START HERE

01

Read a log. Prove one reply.

Use a browser as a serial terminal: connect to the ESP32-S3, read what it says, send one known Python statement and save a useful diagnostic summary. Plan about **25 minutes**, including the hands-on checks.



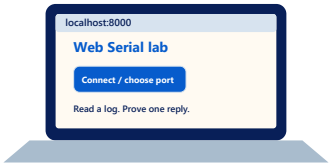
ESP32-S3-N16R8

Use the native USB socket: left in the component-side view, antenna up.



USB data cable

A known data-capable cable that fits your board and computer.



Computer + browser

Use a supported desktop browser and the included local practice page.

Before the live practice

- MicroPython is already installed and known to work; ML-C03 covers firmware setup.
- The computer detects the board's current serial port; ML-C04 covers cables and drivers.
- Desktop Python is available to start a local file server.
- Peripherals and external power are disconnected. Use one USB data cable.

Your finish line: see the received line `ML-C08 OK`, disconnect cleanly, then describe the first missing observation in a failed attempt.

This is Web Serial, not WebREPL. Web Serial carries serial data through the computer's browser. MicroPython WebREPL is a different, network-based feature.

CHECKPOINT 1 · PAGE SUPPORT

Check the page before the board

02

Web Serial is a browser interface for reading and writing a serial port. It needs browser support and a trustworthy page context. Use the desktop version of Chrome for this worked lab; another browser is suitable only when the page reports Web Serial available.

1	2	3	4	5
Page	**Permission**	**Port**	**Runtime**	**Program**
Secure context Web Serial API	Click Connect Select current port	One tool only Open at 115200	Known firmware MicroPython >>>	Send one line Check RX reply

Find the first missing observation before changing anything else.

Permission, an open port and a Python reply are separate observations.

Start the included local page

Keep the course output folder and its `practice` subfolder together. Open a computer terminal **inside the course output folder**, then run:

```
python -m http.server 8000 --bind 127.0.0.1
```

In the browser, open `http://localhost:8000/practice/serial-lab.html`. Leave the terminal running during the exercise. Stop the local server with Ctrl-C when finished.

> **Look for:** "Secure context: yes" and "Web Serial: available." Localhost is treated as trustworthy by supporting browsers. A remote plain-HTTP page is not the same thing.

> **No API or secure context?** Fix this first. Open the localhost address in a supported desktop browser. Do not change firmware or install a random driver to fix a missing browser API.

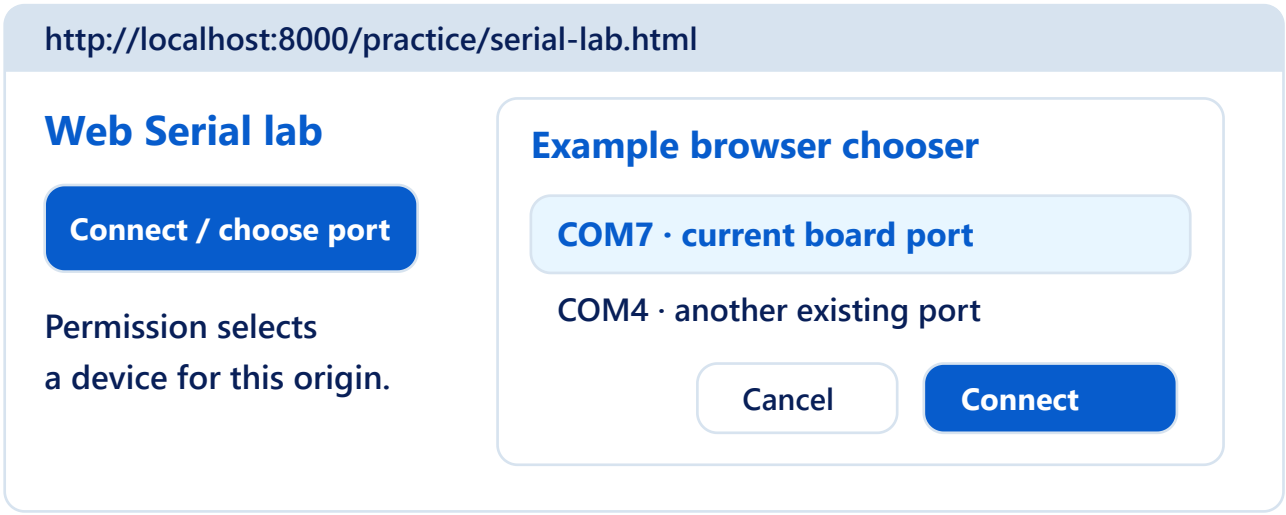
Read the course on mobile if useful; perform this USB browser lab on the supported desktop setup. Browser permissions or organization policy can still restrict a supported API.

