START WITH THE PEOPLE AND TOOLS

Choose a route you can finish

01



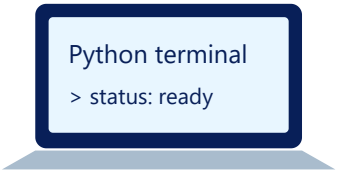
ESP32-S3-N16R8

Runs one chosen receiver application.



USB data cable

Power, direct setup and recovery.



Computer

USB tools or the matching BLE client.



Wi-Fi phone

Local SoftAP page or matching ESPTouch sender.



2.4 GHz lab router

The network the board must eventually join.

Your goal: choose how to deliver Wi-Fi settings, name what must already be installed, and keep a reachable way back if setup fails.

Before you begin

Recognize the board's native USB socket and have a known data cable. Use a temporary, private 2.4 GHz WPA2-Personal lab network for the radio examples.

This is a planning lesson

You can complete the offline decisions without powering hardware. For a live trial, follow the chosen route's full course: ML-C09, C11, C12 or C13.

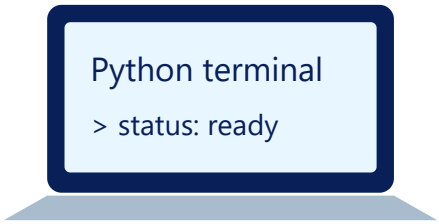
Keep recovery within reach. Front view, antenna up: native USB is bottom left, COM bottom right. Use one data cable, with other circuits and external supplies disconnected.

FOLLOW TWO DIFFERENT PATHS

Delivery is only the first half

02

1



Deliver settings

Sender → chosen transport → receiver application

→

2



Join the network

Board as STA → router → valid local IP address

Provisioning means giving a device the settings it needs. A **sender** supplies them; a **transport** carries them; the installed **receiver application** decides how to validate, store and use them.

The second path is Wi-Fi itself. The board becomes a **station** (STA), joins the target router, and obtains a valid local **IP address**, its address on that network.

1

Receipt

The receiver accepted the message.

2

Saved

The application wrote its settings.

3

Connected

STA joined and has a valid IP.

4

Restarted

A fresh start joins without resending.

> **Point and explain**
> Which arrow carries credentials? Which arrow proves the board joined the router? A form receipt, BLE notification or saved file alone cannot answer the second question.

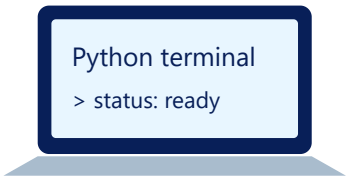
Local STA+IP does not prove internet access or a working cloud service. Test the intended application separately after the local connection works.

ROUTE 1 · ML-C09

At the desk, choose direct USB

03

1

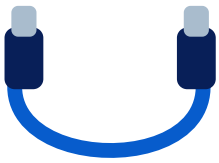


Computer

C09 hidden-input tool

→

2



USB data cable

Direct serial transfer

→

3



Board receiver

C09 MicroPython app

Then: board as STA → target router → valid local IP

STA means station: the board joins a Wi-Fi network.

Choose USB when a computer and data cable can reach the board, especially during development or recovery. You can configure the device while reading its redacted status, without first making a radio setup link work.

1. Confirm the C09 MicroPython receiver files are installed. A USB socket alone does not create a universal configuration command.
2. Close other serial tools, discover the runtime port and use C09's hidden-input computer tool. Keep credentials out of command-line arguments, screenshots and logs.
3. Check the current STA link and IP. Restart with BOOT released, then verify a fresh join without sending settings again.

Good fit

A developer's desk, a service bench, or repeated controlled changes with a known computer.

Fallback stays wired

Rediscover the port after reset. Use the installed application's documented clear/reconfigure helper. Escalate to firmware recovery only if the application/runtime cannot be recovered.

This series uses desktop Python and mpremote. Do not assume that a phone browser supports your USB serial workflow.

ROUTE 2 · ML-C11 SOFTAP

04

A local page can guide first setup

1



Phone browser

Join the setup network

→

2



Board setup AP

Local HTTP POST form

→

3



Board receiver

C11 MicroPython app

Then: board as STA → target router → valid local IP

STA means station: the board joins a Wi-Fi network.

SoftAP means the board temporarily hosts a Wi-Fi access point. The phone joins that setup network and opens a local page stored on the board. No provisioning app or internet page is required.

1. Confirm the C11 receiver is installed. Read its fresh setup network key and local address from private USB.
2. Join the board's setup Wi-Fi. Stay connected despite “No internet,” then open the printed HTTP address manually.
3. POST the target router's details through the form. Read the receipt, then check USB for the separate STA+IP result.

Know the tradeoff. The phone temporarily leaves its usual Wi-Fi. The AP may change channel or disappear as the board joins the router. Read USB before treating a vanished page as failure.

