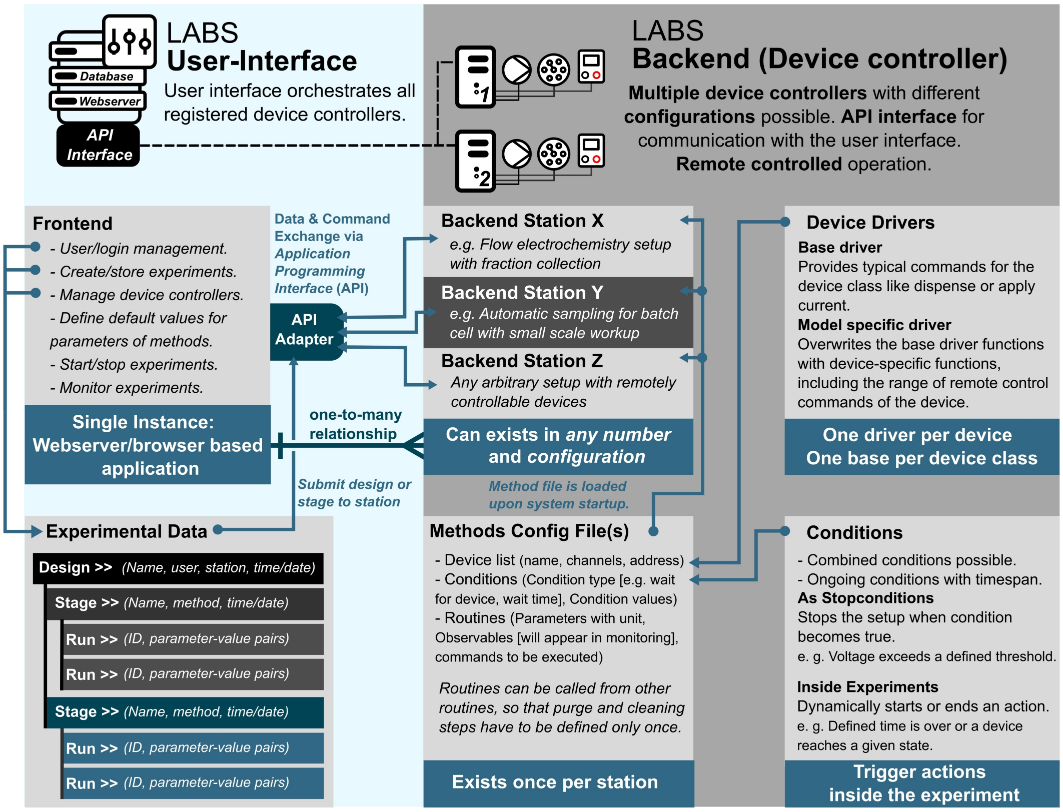

LABS
User-Interface
User interface orchestrates all registered device controllers.
Database
Webserver
API Interface
LABS
Backend (Device controller)
Multiple device controllers with different configurations possible. API interface for communication with the user interface. Remote controlled operation.
1
2
Frontend
- User/login management.
- Create/store experiments.
- Manage device controllers.
- Define default values for parameters of methods.
- Start/stop experiments.
- Monitor experiments.
Single Instance: Webserver/browser based application
Data & Command Exchange via Application Programming Interface (API)
API Adapter
one-to-many relationship
Submit design or stage to station
Backend Station X
e.g. Flow electrochemistry setup with fraction collection
Backend Station Y
e.g. Automatic sampling for batch cell with small scale workup
Backend Station Z
Any arbitrary setup with remotely controllable devices
Can exists in any number and configuration
Method file is loaded upon system startup.
Device Drivers
Base driver
Provides typical commands for the device class like dispense or apply current.
Model specific driver
Overwrites the base driver functions with device-specific functions, including the range of remote control commands of the device.
One driver per device
One base per device class
Experimental Data
Design >> (Name, user, station, time/date)
Stage >> (Name, method, time/date)
Run >> (ID, parameter-value pairs)
Run >> (ID, parameter-value pairs)
Stage >> (Name, method, time/date)
Run >> (ID, parameter-value pairs)
Run >> (ID, parameter-value pairs)
Methods Config File(s)
- Device list (name, channels, address)
- Conditions (Condition type [e.g. wait for device, wait time], Condition values)
- Routines (Parameters with unit, Observables [will appear in monitoring], commands to be executed)
Routines can be called from other routines, so that purge and cleaning steps have to be defined only once.
Exists once per station
Conditions
- Combined conditions possible.
- Ongoing conditions with timespan.
As Stopconditions
Stops the setup when condition becomes true.
e. g. Voltage exceeds a defined threshold.
Inside Experiments
Dynamically starts or ends an action.
e. g. Defined time is over or a device reaches a given state.
Trigger actions inside the experiment

