5Tech Edge Bridge (ESP32) — Phase-1 MVP — User & Operation Manual

Setup, flashing, Wi-Fi / MQTT configuration, Bluetooth LE commands, and the demo flow for the ESP32 Edge Bridge.

`esp32-edge-bridge` · Revision Draft (Preliminary) · 2026-07-08 · Status: **PRELIMINARY**

> **PRELIMINARY.** This manual describes a Phase-1 MVP on an off-the-shelf ESP32 development board. It is a **monitoring and automation prototype**, not a finished or certified product. Procedures and values marked `[TBD]` are pending confirmation. The authoritative engineering documents are in `modules/5tech-edge-iot-bridge/`; always follow the docs shipped with the firmware you flashed.

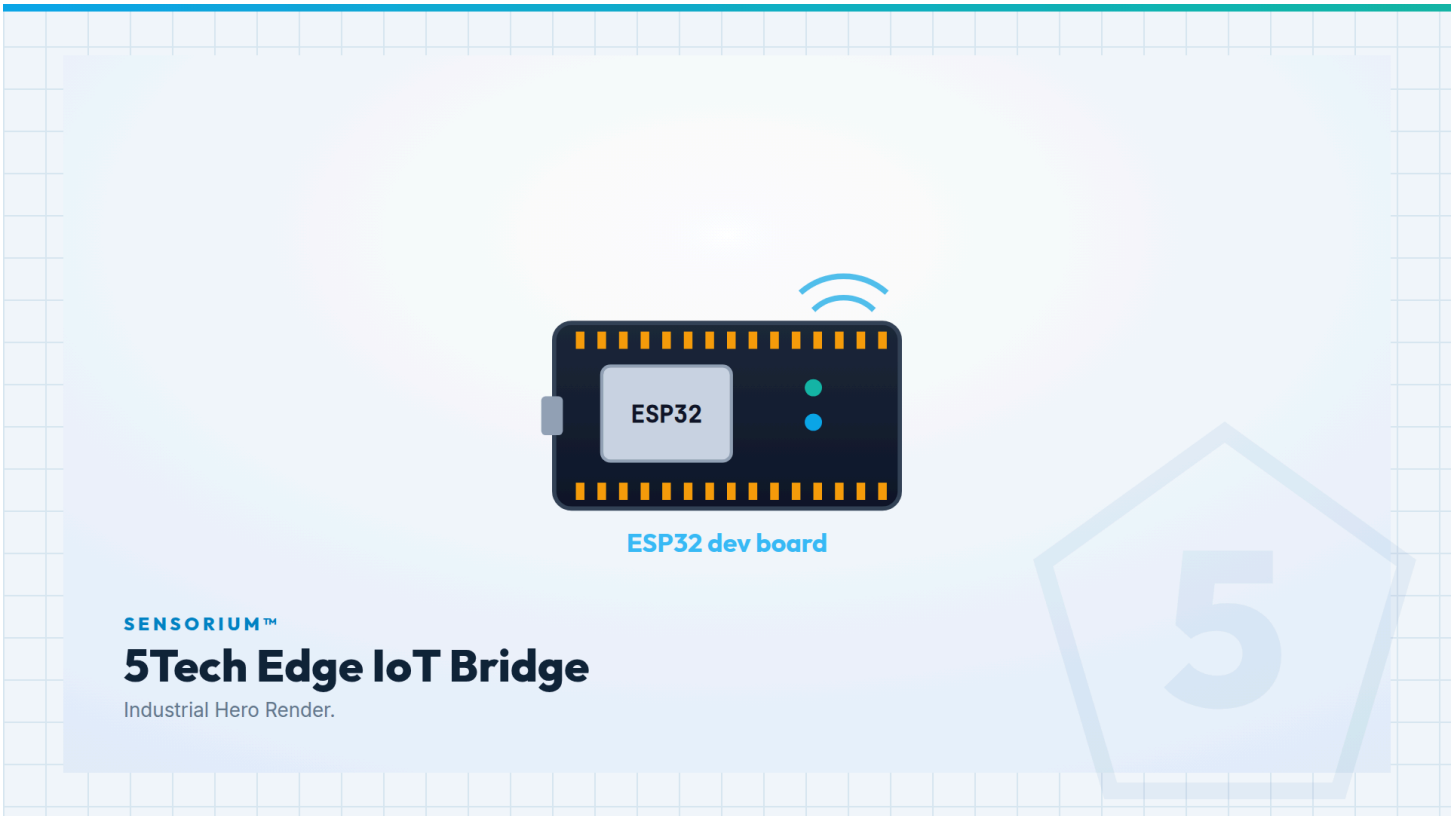

Figure FIG-001 — 5Tech Edge Bridge, the Phase-1 MVP that bridges an input to an automation flow over Wi-Fi, Bluetooth LE, and MQTT.

1. Safety

Read this chapter before you set up, operate, or modify the Edge Bridge.

