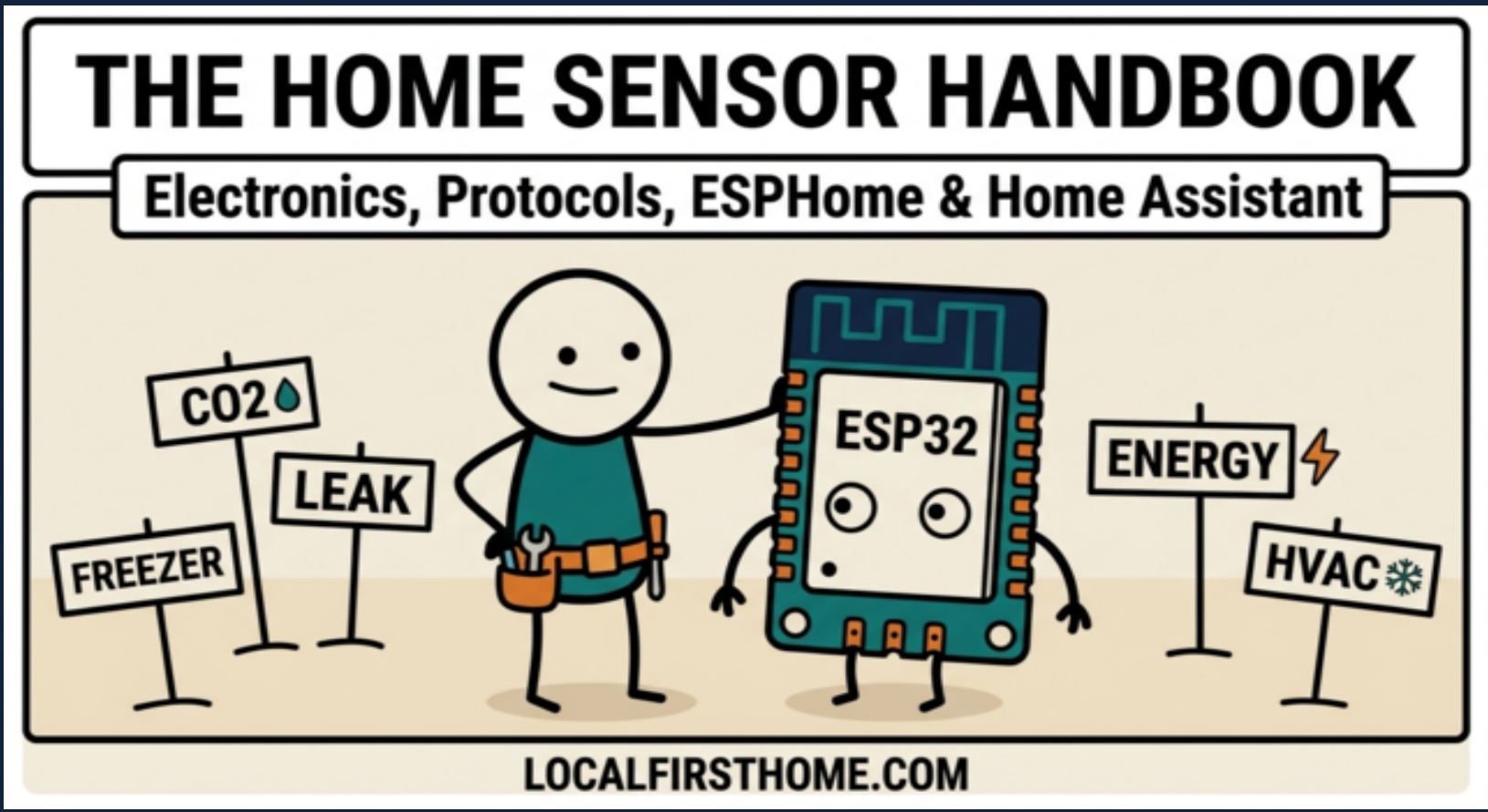
THE HOME SENSOR HANDBOOK

Electronics, Protocols, ESPHome & Home Assistant —
The Core Skills Behind Every Local-First Smart Home Build



A field manual for builders who want no cloud, no subscriptions,
and sensors that keep working when the internet doesn't.

