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

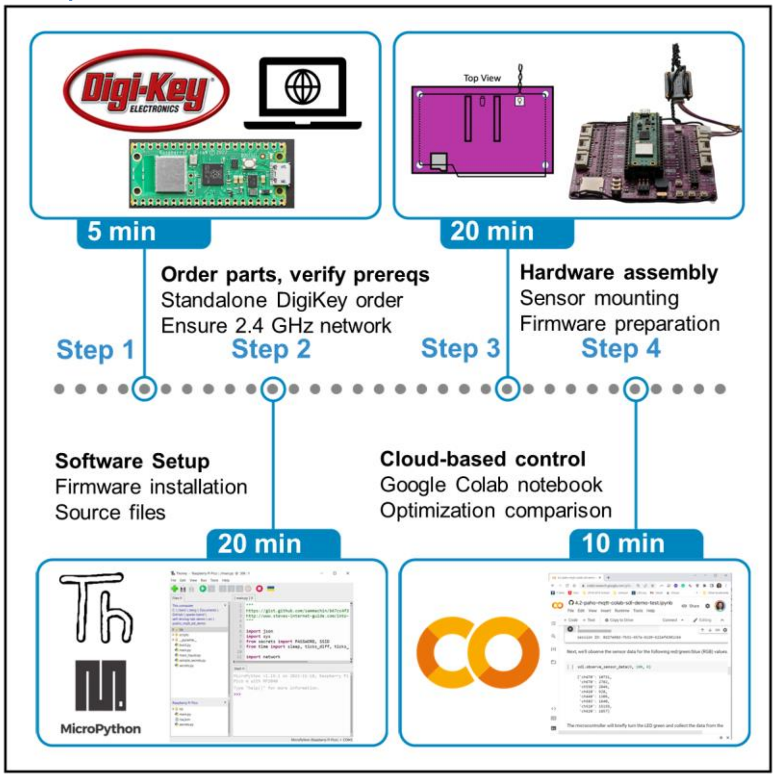
Sterling G. Baird[1,2,*] and Taylor D. Sparks[1,3,]**
[1]Materials Science & Engineering Department, University of Utah, Salt Lake City UT USA, 84108
[2]Technical contact
[3]Lead contact
*Correspondence: sterling.baird@utah.edu
**Correspondence: sparks@eng.utah.edu

Summary

Closed-loop Spectroscopy Lab: Light-mixing Demo (CLSLab:Light) is a teaching and prototyping platform for autonomous scientific discovery. This platform, which consists of a set of LEDs and a light sensor, encapsulates key principles for "self-driving" (i.e. autonomous) research laboratories including sending commands, receiving sensor data, physics-based simulation, and advanced optimization. It serves as 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 a full video build tutorial, please refer to https://youtu.be/D54yfxRSY6s.

For context, please refer to Baird et al. [(1)].

Graphical abstract



Before you begin

The protocol below describes how to set up a "Hello, World!" for a self-driving laboratory[2] 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 parts: (https://www.digikey.com/short/qztj2jt7 AND Pico W with pre-soldered headers) OR https://www.digikey.com/short/vtzjbvr2. For the first option, the total is 68.61 USD (or 73.72 USD including optional parts) + shipping as of 2022-03-06.
   a. The authors plan to periodically check and update the "DigiKey Order" link at https://hackaday.io/project/186289-autonomous-research-laboratories.
   b. In case of part shortages, many products may also be found on the Adafruit website.
   c. If you'd like to avoid soldering, you will need to source a Pico W with headers or a Pico WH separately, such as PiShop's Pico W's with pre-soldered headers. See also Raspberry Pi's supported resellers for the Pico W.

d. The sculpting wire needs to be 14 gauge (2 mm) or thinner, including the insulation jacket, and rigid enough to support the sensor. The sculpting wire is only used for mounting purposes, not to conduct electricity. Sculpting wire is also available at Amazon. Approximately 3' is required. See Problem 5:.
e. The purpose of the wall adapter is so that, after initial setup, the demo can be powered standalone where communication happens purely via Wi-Fi.
f. The hardware and software was designed to work with the Pico W, though the setup can be adapted for other microcontrollers. See Problem 1:.
g. The bill of materials, not including the sculpting wire, is also available at Adafruit.

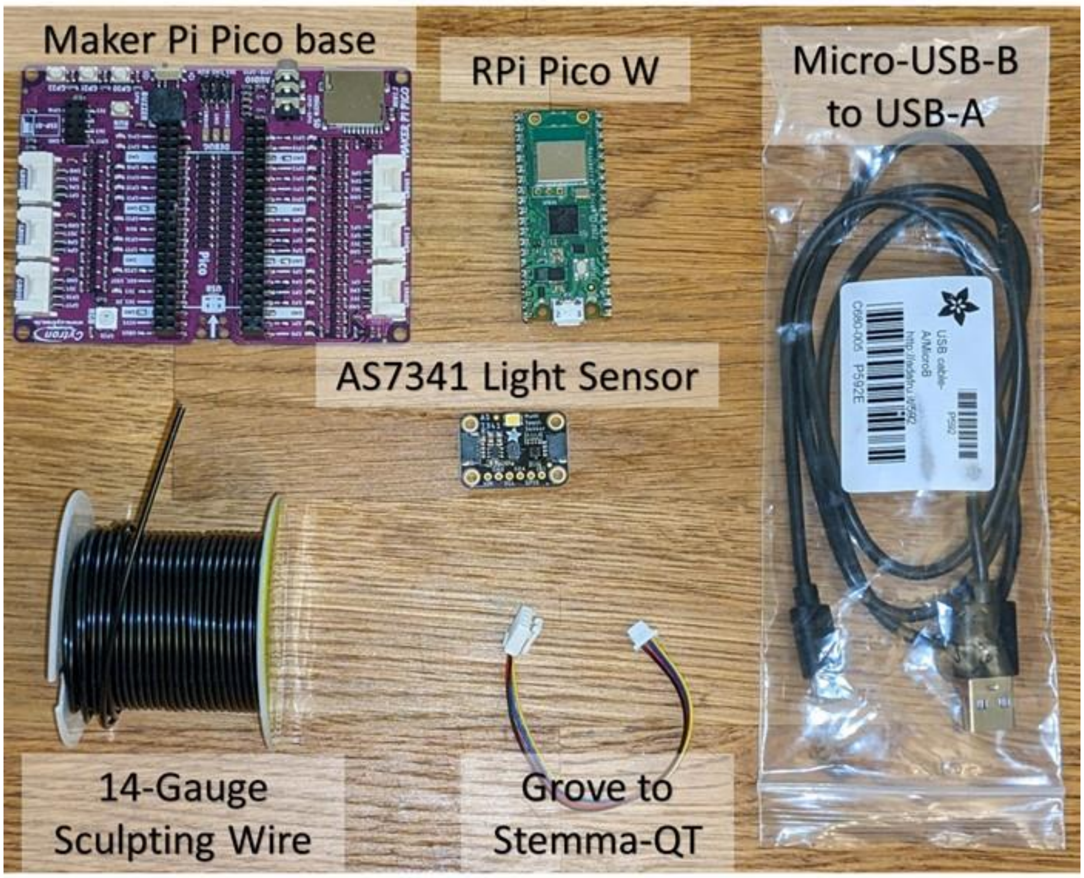


Figure 1

Additional Prerequisites

Timing: N/A

2. Ensure access to a 2.4 GHz Wi-Fi network (SSID + password). The purpose of using a wireless connection rather than a hardwired one is to capture the principles behind “cloud experimentation”, where the host and the client may be separated by large geographical

distances. Additionally, this allows for a computer to only be required for initial setup such that the device can function standalone, waiting to receive commands and send sensor data. This captures best practices of a scaled-up cloud-accessible lab or network of labs. For more context, see https://github.com/sparks-baird/self-driving-lab-demo/discussions/91 and https://github.com/sparks-baird/self-driving-lab-demo/discussions/62. For links to a simple example using a wired connection, see Problem 2:.