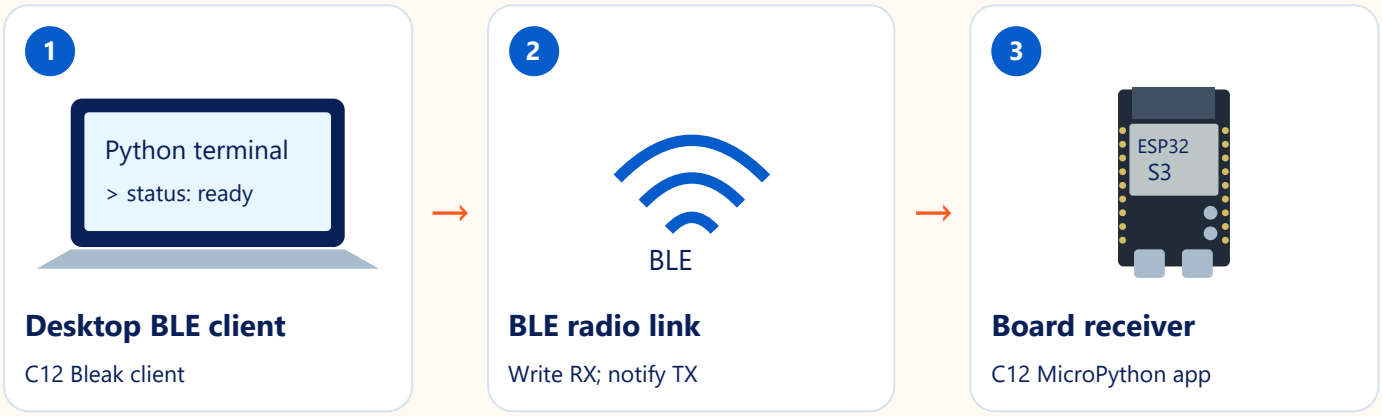
The C11 AP has a fresh WPA2 key, but the local form uses HTTP without TLS. Use temporary lab credentials. POST avoids putting secrets in the URL; it does not provide end-to-end encryption.

> **Fallback**
> Use the reachable USB REPL to read redacted state, reopen setup or clear only the named lab settings. After a failed join, rejoin setup Wi-Fi before retrying the form.

ROUTE 3 · ML-C12

BLE needs a matching service and client

05



Then: board as STA → target router → valid local IP

STA means station: the board joins a Wi-Fi network.

BLE is Bluetooth Low Energy. A **GATT service** is a documented set of data endpoints. C12 provides its own service and a matching desktop Bleak client; it is not a universal Wi-Fi setup feature of every Bluetooth device.

1. Confirm the C12 receiver, supported Bluetooth computer and matching client. Inspect service identifiers and WRITE/NOTIFY properties before sending.
2. Subscribe to TX notifications first. Send commands to RX in the documented chunks; let the client reassemble complete replies.
3. Distinguish accepted, saved and connected. Then restart and check the saved setup without sending credentials again.

Good fit

The sender can keep its normal Wi-Fi while the separate BLE setup link carries settings. The matching client must be available.

Check the security

C12 does not enforce authenticated link encryption. Use isolated, non-sensitive lab credentials. A BLE connection by itself is not a security guarantee.

Fallback

Wrong or missing service? Stop before sending. Disconnect, refresh the OS/app service cache as documented, and rescan. If that fails, use the native USB MicroPython recovery path.

ROUTE 4 · ML-C13 CLASSIC ESPTOUCH

06

SmartConfig has a narrower starting point

1



ESPTouch sender

Phone on target 2.4 GHz

→

2



Target router path

Encoded Wi-Fi packets

→

3



Board receiver

C13 ESP-IDF app

Then: board as STA → target router → valid local IP

STA means station: the board joins a Wi-Fi network.

The matching phone sender uses specially encoded Wi-Fi packets. The listening receiver decodes the settings, then joins the router. For C13, the phone must be on the target **2.4 GHz** network and the classic ESPTouch sender must match the receiver.

1. Confirm the C13 ESP-IDF receiver is installed and in its setup window. It replaces MicroPython; stock MicroPython has no C13 SmartConfig command.
2. Use the matching classic sender and isolated lab credentials. Check phone permissions and required packet delivery on the target network.
3. Observe credentials received, valid STA+IP and sender acknowledgement separately. C13 bounds setup to 90 seconds; do not retry forever.

Choose another plan when a prerequisite is absent. A phone limited to 5 GHz, blocked packet delivery or a mismatched ESPTouchV2 sender is a reason to stop and fix the setup path. It is not proof that the password is wrong.

Fallback

Keep the running C13 native USB console accessible. Its `C` command deliberately clears only its named settings and restarts. That console is not a MicroPython REPL or a USB credential uploader.

Classic ESPTouch is not inherently authenticated provisioning. Do not use sensitive production credentials in this training workflow.

A SHARED CHIP DOES NOT MEAN A SHARED PROTOCOL

07

Match the application before the transport

ML-C09 / C11 / C12

Course-defined MicroPython apps

```
/lab_wifi.json
schema: 1 · ssid · password
```

These three examples share a format by design.