First Edition • 2026 • Cheatsheets, Quick Reference & Use Cases

Table of Contents

1. How to Use This Handbook

This handbook is the common foundation behind every LocalFirstHome.com build: home energy monitoring, leak detection, CO2 and air quality stations, rack/NAS telemetry, freezer monitoring, water infrastructure, perimeter sensing, garage door control, precision irrigation, and HVAC monitoring. Instead of repeating the same basics in every article, this is the reference you keep open on a second screen while building.

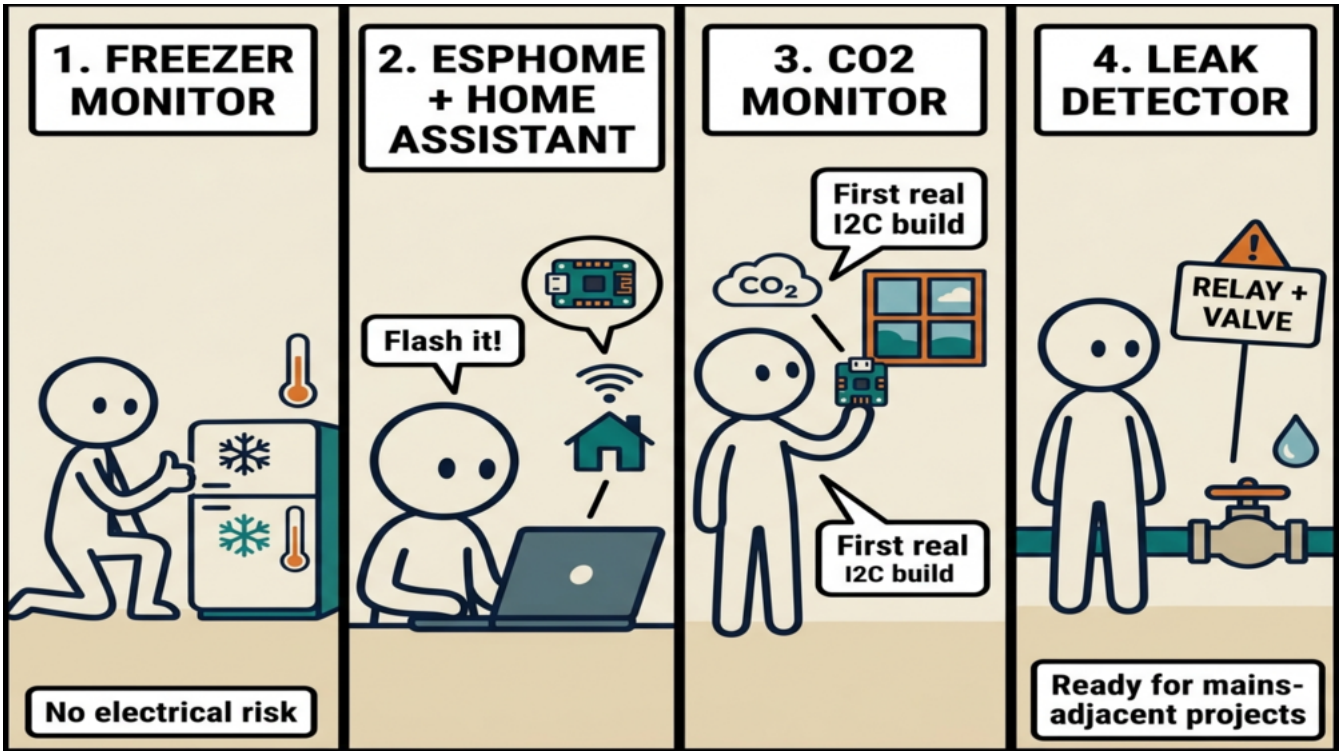
It is organized so you can either read it once as a 2–3 week crash course, or skip straight to a chapter mid-project when you forget a pinout or a YAML syntax. Chapters 3–7 are the theory and reference core. Chapter 8 is a sensor cookbook. Chapter 9 maps every sensor and protocol back to each of the 10 project ideas, with budget and difficulty at a glance.

