

# Build instructions for Closed-loop Spectroscopy Lab: Light-mixing Demo

**Sterling G. Baird<sup>1,2,\*</sup> and Taylor D. Sparks<sup>1,3,\*\*</sup>**

<sup>1</sup>Materials Science & Engineering Department, University of Utah, Salt Lake City UT USA, 84108

<sup>2</sup>Technical contact

<sup>3</sup>Lead contact

\*Correspondence: [sterling.baird@utah.edu](mailto:sterling.baird@utah.edu)

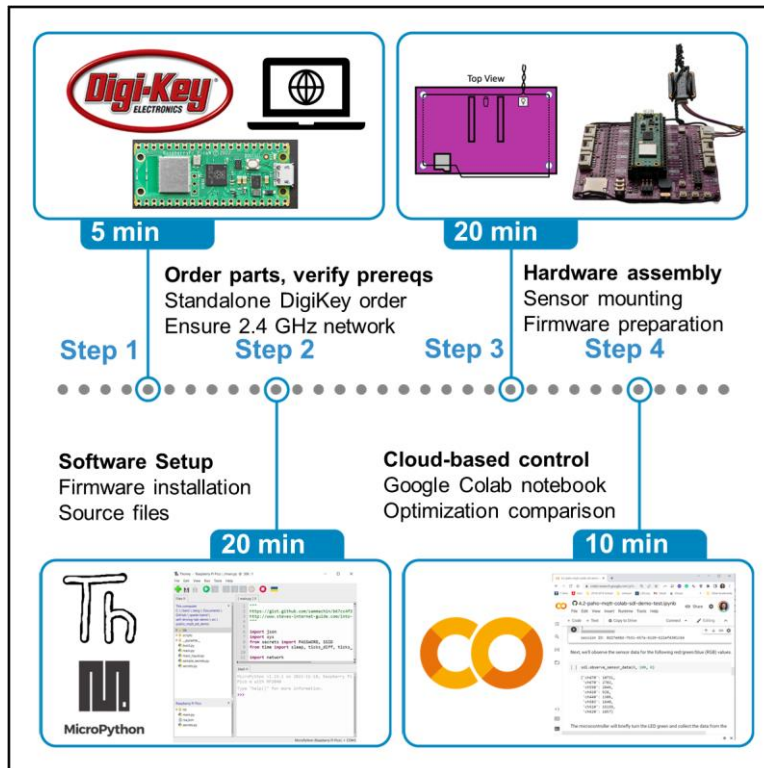
\*\*Correspondence: [sparks@eng.utah.edu](mailto:sparks@eng.utah.edu)

## Summary

Closed-loop Spectroscopy Lab: Light-mixing Demo (CLSLab:Light) is a teaching and prototyping platform for autonomous scientific discovery. It consists of a set of LEDs and a light sensor while encapsulating key principles for "self-driving" (i.e., autonomous) research laboratories, including sending commands, receiving sensor data, physics-based simulation, and advanced optimization. CLSLab:Light is a "Hello, World!" introduction to these topics, accessible by students, educators, hobbyists, and researchers for less than 100 USD, a small footprint, and under an hour of setup time.

**For context, please refer to Baird et al.<sup>1</sup>**

## Graphical abstract



## Before you begin

The protocol below describes how to set up a “Hello, World!” demonstration<sup>1–6</sup> for a self-driving laboratory<sup>7–11</sup> using a Pico W microcontroller, LEDs, a light sensor, and Bayesian optimization.

## Order Required Parts

**Timing: 5 min (not including shipping time)**

1. Order the required parts [[Self-contained Digikey Order](#)] (60.80 USD + shipping as of 2022-10-20)
  - a. The sculpting wire needs to be 14 gauge (2 mm) or thinner, including the insulation jacket, and rigid enough to support the sensor. Sculpting wire is [also available at Amazon](#). Approximately 3' is required.
  - b. The purpose of the wall adapter is so that, after initial setup, the demo can be powered standalone
  - c. The bill of materials is also [available at Adafruit](#), though you may need to source a Pico W with headers or a Pico WH separately. See [Raspberry Pi's supported resellers for the Pico W](#).

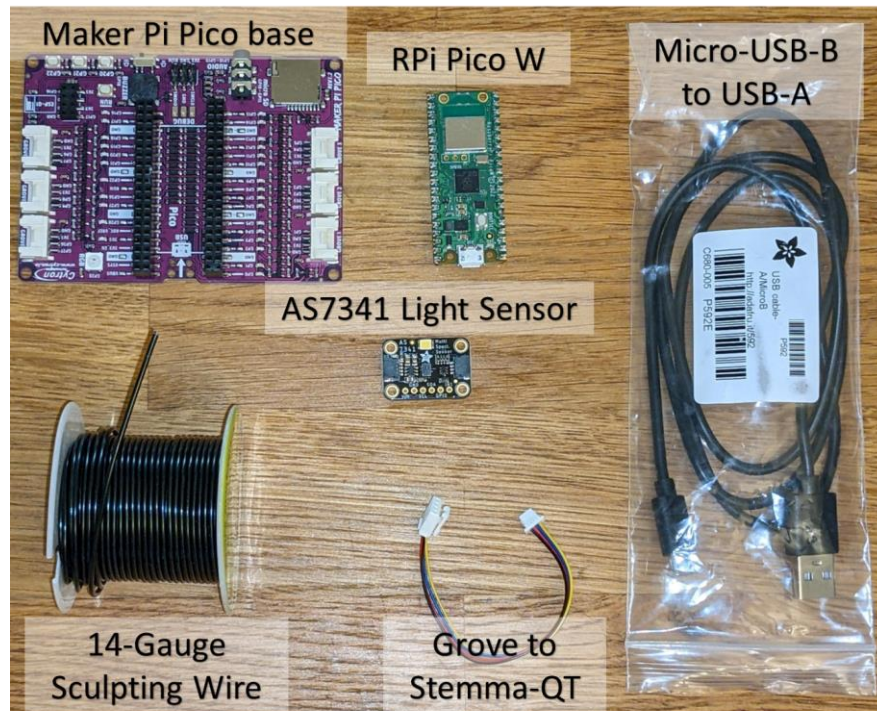


Figure 1

## Additional Prerequisites

**Timing: N/A**

2. Ensure access to a 2.4 GHz WiFi network (SSID + password)
  - a. The Pico W only supports 2.4 GHz WiFi networks. See [self-driving-lab-demo #76](#) for additional context.
    - i WPA enterprise networks such as Eduroam and other networks that use captive portals (most schools, coffee shops, etc.) are not yet supported. It needs to be a network such that on a computer, you can click on the WiFi name (SSID), enter the password, and click connect (no additional steps). Check to see if your institution offers network support for internet of things devices (e.g. ULink at University of Utah).
    - ii Home networks can have both a 5G and a 2.4 GHz network (e.g. "My Network 5G" and "My Network")
    - iii If you use a mobile hotspot, you may need to use your device's "extended compatibility" feature to drop the mobile hotspot from 5G to 2.4 GHz. See also [prepaid, long-expiry hotspot](#) and [classroom demo with standalone network access](#) discussions.
3. Ensure access to a computer (for initial setup only)
  - a. At a minimum, the computer needs to be able to run the Thonny editor (lightweight) and it needs at least one USB-A port
4. Ensure access to a soldering iron and soldering wire (thinner is better in this case)
5. (Optional) Before soldering, ensure the Pico W can successfully connect to a computer

- a. You can do this by holding the BOOTSEL button on the Pico W while connecting the Pico W to your computer via the USB cable. If a new drive appears, that indicates the Pico W is working normally
- b. Be careful only to heat the gold pads while soldering to avoid damaging the circuitry

## Key resources table

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited Data		
Red, Green, and Blue LED Spectral Data	<a href="https://github.com/sparks-baird/self-driving-lab-demo/tree/v0.6.0/src/self_driving_lab_demo/data">https://github.com/sparks-baird/self-driving-lab-demo/tree/v0.6.0/src/self_driving_lab_demo/data</a>	v0.6.0
Software and Algorithms		
self-driving-lab-demo v0.6.0	<a href="https://github.com/sparks-baird/self-driving-lab-demo">https://github.com/sparks-baird/self-driving-lab-demo</a>	
YouTube build tutorial	<a href="https://youtu.be/GVdfJCsQ8vk">https://youtu.be/GVdfJCsQ8vk</a>	
Other		
STEMMA QT AS7341 COLOR SENSOR	DigiKey (Adafruit Product)	Cat#1528-4698-ND
4-PIN STEMMA/GROVE - QT/QWIIC 4"	DigiKey (Adafruit Product)	Cat#1528-4528-ND
RASPBERRY PI PICO W	DigiKey (Adafruit Product)	Cat#2648-SC0918CT-ND
CBL USB2.0 A PLUG-MCR B PLUG 3'	DigiKey (Adafruit Product)	Cat#380-1431-ND
CONN HEADER VERT 20POS 2.54MM	DigiKey (Amphenol CS)	Cat#10129378-920001BLF-ND
MAKER PI PICO BASE (WITHOUT PICO)	DigiKey (Adafruit Product)	Cat#3614-MAKER-PI-PICO-NB-ND
AC/DC WALL MOUNT ADAPTER 5V 5W	DigiKey (Adafruit Product)	Cat#1470-2768-ND
HOOK-UP SOLID 18AWG BLACK 100'	DigiKey (Remington Industries)	Cat#2328-18UL1007SLDBLA-ND
128MB MICRO SD MEMORY CARD (optional)	DigiKey (Adafruit Product)	Cat#1528-5250-ND

## Step-by-step method details

### Hardware Setup

**Timing: 20 min**

Solder the headers onto the Pico W, mount the light sensor so that the pinhole is facing the red green blue (RGB) LED, connect the light sensor to the board, and get the microcontroller ready for firmware installation.