a. The Pico W only supports 2.4 GHz Wi-Fi networks. See self-driving-lab-demo #76 for additional context and recommendations on setting up a 2.4 GHz Wi-Fi network, if not already available.
    i WPA enterprise networks such as Eduroam and other networks that use captive portals (most schools, coffee shops, etc.) are not yet supported by MicroPython. It needs to be a network such that on a computer, you can click on the Wi-Fi 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 demos with standalone network access discussions, which includes a summary of recommendations for prepaid mobile hotspots

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 must have at least one USB-A port
4. If the headers are not already soldered onto the microcontroller, ensure access to a soldering iron and soldering wire (thinner is better in this case)
5. (Optional) 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 that the Pico W is working normally
    b. If soldering, be careful only to heat the gold pads to avoid damaging the circuitry

Key resources table

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited Data		
Red, Green, and Blue LED Spectral Data	https://github.com/sparks-baird/self-driving-lab-demo/tree/v0.6.0/src/self_driving_lab_demo/data	v0.6.0
Software and Algorithms		
self-driving-lab-demo v0.6.0	https://github.com/sparks-baird/self-driving-lab-demo	v0.6.0
Other		
AS7341 Color Sensor	DigiKey (Adafruit Product)	Cat#1528-4698-ND

Grove to Stemma-QT adapter	DigiKey (Adafruit Product)	Cat#1528-4528-ND OR Cat#1528-4528-ND
Raspberry Pi Pico W with pre-soldered headers OR (Raspberry Pi Pico W AND Header pins with 20 positions and 2.54 mm pitch (x2))	PiShop OR (DigiKey-Adafruit Product AND DigiKey-Amphenol CS)	Cat#ASM-1918 OR (Cat#2648-SC0918CT-ND AND Cat#10129378-920001BLF-ND)
USB-A to USB-B Cable	DigiKey (Adafruit Product)	Cat#380-1431-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
18 AWG Hook-up solid black wire, 100' (Outer diameter 14 AWG or higher)	DigiKey (Remington Industries)	Cat#2328-18UL1007SLDBLA-ND
Terminal Binding Post M2.5 (Optional)	DigiKey (Keystone Electronics)	Cat#36-8737-ND
128MB Micro SD Memory Card (Optional)	DigiKey (Adafruit Product)	Cat#1528-5250-ND

Step-by-step method details

Hardware Setup

Timing: 20 min

Unless pre-soldered, attach 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. Unless pre-soldered, Solder headers onto the Pico W or use a hammer header pin install rig for Pico W (note, Pico install rigs are not compatible with the Pico W)
    a. If soldering, insert the Pico W headers into the Maker Pi Pico base, place the Pico W on top of the headers, and solder the headers to the Pico W (MagPi guide, Tom's hardware guide, or YouTube video), and 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. For a more modular alternative of fixturing the wire ends to the Maker Pi Pico base, see Problem 5:

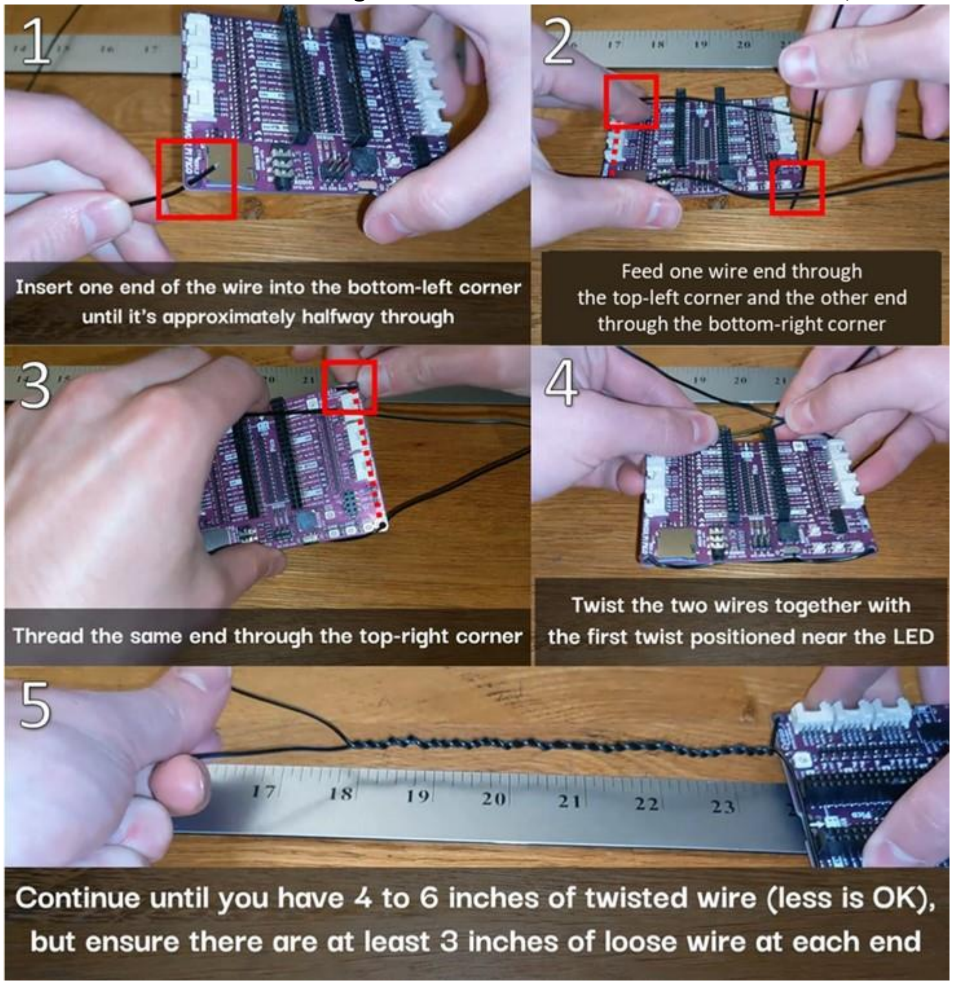

1
Insert one end of the wire into the bottom-left corner until it's approximately halfway through
2
Feed one wire end through the top-left corner and the other end through the bottom-right corner
3
Thread the same end through the top-right corner
4
Twist the two wires together with the first twist positioned near the LED
5
Continue until you have 4 to 6 inches of twisted wire (less is OK), but ensure there are at least 3 inches of loose wire at each end

Figure 2

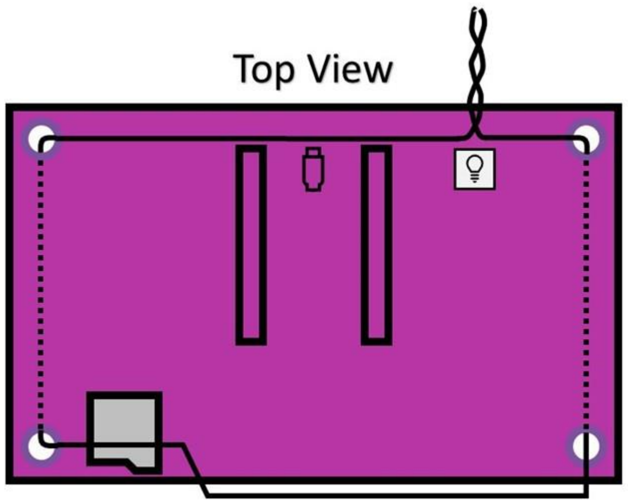


Figure 3

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.