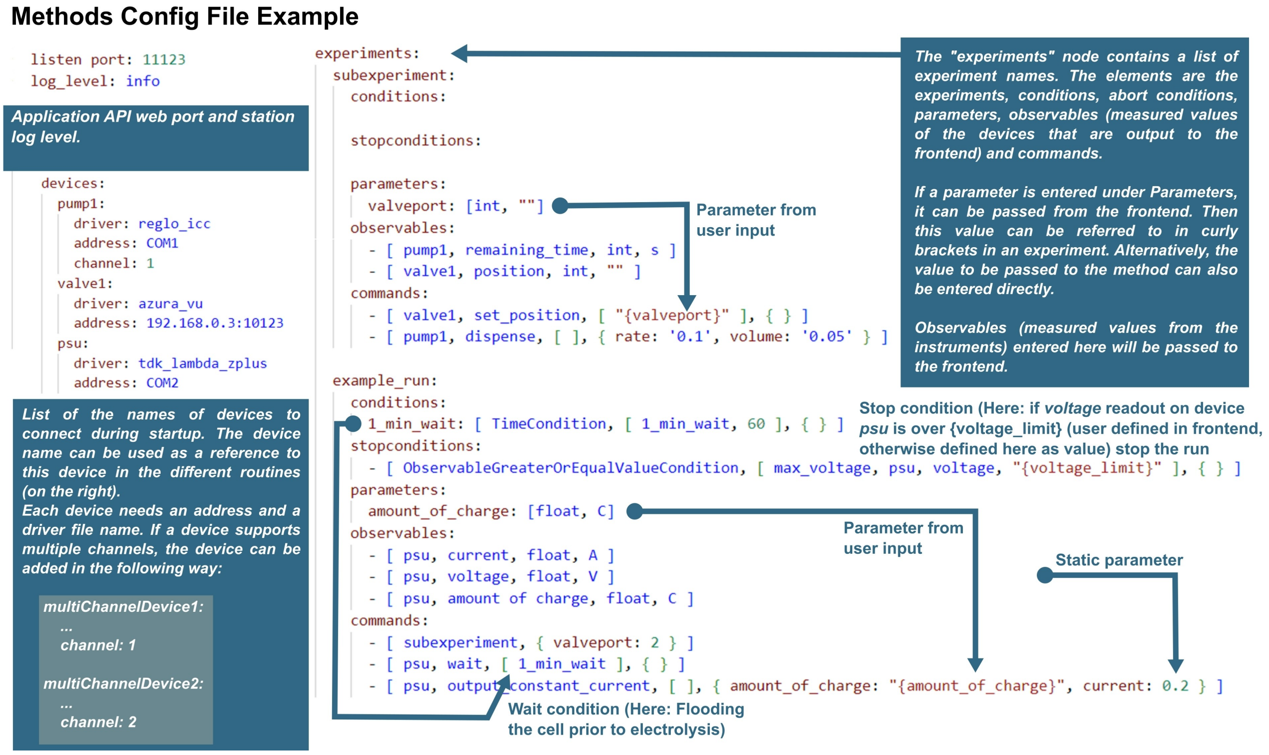
**Figure 2.** Representation of the link between the user frontend and the backends at the experimental stations.

The adapter then accesses a driver file that contains the model-specific control commands and overwrites the generic functions of the adapter so that any device can be used, provided they can map the required basic functions. The user interface is a *Flask*-based[25] Python application that provides a web management platform for users and experimental parameters, as well as monitoring of the various backends. Flask is a popular choice for web development because of its simplicity and flexibility, allowing developers to quickly create scalable and efficient web applications. The rich ecosystem of Flask already provides additional functionalities such as database integration, authentication, API development. It provides a web server so that the user interface can be accessed with common web browsers.

The user interface is designed in a modular fashion, the functionality modules are organized as so-called blueprints, so far there is the module "Users", which provides the functionality to create different users in a database, to authenticate and authorize them to use the user interface. The module "Monitoring" offers an overview page of all integrated backends with their current state, as well as a detailed view, which displays the queued experiments of a selected station in table form. Also displayed there are the observables (which are defined in the configuration file of the station) as line plots (voltage, current, temperature), progress bars (remaining dosing time of the pump, applied charge), or as data tables (positions of the switching valves over time). The module "Experiments" allows the hierarchical management of the experimental data. First, an experimental design is created, which is assigned to a user and a backend. In this design, individual stages can be created, each of which is linked to a method (the experimental sequence to be carried out) provided by the corresponding backend. For example, additional cleaning steps can be added between experiments. Individual runs can then be created within a stage. The stage overview page offers a down- and upload function (Excel) to add, update, or delete the runs of a stage. Any number of backends can be added on an administrative level. The currently loaded configuration can be downloaded from the stations to the user interface's database. For each method known at the particular device, parameters can be set as static and do not have to be defined for each run (e.g., the volume of purging solvent). This should enhance reproducibility in multi-user scenarios.

The configuration syntax (Figure 3) was designed to implement the following features: multichannel devices can be used as one device with the channel as a parameter in each command or each channel as a separate device. Experiments can contain other experiments to allow high reusage of common processes, like purging a compartment or tube. A full description of the configuration file can be found in the supporting information.