1. Solder headers onto the Pico W
  - a. Insert the Pico W headers into the Maker Pi Pico base
  - b. Place the Pico W on top of the headers
  - c. Solder the headers to the Pico W
    - i [MagPi guide](#)
    - ii [Tom's hardware guide](#)
    - iii [YouTube video](#)
  - d. Remove the Pico W from the Maker Pi Pico base
2. Prepare 3 feet of sculpting wire (cut with wire cutters or bend until it breaks)
3. Thread the sculpting wire through each mounting hole on the Maker Pi Pico base, then twist the wires together near the RGB LED. This setup will allow the position and orientation of the sensor to be both adjustable and steady. Continue twisting until you have 4 to 6 inches of twisted wire, and ensure that there are at least 3 inches of loose, untwisted wire at each end (the leftover, untwisted wire will be threaded through the mounting holes of the light sensor)



in the next step). For reference, a diagram is also included below.

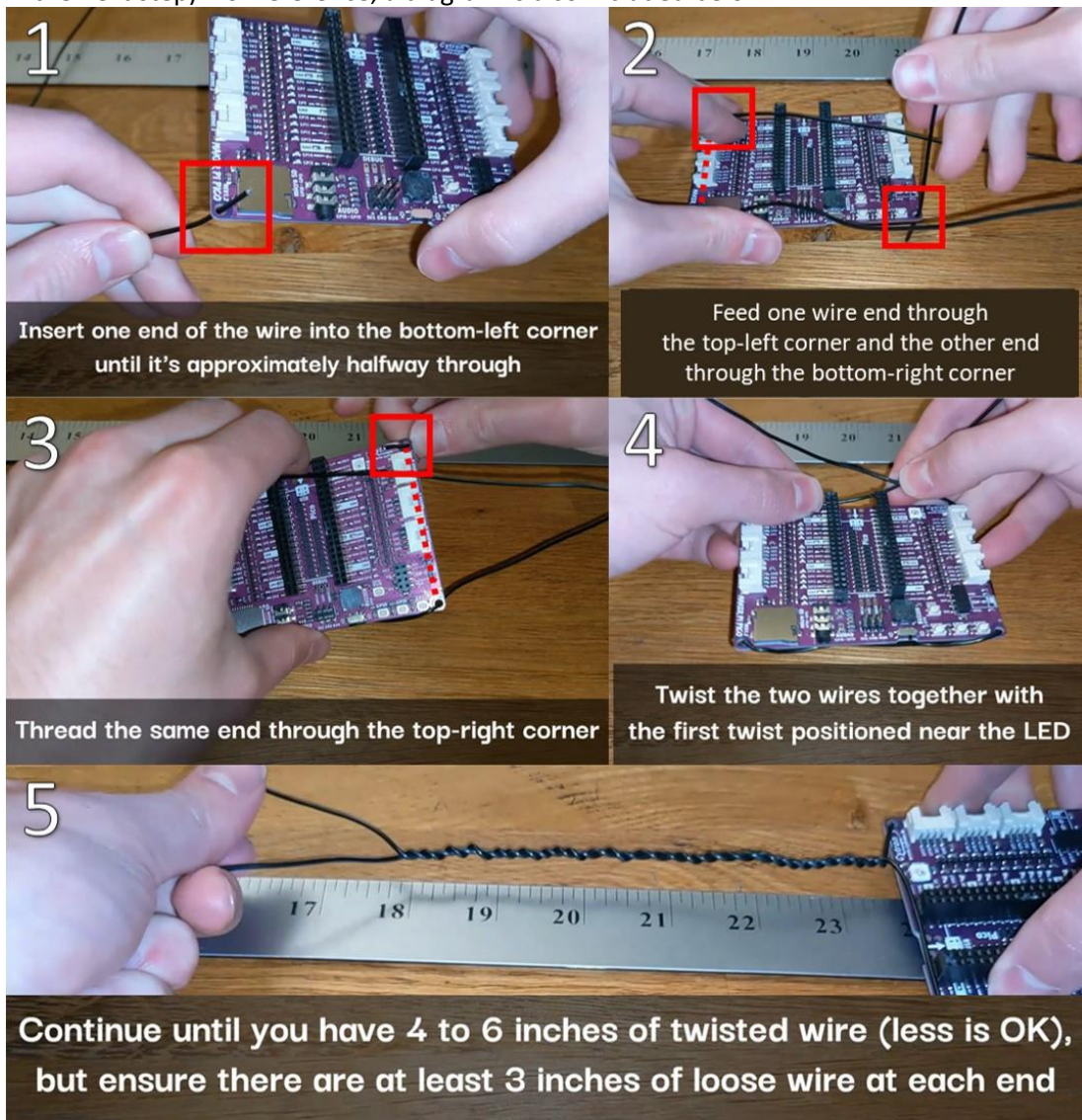
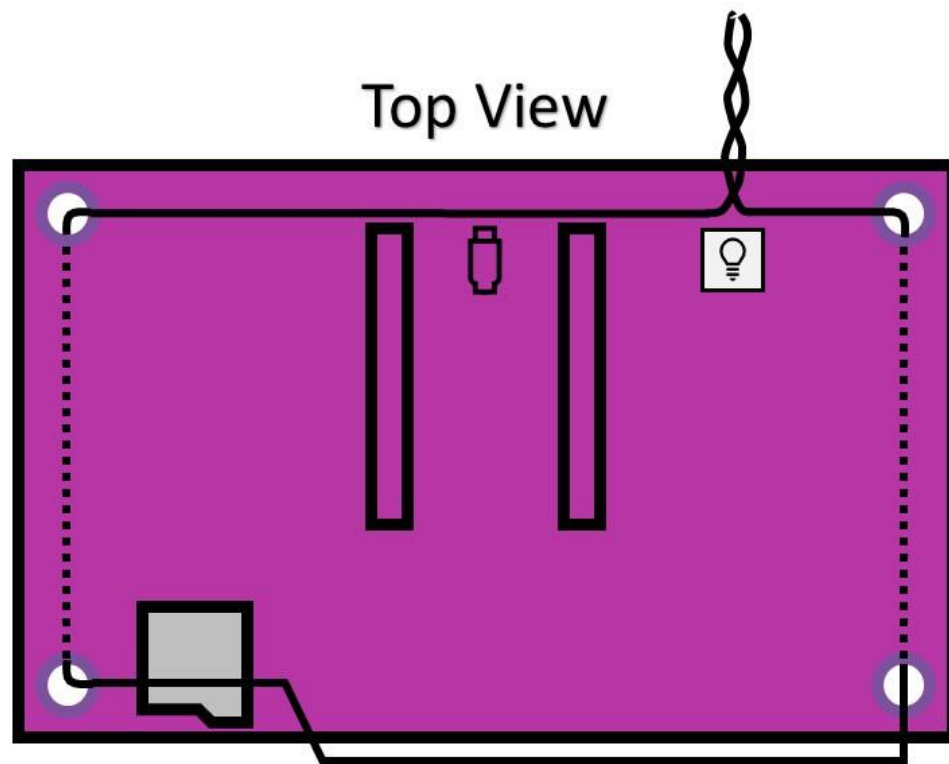


Figure 2



*Figure 3*

4. Thread the same sculpting wire through the AS7341 light sensor and position the sensor so the pinhole is facing approximately 3 to 4 inches away from the RGB LED.