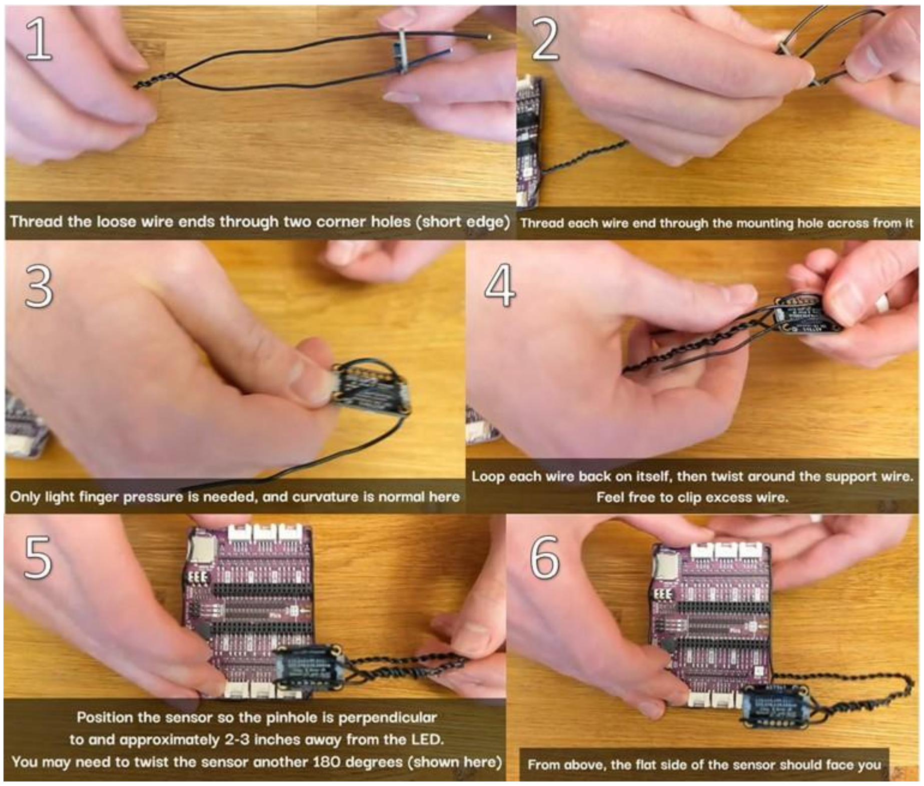

1
Thread the loose wire ends through two corner holes (short edge)
2
Thread each wire end through the mounting hole across from it
3
Only light finger pressure is needed, and curvature is normal here
4
Loop each wire back on itself, then twist around the support wire. Feel free to clip excess wire.
5
Position the sensor so the pinhole is perpendicular to and approximately 2-3 inches away from the LED. You may need to twist the sensor another 180 degrees (shown here)
6
From above, the flat side of the sensor should face you

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.

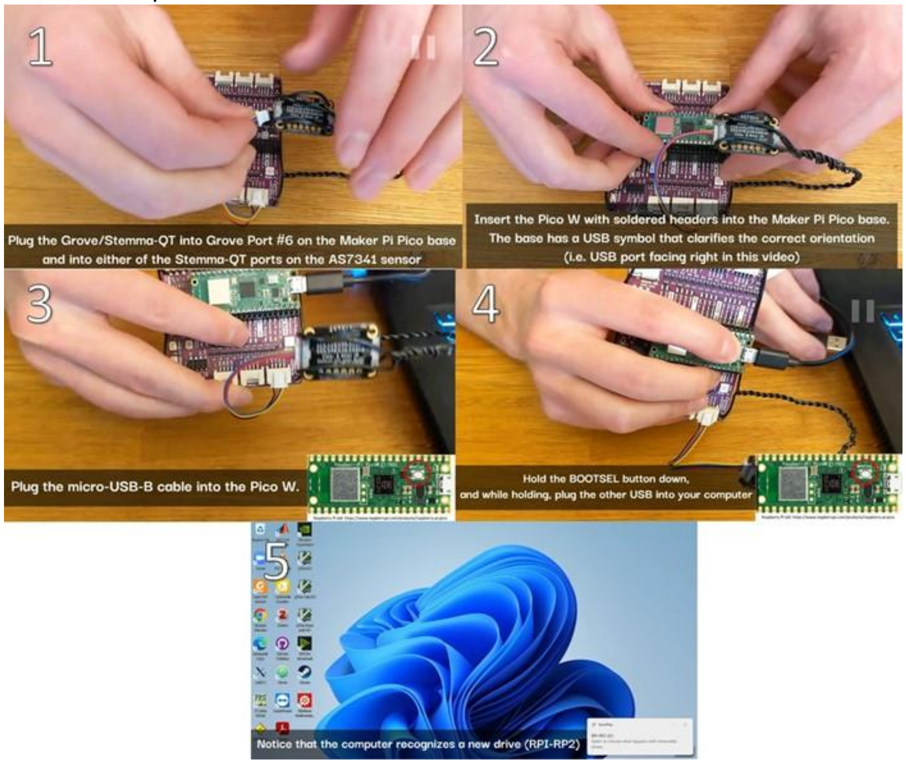

1
Plug the Grove/Stemma-QT into Grove Port #6 on the Maker Pi Pico base and into either of the Stemma-QT ports on the AS7341 sensor
2
Insert the Pico W with soldered headers into the Maker Pi Pico base. The base has a USB symbol that clarifies the correct orientation (i.e. USB port facing right in this video)
3
Plug the micro-USB-B cable into the Pico W.
4
Hold the BOOTSEL button down, and while holding, plug the other USB into your computer
5
Notice that the computer recognizes a new drive (RPI-RP2)

Figure 5

Software Setup

Timing: 20 min

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

6. Download and install Thonny, a Python IDE with native support for microcontrollers, onto your computer. 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)". Thonny comes with its own version of Python located by default at C:\Users\<username>\AppData\Local\Programs\Thonny\python.exe on Windows computers. It is not anticipated that this will cause conflicts with existing installations of Python; however, for conda users, an isolated installation may be performed via the following commands in a conda shell:
```
conda create -n sdl-demo-thonny python==3.10.*
conda activate sdl-demo-thonny
pip install thonny
thonny
```

7. Click on the lower-right dropdown and click "Install MicroPython", which will install the microcontroller firmware onto the Pico W

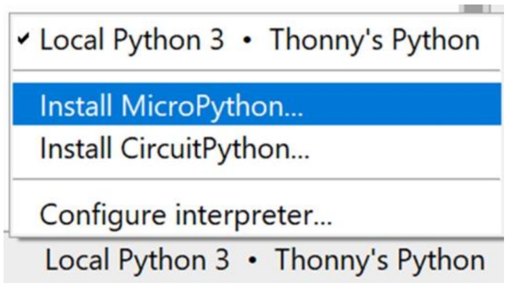
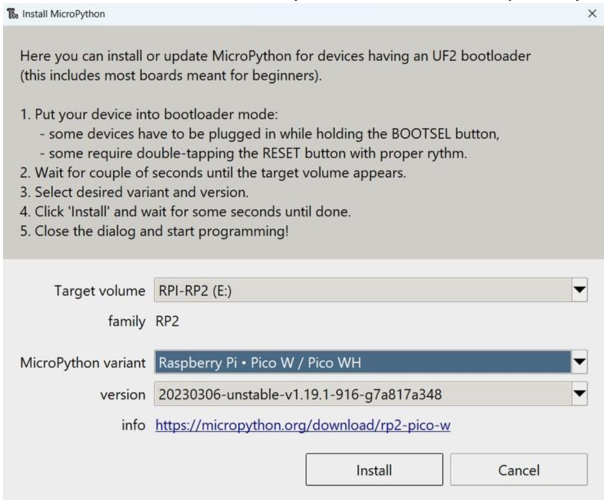


Figure 6

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



Install MicroPython

Here you can install or update MicroPython for devices having an UF2 bootloader
(this includes most boards meant for beginners).

1. Put your device into bootloader mode:
   - some devices have to be plugged in while holding the BOOTSEL button,
   - some require double-tapping the RESET button with proper rythm.
2. Wait for couple of seconds until the target volume appears.
3. Select desired variant and version.
4. Click 'Install' and wait for some seconds until done.
5. Close the dialog and start programming!