What this handbook assumes

- You are comfortable installing software and flashing firmware to a microcontroller.
- You already run (or are willing to run) Home Assistant on local hardware.
- You are not yet comfortable with I2C, UART, 1-Wire, or reading a sensor datasheet — that's exactly the gap this fills.
- You will treat anything touching mains voltage (110/220V) as a job for a licensed electrician, not a weekend experiment.

Recommended learning order

Start with the Freezer Monitor project (1-Wire + GPIO only, zero electrical risk), then the CO2 Monitor (introduces I2C properly), then the Leak Detector (introduces relays and valve actuation), and only after that move to anything involving live mains current, like the whole-home energy monitor.



2. Core Toolkit — What to Buy Once

These tools and components are reused across nearly all 10 projects. Buying them once removes the biggest friction point in getting started.

Physical tools

Tool	Notes	Est. Cost
Soldering iron	Pinecil or similar temp-controlled iron	US$ 20–35
Solder + flux	Lead-free 0.6–0.8mm + flux pen	US$ 8–15
Digital multimeter	Continuity, DC/AC voltage, current (mA range min.)	US$ 15–40
Bench power supply	Adjustable, optional but avoids fried sensors	US$ 30–80
Breadboard + jumper kit	M-M, M-F, F-F jumpers, assorted resistors	US$ 10–20
Wire strippers / flush cutters	Small gauge, 22–26 AWG	US$ 8–15
Heat-shrink tubing kit	Assorted sizes	US$ 5–10
3D printer (optional)	Ender 3 class printer for enclosures	US$ 150–250
Label maker (optional)	For wiring and sensor ID in the field	US$ 15–30

Core electronic components (buy in bulk)

Component	Quantity	Cost	Why
ESP32 dev board	5–10 units	US$ 4–12 each	Common brain for every project
USB power supplies (5V)	3–5 units	US$ 5–15 each	Reliable, avoid cheap no-name bricks
Project enclosures	Assorted sizes, IP65 for outdoor	US$ 5–30 each	3D print or buy off-the-shelf
Perfboard / protoboard	Assorted	US$ 5–15	For semi-permanent builds
Screw terminal blocks	2–4 pin, assorted	US$ 5–10	Cleaner than raw wire splices

Rule of thumb: standardize on ESP32 across all projects. One firmware platform, one set of debugging habits, one flashing workflow — not a zoo of incompatible boards.

3. Electronics Fundamentals Cheatsheet

A weekend of this is enough to unlock every project in this handbook. You do not need a degree — you need these specific concepts, cold.

Ohm's Law (the only formula you truly need)

```
V = I x R        (Voltage = Current x Resistance)
P = V x I        (Power = Voltage x Current)

Rearranged:
I = V / R
R = V / I
```

Use this any time you pick a resistor value, size a pull-up, or check whether a sensor's current draw is safe for a GPIO pin (ESP32 GPIOs typically handle ~12mA safely, 40mA absolute max — never push it).

3.3V vs 5V — the #1 cause of dead boards

Item	Voltage	Gotcha
ESP32 logic level	3.3V	Never feed 5V directly into a GPIO
Many sensors (older ones)	5V logic	Needs a logic-level shifter or voltage divider
I2C sensors (modern)	3.3V native	SCD40/41, BME280, SHT31 — safe direct to ESP32
DS18B20	3.3–5V tolerant	Works fine at 3.3V, simplifies wiring
Relay modules	5V coil, 3.3V-tolerant trigger (check module)	Many boards need a 5V supply + opto-isolated trigger

Pull-up / pull-down resistors

A floating digital input reads random noise. A pull-up resistor ties the pin HIGH by default (signal pulls it LOW to trigger); a pull-down does the opposite. Reed switches, buttons, and many digital sensors need one or the other — ESP32 has usable internal pull-ups/pull-downs you can enable in software, which removes the need for a physical resistor in most binary_sensor use cases.