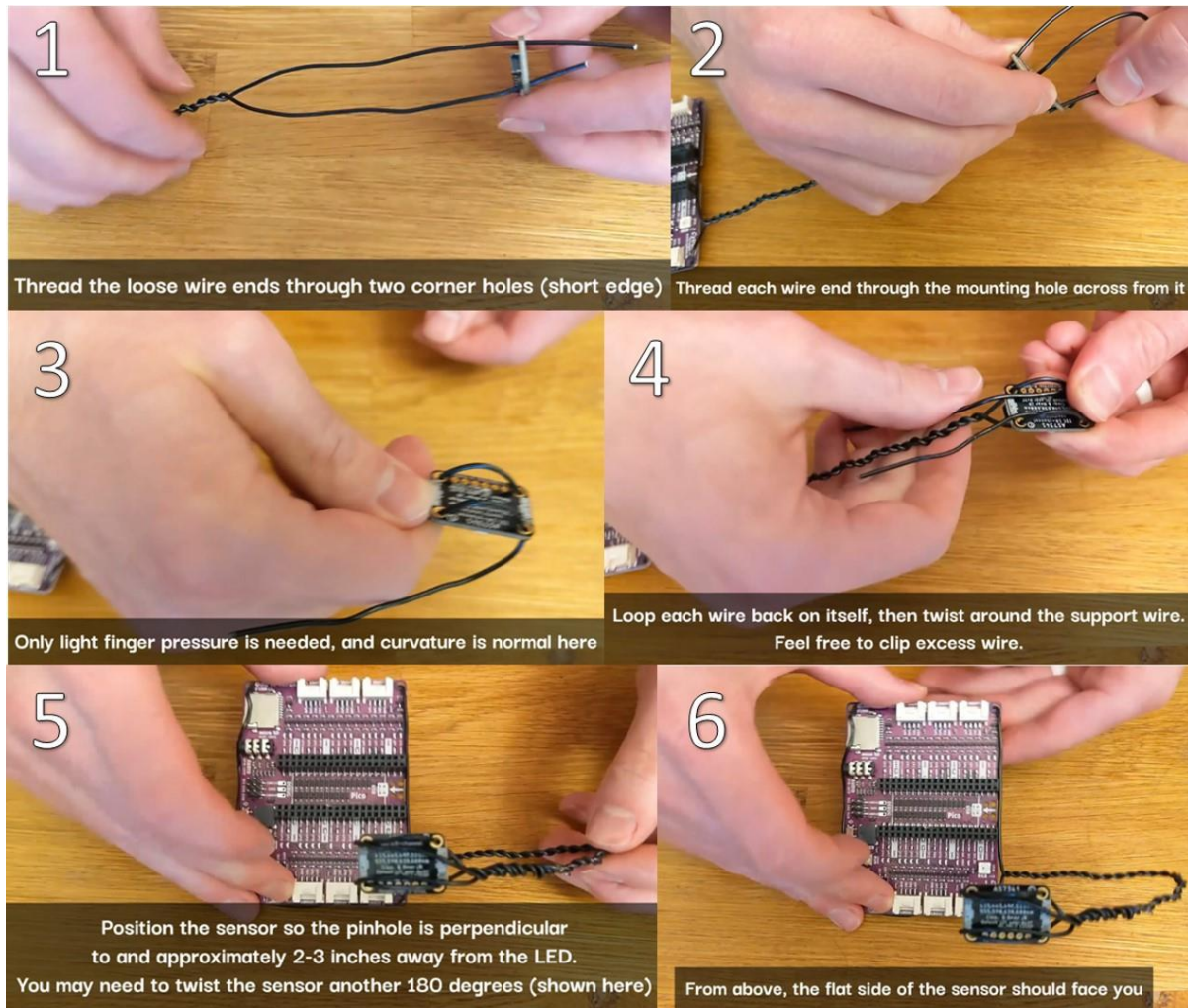


Figure 4

5. Connect the Grove/Stemma-QT connector into Grove port 6 (GP26&27) and the AS7341, insert the Pico W, and while holding the BOOTSEL button, connect the Pico W to the



computer.

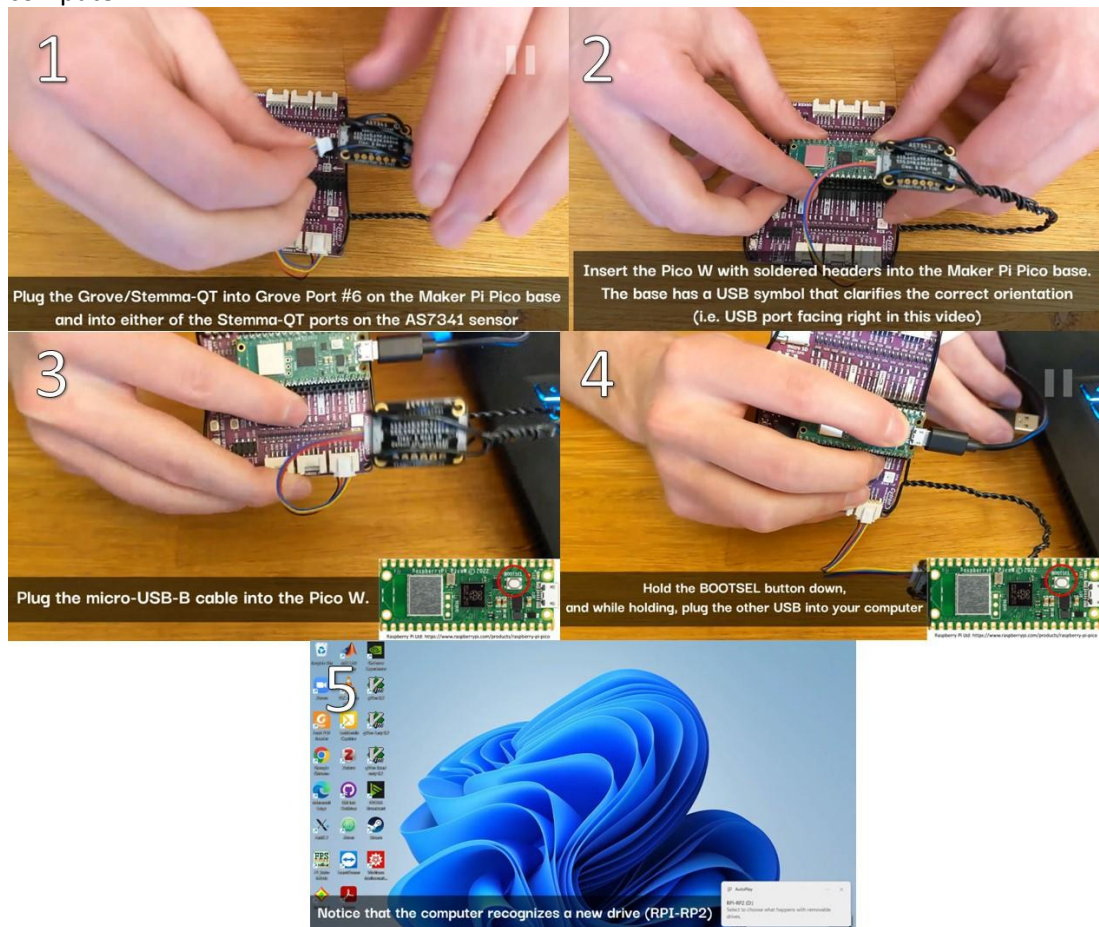


Figure 5

## Software Setup

**Timing: 20 min**

Install the MicroPython firmware onto the Pico W microcontroller, enter the WiFi credentials, and upload the source code files.

6. Download and install [Thonny](#), a Python IDE with native support for microcontrollers. Choose the platform appropriate for you (in my case, this is Windows 64-bit, Python 3.10). When installing, use the default settings: "Standard (default)".
7. Click on the lower-right dropdown and click "Install MicroPython"