Target volume RPI-RP2 (E:)
family RP2
MicroPython variant Raspberry Pi • Pico W / Pico WH
version 20230306-unstable-v1.19.1-916-g7a817a348
info https://micropython.org/download/rp2-pico-w

Figure 7

9. Change the interpreter from Local Python 3 to MicroPython (Raspberry Pi Pico), which will open a shell that can be used to enter MicroPython commands that run directly on the Pico W

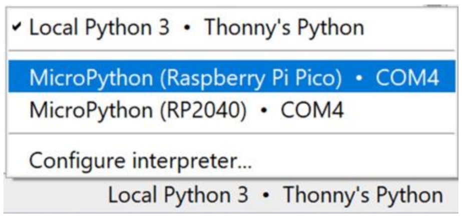

✔ Local Python 3 • Thonny's Python
MicroPython (Raspberry Pi Pico) • COM4
MicroPython (RP2040) • COM4
Configure interpreter...
Local Python 3 • Thonny's Python

Figure 8

10. In Thonny's menubar, click "View" then "Files" to open a sidebar which shows both your local computer's files (top) and the files on the Pico W (bottom)

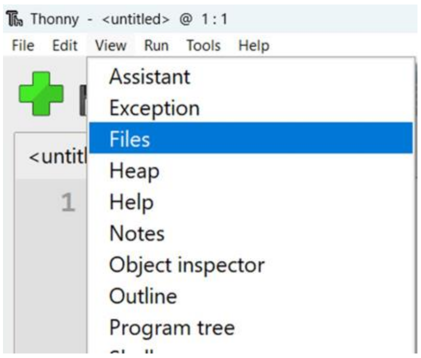
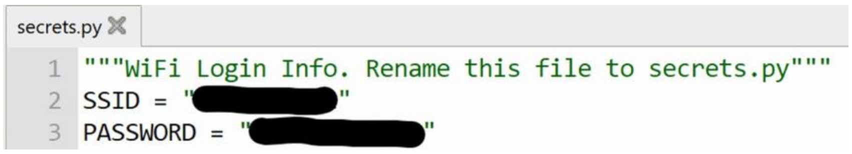


Figure 9

11. Download *sdl_demo.zip* from the latest release at self-driving-lab-demo to your computer and unzip it
12. In Thonny, navigate to the unzipped *sdl_demo* folder, open *secrets.py*, and enter your Wi-Fi 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.*


secrets.py

```
"""WiFi Login Info. Rename this file to secrets.py"""
SSID = "[redacted]"
PASSWORD = "[redacted]"
```

Figure 10

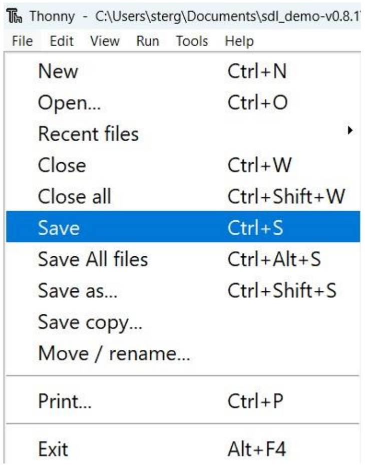
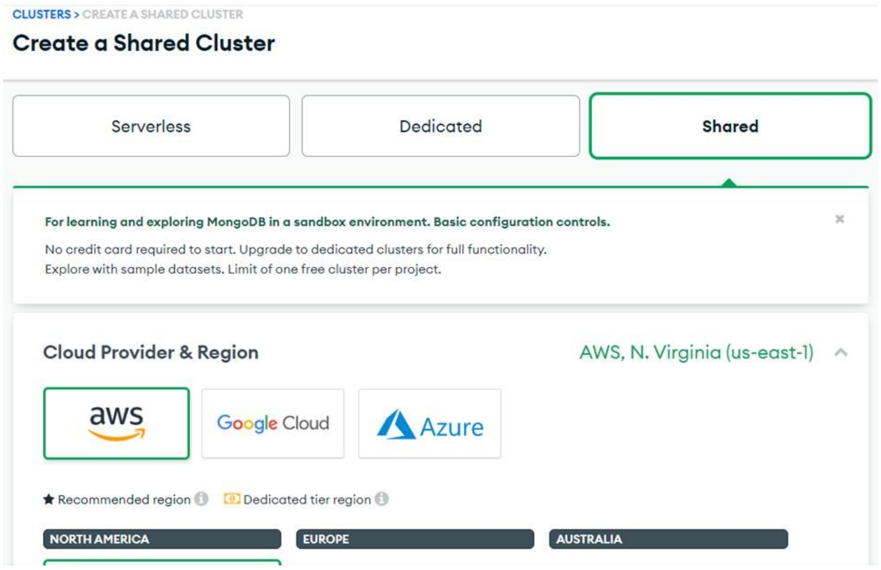


Figure 11

a. (Optional) Set up a MongoDB database backend. If ignored, the demo will function, just without logging data to a database (i.e., the user becomes responsible for saving the data on the client side). See Problem 3:.
   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)

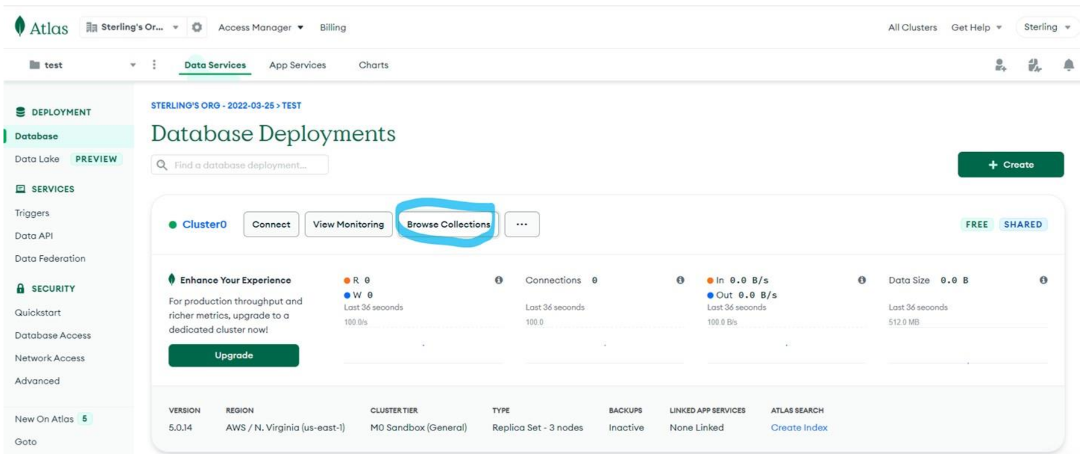


For learning and exploring MongoDB in a sandbox environment. Basic configuration controls.

No credit card required to start. Upgrade to dedicated clusters for full functionality.
Explore with sample datasets. Limit of one free cluster per project.

Figure 12

   iii. Navigate to "Data Services" → "Deployment" → "Database" and click "Browse Collections" then "Add My Own Data". Enter a database name (e.g., clslab-light-mixing) and collection name (e.g., test). Copy the names into MONGODB_DATABASE_NAME and MONGODB_COLLECTION_NAME in secrets.py.