To test the sequence of the methods in a dry run, we created virtual "dummies" of the devices used. These behave like the devices in terms of device communication. The readouts can be counting down times or randomized values, the cell voltage of a current source, etc. As of now, the core functions for the devices we use are programmed into the



Methods Config File Example

```
listen port: 11123
log_level: info
```

Application API web port and station log level.

```
devices:
  pump1:
    driver: reglo_icc
    address: COM1
    channel: 1
  valve1:
    driver: azura_vu
    address: 192.168.0.3:10123
  psu:
    driver: tdk_lambda_zplus
    address: COM2
```

List of the names of devices to connect during startup. The device name can be used as a reference to this device in the different routines (on the right).
Each device needs an address and a driver file name. If a device supports multiple channels, the device can be added in the following way:

```
multiChannelDevice1:
  ...
  channel: 1

multiChannelDevice2:
  ...
  channel: 2
```

```
experiments:
  subexperiment:
    conditions:

    stopconditions:

    parameters:
      valveport: [int, ""]
    observables:
      - [ pump1, remaining_time, int, s ]
      - [ valve1, position, int, "" ]
    commands:
      - [ valve1, set_position, [ "{valveport}" ], { } ]
      - [ pump1, dispense, [ ], { rate: '0.1', volume: '0.05' } ]

  example_run:
    conditions:
      1_min_wait: [ TimeCondition, [ 1_min_wait, 60 ], { } ]
    stopconditions:
      - [ ObservableGreaterOrEqualValueCondition, [ max_voltage, psu, voltage, "{voltage_limit}" ], { } ]
    parameters:
      amount_of_charge: [float, C]
    observables:
      - [ psu, current, float, A ]
      - [ psu, voltage, float, V ]
      - [ psu, amount of charge, float, C ]
    commands:
      - [ subexperiment, { valveport: 2 } ]
      - [ psu, wait, [ 1_min_wait ], { } ]
      - [ psu, output_constant_current, [ ], { amount_of_charge: "{amount_of_charge}", current: 0.2 } ]
```

The "experiments" node contains a list of experiment names. The elements are the experiments, conditions, abort conditions, parameters, observables (measured values of the devices that are output to the frontend) and commands.

If a parameter is entered under Parameters, it can be passed from the frontend. Then this value can be referred to in curly brackets in an experiment. Alternatively, the value to be passed to the method can also be entered directly.

Observables (measured values from the instruments) entered here will be passed to the frontend.

Parameter from user input

Stop condition (Here: if *voltage* readout on device *psu* is over {voltage_limit} (user defined in frontend, otherwise defined here as value) stop the run

Parameter from user input

Static parameter

Wait condition (Here: Flooding the cell prior to electrolysis)

**Figure 3.** Configuration file example. Left: General settings and device configuration; demonstrated features: channel of a multichannel device is treated as a single device, COM- and TCP/IP ports can be used. Top right: First experiment named *subexperiment* (demonstrated features: experiment has a variable parameter named *valveport*). Bottom right: Second experiment named *example_run* (demonstrated features: condition 1_min_wait used in wait command, stop-condition utilized to cancel experiment if voltage is above configurable threshold, parameter valve port handed to *subexperiment*).

dummies. The function for generating random numbers can be coupled with a user-defined function that simulates, for example, a time-dependent increase in voltage. In this way, more complex stop conditions for the automated run observation can be tested efficiently prior to the actual experiment. For the devices used in this project we provide virtual devices in a simple framework. We employed this to test changes in the configuration before running it in the laboratory. This "virtual experimental setup" can also be integrated into the user interface to control it. This facilitates the further development of the user interface, as no real devices are needed to test new features.

Due to our experience with studying dehydrogenative C–C coupling of and with phenols, we chose the anodic oxidative homocoupling reaction of 2,4-dimethylphenol as an example for the first test runs.[19,26,27,28] This electrochemical transformation to the corresponding 2,2'-biphenol is particularly challenging as it is very prone to side reactions, such as the formation of polycyclic by-products.[28,29] Therefore, our first experimental setup (Figure 4) consisted of a dead-end selector switching valve (*Knauer*) to which 2,4-dimethylphenol, pyridine, hexafluoro-isopropanol (HFIP), and isopropanol reservoirs were connected. The common outlet connects to one channel of a four-channel Reglo ICC peristaltic pump with independent channel control (*Ismatec*) into a mixing chamber, in our case a 50 mL *Schott* glass screw-cap bottle with a lid with a plug-in insert for four 1/8" OD tubes. An additional hole was drilled in the center of the lid to allow flushing with compressed air. The compressed air was set to 1.5 bar using a pressure regulator and fed in via a switchable normally closed solenoid valve. An Arduino Uno microcontroller (*Arduino*) opens and closes the valve via a relay. The reservoir was clamped at a slight angle to enable full drainage via the second channel of the pump. This channel leads to a flow electrolysis cell with a 2×6 $cm^2$ anode made of boron-doped diamond on silica and a stainless-steel cathode block with temperature control. A spacer made of PTFE adjusted the electrode gap to 0.25 mm.[9] The outlet of the flow cell was connected to the common port of another switching valve, from where the flow can be directed back to the reservoir, to a waste canister, or to a fraction collector. The fraction collector used was a low-cost Lambda Omnicoll fraction collector (*Lambda Instruments*), which can be operated with different sized collection vessels.