CHECKPOINT 2 · PERMISSION

Choose the current device

03

Connect the data cable to native USB. Click **Connect / choose port** in the lab. A user click lets the browser ask which serial device this page may access.



Illustrated example. Device names and chooser wording vary by operating system and browser.

1. Identify the entry that appears and disappears with this board, as in ML-C04.
2. Select that entry in the chooser. Do not select an unrelated device.
3. Confirm the browser's connection choice. Watch the lab's session status.

Cancel is an observation

If you cancel, no new port opens. Click Connect again when ready; cancellation does not prove a cable or firmware fault.

Permission belongs to an origin

A remembered grant may let that same page request access again. It does not mean a port is currently open or another application has released it.

Next checkpoint: the lab reaches **open**. If the chooser appeared but opening failed, investigate the port/open layer before debugging Python.

CHECKPOINT 3 · OPEN SESSION

04

Give one tool the port

Close editor serial monitors, mpremote sessions, other serial pages and duplicate tabs before opening this lab. A serial port normally has one open owner at a time.

One serial port

Current board runtime connection

→ **Browser owns the open session**
Read and write here.

✕ **Editor / mpremote / another tab**
Close these before opening the browser.

A permission grant is not proof that the port opened successfully.

Granting browser permission does not close another program's serial session.

1 · Release
Close every other serial session.

2 · Select
Choose the board's current port.

3 · Observe
Wait for the lab to show open.

Why 115200?

The lab opens a serial session with `baudRate: 115200`, matching MicroPython's documented serial-console setting. It is a communication setting, not a firmware version or a USB power rating.

Reset can change the port. ROM download mode and normal MicroPython USB can enumerate under different names. Release BOOT, tap RST for normal startup, then identify and select the current runtime port again.

Open failed? Record the error name and the last successful step. Check competing tools, the current port, cable and device state. Do not treat every open error as a missing driver.

CHECKPOINT 4 · RUNTIME

05

Read before sending

An open serial port tells you that the browser opened a connection. Read the received text to learn which firmware or device state is responding.

ROM download · example

```
DOWNLOAD ...
waiting for firmware
```

Do not send Python here.

MicroPython · example

```
MicroPython ...
>>>
```

Python can accept a statement.

Read the state first. A blank log is not proof of a dead board.

Example states, not a transcript from your board. Exact boot messages vary.

Find the Python prompt

1. Click **Send Enter** to request a fresh line.
2. If a known MicroPython program is running, **Interrupt (Ctrl-C + Enter)** can stop it and return to the prompt.
3. Look for >>> from the known MicroPython setup.

Interrupt changes execution. It stops a running Python program; it is not a passive log read. Save any observation you need first. Keep attached hardware disconnected for this exercise.

No readable reply? First check whether the correct runtime port is open. ROM download text means you need normal startup before Python. A blank log alone is not proof the board is broken.

Do not send Python statements to unknown firmware. If you cannot establish the known MicroPython state, return to the firmware or USB lesson before continuing.

CHECKPOINT 5 · APPLICATION REPLY

Send one harmless test

06

When the known MicroPython prompt is visible, tick **I can see the >>> prompt** in the lab. Then send the fixed print test.

Example exchange · known MicroPython

```
>>>
[TX] print("ML-C08 OK") + Enter
ML-C08 OK
>>>
```

Check received text, not just the transmitted command echo.