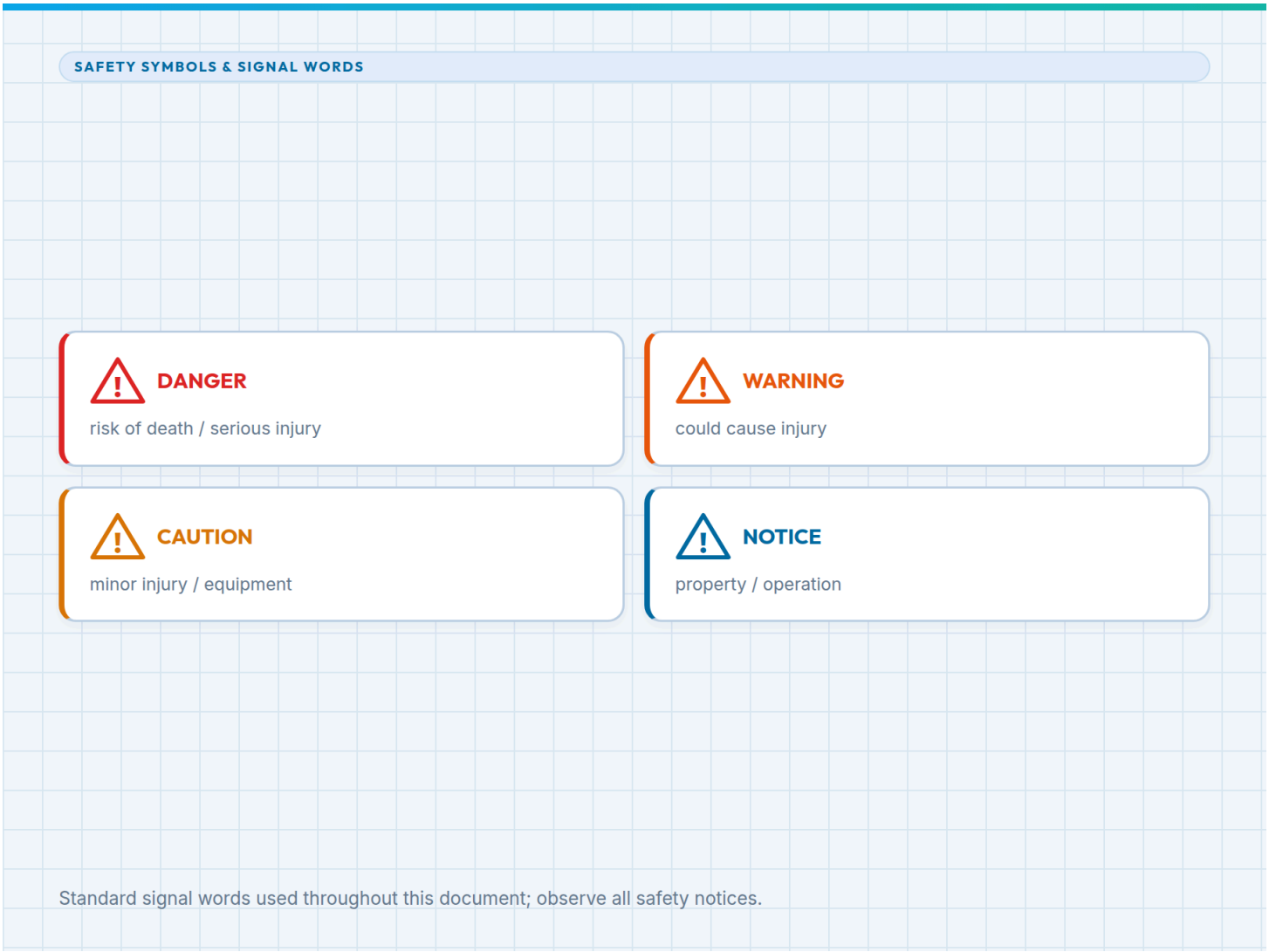

Figure FIG-002 — Signal words (DANGER · WARNING · CAUTION · NOTICE) and the pictograms used in this manual.

1.1 Signal words

SIGNAL WORD	MEANING
DANGER	Hazard that **will** cause death or serious injury if not avoided.
WARNING	Hazard that **could** cause death or serious injury if not avoided.
CAUTION	Hazard that could cause minor or moderate injury if not avoided.
NOTICE	Practice **not related to personal injury** — property or equipment damage, or an important advisory (scope / limitation) the reader must heed.

1.2 This is not a safety device

> **NOTICE — Not a safety device.** The 5Tech Edge Bridge is a **monitoring and automation prototype** on a development board. It carries **no safety certification of any kind** and must **not** be used to protect people or equipment, to stop machinery, to create a protective zone, or as any part of a functional-safety function. Its hazard states (`WARNING` / `DANGER`) are demonstration signals in a data pipeline, not safety outputs. Never rely on it where a failure could cause injury or loss. *(This is a scope advisory about what the device is not — a `NOTICE` per §1.1, not a hazard warning about the device itself.)*

1.3 Intended use and audience

The Edge Bridge is intended for use **by developers, technicians, and integrators on a trusted test bench** to validate the sensor-to-automation pipeline and to demonstrate it to clients and stakeholders. Any other use — in particular field deployment, safety use, or connection to an untrusted network — is considered misuse for this Phase-1 MVP.