In previous studies on electrochemical conversions in flow with active electrode areas of 2×6 $cm^2$ peristaltic pumps were typically used. These offer the possibility to work in divided cells with separate channels for anolyte and catholyte solutions, or to operate several cells in parallel with one pump. In the past, we used PharMed® BPT tubing as peristaltic pump tubing, which has a high mechanical stability but only moderate chemical stability. We therefore also tested peristaltic tubing made of fluororubber (FPM, Viton® *Optubus*). However, these were not mechanically stable in the original cassettes that came with the pumps and ruptured within the first few minutes.

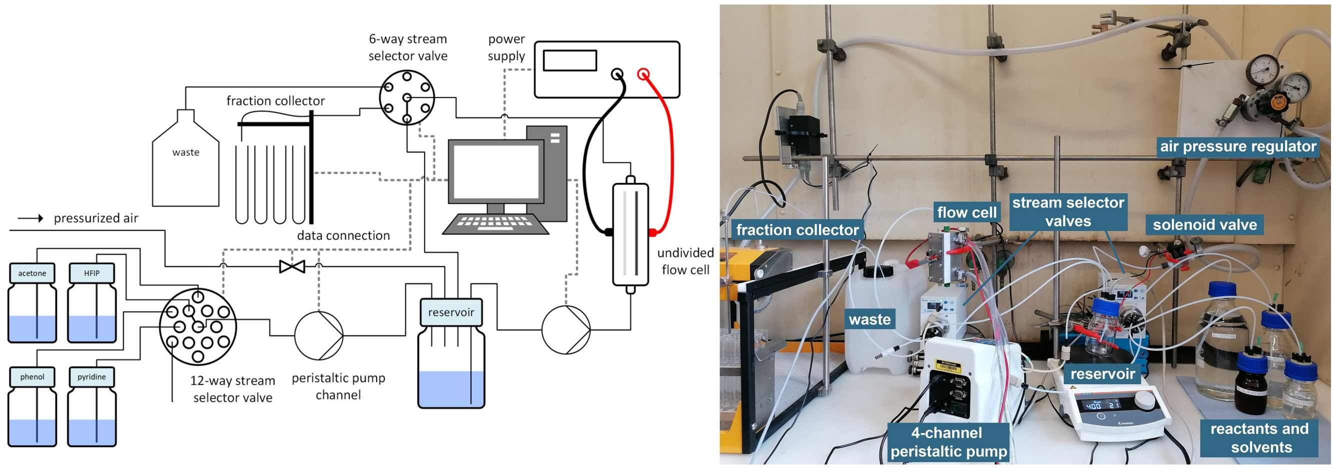


**Figure 4.** Left: Scheme of the automated flow setup enabling continuous and cyclic electrolysis. Right: Picture of the setup in use, the components are labelled. Electrolyte flow goes from right to left. Picture on the right by J. Schneider.

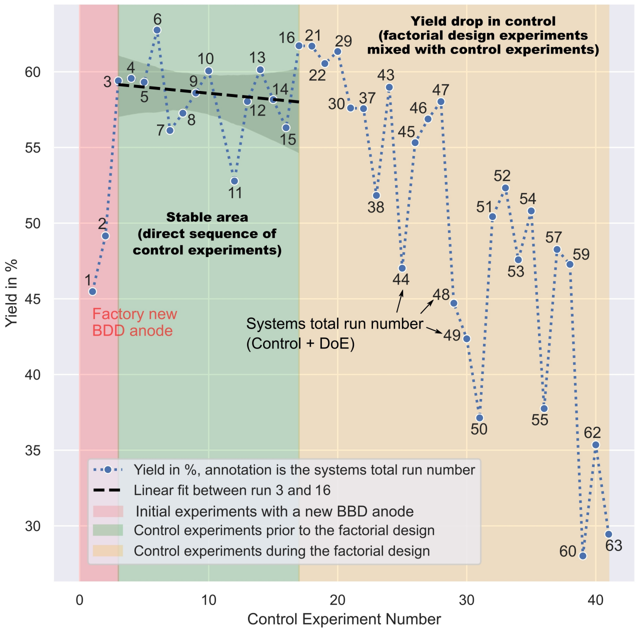
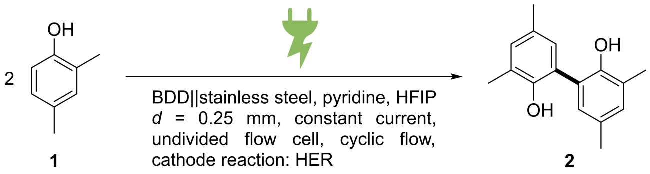


**Figure 5.** Yield of the control experiment with increasing number of experiments of the automation system. The annotations at the markers indicate the actual run number of the data point. In the green highlighted area, the yields of the consecutive control experiments are plotted. In the brown highlighted area, the yields of the control experiments that were performed between the experiments for the DoE design are plotted. The number next to the markers indicates the total consecutive experiment number associated with the data point.