GPIO, PWM, ADC — the three pin behaviors

Pin mode	What it does	Used for
GPIO digital	Reads/writes HIGH or LOW only	Reed switches, buttons, relay triggers
PWM (pulse-width modulation)	Simulates analog output by rapid switching	Dimming, motor speed, some valve control
ADC (analog-to-digital)	Reads a continuous voltage as a number	Some moisture sensors, analog gas sensors

ESP32 ADC note: the ADC2 pins conflict with Wi-Fi — always prefer ADC1 pins (GPIO32–39 range on most dev boards) for analog sensors.

4. Communication Protocols Cheatsheet

This is the real core of the handbook. Learn these four and you can read almost any sensor datasheet in this space.

4.1 — I2C (Inter-Integrated Circuit)

Two-wire bus (SDA = data, SCL = clock) that lets you connect multiple sensors to the same two ESP32 pins, each identified by a unique address. This is the protocol behind most of your “premium” sensors.

Sensor	Typical I2C Address	Measures
SCD40 / SCD41	0x62	CO2, temperature, humidity
BME280	0x76 or 0x77	Temperature, humidity, pressure
SHT31	0x44 or 0x45	Temperature, humidity (higher accuracy)
BME680	0x76 or 0x77	Temp/humidity/pressure + VOC
SGP40	0x59	VOC index
SSD1306 OLED display	0x3C (typ.)	Small status display

Wiring pattern (always the same 4 wires):

```
Sensor VCC  -> ESP32 3.3V
Sensor GND  -> ESP32 GND
Sensor SDA  -> ESP32 GPIO21 (default SDA)
Sensor SCL  -> ESP32 GPIO22 (default SCL)
```

Multiple I2C sensors can share the same SDA/SCL pins as long as their addresses don't collide. If two sensors share an address, check the datasheet for an address-select pin/solder jumper.

4.2 — UART / Serial

Two-wire, point-to-point (TX/RX) protocol. Used by devices that talk in a constant data stream rather than being “polled” like I2C.

Module	Baud rate	Measures
PZEM-004T V3	9600	Voltage, current, power, energy (with CT clamp)
PMS5003 (particulate)	9600	PM1.0 / PM2.5 / PM10
Some MH-Z19 CO2 sensors	9600	CO2 (NDIR, budget alternative to SCD4x)

```
Sensor TX  -> ESP32 RX (choose any free UART-capable GPIO)
Sensor RX  -> ESP32 TX
Sensor VCC -> matches sensor spec (5V for PZEM, check others)
Sensor GND -> ESP32 GND
```

Remember TX/RX are crossed: a module's TX goes to the ESP32's RX, and vice versa. Wiring TX-to-TX is the single most common “why isn't this working” mistake with serial sensors.

4.3 — 1-Wire

A single data line (plus power and ground) that can address multiple devices on the same wire using a unique 64-bit ID burned into each chip. The defining protocol of the DS18B20 temperature probe — the sensor behind freezer and HVAC monitoring.