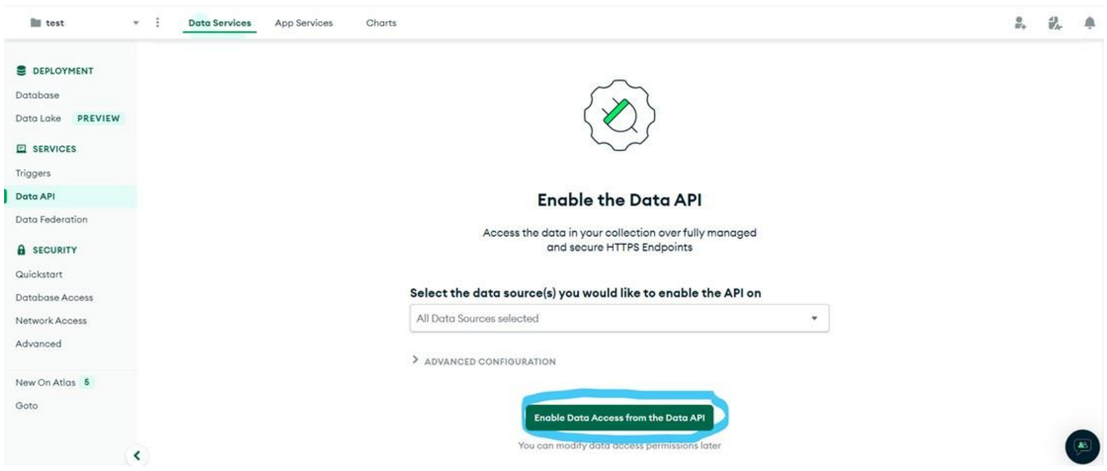
Enhance Your Experience

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

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

Figure 13

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

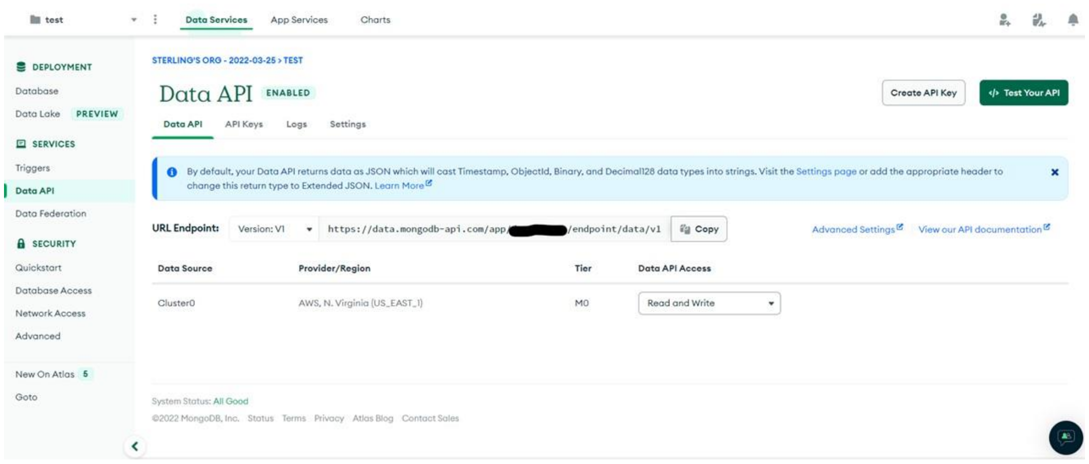
ADVANCED CONFIGURATION

Enable Data Access from the Data API

You can modify data access permissions later

Figure 14

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

By default, your Data API returns data as JSON which will cast Timestamp, ObjectId, Binary, and Decimal128 data types into strings. Visit the Settings page or add the appropriate header to change this return type to Extended JSON. Learn More

URL Endpoint: Version: V1 `https://data.mongodb-api.com/app/`[redacted]`/endpoint/data/v1` Copy

Advanced Settings View our API documentation

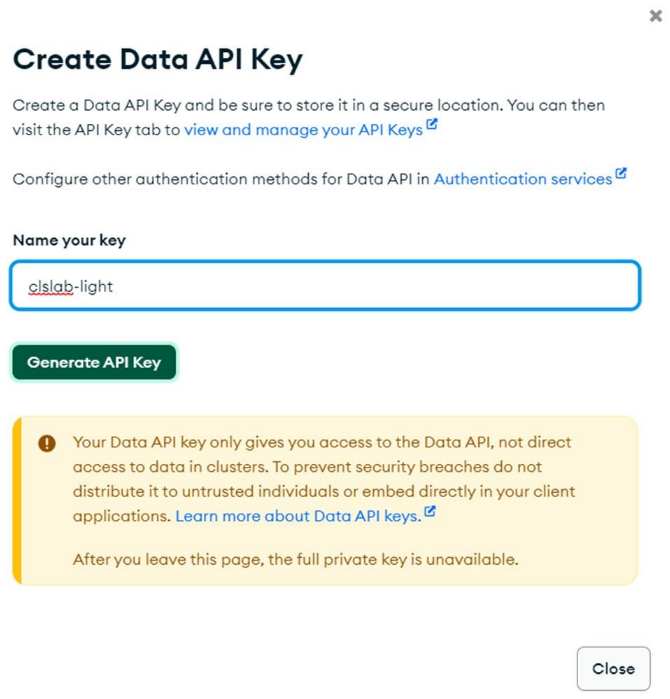
Data Source	Provider/Region	Tier	Data API Access
Cluster0	AWS, N. Virginia (US_EAST_1)	M0	Read and Write

System Status: All Good

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

Figure 15

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

clslab-light

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.

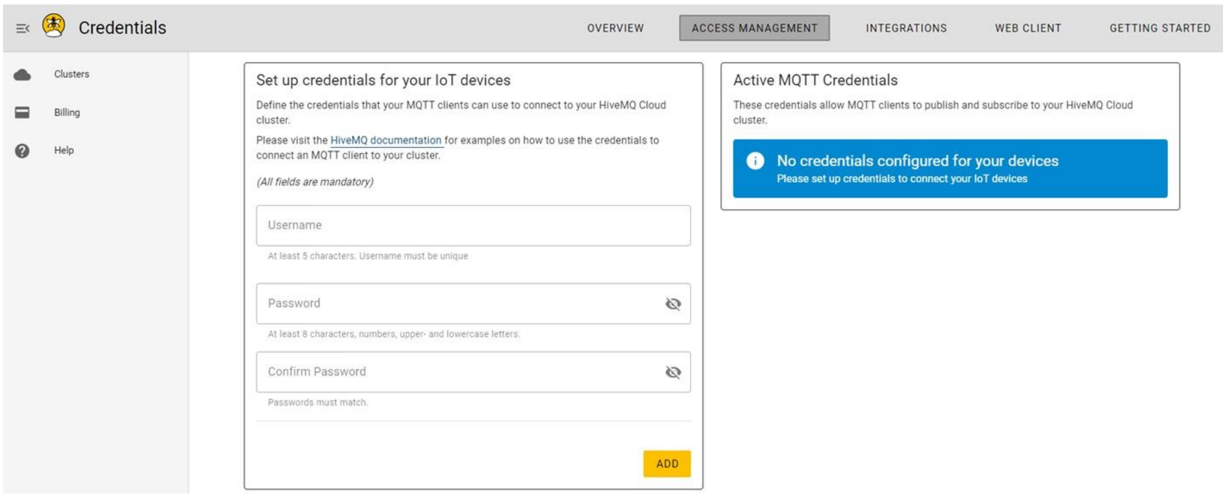
After you leave this page, the full private key is unavailable.

Figure 16

b. (Optional) Create your own HiveMQ instance. If this setup is ignored, the demo will function properly; however, the hardware commands and sensor data will be transmitted via a default HiveMQ instance for which the credentials are public. Setting up your own HiveMQ instance ensures that the data you transfer remains private and secure. Other MQTT brokers such as Mosquitto or Adafruit IO are available. At the time of writing, we recommend HiveMQ because it provides free instances with generous limits. Setting up a private MQTT broker is in line with best practices for internet of things (IoT) security.

i. Navigate to https://www.hivemq.com/mqtt-cloud-broker/, click "Try out for free", and create an account

ii. Set up credentials by entering a username and password and press "ADD"



Set up credentials for your IoT devices

Define the credentials that your MQTT clients can use to connect to your HiveMQ Cloud cluster.