1.4 Qualified personnel and required competency

Set-up, operation, wiring, and modification of the Edge Bridge are for **qualified personnel** only. For this Phase-1 MVP that means a person who:

- is a **developer, technician, or integrator** competent in low-voltage electronics and ESD-safe handling of a bare PCB;
- can flash firmware and read a serial console, and understands 3.3 V logic and that the GPIO pins are **not** 5 V tolerant (§1.5);
- understands MQTT / Bluetooth LE on a trusted LAN and the plaintext, unauthenticated nature of this prototype (§1.7);
- and, before wiring anything to the `Con1` output or the `Det1` / `Det2` inputs, can confirm the external circuit's voltage, current, and isolation against the datasheet.

A person without this competency must not set up, wire, or modify the device unsupervised.

1.5 Electrical safety

The board is powered from a **USB 5 V** source and operates at low voltage (3.3 V logic). Electrical-shock risk from the board itself is minimal, but:

> **NOTICE — Equipment damage.** GPIO pins are **3.3 V logic and are not 5 V tolerant**. Applying more than 3.3 V to a GPIO input, shorting pins, or reversing a connection can

destroy the board. Power the board from one source at a time (USB *or* an external 3.3 V/5 V rail, not both). Observe ESD precautions when handling the bare board.

- Use a data-capable USB cable and a known-good USB port or supply.
- If you connect an external sensor or a load to the `Con1` output, confirm its voltage and current are within the pin's rating (`[TBD]`) before wiring. Do not switch mains or an inductive load directly from a GPIO pin.

1.6 Radio-frequency (RF) note

The board transmits on **Wi-Fi (2.4 GHz)** and **Bluetooth LE** through the module's onboard antenna. Transmit power is low and typical of a consumer ESP32 module. Regional radio approvals for the *assembled prototype* are `[TBD]` — this is a development device, so operate it only where such use is permitted and keep it on a trusted bench network.

1.7 Residual risks

Even when used as intended, the following remain: the device may **latch a hazard state** and keep reporting `DANGER` after a sensor recovers (this is deliberate — see §3.3); on a plaintext network anyone who can reach the broker can command the device; and Bluetooth LE and Wi-Fi share one antenna, so BLE responsiveness varies with Wi-Fi load. None of these is a safety mitigation.

---

2. About this manual

Scope. This manual covers setting up, flashing, configuring, operating, and troubleshooting the Edge Bridge Phase-1 MVP on an ESP32 development board. It does **not** replace the detailed engineering documentation in the firmware repository.

Audience. Developers, technicians, and integrators as described in §1.3.

Conventions.

- Safety notices use the signal words in §1.1 and appear **before** the step they apply to.
- Procedures are numbered, one action per step.
- `[TBD]` marks a value pending confirmation. Never substitute a guessed value for a `[TBD]`.
- `Monospace` denotes commands, file names, pin names, topics, and UI/console text.

Related documents.

DOCUMENT	LOCATION	USE
Datasheet	`datasheets/esp32-edge-bridge-datasheet/`	Dev-board specs, interfaces, pinout, ratings
Product overview	`products/esp32-edge-bridge/`	Positioning and highlights
Firmware — setup	`modules/5tech-edge-iot-bridge/docs/setup.md`	Toolchains, secrets, build, flash
Firmware — MQTT API	`modules/5tech-edge-iot-bridge/docs/mqtt-api.md`	Topics and payload schemas
Firmware — Bluetooth API	`modules/5tech-edge-iot-bridge/docs/bluetooth-api.md`	GATT table, BLE commands
Firmware — demo script	`modules/5tech-edge-iot-bridge/docs/demo-script.md`	The client / investor walkthrough