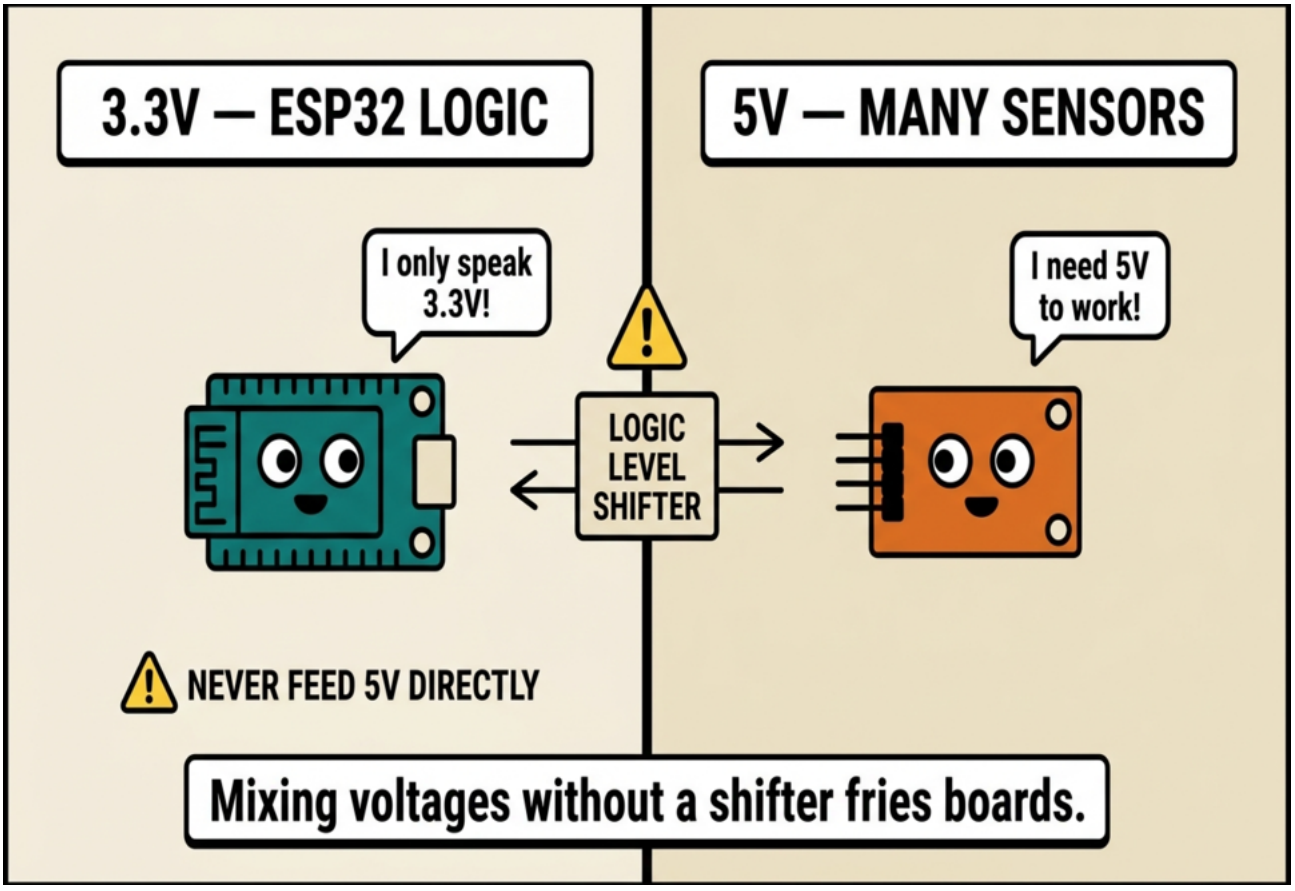
```
DS18B20 Red   -> 3.3V (or 5V, both work)
DS18B20 Black -> GND
DS18B20 Yellow-> ESP32 data GPIO

Required: 4.7k ohm pull-up resistor between data line and VCC
(many waterproof probes ship with this already built in — check datasheet)
```

One data pin can read many DS18B20 probes at once — handy for multi-zone freezer or HVAC supply/return temperature deltas.

4.4 — Plain GPIO digital sensing

The simplest protocol: no protocol at all. A reed switch, water-leak probe, or push button is just a circuit that opens or closes. ESPHome reads it as a `binary_sensor`.

Component	How it behaves
Reed switch (door/window)	Contact opens/closes with magnet proximity
Water leak probe	Two exposed contacts — water bridges them, changes state
Tilt sensor	Contact changes when the sensor is tilted past a threshold
PIR / motion (digital output)	HIGH/LOW pulse on motion detection



5. ESPHome Quick Reference

ESPHome is the fastest path from sensor-in-hand to Home-Assistant-entity. Most of the time you are not writing C++ — you are writing YAML that references already-built components.

Minimal base config (every project starts here)

```
esphome:
  name: freezer-monitor
  friendly_name: Freezer Monitor

esp32:
  board: esp32dev
  framework:
    type: arduino

wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password

api:      # enables native Home Assistant integration
logger:   # enables USB/OTA log viewing
ota:
  - platform: esphome
```

I2C sensor example (SCD41 — CO2 monitor)

```
i2c:
  sda: GPIO21
  scl: GPIO22

sensor:
  - platform: scd4x
    co2:
      name: "CO2"
    temperature:
      name: "Temperature"
    humidity:
      name: "Humidity"
    update_interval: 60s
```