Consequently, we equipped the pump with PharMed® BPT tubing for the initial experiments (Scheme 1).

We started with a test series of the same electrolysis conditions to investigate the overall behavior of the system in terms of yield (Conditions: $j=10$ mA/cm$^2$, $Q=0.9$ $F$, $v=1.99$ mL/min (15 × Single pass rate), $C(\mathbf{1})=1.25$ mol/L, vol.%(Pyridine) = 5%, $T=25$ °C). This included a purge phase in which the system was purged with acetone then primed with HFIP. The electrolyte was then prepared in the reservoir, electrolyzed in cyclic flow, and collected by the fraction collector.

By starting with a new, pristine BDD anode, we observed a sharp increase in yield in the first three experiments (red area, Figure 5). We attribute this to the known change in surface termination from H-type to OH-termination.[30] Runs 7, 11, and 15 were each the first run after the cell was idle overnight during the initial test phase where the interface was open to the air. From previous studies, deposit formation (fouling) is known to occur while electrolysis, and its modification by oxidation may contribute to the observed effect.[31] The system subsequently behaved in a largely stable manner over 15 electrolysis reactions with respect to the yield of coupling product (green area). We equipped the pump with new peristaltic tubing and began submission of all runs necessary for the factorial design Table 1) in sections mixed with the control experiments (orange area). The system initially continued to behave stable in terms of yield, but a steady downward trend was observed after control experiment 30. The anode did not show any deposit formation or fouling during a visual inspection, and it was concluded that the drift could be explained by the deterioration of the peristaltic tubing.


OH
2
1
BDD||stainless steel, pyridine, HFIP
d = 0.25 mm, constant current, undivided flow cell, cyclic flow, cathode reaction: HER
OH
OH
2

**Scheme 1.** Anodic dehydrogenative C–C homocoupling reaction of 2,4-dimethylphenol (**1**) to 3,3',5,5'-Tetramethyl-2,2'-biphenol (**2**).

Since our default peristaltic tubing choice was not suitable for continuous operation, we equipped the peristaltic pump with a new cassette that provided control over the contact pressure of the tubing. This allowed us to use the fluorocarbon rubber (FPM) for a longer operating time. We repeated the experimental design but halved the concentration range from 0.75–1.25 mol/L to 0.38–0.63 mol/L. This forced us to reduce the flow multiplier in the high setting from 15 to 10 times, as the

**Table 1.** Fractional factorial $2^{6-2}$ design for the investigation of the homocoupling of 2,4-dimethylphenol. Each experiment was performed in duplicate, with a total of two central points.

| Parameter | Unit | Low level | High level |
|---|---|---|---|
| Current density | $mA/cm^2$ | 20 | 40 |
| Applied charge | *F* | 0.70 | 0.90 |
| Flowrate multiplier[a] | – | 5 | 15 |
| Concentration of 2,4-dimethylphenol | mol/L | 0.75 | 1.25 |
| Concentration of pyridine | vol.% | 5 | 10 |
| Temperature | °C | 10 | 25 |

[a] We calculated the flowrate needed for a single pass electrolysis and multiplied it by this parameter value.

flow rates were otherwise greater than the pumps could handle. The 32 experiments of the design were divided into 4 blocks of eight experiments each. Three central point experiments were performed before each block. For this, a new tube was inserted into each pump channel. The dosing tube was calibrated to 1 mL/min, and the tube at the flow cell was calibrated to 6 mL/min. After each block of 11 experiments, the flow rate was determined, the tubing was changed and recalibrated. Since flow rate deviations to lower but also higher flow rates were determined, we determined a correction factor for all experiments. This was determined from a linear fit between the actual flow rate before and after each block of experiments (See SI for more information). The concentrations of **1** and the yields of **2** were corrected using this factor. In addition, the deviation of the dosing tube was included as a covariate in the regression and analysis of variance.

No parameter could be excluded by ANOVA from the obtained regression model ($R^2 = 74\%$) with $\alpha = 10\%$, however, current density, charge quantity, and the flow rate multiplier are below the significance threshold. However, these are involved in two-parameter interactions. The model obtained, subject to the corrections discussed, shows values in agreement with our previous studies with respect to the main effects and the curvature analysis (Figure 6).[27]

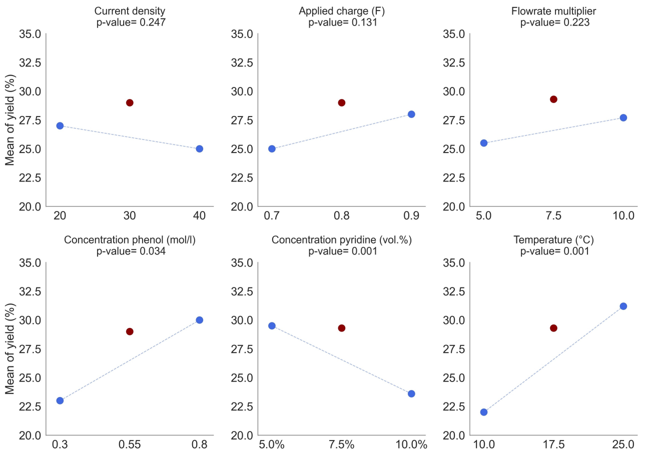


**Figure 6.** Main effects plot for the yield of **2**. The yield was determined using GC-FID with *n*-octylbenzene as internal reference material.