Figure 6

8. Choose "MicroPython variant: Raspberry Pi - Pico W / Pico WH" and click install

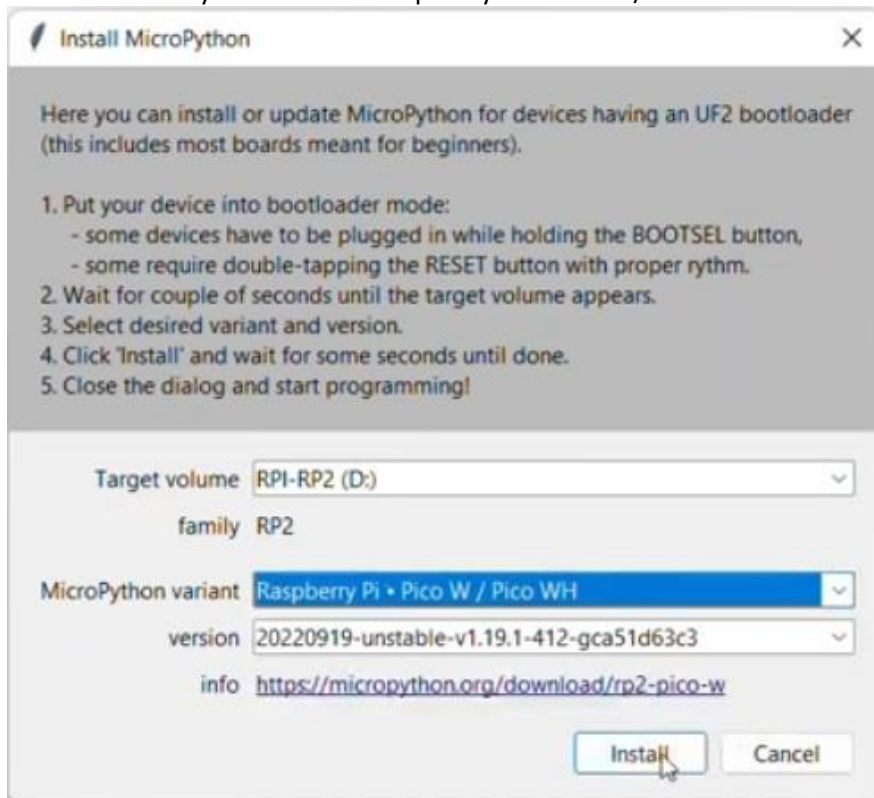


Figure 7

9. Change the interpreter from Local Python 3 to MicroPython (Raspberry Pi Pico)

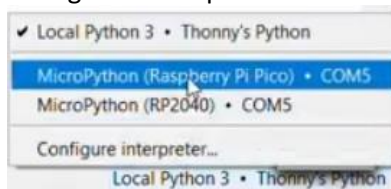


Figure 8

10. In Thonny's menubar, click "View" then "Files" to open a sidebar

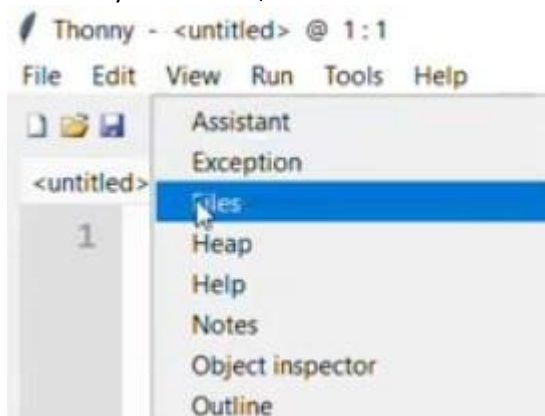


Figure 9

11. Download *sdl\_demo.zip* from [the latest release at self-driving-lab-demo](#) and unzip it
12. In Thonny, navigate to the unzipped *sdl\_demo* folder, open *secrets.py*, enter your WiFi network name (SSID) and password as Python strings. Optionally, you can create your own MongoDB Atlas database and enter values for `MONGODB_API_KEY`, `MONGODB_COLLECTION_NAME`, and `DEVICE_NICKNAME` (see below). Optionally, you can create your own HiveMQ instance and enter the credentials there (see below).  
Save *secrets.py*

A screenshot of the Thonny IDE showing the `secrets.py` file. The file contains three lines of Python code: a docstring, an assignment for `SSID`, and an assignment for `PASSWORD`. The values are redacted with black boxes, except for the word "password" in the password string.

```
<untitled> - secrets.py *  
1 """WiFi Login Info"""  
2 SSID = "  
3 PASSWORD = "password here"
```

Figure 10

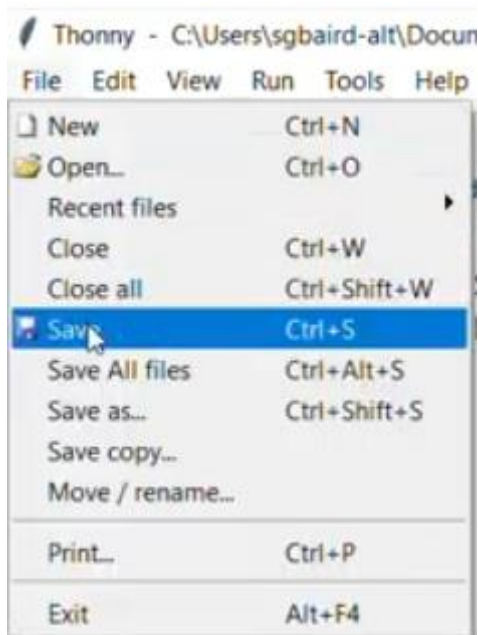


Figure 11

- a. (Optional) Set up a MongoDB database backend
- i. Create an account at <https://www.mongodb.com/cloud/atlas/register>
  - ii. Create a free, Shared Cluster (optionally rename Cluster0 to something of your choice, e.g. self-driving-labs. You can leave the default provider as-is)

[CLUSTERS](#) > [CREATE A SHARED CLUSTER](#)

### Create a Shared Cluster

Serverless

Dedicated

Shared

For learning and exploring MongoDB in a sandbox environment. Basic configuration controls.

No credit card required to start. Upgrade to dedicated clusters for full functionality.

Explore with sample datasets. Limit of one free cluster per project.

Cloud Provider & Region

AWS, N. Virginia (us-east-1)

aws

Google Cloud

Azure

★ Recommended region ⓘ 🏷️ Dedicated tier region ⓘ

NORTH AMERICA

EUROPE

AUSTRALIA

- iii. Navigate to “Data Services” → “Deployment” → “Database” and click “Browse Collections” then “Add My Own Data”. Enter a database name (e.g., cls-lab-light-mixing) and collection name (e.g., test). Copy the names into MONGODB\_DATABASE\_NAME and MONGODB\_COLLECTION\_NAME in

secrets.py.

STERLING'S ORG - 2022-03-25 • TEST

## Database Deployments

Find a database deployment...

+ Create

Cluster0 Connect View Monitoring **Browse Collections** ...

FREE SHARED

**Enhance Your Experience**

For production throughput and richer metrics, upgrade to a dedicated cluster now!

Upgrade

VERSION	REGION	CLUSTER TIER	TYPE	BACKUPS	LINKED APP SERVICES	ATLAS SEARCH
5.0.14	AWS / N. Virginia (us-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inactive	None Linked	Create Index

- iv. Navigate to “Data Services” → “Services” → “Data API”, use the dropdown to select your cluster, and click “Enable Data Access from the Data API”

Enable the Data API

Access the data in your collection over fully managed and secure HTTPS Endpoints

Select the data source(s) you would like to enable the API on

All Data Sources selected

ADVANCED CONFIGURATION

**Enable Data Access from the Data API**

You can modify data access permissions later

- v. Note the app name in the “URL Endpoint” box of the form “https://data.mongodb-api.com/app/<data-abc123>/endpoint/data/v1” where <data-abc123> is the app name. Copy the app name into the MONGODB\_APP\_NAME variable in secrets.py.

STERLING'S ORG - 2022-03-25 • TEST

## Data API ENABLED

Create API Key Test Your API

Data API API Keys Logs Settings

By default, your Data API returns data as JSON which will cast Timestamp, ObjectId, Binary, and Decimal28 data types into strings. Visit the Settings page or add the appropriate header to change this return type to Extended JSON. Learn More

URL Endpoints: Version: V1 https://data.mongodb-api.com/app/[redacted]/endpoint/data/v1 Copy Advanced Settings View our API documentation

Data Source	Provider/Region	Tier	Data API Access
Cluster0	AWS, N. Virginia (US_EAST_1)	M0	Read and Write

System Status: All Good

©2022 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

- vi. Click “Create API Key”, enter a name of your choice (e.g. clslab-light), and click “Generate API key”. Copy the API key and store it somewhere secure. Paste



the API key into the MONGODB\_API\_KEY variable in secrets.py.



## Create Data API Key

Create a Data API Key and be sure to store it in a secure location. You can then visit the API Key tab to [view and manage your API Keys](#)

Configure other authentication methods for Data API in [Authentication services](#)

Name your key

cls-lab-light

Generate API Key

**!** Your Data API key only gives you access to the Data API, not direct access to data in clusters. To prevent security breaches do not distribute it to untrusted individuals or embed directly in your client applications. [Learn more about Data API keys.](#)

After you leave this page, the full private key is unavailable.

Close

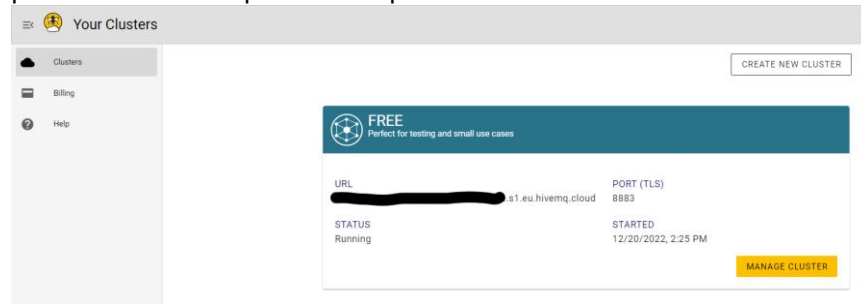
b. (Optional) Create your own HiveMQ instance

- Navigate to <https://www.hivemq.com/mqtt-cloud-broker/>, click “Try out for free”, and create an account
- Set up credentials by entering a username and password and press “ADD”

The screenshot shows the 'Credentials' page in the HiveMQ interface. The sidebar on the left has a menu icon and links to 'Clusters', 'Billing', and 'Help'. The main content area is titled 'Set up credentials for your IoT devices' and includes instructions to define credentials for MQTT clients. It features three input fields: 'Username' (with a note 'At least 5 characters. Username must be unique'), 'Password' (with a note 'At least 8 characters, numbers, upper- and lowercase letters'), and 'Confirm Password' (with a note 'Passwords must match'). An 'ADD' button is at the bottom right. On the right side, there is a box titled 'Active MQTT Credentials' with a blue banner stating 'No credentials configured for your devices' and a sub-note 'Please set up credentials to connect your IoT devices'.

- Navigate to the “Clusters” tab and copy the URL (e.g., abc123.s2.eu.hivemq.cloud) to HIVEMQ\_HOST in secrets.py. Also update HIVEMQ\_USERNAME and HIVEMQ\_PASSWORD with the username and

password from the previous step.



- iv. Create a certificate using the Google Colab notebook at <https://github.com/sparks-baird/self-driving-lab-demo/blob/v0.7.3/notebooks/7.2.1-hivemq-openssl-certificate.ipynb>. Enter the server address (same as HIVEMQ\_HOST), run the Google Colab cells, and follow the instructions to download the `hivemq-com-chain.der` file to the unzipped `sdl_demo` folder. This file is used to do secure authentication via HiveMQ.