1-Wire sensor example (DS18B20 — freezer monitor)

```
one_wire:
  - platform: gpio
    pin: GPIO4

sensor:
  - platform: dallas_temp
    address: 0x1234567890ABCDEF   # find via logs on first boot
    name: "Freezer Temperature"
    update_interval: 60s
```

GPIO binary sensor example (door / leak / reed switch)

```
binary_sensor:
  - platform: gpio
    pin:
      number: GPIO14
      mode: INPUT_PULLUP
      inverted: true
    name: "Freezer Door"
    device_class: door
```

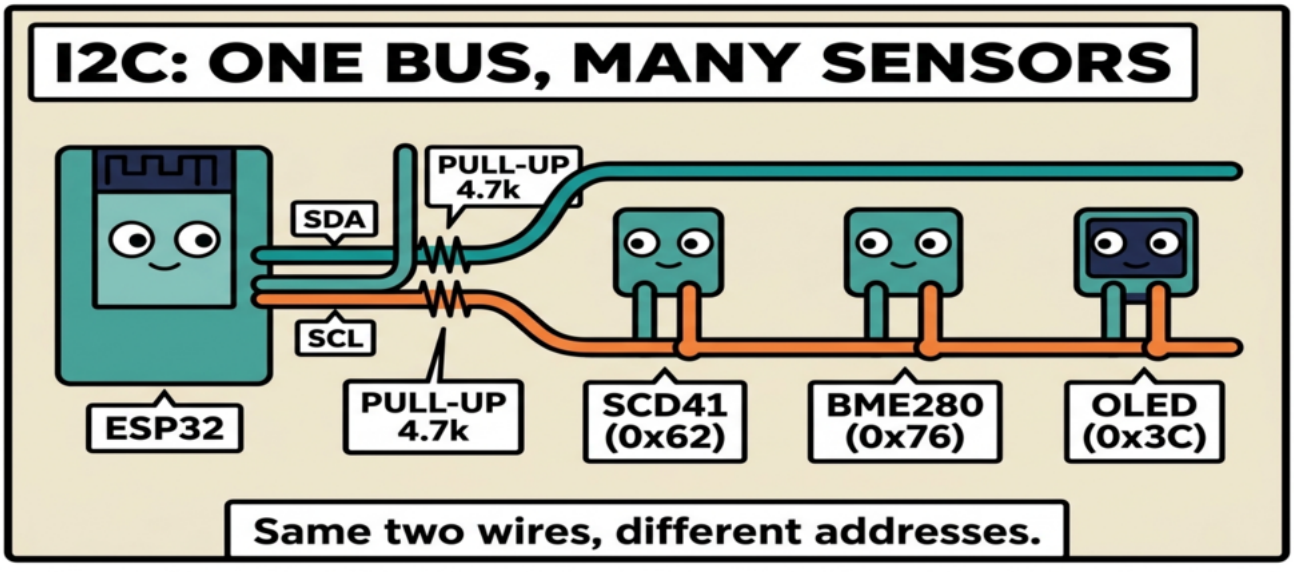
Local automation on-device (no cloud, no HA required to react)

```
binary_sensor:
  - platform: gpio
    pin: GPIO14
    name: "Leak Sensor"
    on_press:
      then:
        - switch.turn_on: water_valve_relay
        - output.turn_on: siren_gpio
```

This is the key local-first idea: critical safety actions (closing a valve, sounding a siren) should trigger directly on the ESP32, not depend on Home Assistant, Wi-Fi, or the cloud being available at that exact second.

ESPHome cheatsheet — component to use per protocol

Sensor Type	ESPHome Building Block
I2C sensors	i2c: + platform-specific block (scd4x, bme280, sht3xd, bme680, sgp40)
UART sensors	uart: bus + platform (pzem004t, pmsx003, mhz19)
1-Wire probes	one_wire: + platform: dallas_temp
Digital contact	binary_sensor: platform: gpio
Relay / valve / siren	switch: platform: gpio, or output: for raw pins
Analog moisture/gas	sensor: platform: adc



6. Home Assistant Integration Cheatsheet

Once an ESPHome device has `api:` enabled, Home Assistant discovers it automatically on the local network — no cloud account, no app pairing.