3. Product overview & theory of operation

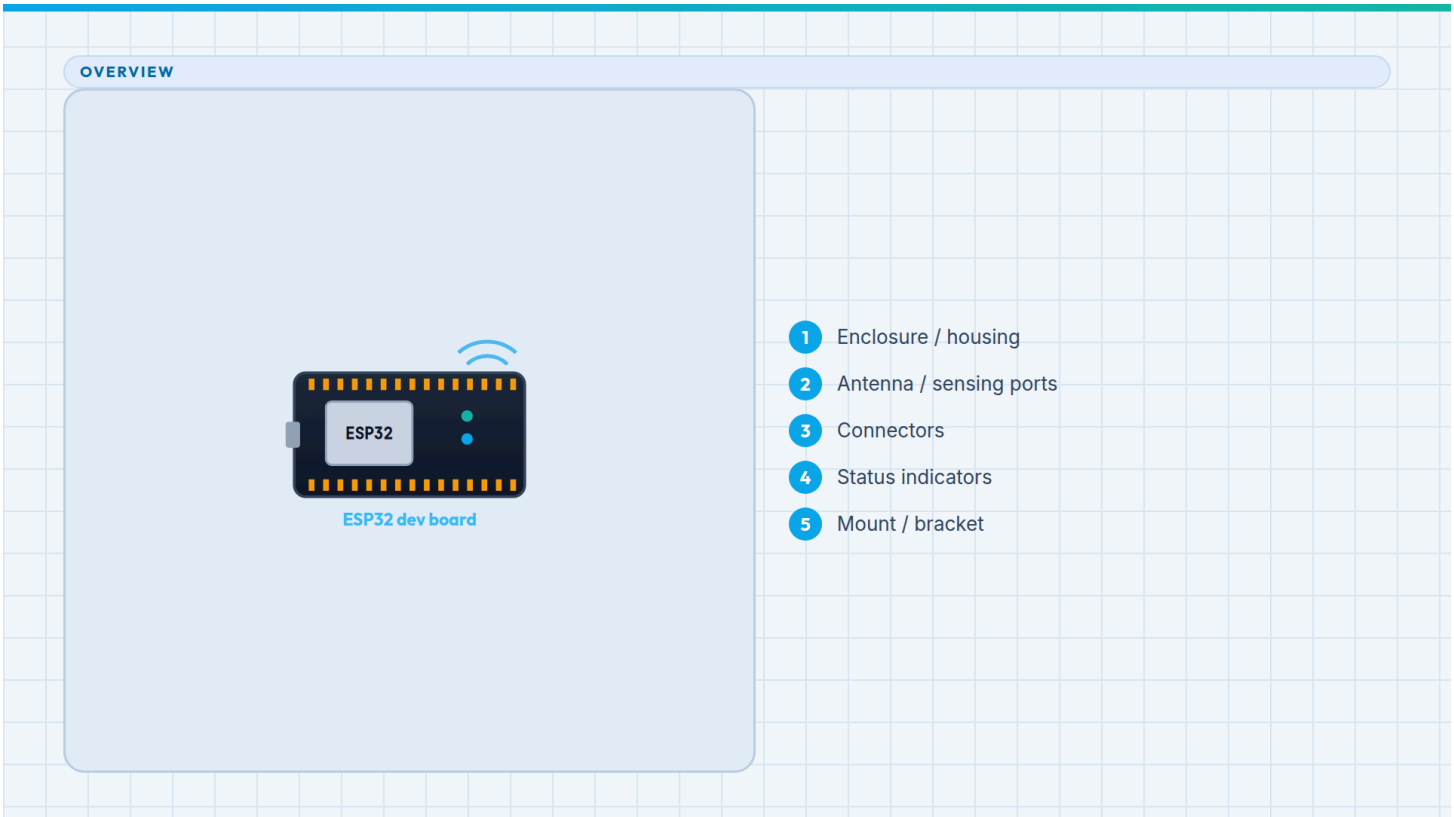

Figure FIG-003 — The Edge Bridge and its connections: USB, optional inputs, and the two 2.4 GHz

radios.

3.1 What it does

The Edge Bridge reads an input, decides whether that input represents a `WARNING` or `DANGER` condition, and republishes the decision as structured JSON over Bluetooth LE and MQTT so that an automation flow can act on it. It runs on a stock ESP32 dev board and needs no sensor hardware to demonstrate.

3.2 Where it sits — Sense → Connect → Decide

- **Sense** — a GPIO sensor (Det1 / Det2), the onboard button, a simulated sensor, or a remote command indicates a condition.
- **Connect** — the ESP32 resolves its state and publishes over Wi-Fi to an MQTT broker; a Bluetooth LE channel gives local control even with no network.
- **Decide** — an automation flow (n8n / Node-RED / custom) subscribes and drives dashboards, alerts, and logs.

3.3 Theory of operation

1. **Device identity.** On boot the device derives its identity from the module's MAC: the MQTT id `5tech-bridge-a1b2c3` and the BLE name `5Tech-IoT-Bridge-a1b2c3` (last six MAC hex digits).
2. **State machine.** Nine states — `BOOTING`, `WIFI_CONNECTING`, `MQTT_CONNECTING`, `READY`, `WARNING`, `DANGER`, `OFFLINE`, `CONFIG_MODE`, `ERROR`. Once operational, the state is resolved in strict priority: `error > danger > warning > config_mode > !wifi > READY`. Each state has its own onboard-LED blink pattern.
3. **Latching hazards.** Hazard levels are ordered `NORMAL < WARNING < DANGER`. Raising to a higher level wins and publishes an event; raising to an equal or lower level is a no-op. A hazard **latches** — a recovering sensor does **not** auto-clear it. Only an explicit `clear` (or a configured timeout, off by default) returns to `NORMAL`.
4. **Publishing.** Telemetry publishes every 5 s, an event publishes immediately on any change, and status is retained on the broker with a Last Will so a late subscriber always sees the true state. A hazard outranks connectivity: a device in `DANGER` keeps reporting `DANGER` even if Wi-Fi drops.
5. **Commands.** The same command grammar works on serial, Bluetooth LE, and MQTT. Commands are queued and executed on the main loop, never on a radio callback task.