13. While holding Ctrl (Windows) or Cmd (Mac), select "lib", "main.py", "hivemq-com-chain.der", and "secrets.py", right click in the gray region, and click "Upload to /"

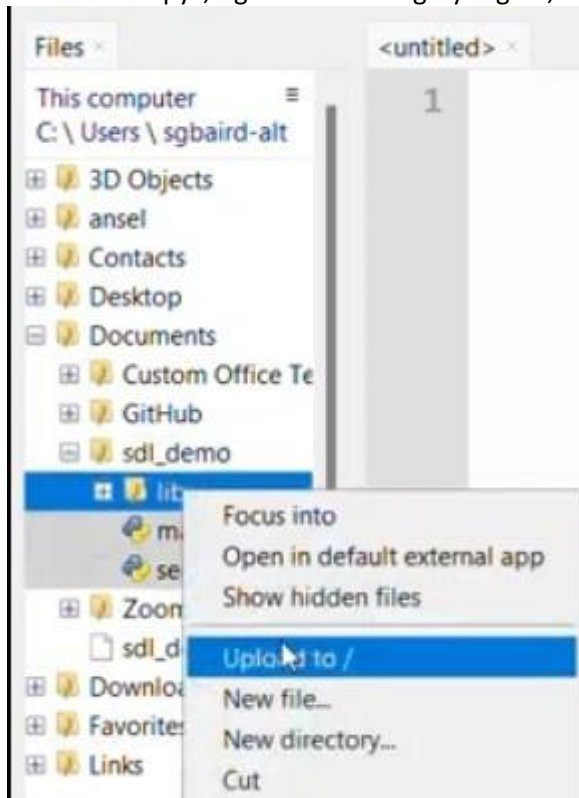


Figure 12

14. Double click to open `main.py`, click the green play button, and note the PICO ID that prints to the command window ("prefix/picow/<PICO\_ID>/"). This will act as the "password" to control

the demo.

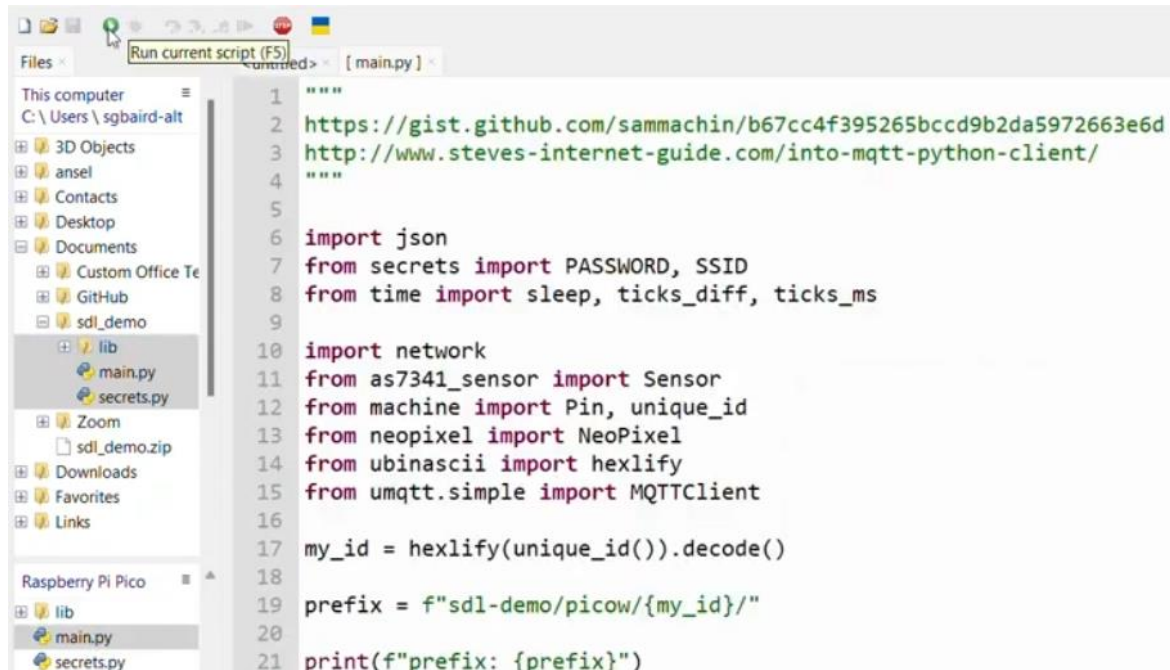


Figure 13

## Control from the cloud

**Timing: 10 min**

Bayesian optimization is commonly used for computational and experimental discovery of new materials, and is often used with low experimental budgets in self-driving laboratory settings. This section covers controlling the device in a closed-loop fashion via internet-of-things style communication (MQTT) and run a basic optimization comparison of grid search vs. random search vs. Bayesian optimization.

16. [Open notebooks/4.2-paho-mqtt-colab-sdl-demo-test.ipynb in Google Colab](#)
17. Scroll to the first code cell and click the play button to install the self-driving-lab-demo Python package

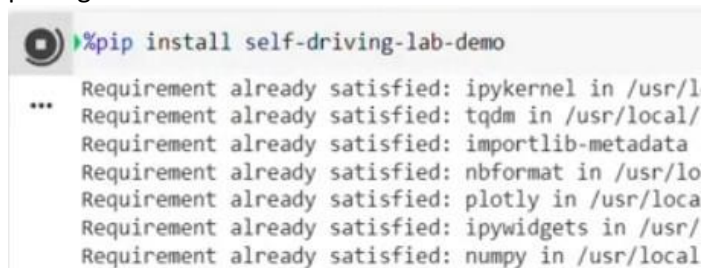


Figure 14

18. Copy the PICO ID from the Thonny editor and paste it in place of "test" (without quotes). The following is an example image of the output; the actual output to the command window may

vary in future releases.



Figure 15



Figure 16

19. Run the remaining code cells
  - a. Instantiate a SelfDrivingLabDemo class
  - b. Perform optimizations for grid search, random search, and Bayesian optimization
20. Additional notebooks that cover advanced optimization topics<sup>12</sup> such as constrained<sup>13–15</sup>, high-dimensional<sup>16,17</sup>, multi-fidelity<sup>18</sup>, and multi-objective<sup>11,19–22</sup> optimization are also available.

## Expected outcomes

1. Successfully set up the hardware and software for a closed-loop experiment
2. Run the first "autonomous drive" given in an example interactive notebook
3. Explore [additional example notebooks](#)

Figure 17 shows a comparison of optimization results for grid search vs. random search vs. Bayesian optimization averaged over repeat campaigns with standard deviation error bands, where Bayesian optimization, on average, performs the best. Figure 18 shows one of the outputs from the cloud-based control notebook of best error so far vs. iteration number comparing grid search vs. random search vs. Bayesian optimization. Typically, grid search is the least efficient, Bayesian optimization is the most efficient, and random search is somewhere in-between. Figure 19, Figure 20, and Figure 21 show the points that were searched for a given campaign for grid search, random search, and Bayesian optimization, respectively. Finally, Figure 22 shows the true, underlying target color (defined by red, green, and blue values) and the best parameter set based on minimizing error between the observed spectrum and the target spectrum for each of the optimization methods.