Please visit the HiveMQ documentation for examples on how to use the credentials to connect an MQTT client to your cluster.

(All fields are mandatory)

Username

At least 5 characters. Username must be unique

Password

At least 8 characters, numbers, upper- and lowercase letters.

Confirm Password

Passwords must match.

Active MQTT Credentials

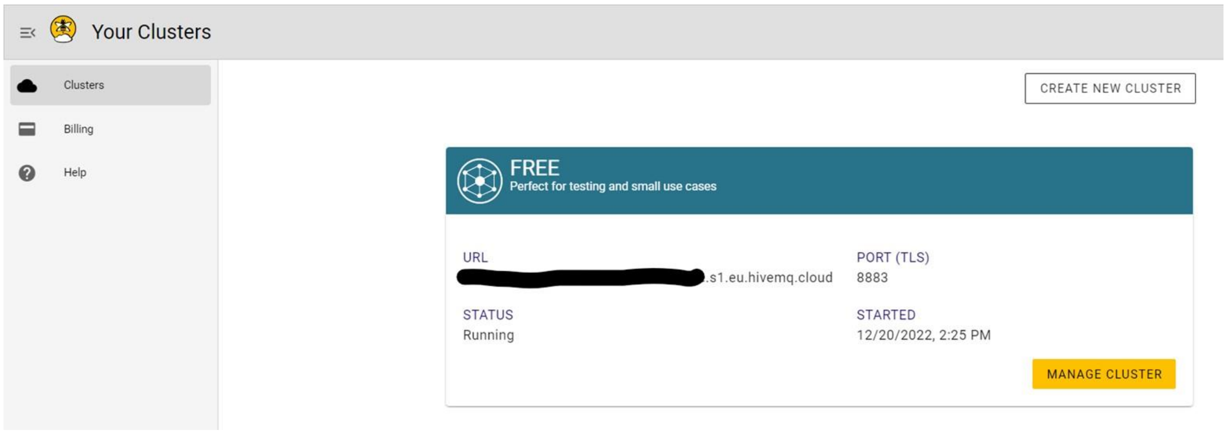
These credentials allow MQTT clients to publish and subscribe to your HiveMQ Cloud cluster.

No credentials configured for your devices

Please set up credentials to connect your IoT devices

Figure 17

iii. 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.



FREE

Perfect for testing and small use cases

URL
[redacted].s1.eu.hivemq.cloud

PORT (TLS)
8883

STATUS
Running

STARTED
12/20/2022, 2:25 PM

Figure 18

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 /". This will upload the files from your computer to the Pico W microcontroller

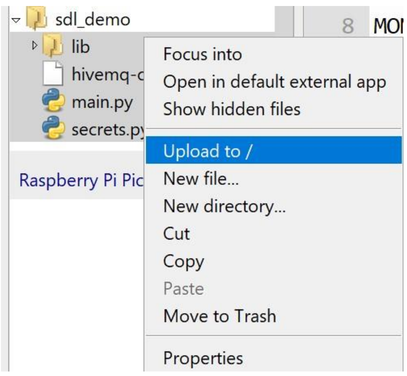
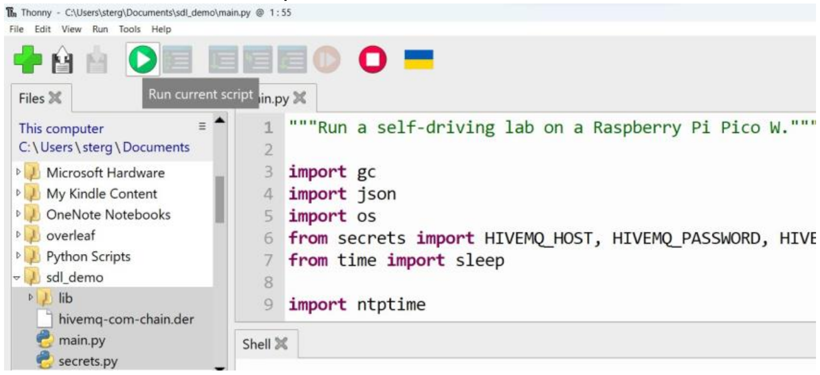


Figure 19

14. Double click to open *main.py*, click the green play button (i.e., run the code on the Pico W), 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.



```
"""Run a self-driving lab on a Raspberry Pi Pico W."""

import gc
import json
import os
from secrets import HIVEMQ_HOST, HIVEMQ_PASSWORD, HIVE
from time import sleep

import ntptime
```

Figure 20

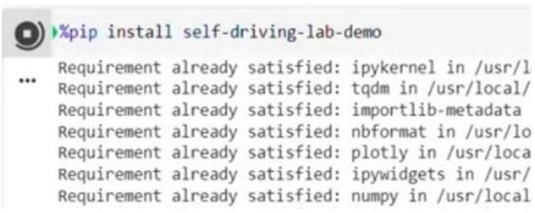
Control from the cloud

Timing: 10 min

Control the device 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]

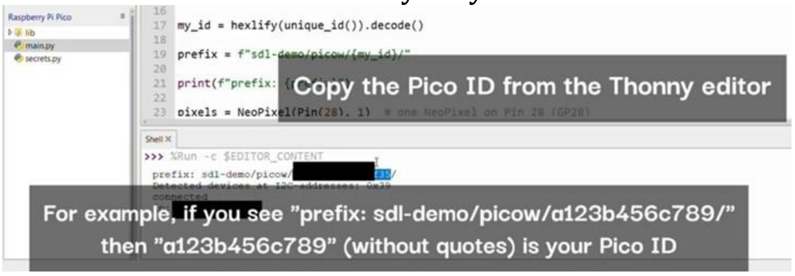
17. Scroll to the first code cell and click the play button to install the self-driving-lab-demo Python package



```
%pip install self-driving-lab-demo
Requirement already satisfied: ipykernel in /usr/l
Requirement already satisfied: tqdm in /usr/local/
Requirement already satisfied: importlib-metadata
Requirement already satisfied: nbformat in /usr/lo
Requirement already satisfied: plotly in /usr/loca
Requirement already satisfied: ipywidgets in /usr/
Requirement already satisfied: numpy in /usr/local
```

Figure 21

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.



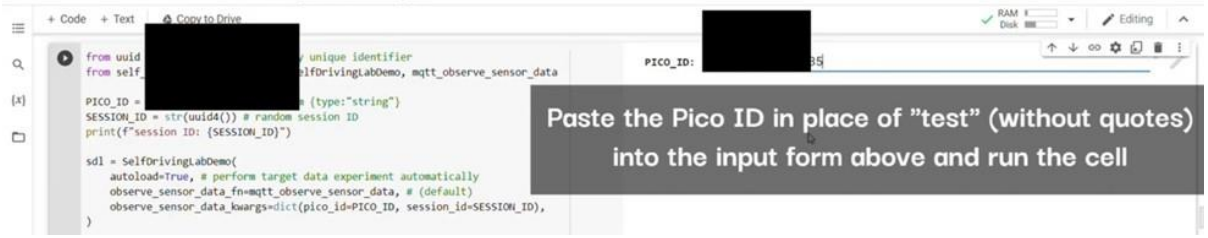
```
my_id = hexlify(unique_id()).decode()

prefix = f"sdl-demo/picow/{my_id}/"

print(f"prefix: {

pixels = NeoPixel(Pin(28), 1)  # one NeoPixel on Pin 28 (GP28)
```

```
>>> %Run -c $EDITOR_CONTENT
prefix: sdl-demo/picow/[redacted]35/
Detected devices at I2C-addresses: 0x39
connected
[redacted]
```

Copy the Pico ID from the Thonny editor

For example, if you see "prefix: sdl-demo/picow/a123b456c789/" then "a123b456c789" (without quotes) is your Pico ID

Figure 22



```
from uuid [redacted] unique identifier
from self_ elfDrivingLabDemo, mqtt_observe_sensor_data

PICO_ID = {type:"string"}
SESSION_ID = str(uuid4()) # random session ID
print(f"session ID: {SESSION_ID}")

sdl = SelfDrivingLabDemo(
    autoload=True, # perform target data experiment automatically
    observe_sensor_data_fn=mqtt_observe_sensor_data, # (default)
    observe_sensor_data_kwargs=dict(pico_id=PICO_ID, session_id=SESSION_ID),
)
```

PICO_ID: [redacted]35

Paste the Pico ID in place of "test" (without quotes) into the input form above and run the cell

Figure 23

19. Run the remaining code cells
    a. Instantiate a SelfDrivingLabDemo class
    b. Perform optimizations for grid search, random search, and Bayesian optimization

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 24 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 25 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 26, 27, and 28 show the points that were searched for a given campaign for grid search, random search, and Bayesian optimization, respectively. Finally, Figure 29 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.