The optimum of the applied charge is equal to the central point of the model at 0.8 *F*. Higher concentrations of **1** favor the dehydrodimerization reaction, in agreement with the previously performed reactions at 1.25 mol/l of phenol **1**. The positive effect of the higher temperature could be an artifact since there was not sufficient condensation of the solvent in the reservoir. Turbulence in the tube with the cathodically generated hydrogen facilitates the evaporation of the solvent.

We have refrained from expanding the model to a CCD due to the poor suitability of the pump. Nonetheless, automation with LABS has worked successfully. In our next project, we want to implement a driver for piston pumps to replace the peristaltic pumps.

## Conclusion

We have developed and successfully applied LABS, a Python-based software for automating laboratory setups. It can simultaneously control any number of devices, to run any combination of supported devices, in this work an electrochemical flow setup capable of single-pass and cyclic electrolysis. The focus of this project goes beyond the previously published frameworks. With only the configuration of the setup and experiments left to the user, we enable scientists with less programming experience to get in touch with automated synthesis. A centralized user front-end allows the submission of experiments and the monitoring of any number of experimental stations. The ability to implement stop conditions allows for safe, unattended operation. The open programming allows the user to customize LABS to use additional sensors or measurement devices for their specific use case to enable more complex feedback loops. The program code for front and backend has been published as open source on GitHub.[31–33] The continuous operation has enabled us to increase our sample throughput for electrosynthetic screening in flow cells enormously.

This project lays the foundation for future implementation of advanced autonomous systems, including methods like Bayesian optimization. As such we will base upcoming projects on LABS and gradually support more devices not limited to but showcased in electrochemical setups. The use of an API interface for communication between the two modules also allows the control of third stations, provided that they offer corresponding API interfaces. Our main goal is to make LABS easier accessible and widely applicable in many areas of chemistry. Enabling future releases of LABS to be compatible with *Cronins* χDL[20,34] for example, would vastly improve the accessibility of LABS and the reusability of procedures run by LABS driven setups.

## Data and materials availability

Full details of the experimental methods and equipment used are given in the electronic supporting information. The program code for the controller was published in three separate repositories: *LABS-Backend*,[32] *LABS-Frontend*,[31] and *LABS-DeviceDummies*.[33]

## *Acknowledgements*

*Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under FOR 2982/1–UNODE (WA1276/24-1). Support by Forschungsinitiative Rheinland-Pfalz within the frame of SusInnoScience was very helpful. We would like to express our sincere thanks to R. M. Hamill for the effort she put into improving the language of this manuscript. We would like to thank F. Weyer for fruitful discussions regarding programming. Open Access funding enabled and organized by Projekt DEAL.*

## *Conflict of Interests*

*The authors declare no conflict of interest.*

## Data Availability Statement

The data that support the findings of this study are openly available in LABS-Backend: Beta-Publication_submission at https://doi.org/10.5281/zenodo.7879589, reference number 7879589, Labs-Frontend: Beta-Publication submission at https://doi.org/10.5281/zenodo.7879586, reference number: 7879586, LABS-DeviceDummys at https://doi.org/10.5281/zenodo.7879590, reference number 7879590.

**Keywords:** electrochemistry · automation · anodic oxidation · cheminformatics · phenol coupling reaction

[1] a) P. Anastas, N. Eghbali, *Chem. Soc. Rev.* **2010**, *39*, 301–312; b) B. A. Frontana-Uribe, R. D. Little, J. G. Ibanez, A. Palma, R. Vasquez-Medrano, *Green Chem.* **2010**, *12*, 2099.

[2] a) A.-K. Seitz, P. J. Kohlpaintner, T. van Lingen, M. Dyga, F. Sprang, M. Zirbes, S. R. Waldvogel, L. J. Gooßen, *Angew. Chem. Int. Ed.* **2022**, *61*, e202117563; *Angew. Chem.* **2022**, *134*, e202117563; b) N. Kurig, J. Meyers, F. J. Holzhäuser, S. Palkovits, R. Palkovits, *ACS Sustainable Chem. Eng.* **2021**, *9*, 1229–1234; c) S. Arndt, D. Weis, K. Donsbach, S. R. Waldvogel, *Angew. Chem. Int. Ed.* **2020**, *59*, 8036–8041; *Angew. Chem.* **2020**, *132*, 8112–8118; d) D. Pollok, S. R. Waldvogel, *Chem. Sci.* **2020**, *11*, 12386–12400; e) S. Möhle, M. Zirbes, E. Rodrigo, T. Gieshoff, A. Wiebe, S. R. Waldvogel, *Angew. Chem. Int. Ed.* **2018**, *57*, 6018–6041; *Angew. Chem.* **2018**, *130*, 6124–6149; f) S. R. Waldvogel, B. Janza, *Angew. Chem. Int. Ed.* **2014**, *53*, 7122–7123; *Angew. Chem.* **2014**, *126*, 7248–7249.