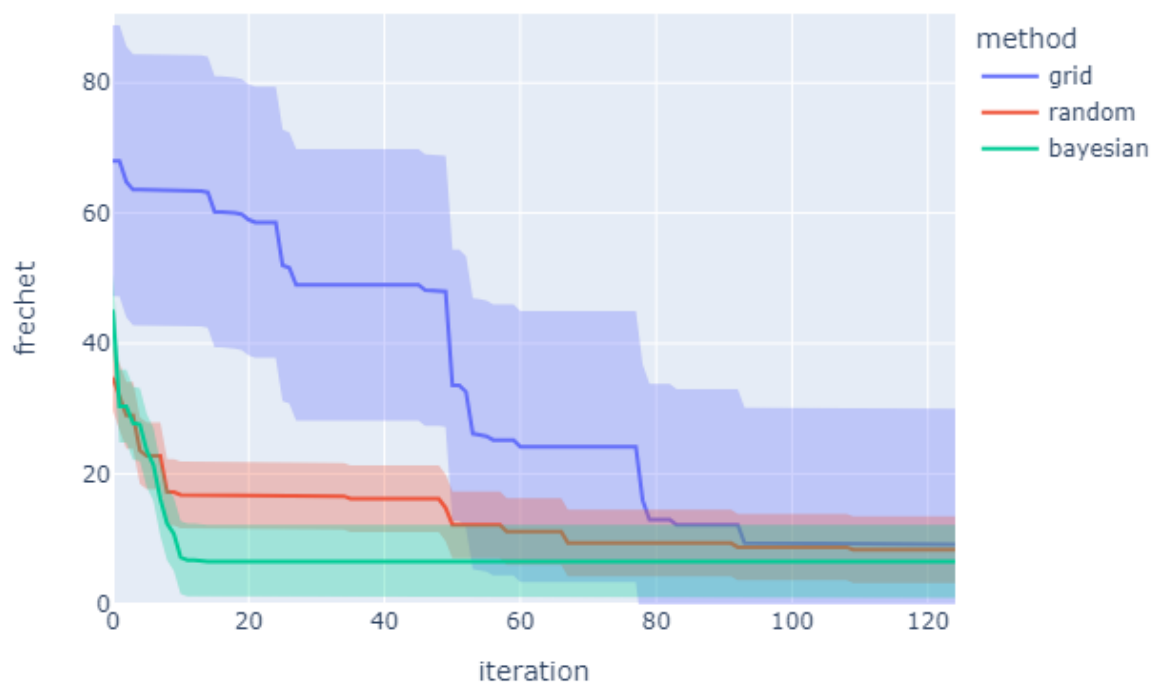


Figure 17

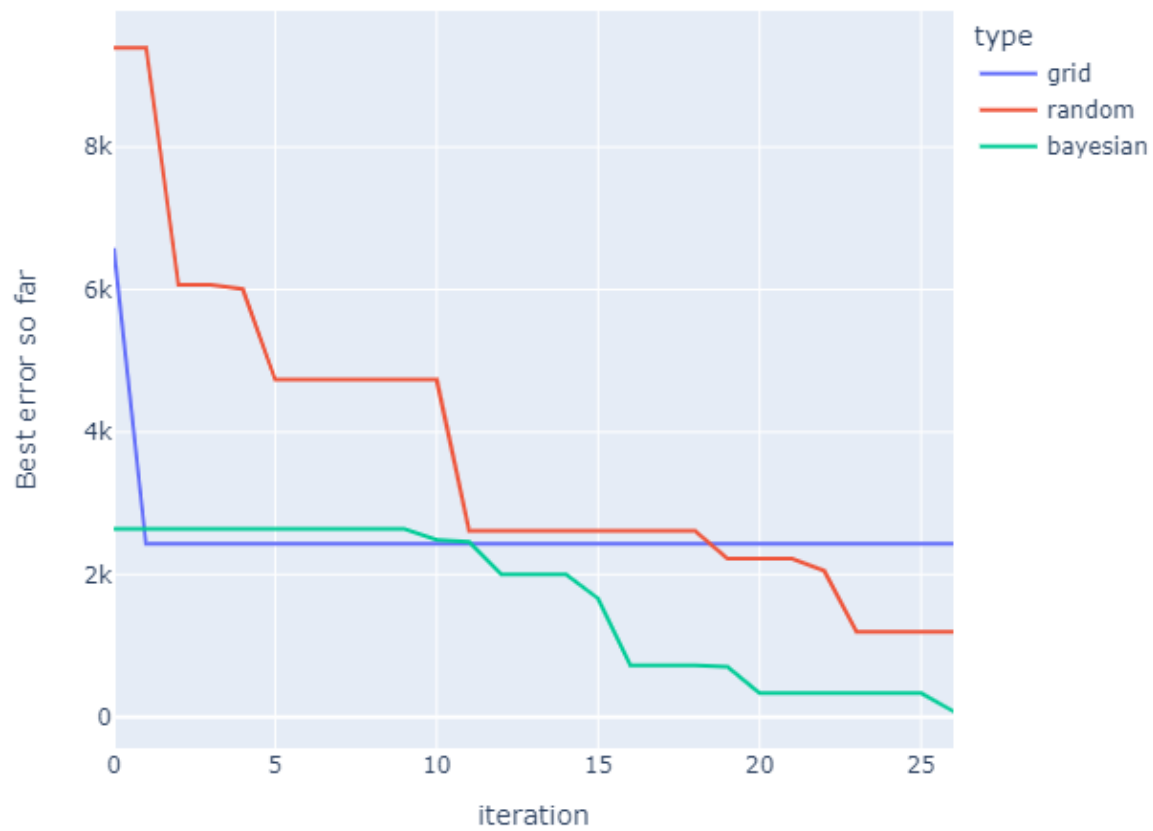


Figure 18



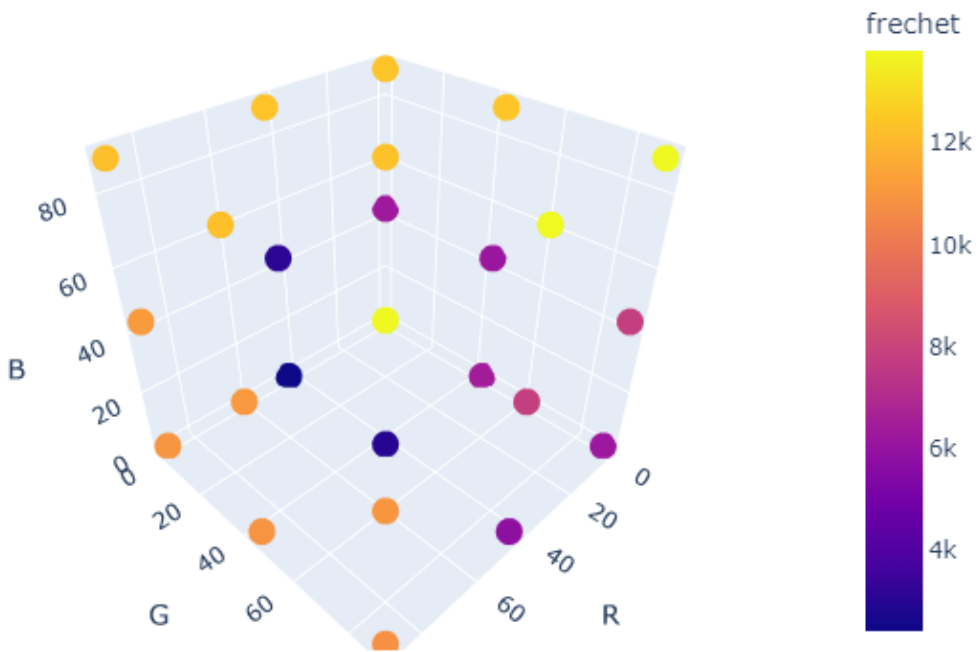


Figure 19

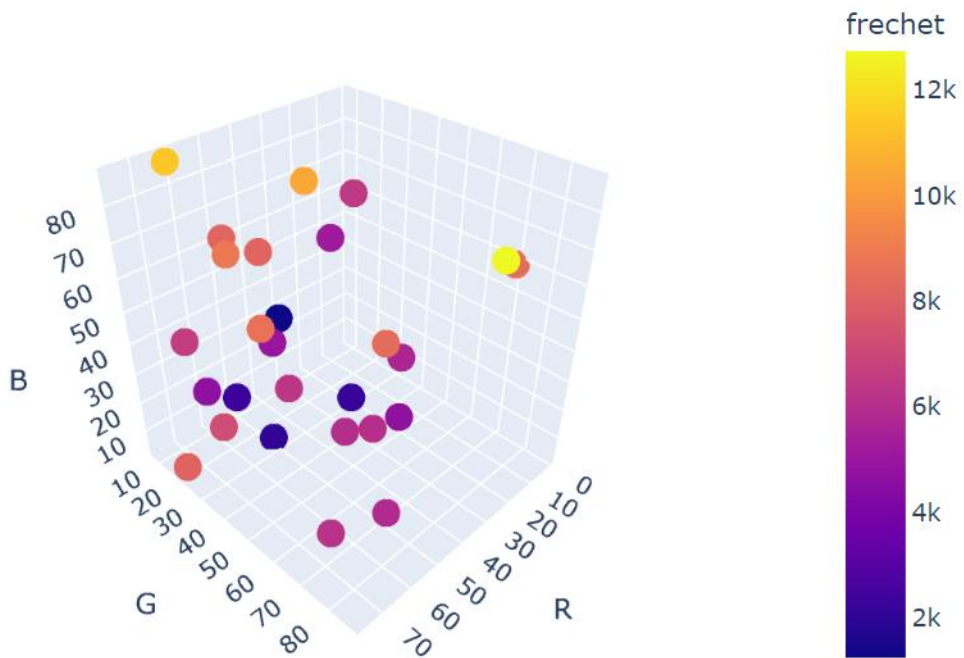


Figure 20

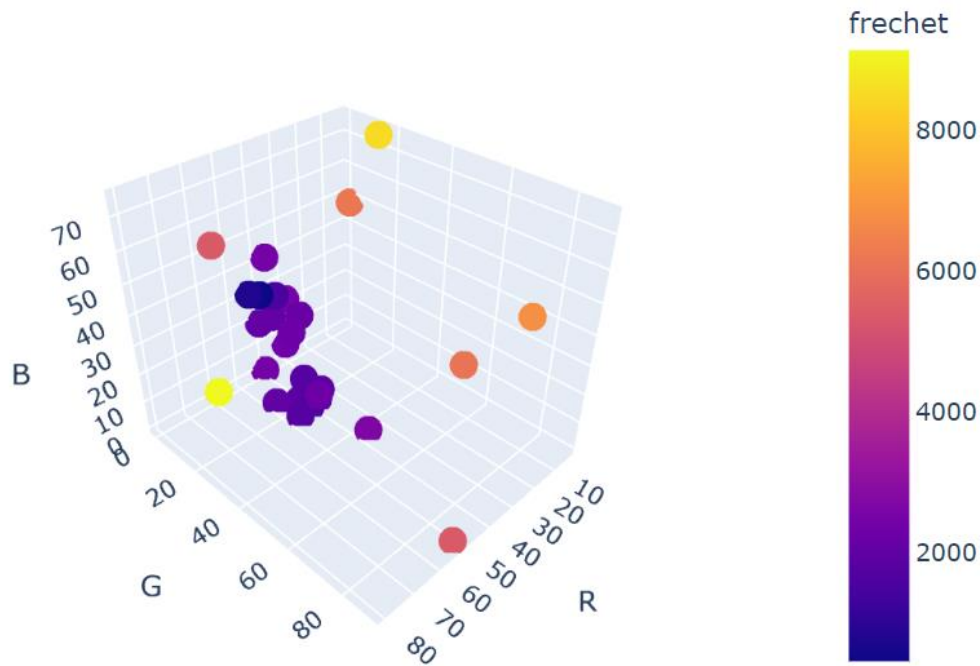


Figure 21

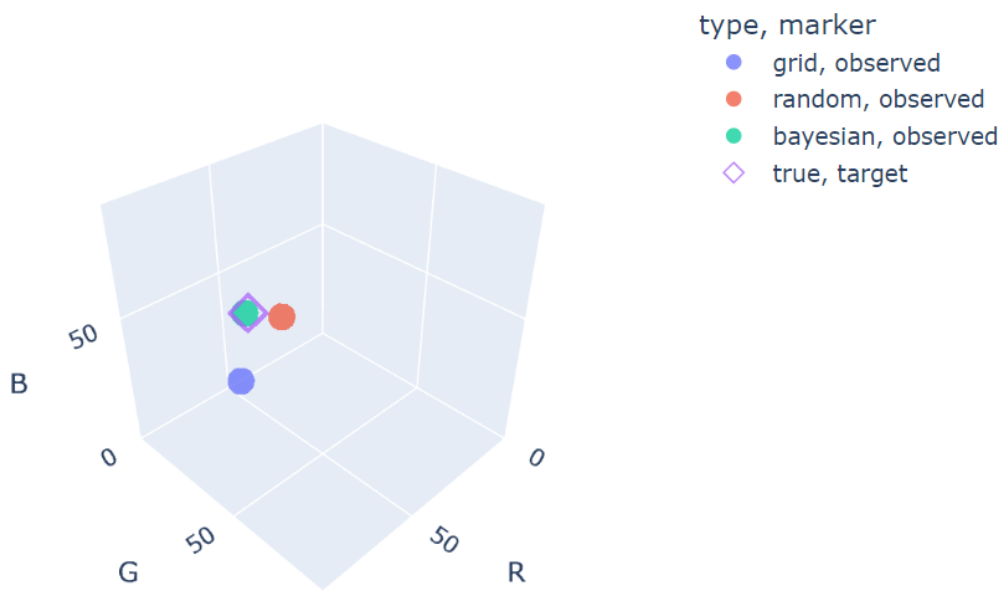


Figure 22

## Quantification and statistical analysis

Discrete Fréchet distance, as implemented in [https://github.com/cjekel/similarity\\_measures](https://github.com/cjekel/similarity_measures)<sup>23</sup>, is used to assess the mismatch between the currently observed spectrum and the target spectrum, where the target spectrum is determined by arbitrarily choosing a random set of RGB values and measuring the sensor data for the fixed, random set of RGB values. Lower Fréchet distances correspond to better matches between the observed and target spectra (i.e. lower error).

An example JSON document logged to a MongoDB database backend containing experimental data for a single run is given as follows:

```
{
  "utc_timestamp": "2022-11-4 06:51:16",
  "ch510": 354,
  "ch620": 5671,
  "ch410": 188,
  "ch440": 3675,
  "ch583": 2756,
  "_input_message": {
    "_session_id": "542e6e80-9c50-4c41-95a5-832603b96238",
    "B": 31,
    "atime": 100,
    "gain": 128,
    "astep": 999,
    "_experiment_id": "9b50c819-db8f-476f-b601-dbe79e871a46",
    "G": 3,
    "integration_time": 280.78,
    "R": 41,
  },
  "onboard_temperature_K": 294.1085,
  "sd_card_ready": True,
  "ch470": 2827,
  "ch550": 498,
  "ch670": 277,
}
```

The experimental parameters for two JSON documents are given in Table 1.

*Table 1. Example of data obtained from two experiments. The LED parameters are red (R), green (G), blue (B). The sensor settings are atime, gain, astep (affects integration time and intensity). The measured output values are of the form "ch###" where the three digit number corresponds to the full-width half-max (FWHM) wavelength being measured.*

utc_timestamp	onboard_temperature_K	R	G	B	atime	gain	aste	ch410	ch440	ch470	ch510	ch550	ch583	ch620	ch670
11/4/2022 6:40	292.7041	41	3	31	100	128	999	188	3674	2828	354	498	2748	5661	276
11/4/2022 6:51	294.1085	41	3	31	100	128	999	188	3675	2827	354	498	2756	5671	277

## Limitations

Environmental noise (e.g. light conditions) and hardware variation (LED, sensor, sensor positioning, etc.) may affect the results obtained.

## Troubleshooting

See the [GitHub issue tracker](#) for existing known issues or to post a new issue. See the [GitHub discussions](#) for general questions and discussion.

### Problem 1:

Can I use this with alternate microcontrollers or firmware?

### Potential solution:

The hardware configuration and software were designed based on Raspberry Pi's Pico Wireless (Pico W) microcontroller. Libraries exist for LED control and the AS7341 light sensor in CircuitPython and Arduino. The hardware and configuration and software can be adapted for other microcontrollers. Contributions at <https://github.com/sparks-baird/self-driving-lab-demo/> are welcome.