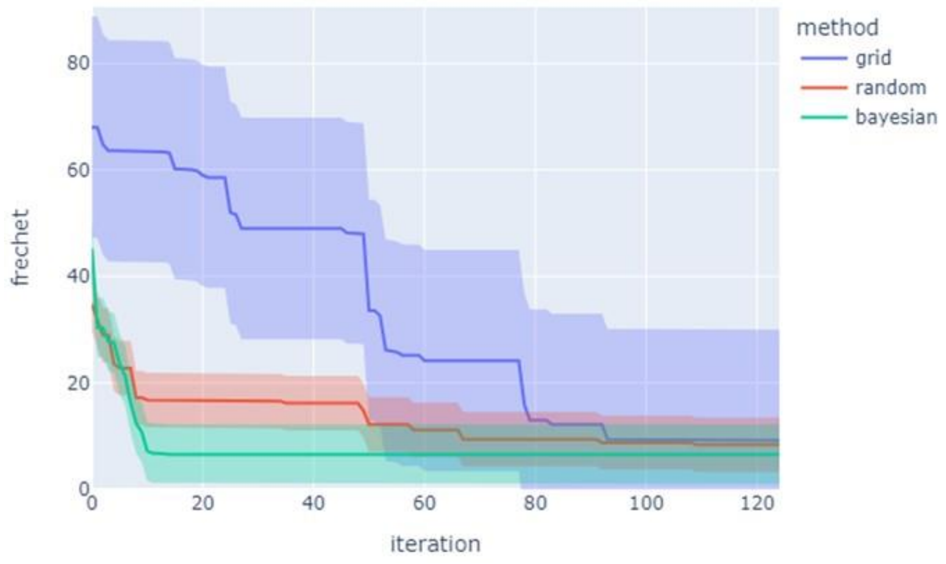


Figure 24

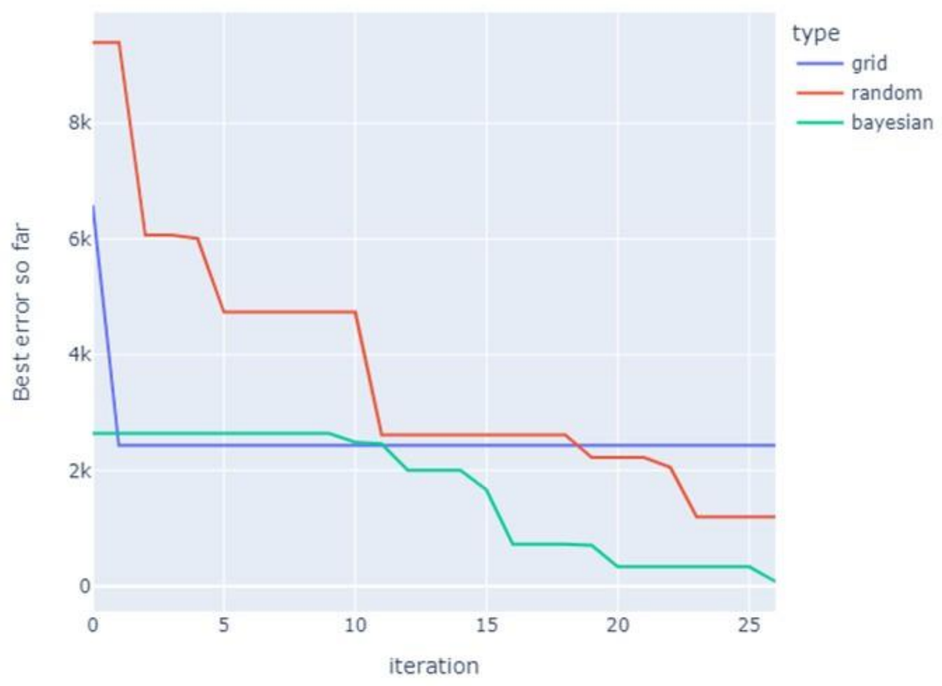


Figure 25

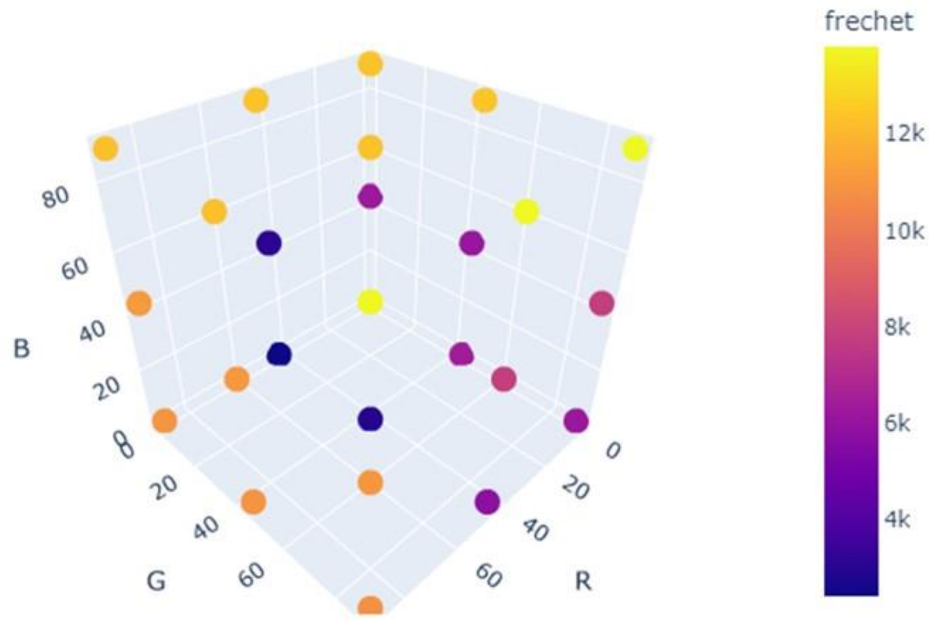


Figure 26

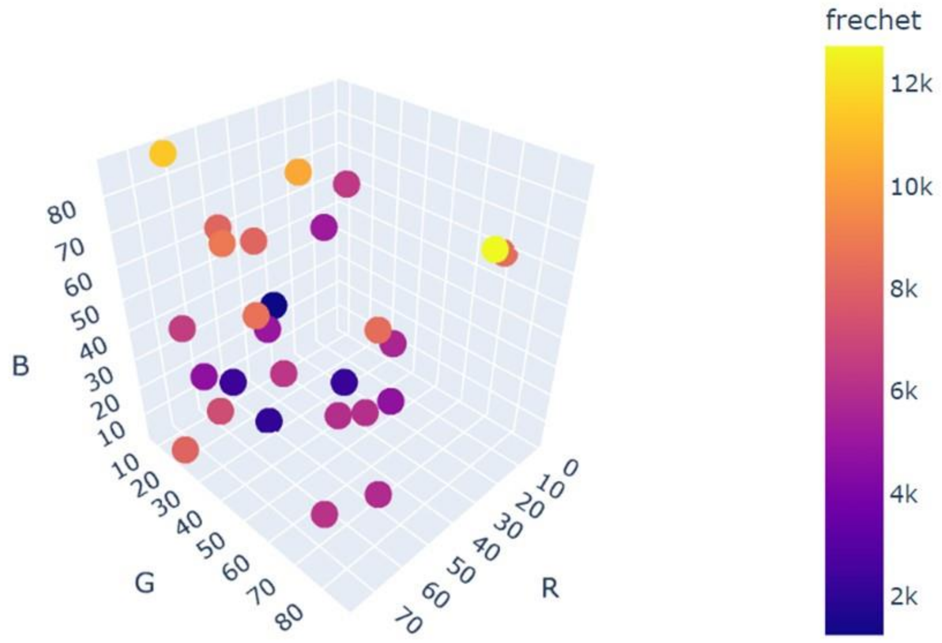


Figure 27

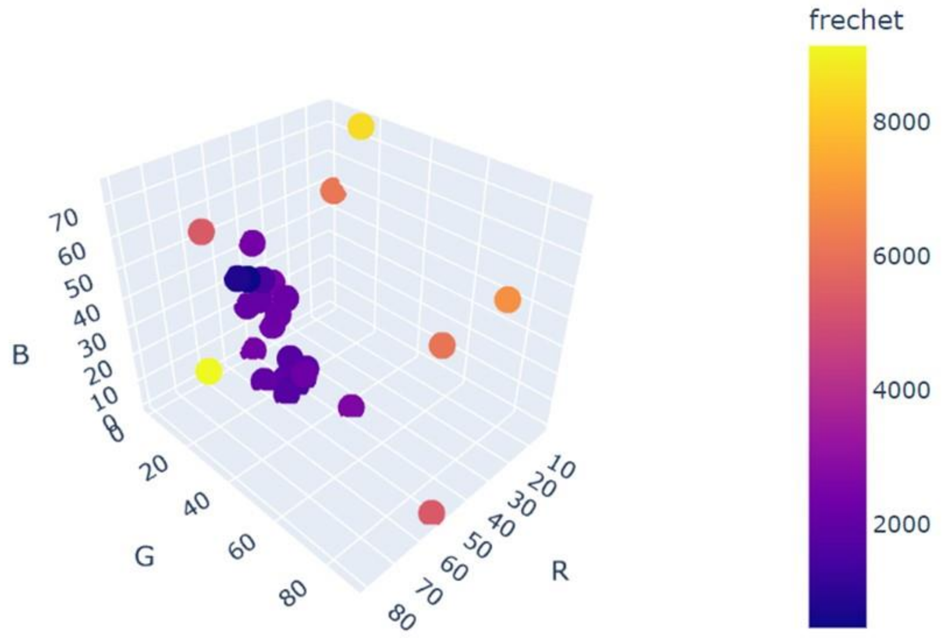


Figure 28

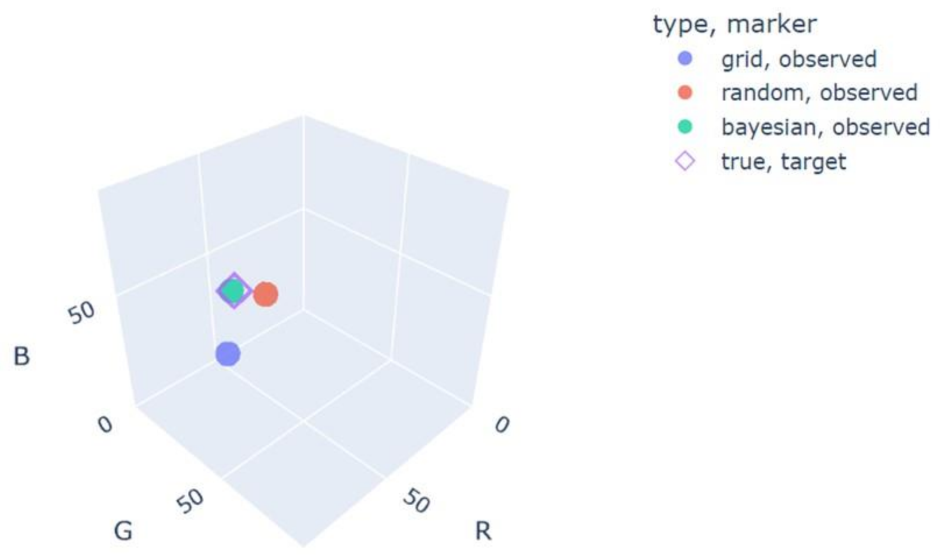


Figure 29

Quantification and statistical analysis

Discrete Fréchet distance, as implemented in https://github.com/cjekel/similarity_measures, 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	astep	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

The code for grid search, random search, and Bayesian optimization is hosted at https://github.com/sparks-baird/self-driving-lab-demo/blob/main/src/self_driving_lab_demo/utils/search.py [permalink].

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. See page 3.

Problem 2:

Can I use this without connecting to the internet?

Potential solution:

A simple example of wired communication between a computer and the microcontroller for the microcontroller host code and a Jupyter notebook tutorial (client) can be found at https://github.com/sparks-baird/self-driving-lab-demo/tree/main/src/extra/nonwireless [permalink] and https://github.com/sparks-baird/self-driving-lab-demo/blob/main/notebooks/5.0-nonwireless-search.ipynb [permalink], respectively. While possible with some modification, data communication via a USB cable is not actively supported for new releases of microcontroller host code nor the advanced tutorials. For private, secure, wireless 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. For recommendations regarding connecting to a 2.4 GHz network (e.g., in university classroom settings) see https://github.com/sparks-baird/self-driving-lab-demo/discussions/83 and https://github.com/sparks-baird/self-driving-lab-demo/discussions/88. See also page 3.

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. In other words, the device will function normally without database logging. 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 this step will be skipped. See page 11.

Problem 4:

The Stemma-QT to Grove connectors are out-of-stock.

Potential solution:

First, look at Adafruit and other vendors to see if it is available. Note that Cat#1528-4424-ND is incompatible with the Maker Pi Pico base due to the adapter housing blocking it from being plugged in fully. If no Stemma-QT to Grove connectors can be located, 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. See page 2.

Problem 5:

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. Alternatively, for a more modular setup, a single M2.5 binding post (Digikey Cat#36-8737-ND) can be used to clamp the wire via a single mounting hole instead of looping the wire through each of the mounting holes. See page 3.

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.

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. The recommended option for ordering parts is https://www.digikey.com/short/qztj2jt7 AND Pico W with pre-soldered headers to avoid soldering. Alternatively, https://www.digikey.com/short/vtzjbvr2 is a standalone DigiKey order, but requires soldering headers onto the Pico W. For a full video build tutorial, please refer to https://youtu.be/D54yfxRSY6s. Code for grid search, random search, and Bayesian optimization is hosted at https://github.com/sparks-baird/self-driving-lab-demo/blob/main/src/self_driving_lab_demo/utils/search.py.

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 have been exploring selling at-cost kits via a crowdfunding platform called GroupGets. As of 2023-03-04, the cost-breakdown is as follows: List price: 75 USD + shipping (depends on location). GroupGets fee: 15.57 USD (10%+2.9%+5 USD). Hardware cost: 56.19 USD. Labor/testing: 20 min (est. 6 USD). Profit: -2.76 USD, offset somewhat by bulk pricing discounts. Note that this assumes purchasing a Pico W with presoldered headers. See round 1 and round 2 and a discussion of packaging open-source hardware as commercial kits.

References

a. S.G. Baird, T.D. Sparks, "What is a Minimal Working Example for a Self-driving Laboratory?" Matter, Cell Press, 2022. 5 (12), 4170–4178. https://doi.org/10.1016/j.matt.2022.11.007.

b. 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, Cell Press, 2022. 5 (7), 1972–1976. https://doi.org/10.1016/j.matt.2022.05.035.