Integration checklist

- Settings → Devices & Services → the ESPHome device should appear under “Discovered”.
- If not discovered, add manually: Settings → Devices & Services → Add Integration → ESPHome → enter device IP.
- Every sensor/binary_sensor/switch defined in YAML becomes an entity automatically — no extra config needed.
- Use Settings → Areas to assign each device to a room/zone for cleaner dashboards.

Automation cheatsheet (YAML mode)

```
automation:
  - alias: "Freezer too warm"
    trigger:
      - platform: numeric_state
        entity_id: sensor.freezer_temperature
        above: -10
        for:
          minutes: 10
    action:
      - service: notify.mobile_app_your_phone
        data:
          message: "Freezer above -10C for 10 minutes"
```

Dashboards

- Use the built-in **History** and **Statistics** graph cards for any numeric sensor — no add-ons needed for basic trend lines.
- For richer graphs (multi-day energy, CO2 trend overlays), install **ApexCharts Card** via HACS.
- Group related entities per project with **Areas** + **Entities card** rather than one giant dashboard.

Optional add-ons worth knowing

Add-on	Why you'd use it
Node-RED	Visual automation builder, better for complex multi-condition logic than YAML
MQTT broker (Mosquitto)	Needed for Shelly devices and any MQTT-native sensor
ESPHome add-on	Compile/flash devices directly from the HA UI, no separate laptop tool needed
InfluxDB + Grafana	Long-term storage and advanced graphing beyond HA's built-in history

7. Electrical Safety Rules (Non-Negotiable)

These rules apply to any project that reads mains current (energy monitoring) or switches mains-powered loads (valves, pumps, garage motors). Ignore them and you risk fire, electrocution, or both.

The rules

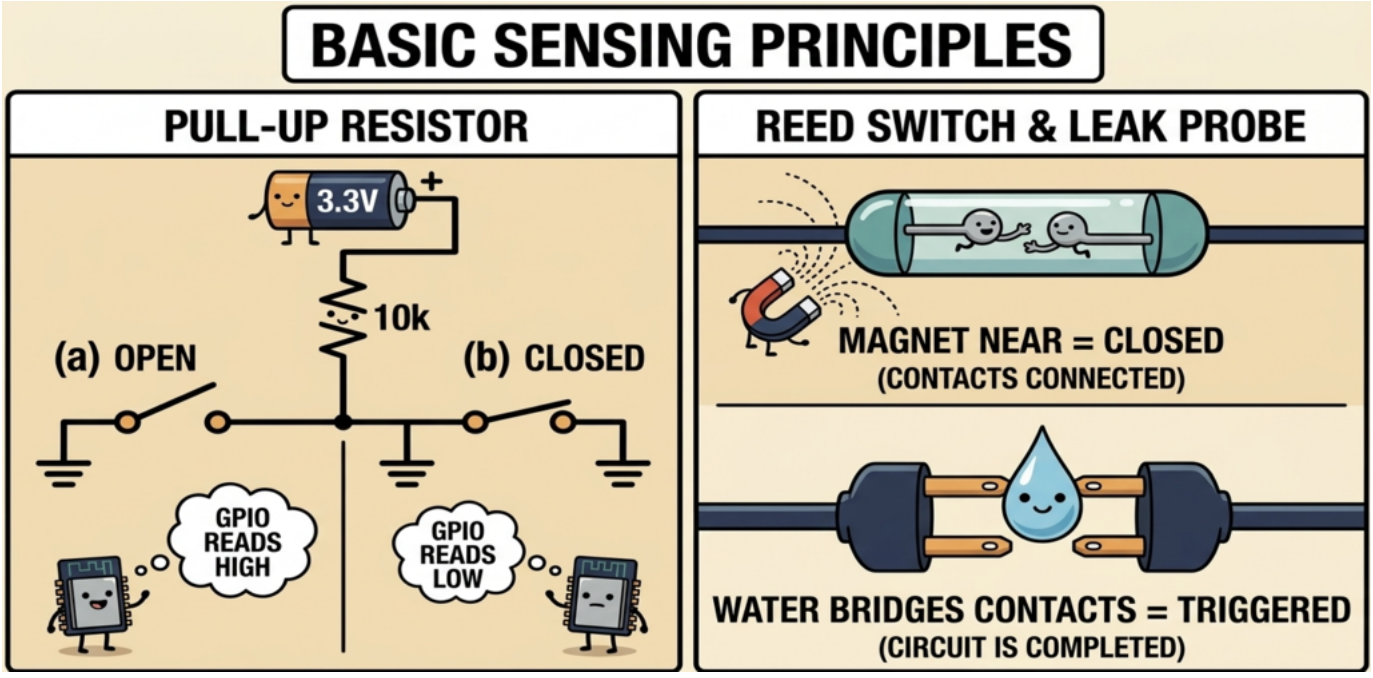
- **Never work on a live circuit.** Turn off the breaker, then verify with a multimeter before touching anything.
- **Non-invasive current sensing (CT clamps) is always preferred** over splicing into a live wire — clamp around the conductor, no cutting required.
- **Galvanic isolation is mandatory** between your ESP32 and anything touching mains: opto-isolated relay modules, not direct GPIO-to-mains contact.
- **Any load switching a motor, pump, or valve above a few watts needs a correctly rated relay or contactor** — check the current rating against the load's inrush current, not just its running current.
- **When in doubt, hire a licensed electrician** for the panel-side work, and keep your own project on the low-voltage side of a clearly defined boundary.
- **Never publish or follow a tutorial that says "just wire it into the panel"** without a breaker, an electrician sign-off, and local code compliance.