[3] a) J. L. Röckl, D. Pollok, R. Franke, S. R. Waldvogel, *Acc. Chem. Res.* **2020**, *53*, 45–61; b) J. L. Röckl, D. Schollmeyer, R. Franke, S. R. Waldvogel, *Angew. Chem. Int. Ed.* **2020**, *59*, 315–319; *Angew. Chem.* **2020**, *132*, 323–327; c) S. R. Waldvogel, S. Lips, M. Selt, B. Riehl, C. J. Kampf, *Chem. Rev.* **2018**, *118*, 6706–6765; d) A. Wiebe, S. Lips, D. Schollmeyer, R. Franke, S. R. Waldvogel, *Angew. Chem. Int. Ed.* **2017**, *56*, 14727–14731; *Angew. Chem.* **2017**, *129*, 14920–14925; e) E. J. Horn, B. R. Rosen, P. S. Baran, *ACS Cent. Sci.* **2016**, *2*, 302–308.

[4] M. Yan, Y. Kawamata, P. S. Baran, *Chem. Rev.* **2017**, *117*, 13230–13319.

[5] A. Wiebe, T. Gieshoff, S. Möhle, E. Rodrigo, M. Zirbes, S. R. Waldvogel, *Angew. Chem. Int. Ed.* **2018**, *57*, 5594–5619; *Angew. Chem.* **2018**, *130*, 5694–5721.

[6] D. Pollok, B. Gleede, A. Stenglein, S. R. Waldvogel, *Aldrichimica Acta* **2021**, *54*, 3–15.

[7] a) N. Amri, T. Wirth, *J. Org. Chem.* **2021**, *86*, 15961–15972; b) N. Amri, T. Wirth, *Synlett* **2020**, *31*, 1894–1898.

[8] a) E. Sato, M. Fujii, H. Tanaka, K. Mitsudo, M. Kondo, S. Takizawa, H. Sasai, T. Washio, K. Ishikawa, S. Suga, *J. Org. Chem.* **2021**, *86*, 16035–16044; b) M. Elsherbini, T. Wirth, *Acc. Chem. Res.* **2019**, *52*, 3287–3296.

[9] C. Gütz, A. Stenglein, S. R. Waldvogel, *Org. Process Res. Dev.* **2017**, *21*, 771–778.

[10] A. G. Wills, S. Charvet, C. Battilocchio, C. C. Scarborough, K. M. P. Wheelhouse, D. L. Poole, N. Carson, J. C. Vantourout, *Org. Process Res. Dev.* **2021**, *25*, 2587–2600.

[11] N. Kurig, J. Meyers, E. Richter, S. Palkovits, R. Palkovits, *Chem. Ing. Tech.* **2022**, *94*, 786–790.

[12] a) E. Sato, G. Tachiwaki, M. Fujii, K. Mitsudo, T. Washio, S. Takizawa, S. Suga, *Org. Process Res. Dev* **2023**; b) M. Dörr, M. M. Hielscher, J. Proppe, S. R. Waldvogel, *ChemElectroChem* **2021**, *8*, 2621–2629; c) M. M. Hielscher, B. Gleede, S. R. Waldvogel, *Electrochim. Acta* **2021**, *368*, 137420; d) M. Dörr, J. L. Röckl, J. Rein, D. Schollmeyer, S. R. Waldvogel, *Chem. Eur. J.* **2020**, *26*, 10195–10198; e) M. Santi, J. Seitz, R. Cicala, T. Hardwick, N. Ahmed, T. Wirth, *Chem. Eur. J.* **2019**, *25*, 16230–16235; f) R. Möckel, J. Hille, E. Winterling, S. Weidemüller, T. M. Faber, G. Hilt, *Angew. Chem. Int. Ed.* **2018**, *57*, 442–445; *Angew. Chem.* **2018**, *130*, 450–454.

[13] a) Z. Crandall, K. Basemann, L. Qi, T. L. Windus, *React. Chem. Eng.* **2022**, *7*, 416–428; b) C. J. van der Westhuizen, J. Du Toit, N. Neyt, D. Riley, J.-L. Panayides, *Digital Discovery* **2022**, *1*, 596–604; c) S. Chatterjee, M. Guidi, P. H. Seeberger, K. Gilmore, *Nature* **2020**, *579*, 379–384; d) N. Cherkasov, Y. Bai, A. J. Expósito, E. V. Rebrov, *React. Chem. Eng.* **2018**, *3*, 769–780; e) D. E. Fitzpatrick, C. Battilocchio, S. V. Ley, *Org. Process Res. Dev.* **2016**, *20*, 386–394.

[14] L. M. Roch, F. Häse, C. Kreisbeck, T. Tamayo-Mendoza, L. P. E. Yunker, J. E. Hein, A. Aspuru-Guzik, *Sci. Robot.* **2018**, *3*.