ML-C13

Course-defined ESP-IDF app

```
NVS: ml_c13 / wifi
Application-owned binary settings
```

A different format. No Python REPL in this firmware.

The storage format belongs to the application. USB, SoftAP and BLE can carry many formats; their names do not specify a filename, a password policy or a clear command.

The C09, C11 and C12 lab file contains exactly `schema`, `ssid` and `password`. Schema is `1`; SSID is 1–32 UTF-8 bytes; the lab WPA2 passphrase is 8–63 printable ASCII characters. These labs exclude open networks and enterprise login.

Choose	Have this sender	Confirm this first
USB	Computer + data cable	C09 app, correct runtime port
SoftAP	Wi-Fi browser	C11 app, setup key + local page
BLE	C12 desktop client	C12 service, subscription + version
SmartConfig	Matching classic sender	C13 ESP-IDF app, 2.4 GHz path

Do not switch firmware casually. Back up your project and private settings before replacing a receiver. Restoring MicroPython requires its documented merged image and recovery workflow; an ESP-IDF project uses its own generated flash addresses.

FOUR WORKED DECISIONS

Choose from the constraints you can see

08

Desk development → USB

Computer and cable are already present. C09 is installed. Choose direct USB; verify STA+IP, then restart. Keep the USB application helper as recovery.

First-run phone → SoftAP

The user has Wi-Fi and a browser, but no setup app. C11 is installed. Choose its protected local page; keep technician USB access available.

Keep sender Wi-Fi → BLE

C12 is installed and its desktop client is available. Choose BLE; subscribe before sending. Refresh stale service cache or use USB if discovery fails.

Classic sender ready → ESPTouch

C13 is listening, the matching phone is on target 2.4 GHz and packet delivery is allowed. Choose classic SmartConfig; keep the C13 console reachable.

Use these gates in order

1. **Receiver:** identify the installed application and supported version.
2. **Sender:** confirm the actual computer/app/browser and permissions.
3. **Network:** check the lab's 2.4 GHz path and required transport conditions.
4. **Recovery:** name the reachable interface, specific action and expected state.

> **Your turn**
> The phone only has 5 GHz access, and the board runs C13. Should you try a C11 page? No: that page is not installed. Pause, arrange a compatible sender/network, and use C13's console for redacted status or deliberate named-config reset.

PLAN RECOVERY BEFORE SETUP

09

A fallback must survive the failure

1. **Stop at the failed layer**

   Receipt only? Check the router join and IP result.

2. **Retry one known setup path**

   Check 2.4 GHz, spelling, range and the documented sender.

3. **Use native USB**

   Close other tools; rediscover the runtime port; read redacted state.

4. **Recover the installed application**

   MicroPython: course helper. ESP-IDF C13: console C only when running.

Use a bounded retry with one changed condition. Record only the failed layer and redacted state. Never send real credentials to a support chat or erase all flash just because a setup page disappeared.

Keep the runtime-specific USB recovery card available offline.

MicroPython C09 / C11 / C12

Connect native USB, close other tools, find the runtime port and stop the application at REPL. Use that course's named clear/reconfigure helper. It does not erase all application data.

ESP-IDF C13

Use its running native USB console. `C` clears NVS namespace `ml_c13`, key `wifi`, then restarts. If console/app cannot start, get help with the documented firmware recovery; the command cannot fix every storage fault.

No physical access? A sealed USB socket plus unavailable radio setup is not a working fallback. Arrange service access. For future units, preserve a usable USB/service path and a known receiver image.

After reset or firmware changes, the USB port name may change. Keep BOOT released for normal restart. Use ML-C10 only when firmware recovery is actually needed, with a backup and its explicit erase/flash steps.

PRACTICE, FEEDBACK AND SUCCESS

Write one complete setup plan

10

Situation	C11 is installed; first-time user has a Wi-Fi phone, no app; technician can reach USB.
Primary route	SoftAP: join setup Wi-Fi, open local address, submit a POST form.
Proof	Read receipt, then current STA+IP; restart without submitting again.
Bounded retry	After a join failure, check lab SSID/key/2.4 GHz and rejoin setup before one corrected attempt.
Fallback	Technician uses USB REPL and C11 helper to reopen setup or clear only named lab settings.

Now work through the six cases in the offline decision practice. Save decision_lab.py to a computer and use Python 3.13+. No hardware or credentials are needed.

```
python decision_lab.py
python decision_lab.py --case desk --route usb --fallback usb-app
```

The first command lists every scenario. Each answer needs both a route and a reachable fallback. `pause` is the right route when a required path is missing. PASS confirms the pair; TRY AGAIN explains the failed choice.

Two quick checks

1. BLE says “saved,” but no IP. Is provisioning complete?

2. C13 runs ESP-IDF. Can C09's Python helper clear it?

Answers

1. No. Storage worked; check station join, router and IP.

2. No. Use the installed C13 console workflow, not a different runtime's helper.

You are ready when

You can justify a route, name its sender and installed receiver, distinguish receipt/storage from STA+IP, prove restart, and describe a fallback that remains reachable after the likely failure.