`[TX]` marks text sent by the page. The received output after it is the check that matters.

Send

`print("ML-C08 OK")` followed by a carriage return.

Receive

Look for `ML-C08 OK`.

Ready again

Look for another `>>>` prompt.

Success: the marker arrives from the board and the prompt returns. This verifies a small browser → serial → Python → browser exchange.

An echoed command is not its result. Seeing `[TX] print(...)` only proves the page attempted a send. A Python error, missing marker or missing prompt needs a follow-up check.

This test prints text. It does not erase firmware, save a file, change Wi-Fi configuration or drive a GPIO.

FINISH THE SESSION

07

Disconnect before switching tools

Click **Disconnect** and wait for **closed** before unplugging, resetting for a different mode or opening another serial tool.

1 · Disconnect

Cancel pending read

Release reader + writer

2 · Wait for closed

The browser closes

its serial session

3 · Change tools

Then unplug or open

another serial tool

Close the session before unplugging or switching applications.

The lab ends a pending read, releases stream locks and closes the serial port.

Close is not forget

Disconnect closes this open session. A browser may retain the page's device permission. Remove that grant in browser site settings when needed on a shared computer.

Unexpected unplug

The stream may fail or stop. Use Disconnect to release the session, reconnect the device and identify its current port before trying again.

Repeat a clean cycle

1. Disconnect and confirm closed.
2. Reconnect using the current runtime port.
3. Read the prompt, send the print test and check the received marker.
4. Disconnect again before leaving the exercise.

> **Explain it:** why should an editor be able to open the port after the browser shows closed? The browser has released its open session.

USEFUL EVIDENCE

08

Capture the first failing layer

A useful report names the last successful observation and the first missing one. Keep it small enough for another person to repeat the check.

Example diagnostic summary

Secure context: true · Web Serial present: true

Last step: open · Last error: NetworkError

First missing observation: no open session

Browser/version: add from the browser About page

Raw serial log is excluded. Review before sharing.

The lab's summary excludes the raw receive log. Add only the relevant, reviewed detail.

Prepare a short report

1. Click **Prepare summary** in the lab.
2. Add your browser/version from its About page.
3. Replace the first-missing-observation placeholder with one short fact, such as "chooser appeared; opening failed."
4. Review the report, then copy it. Include a small redacted log excerpt only if needed.

Review logs before sharing. Remove Wi-Fi passwords, tokens, personal paths, account details and unrelated device identifiers. Do not copy a full startup log into a public issue or AI prompt by default.

Practice with an easy failure: disconnect, click Connect and cancel the chooser. Record "chooser cancelled; no port opened." Then reconnect normally. This exercises reporting without damaging the board.

TROUBLESHOOTING AND COMPLETION

Use the first missing observation

09

Page / API missing

Use the localhost address and a supported desktop browser. Check policy or permissions. The board's firmware cannot add an API to a browser.

No board in chooser

Check the USB socket, data cable, OS detection and current mode. Return to ML-C04 before changing application code.

Permission granted; open fails

Close competing tools. Identify the current port again. Capture the error name and the open checkpoint.

Open; no Python prompt

Read existing text, send Enter and interrupt only the known MicroPython program. Check normal startup and runtime port.

TX appears; no marker

Confirm the prompt and known firmware. Distinguish command echo from output. Read the Python error if one arrived.

Disconnect or mode change

Close the old session, reconnect and rediscover the port. A remembered permission does not guarantee the same port still exists.

Your completion check

- □ Recognize the page-support, permission, port, runtime and program checkpoints.
- □ Read the known MicroPython prompt and receive `ML-C08 OK`.
- □ Repeat a clean connect → test → disconnect cycle.
- □ Write a short redacted report that names the first failing layer.

> **Explain the difference:** choosing a port grants access; opening creates a session; receiving the expected marker proves the small application exchange.

Keep the native USB route and working firmware from the prerequisite lessons. For deliberate firmware recovery, continue with ML-C10.