[15] S. Steiner, J. Wolf, S. Glatzel, A. Andreou, J. M. Granda, G. Keenan, T. Hinkley, G. Aragon-Camarasa, P. J. Kitson, D. Angelone et al., *Science* **2019**, *363*.

[16] L. M. Roch, F. Häse, C. Kreisbeck, T. Tamayo-Mendoza, L. P. E. Yunker, J. E. Hein, A. Aspuru-Guzik, *PLoS One* **2020**, *15*, e0229862.

[17] T. Wirtanen, T. Prenzel, J.-P. Tessonnier, S. R. Waldvogel, *Chem. Rev.* **2021**, *121*, 10241–10270.

[18] M. Seifrid, R. Pollice, A. Aguilar-Granda, Z. Morgan Chan, K. Hotta, C. T. Ser, J. Vestfrid, T. C. Wu, A. Aspuru-Guzik, *Acc. Chem. Res.* **2022**, *55*, 2454–2466.

[19] M. M. Hielscher, B. Gleede, S. R. Waldvogel, *Electrochim. Acta* **2021**, *368*, 137420.

[20] A. J. S. Hammer, A. I. Leonov, N. L. Bell, L. Cronin, *JACS Au* **2021**, *1*, 1572–1587.

[21] a) GitHub, "ChemputerSoftware/client at master·croningp/ChemputerSoftware", can be found under https://github.com/croningp/ChemputerSoftware, **2023**; b) GitHub, "GitHub - aspuru-guzik-group/ChemOS", can be found under https://github.com/aspuru-guzik-group/ChemOS, **2023**; c) GitHub, "GitHub - RxnRover/RxnRover", can be found under https://github.com/RxnRover/RxnRover, **2023**; d) SourceForge, "OpenFlowChem", can be found under https://sourceforge.net/projects/openflowchem/, **2023**.

[22] GitHub, "GitHub – cambiegroup/flowchem: Flowchem is an application to simplify the control of instruments and devices commonly found in chemistry labs", can be found under https://github.com/cambiegroup/flowchem, **2023**.

[23] "Twisted", can be found under https://twisted.org/, **2023**.

[24] a) GitHub, "richardingham/octopus: Real-time laboratory automation and monitoring in Python", can be found under https://github.com/richardingham/octopus, **2023**; b) R. J. Ingham, C. Battilocchio, J. M. Hawkins, S. V. Ley, *Beilstein J. Org. Chem.* **2014**, *10*, 641–652.

[25] "Welcome to Flask – Flask Documentation (2.2.x)", can be found under https://flask.palletsprojects.com/en/2.2.x/, **2023**.

[26] a) M. Selt, S. Mentizi, D. Schollmeyer, R. Franke, S. R. Waldvogel, *Synlett* **2019**, *30*, 2062–2067; b) B. Riehl, K. Dyballa, R. Franke, S. Waldvogel, *Synthesis* **2016**, *49*, 252–259.

[27] M. Selt, R. Franke, S. R. Waldvogel, *Org. Process Res. Dev.* **2020**, *24*, 2347–2355.

[28] I. M. Malkowsky, U. Griesbach, H. Pütter, S. R. Waldvogel, *Eur. J. Org. Chem.* **2006**, *2006*, 4569–4572.

[29] a) J. Barjau, G. Schnakenburg, S. R. Waldvogel, *Angew. Chem. Int. Ed.* **2011**, *50*, 1415–1419; *Angew. Chem.* **2011**, *123*, 1451–1455; b) I. M. Malkowsky, C. E. Rommel, K. Wedeking, R. Fröhlich, K. Bergander, M. Nieger, C. Quaiser, U. Griesbach, H. Pütter, S. R. Waldvogel, *Eur. J. Org. Chem.* **2006**, *2006*, 241–245.

[30] a) S. Palmas, A. M. Polcaro, A. Vacca, M. Mascia, F. Ferrara, *J. Appl. Electrochem.* **2006**, *37*, 63–70; b) I. Duo, C. Levy-Clement, A. Fujishima, C. Comninellis, *J. Appl. Electrochem.* **2004**, *34*, 935–943.
[31] J. Fangmeyer, A. Behrens, B. Gleede, S. R. Waldvogel, U. Karst, *Angew. Chem. Int. Ed.* **2020**, *59*, 20428–20433; *Angew. Chem.* **2020**, *46*, 20608–20613.
[32] M. M. Hielscher, M. Dörr, *LABS-Frontend* **2023**, DOI 10.5281/ZENODO.7879586.
[33] M. Dörr, M. M. Hielscher, *LABS-Backend* **2023**, DOI 10.5281/ZENODO.7879588.
[34] M. Dörr, M. M. Hielscher, *LABS-DeviceDummys* **2023**, DOI 10.5281/ZENODO.7879590.
[35] "Cronin Group/Chemputer/XDL GitLab", can be found under https://gitlab.com/croningroup/chemputer/xdl, **2023**.

Manuscript received: April 30, 2023
Revised manuscript received: May 27, 2023
Accepted manuscript online: June 3, 2023
Version of record online: June 12, 2023