### Problem 2:

Can I use this without connecting to the internet?

### Potential solution:

While possible with minor modification, connecting via USB cable is not directly supported. The emphasis is on using this with sophisticated software packages (e.g., [Meta's Adaptive Experimentation platform](#)) that are not typically supported via the lightweight MicroPython firmware that runs on the microcontroller. For private, secure communication between the Pico W microcontroller and the client (e.g., Jupyter notebook running locally), a free, private HiveMQ instance can be set up per the instructions in [Software Setup](#).

### Problem 3:

Can I use this without logging to a MongoDB backend?

### Potential solution:

If the MongoDB credentials are left to their default dummy values in secrets.py, then logging to the MongoDB backend will fail and the device will simply notify the user rather than exit the program. The same applies for logging to an onboard SD card. If an SD card is detected, the microcontroller will write backup data to it, otherwise it will be skipped.

### Problem 3:

The Stemma-QT to Grove connector is out-of-stock.

### Potential solution:

An alternative connector that can be used in place of the Stemma-QT to Grove connector is a 4-pin JST PH to JST SH Cable (DigiKey Cat#1528-4424-ND). Another alternative is using a Stemma-QT to header pin cable (DigiKey Cat#1528-4209-ND) and plugging directly into the GPIO pins that correspond to Grove Port #6 of the Maker Pi Pico base.

### Problem 3:

The sculpting wire doesn't fit through the mounting holes.

### Potential solution:

Ensure that the outer diameter of the sculpting wire is 14 AWG or higher (i.e., 1.628 mm or thinner). Enameled wire (often advertised as sculpting wire) has a very thin coating, whereas electrical wiring typically has a non-negligible insulation thickness.

## Resource availability

### **Lead contact**

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Taylor D. Sparks [sparks@eng.utah.edu](mailto:sparks@eng.utah.edu).

### **Materials availability**

This study did not generate new unique reagents.

### **Data and code availability**

The datasets and code generated during this study are available on GitHub:

<https://github.com/sparks-baird/self-driving-lab-demo>. A standalone DigiKey order is available at <https://www.digikey.com/short/c05d10fd>.

## Acknowledgments

This work was supported by the National Science Foundation under Grant No. DMR-1651668.

## Author contributions

Sterling G. Baird: Conceptualization, Methodology, Software, Writing – Original Draft, Writing – Review & Editing, Visualization, Taylor D. Sparks: Supervision, Funding Acquisition

## Declaration of interests

The authors declare no competing interests.