---

4. Controls & indicators

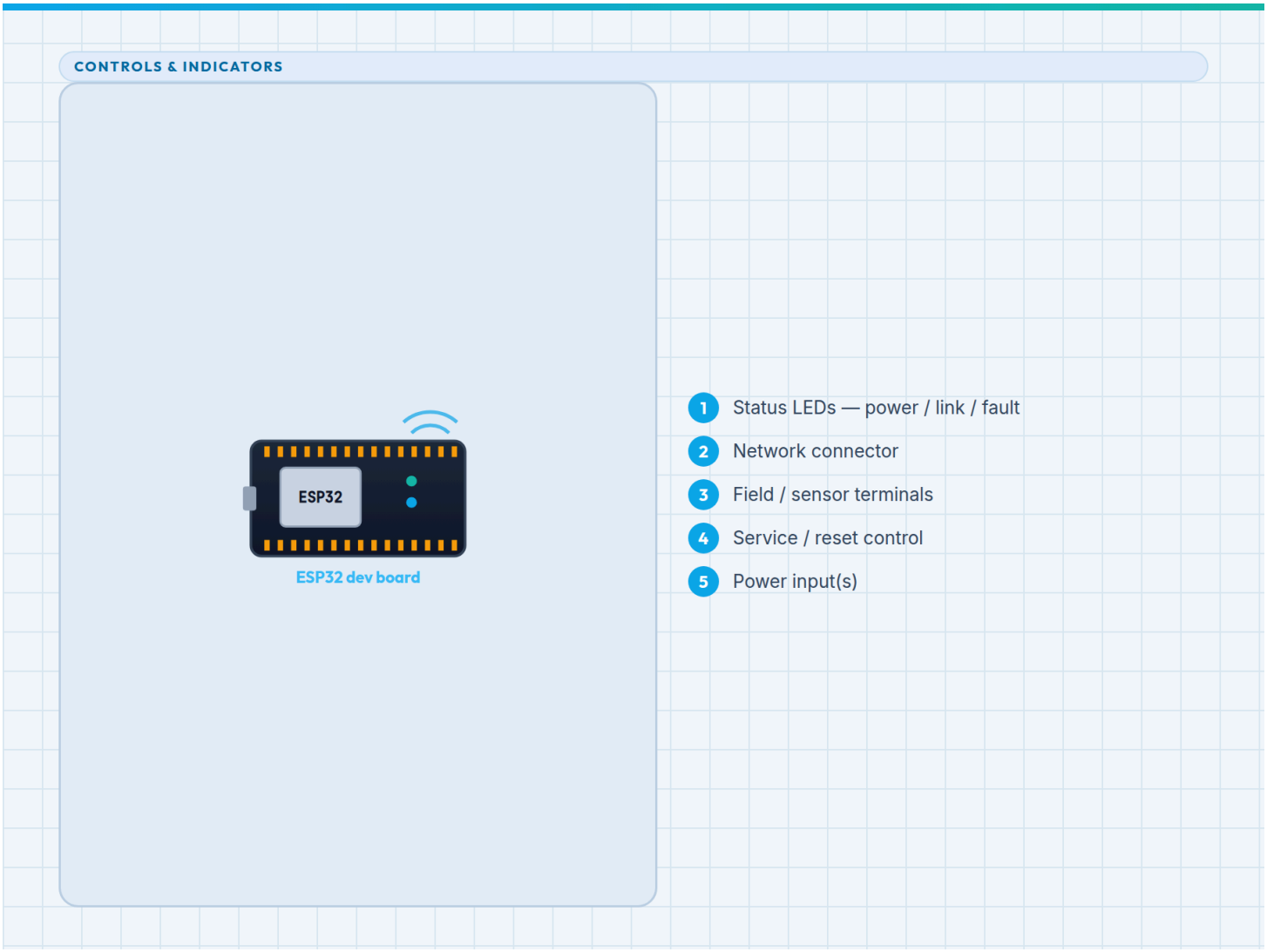

Figure FIG-004 — Pins, the onboard button, the status LED, and the USB connector, with a numbered legend.

Exact pin numbers are the Phase-1 defaults; confirm against the datasheet and your board's silkscreen.

CONTROL / INDICATOR	FUNCTION	STATES
USB connector	Power, flashing, serial console (115 200 baud)	Connected / disconnected
Status LED (`GPIO2`)	Device state	A distinct blink pattern per state (READY, WARNING, DANGER, OFFLINE, CONFIG_MODE, …)
Button (`GPIO0`, onboard BOOT)	Manual trigger	**Short press** = raise warning / clear an active hazard · **Long press (≥1.5 s)** = raise danger
Det1 input (`GPIO18`)	Digital input, active-low	Asserted → WARNING; value mirrored over BLE
Det2 input (`GPIO19`)	Digital input, active-low	Asserted → DANGER; value mirrored over BLE
Con1 output (`GPIO23`)	Remotely settable contact	Set over BLE / MQTT / serial (`set_con1 0|1`)
Buzzer (`GPIO25`, optional)	Local annunciation	Disabled by default (`ENABLE_BUZZER = 0`)

> **NOTICE.** On most ESP32 devkits `GPIO2` is the onboard LED. Phase-1 uses it for device state, so `Con1` was moved to `GPIO23`. To restore the original wiring, set `CON1_GPIO = 2` and `ENABLE_STATUS_LED = 0` (a compile-time guard enforces that they never share a pin).

---

5. Operation

5.1 Before you start

You need: an ESP32-WROOM-32 dev board, a data-capable USB cable, a host computer with the toolchain, and (for the Wi-Fi/MQTT path) an MQTT broker such as `mosquitto` on the same trusted LAN. No sensor hardware is required.

Two firmware variants are available and are interchangeable on the wire:

	ESP-IDF VARIANT	PLATFORMIO / ARDUINO VARIANT
Toolchain	ESP-IDF (builds on v5.4.4)	PlatformIO (builds on 6.1.19)
Build	`idf.py build`	`pio run`

Full toolchain setup is in `modules/5tech-edge-iot-bridge/docs/setup.md`.

5.2 Configure secrets and flash

> **NOTICE.** `secrets.h` is gitignored and must never be committed. Only `secrets.example.h` is tracked.

ESP-IDF variant:

1. Copy the template: `cp firmware/esp-idf/main/config/secrets.example.h firmware/esp-idf/main/config/secrets.h`
2. Edit `secrets.h` — set `WIFI_SSID`, `WIFI_PASSWORD`, and `MQTT_HOST`.
3. Build and flash: `cd firmware/esp-idf && idf.py set-target esp32 && idf.py build`
4. `idf.py -p /dev/ttyUSB0 flash monitor`

Arduino / PlatformIO variant:

1. `cp firmware/platformio/src/config/secrets.example.h firmware/platformio/src/config/secrets.h`
2. Edit `secrets.h` as above.
3. `cd firmware/platformio && pio run && pio run -t upload`
4. `pio device monitor`

5.3 First power-on and status check

1. Connect the board over USB and open the serial monitor at **115 200 baud**.
2. Confirm the device prints its identity (`5tech-bridge-…`) and begins its boot sequence.
3. Watch it associate with Wi-Fi, connect to the broker, and start Bluetooth LE advertising.
4. Confirm the status LED settles into the `READY` pattern.
5. If Wi-Fi does not connect within ~15 s the device continues **offline**: BLE and serial keep working and the state reads `OFFLINE`. This is expected, not a fault.

5.4 Bench mode (no Wi-Fi)

Leaving `WIFI_SSID` at its placeholder is a **supported bench mode**: the device skips Wi-Fi, boots into `CONFIG_MODE`, and stays fully usable over Bluetooth LE and the serial console. Every command works; only MQTT publishing is unavailable.

5.5 The demo flow

1. On a laptop on the same LAN, subscribe to the broker: `mosquitto_sub -h <broker> -t "5tech/iotbridge/+/#" -v`
2. Trigger a warning — from the serial console, from a phone over Bluetooth LE (write `test_warning` to the **Cmd** characteristic), or by publishing to the command topic. All three are equivalent.
3. Watch `[STATE] READY -> WARNING`, the LED cadence change, and the event land on the broker.
4. Trigger `test_danger`. The LED goes faster and the danger **latches**.
5. Send `clear`. The device returns to `READY`.
6. Explain the point: only the trigger is simulated — swap it for a real sensor and nothing downstream changes.

5.6 Orderly shutdown

Simply unplug USB. On an ungraceful disconnect the broker publishes the retained **Last Will**, so the device's status flips to `OFFLINE` for any subscriber. On the next boot the device overwrites it with a live status.

---

6. Configuration

6.1 Configuration methods

METHOD	USE	NOTES
`secrets.h` (build-time)	Wi-Fi and broker credentials	Gitignored; requires a re-flash
Feature flags (build flags)	Enable/disable subsystems	`#ifndef`-guarded; override with `-D`
Runtime commands (serial / BLE / MQTT)	Trigger events, set thresholds, drive `Con1`, reboot	Identical grammar on all three transports

6.2 Runtime commands

Accepted as bare text **or** JSON, on any transport.

COMMAND	EFFECT
`help`	List commands
`status`	Current status, as JSON
`config`	Running configuration, as JSON
`test_warning`	Raise a `WARNING` event
`test_danger`	Raise a `DANGER` event
`clear`	Clear the active event
`set_threshold <n>`	Set the danger threshold (JSON form can move both bounds)
`set_con1 <0|1>`	Drive the Con1 contact
`reboot`	Restart (the reply is flushed first)

Examples:

`test_danger`
`{"command":"set_threshold","danger_threshold":40,"warning_threshold":90}`

6.3 Thresholds

The sensor value is modelled as a **distance in cm** — smaller means closer means worse. A reading below `warning_threshold` (default 100) raises WARNING; below `danger_threshold` (default 50) raises DANGER. `set_threshold` rejects any pair where warning is not strictly above danger or that leaves the valid range `[0, 200]`. **Thresholds are not persisted** — a reboot restores the compiled defaults.