## References



- (1) Baird, S. G.; Sparks, T. D. What Is a Minimal Working Example for a Self-Driving Laboratory? *Matter* **2022**, 5 (12), 4170–4178. <https://doi.org/10.1016/j.matt.2022.11.007>.
- (2) Saar, L.; Liang, H.; Wang, A.; McDannald, A.; Rodriguez, E.; Takeuchi, I.; Kusne, A. G. A Low-Cost Robot Science Kit for Education with Symbolic Regression for Hypothesis Discovery and Validation. arXiv June 13, 2022. <https://doi.org/10.48550/arXiv.2204.04187>.
- (3) Vargas, S.; Zamirpour, S.; Menon, S.; Rothman, A.; Häse, F.; Tamayo-Mendoza, T.; Romero, J.; Sim, S.; Menke, T.; Aspuru-Guzik, A. Team-Based Learning for Scientific Computing and Automated Experimentation: Visualization of Colored Reactions. *J. Chem. Educ.* **2020**, 97 (3), 689–694. <https://doi.org/10.1021/acs.jchemed.9b00603>.
- (4) Gutierrez, J. M. P.; Hinkley, T.; Taylor, J. W.; Yanev, K.; Cronin, L. Evolution of Oil Droplets in a Chemorobotic Platform. *Nat Commun* **2014**, 5 (1), 5571. <https://doi.org/10.1038/ncomms6571>.
- (5) Caramelli, D.; Salley, D.; Henson, A.; Camarasa, G. A.; Sharabi, S.; Keenan, G.; Cronin, L. Networking Chemical Robots for Reaction Multitasking. *Nat Commun* **2018**, 9 (1), 3406. <https://doi.org/10.1038/s41467-018-05828-8>.
- (6) Fuhrmann, T.; Ahmed, D. I.; Arikson, L.; Wirth, M.; Miller, M. L.; Li, E.; Lam, A.; Blikstein, P.; Riedel-Kruse, I. Scientific Inquiry in Middle Schools by Combining Computational Thinking, Wet Lab Experiments, and Liquid Handling Robots. In *Interaction Design and Children*; ACM: Athens Greece, 2021; pp 444–449. <https://doi.org/10.1145/3459990.3465180>.
- (7) Seifrid, M.; Hattrick-Simpers, J.; Aspuru-Guzik, A.; Kalil, T.; Cranford, S. Reaching Critical MASS: Crowdsourcing Designs for the next Generation of Materials Acceleration Platforms. *Matter* **2022**, 5 (7), 1972–1976. <https://doi.org/10.1016/j.matt.2022.05.035>.
- (8) Hickman, R. J.; Ru, J. Equipping Data-Driven Experiment Planning for Self-Driving Laboratories with Semantic Memory: Case Studies of Transfer Learning in Chemical Reaction Optimization. 27. <https://doi.org/10.26434/chemrxiv-2022-jt4sm>.
- (9) MacLeod, B. P.; Parlane, F. G. L.; Morrissey, T. D.; Häse, F.; Roch, L. M.; Dettelbach, K. E.; Moreira, R.; Yunker, L. P. E.; Rooney, M. B.; Deeth, J. R.; Lai, V.; Ng, G. J.; Situ, H.; Zhang, R. H.; Elliott, M. S.; Haley, T. H.; Dvorak, D. J.; Aspuru-Guzik, A.; Hein, J. E.; Berlinguette, C. P. Self-Driving Laboratory for Accelerated Discovery of Thin-Film Materials. *Sci. Adv.* **2020**, 6 (20), eaaz8867. <https://doi.org/10.1126/sciadv.aaz8867>.
- (10) Bennett, J. A.; Abolhasani, M. Autonomous Chemical Science and Engineering Enabled by Self-Driving Laboratories. *Current Opinion in Chemical Engineering* **2022**, 36, 100831. <https://doi.org/10.1016/j.coche.2022.100831>.
- (11) Häse, F.; Roch, L. M.; Aspuru-Guzik, A. Chimera: Enabling Hierarchy Based Multi-Objective Optimization for Self-Driving Laboratories. *Chem. Sci.* **2018**, 9 (39), 7642–7655. <https://doi.org/10.1039/C8SC02239A>.
- (12) Arróyave, R.; Khatamsaz, D.; Vela, B.; Couperthwaite, R.; Molkeri, A.; Singh, P.; Johnson, D. D.; Qian, X.; Srivastava, A.; Allaire, D. A Perspective on Bayesian Methods Applied to Materials Discovery and Design. *MRS Communications* **2022**. <https://doi.org/10.1557/s43579-022-00288-0>.
- (13) Griffiths, R.-R.; Hernández-Lobato, J. M. Constrained Bayesian Optimization for Automatic Chemical Design Using Variational Autoencoders. *Chem. Sci.* **2020**, 11 (2), 577–586. <https://doi.org/10.1039/C9SC04026A>.
- (14) Baird, S.; Hall, J. R.; Sparks, T. D. *Effect of Reducible and Irreducible Search Space Representations on Adaptive Design Efficiency: A Case Study on Maximizing Packing Fraction for Solid Rocket Fuel Propellant Simulations*; preprint; Chemistry, 2022. <https://doi.org/10.26434/chemrxiv-2022-nz2w8>.

- (15) Hickman, R. J.; Aldeghi, M.; Häse, F.; Aspuru-Guzik, A. Bayesian Optimization with Known Experimental and Design Constraints for Chemistry Applications. *arXiv:2203.17241 [cond-mat]* **2022**.
- (16) Baird, S. G.; Liu, M.; Sparks, T. D. High-Dimensional Bayesian Optimization of 23 Hyperparameters over 100 Iterations for an Attention-Based Network to Predict Materials Property: A Case Study on CrabNet Using Ax Platform and SAASBO. *Computational Materials Science* **2022**, 211, 111505. <https://doi.org/10.1016/j.commatsci.2022.111505>.
- (17) Eriksson, D.; Jankowiak, M. High-Dimensional Bayesian Optimization with Sparse Axis-Aligned Subspaces. *arXiv:2103.00349 [cs, stat]* **2021**.
- (18) Tran, A.; Tranchida, J.; Wildey, T.; Thompson, A. P. Multi-Fidelity Machine-Learning with Uncertainty Quantification and Bayesian Optimization for Materials Design: Application to Ternary Random Alloys. *J. Chem. Phys.* **2020**, 153 (7), 074705. <https://doi.org/10.1063/5.0015672>.
- (19) Chen, Y.; Tian, Y.; Zhou, Y.; Fang, D.; Ding, X.; Sun, J.; Xue, D. Machine Learning Assisted Multi-Objective Optimization for Materials Processing Parameters: A Case Study in Mg Alloy. *Journal of Alloys and Compounds* **2020**, 844, 156159. <https://doi.org/10.1016/j.jallcom.2020.156159>.
- (20) Hanaoka, K. Comparison of Conceptually Different Multi-Objective Bayesian Optimization Methods for Material Design Problems. *Materials Today Communications* **2022**, 103440. <https://doi.org/10.1016/j.mtcomm.2022.103440>.
- (21) Daulton, S.; Eriksson, D.; Balandat, M.; Bakshy, E. Multi-Objective Bayesian Optimization over High-Dimensional Search Spaces. *arXiv June 15, 2022*. <https://doi.org/10.48550/arXiv.2109.10964>.
- (22) del Rosario, Z.; Rupp, M.; Kim, Y.; Antono, E.; Ling, J. Assessing the Frontier: Active Learning, Model Accuracy, and Multi-Objective Candidate Discovery and Optimization. *J. Chem. Phys.* **2020**, 153 (2), 024112. <https://doi.org/10.1063/5.0006124>.
- (23) Jekel, C. F.; Venter, G.; Venter, M. P.; Stander, N.; Haftka, R. T. Similarity Measures for Identifying Material Parameters from Hysteresis Loops Using Inverse Analysis. *Int J Mater Form* **2019**, 12 (3), 355–378. <https://doi.org/10.1007/s12289-018-1421-8>.

## Figure legends

Figure 1: Visual bill of materials

Figure 2: Wire mounting instructions

Figure 3: Wire mounting schematic

Figure 4: Light sensor mounting instructions

Figure 5: Hardware connections

Figure 6: Firmware installation dropdown

Figure 7: MicroPython installation dialogue box

Figure 8: Interpreter dropdown

Figure 9: Opening the files sidebar

Figure 10: Editing secrets.py

Figure 11: Saving secrets.py

Figure 12: Uploading source files to microcontroller

Figure 13: Running main.py

Figure 14: Python package installation

Figure 15: Copying the Pico ID from the Thonny editor

Figure 16: Pasting the Pico ID into the Google Colab form box

Figure 17: Example optimization comparison between grid search, random search, and Bayesian optimization averaged over repeated campaigns. Lower Fréchet distance between observed and target spectra is better.

Figure 18: Example optimization comparison between grid search, random search, and Bayesian optimization. Lower error is better.

Figure 19: Twenty-seven grid search points colored by the Fréchet distance between the target spectrum and the sensor data evaluated at each grid point.

Figure 20: Twenty-seven random search points colored by the Fréchet distance between the target spectrum and the sensor data evaluated at each grid point.

Figure 21: Twenty-seven Bayesian optimization points colored by the Fréchet distance between the target spectrum and the sensor data evaluated at each grid point.

Figure 22: The true, underlying RGB target (purple diamond) and the best observed points for grid search (blue circle), random search (red circle), and Bayesian optimization (green circle). Bayesian optimization gave the closest match to the true target.

Methods Video S1: Thread the mounting wire through the mounting holes of the Maker Pi Pico base, related to step 3

Methods Video S2: Thread the remaining mounting wire through the mounting holes of the AS7341 light sensor and position the sensor above the LEDs, related to step 4

Methods Video S3: Attach the Pico W and the AS7341 light sensor to the Maker Pi Pico base, then connect the USB cable from the Pico W to the computer while holding down the BOOTSEL button, related to step 5

Methods Video S4: Download the Thonny editor and install the MicroPython firmware onto the Pico W, related to steps 6, 7, 8, and 9

Methods Video S5: Download the source code from GitHub, unzip it, and enter WiFi credentials, related to steps 10, 11, 12, and 13

Methods Video S6: Upload the source code to the Pico W and run the main.py script, related to steps 14 and 15

Methods Video S7: Open the cloud-control Jupyter notebook via Google Colab and install the self-driving-lab-demo Python package, related to steps 16 and 17

Methods Video S8: Copy-paste the PICO ID from Thonny to Colab and control the setup remotely through the “evaluate” command, related to steps 18 and 19.

Methods Video S9: Perform the “Hello, World!” of optimization, comparing grid search vs. random search vs. Bayesian optimization, related to step 19

Methods Video S10: Visualize the results of the optimization comparison, related to step 19