6.4 Bluetooth LE commands

1. Scan for `5Tech-IoT-Bridge-…`, or filter on the service UUID `4fafc201-1fb5-459e-8fcc-c5c9c331914b` (more reliable — the name is in the scan response).
2. Connect and expand the custom service.
3. **Enable notifications on the `Resp` characteristic before writing to `Cmd`.** Replies to an unsubscribed `Resp` are dropped — this is the most common cause of "no response".
4. Write a command as UTF-8 text (e.g. `test_danger`) to `Cmd`. Replies arrive as notifications on `Resp`, shaped `OK <cmd>: <message>` or `ERR <cmd>: <message>`.
5. Write `0x01` (a raw byte, not the text "1") to `Con1` to drive `GPIO23` high; `0x00` drives it low.

6.5 Firmware update

Flash over USB (§5.2). There is **no OTA and no BLE update** in Phase 1.

---

7. Maintenance

The Edge Bridge is firmware on a development board; there are no scheduled physical maintenance tasks. "Maintenance" here means keeping the firmware and its configuration sound.

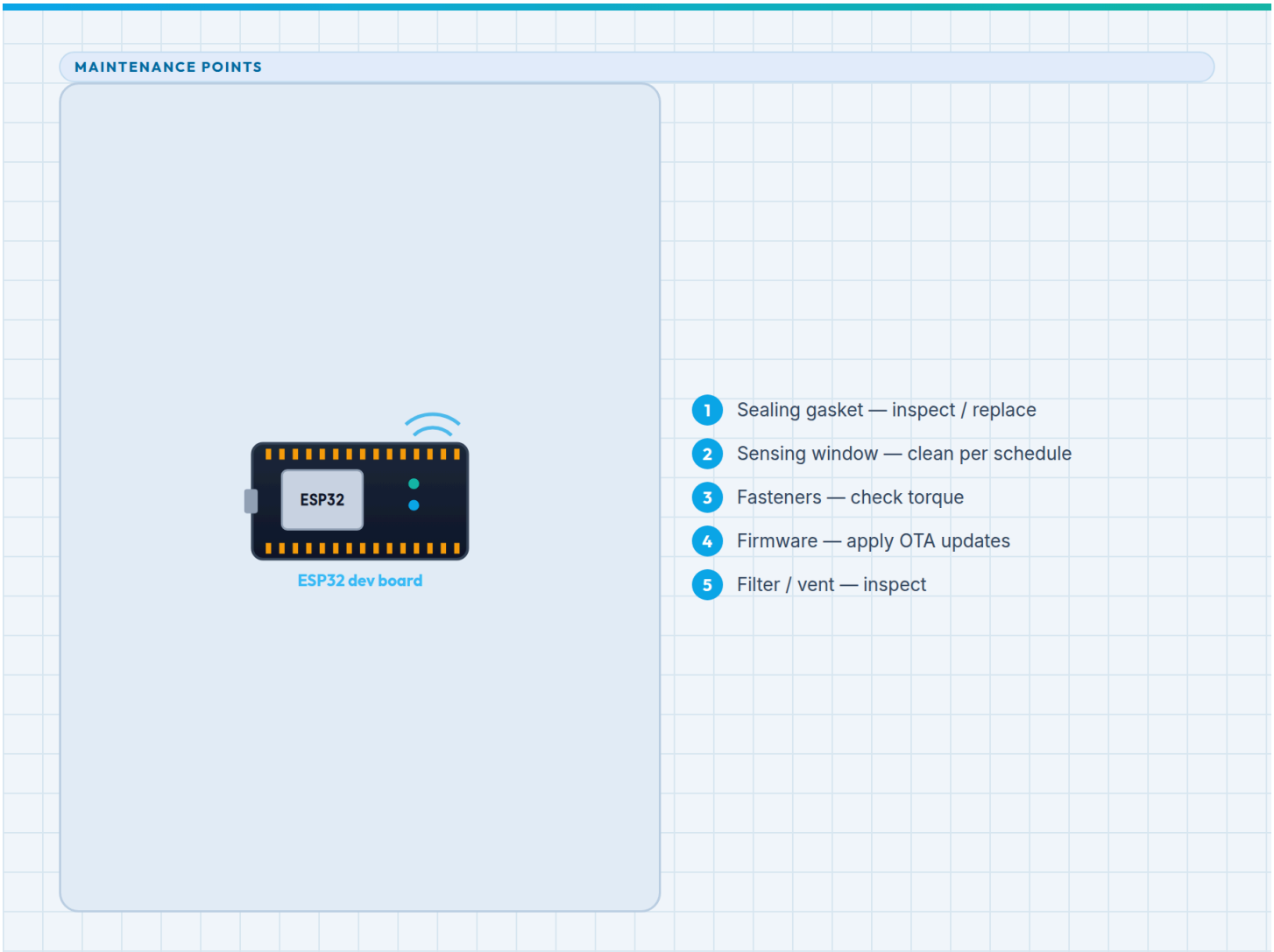

Figure FIG-005 — Points of routine attention: the USB connection, the secrets file, and the automated test suites.

INTERVAL	TASK	NOTES
As needed	Re-flash after a firmware change	§5.2
As needed	Keep `secrets.h` current and uncommitted	Verify with `git check-ignore`
Before tagging a build	Run the host test suite (`make -C tests/host`)	77 assertions, no hardware
Before tagging a build	Run the wire-parity check	Confirms both variants still agree
Routine	Inspect the USB cable and connector	A charge-only cable is a common failure
Routine	Keep the board dry, static-safe, and clear of shorts	Bare PCB, no enclosure

8. Troubleshooting

SYMPTOM	POSSIBLE CAUSE	ACTION
No serial output	Wrong baud or charge-only cable	Use 115 200 baud and a data cable; confirm the serial port
No BLE response to a command	`Resp` notifications not enabled	Enable notifications on `Resp` **before** writing `Cmd` (§6.4)
Device not on the broker	Wrong `MQTT_HOST`, broker down, or wrong port	Check `secrets.h`, confirm the broker on port `1883`, check the LAN
Stuck in `CONFIG_MODE`	Placeholder or missing Wi-Fi credentials	Set real `WIFI_SSID` / `WIFI_PASSWORD` in `secrets.h` and re-flash
State reads `OFFLINE`	Wi-Fi not connected	Expected without Wi-Fi; BLE and serial still work; check credentials/coverage
Broker shows the device `OFFLINE`	Last Will fired on an ungraceful disconnect	Normal after power loss; it self-repairs on reconnect
Hazard will not clear	Latching hazard model	Send `clear` explicitly — a recovering sensor does not auto-clear (§3.3)
`Con1` seems to toggle the LED	`Con1` on `GPIO2` (original wiring)	Phase-1 uses `GPIO23`; see the §4 NOTICE
BLE stutters under load	Wi-Fi and BLE share one antenna	Expected on ESP32; reduce Wi-Fi load during BLE work
ESP-IDF build fails on partition size	App overflows the default 1 MB partition	Use `CONFIG_PARTITION_TABLE_SINGLE_APP_LARGE` (Arduino: `huge_app.csv`)
A command is silently dropped	Command queue (depth 8) full	Reduce command rate; the drop is logged on serial

If a fault persists, capture the serial log, the `status` / `config` output, and the broker messages, then consult the engineering docs in `modules/5tech-edge-iot-bridge/`.

9. Specifications

This manual does **not** duplicate specification values. All dev-board, interface, electrical, and environmental specifications are held in the datasheet:

- **Datasheet:** `datasheets/esp32-edge-bridge-datasheet/document.md`

AREA	WHERE TO FIND IT
Hardware platform (ESP32 dev board)	Datasheet §1, §4
Mechanical (form factor)	Datasheet §5
Electrical (USB power, logic level)	Datasheet §6
Interfaces & protocols (Wi-Fi, BLE, MQTT, GPIO, serial)	Datasheet §7
Environmental & ratings	Datasheet §8
Compliance (n/a — development prototype)	Datasheet §9

---

10. Warranty & support

Warranty. This is a **Phase-1 MVP / development prototype**, provided as-is for evaluation and demonstration. It carries no commercial product warranty and no certification. Production warranty terms are `[TBD]` and belong to Phase 2.

Support.

- 5Tech — Vancouver, BC, Canada
- Email: `info@5tech.ca`
- Web: `https://5tech.ca`

When contacting support, have ready: the firmware variant and version, the device id (`5tech-bridge-…`), the serial log, and the broker messages around the issue.

> **NOTICE — Not a safety device.** Do not deploy this prototype in any role where a failure could harm people or equipment. It is a monitoring and automation MVP, nothing more.
> *(A scope advisory — a `NOTICE` per §1.1, not a hazard warning about the device itself.)*

Figures

ID	TITLE	STATUS	FILENAME	PURPOSE
FIG-001	Edge Bridge — Prototype Hero Render	Missing	`renders/hero_edge-bridge.png`	Manual cover / hero
FIG-002	Safety Symbols & Signal Words	Missing	`figures/safety_symbols.png`	Signal words and pictograms for the safety chapter
FIG-003	Product Overview	Missing	`figures/overview.png`	Orient the user to the board and its connections
FIG-004	Controls & Indicators	Missing	`figures/controls_indicators.png`	Identify pins, the button, the LED, and the USB connector
FIG-005	Maintenance Points	Missing	`figures/maintenance_points.png`	Locate points of routine attention

---

This manual is preliminary and subject to change. See `revision.md` for change control, `references.md` for sources, and `image_prompts.md` for the figure/prompt index.