What's safe to DIY vs what isn't

Task	Risk Tier	Why
CT clamp current sensing	Safe DIY	No wire cutting, clamp goes around insulated conductor
Smart plug / Shelly on existing outlet	Safe DIY	Uses existing certified outlet wiring
Low-voltage valve actuator (battery/12V)	Safe DIY	No mains contact at all
Wiring a relay into a breaker panel	Electrician	Direct mains contact, code compliance required
Motorized valve on a boiler/mains water heater	Electrician + plumber	Combines electrical and pressurized-system risk
Garage door opener rewiring	Usually DIY-safe	Most openers expose a low-voltage accessory terminal

8. Sensor Library — Specs & Wiring at a Glance

Sensor	Protocol	Measures	Cost	Key Note
SCD40 / SCD41	I2C	CO2, temp, humidity	US$ 20–60	True NDIR/photoacoustic CO2 — do not confuse with cheap eCO2 estimators
BME280	I2C	Temp, humidity, pressure	US$ 3–10	No CO2 — pair with SCD4x for full air quality
SHT31	I2C	Temp, humidity (high accuracy)	US$ 6–15	Better accuracy than BME280 for humidity-critical builds
BME680	I2C	Temp, humidity, pressure, VOC	US$ 12–25	VOC reading is an index, not calibrated ppm
SGP40	I2C	VOC index	US$ 8–18	Pair with BME280/SHT31 for temp/humidity compensation
PMS5003 / SPS30	UART	PM1.0 / PM2.5 / PM10	US$ 15–45	Has a small fan — factor in noise and dust buildup
PZEM-004T V3	UART	Voltage, current, power, kWh	US$ 8–15	Needs a matching CT clamp rated for your max current
DS18B20 (waterproof)	1-Wire	Temperature (single point)	US$ 2–8	Multiple probes share one data pin — great for multi-zone
Reed switch	GPIO digital	Open/closed contact	US$ 1–5	Use INPUT_PULLUP + inverted: true in ESPHome
Water leak probe	GPIO digital	Presence of water	US$ 2–8	Keep contacts elevated off the floor to avoid false triggers from humidity
Capacitive soil moisture	Analog (ADC)	Soil moisture	US$ 2–8	Degrades over 1–2 years — budget for replacement, calibrate per soil type
Ultrasonic distance (JSN-SR04T)	GPIO (trigger/echo)	Tank/reservoir level	US$ 5–12	Waterproof transducer version needed for tank use



9. Per-Project Quick Reference (10 Builds)

Every project below maps back to the protocols and components covered in Chapters 3–8. Difficulty is rated on electrical risk and protocol complexity, not on total build time.

Protocol / stack	UART (PZEM) / vendor local API (Shelly)
Budget	US$ 15–220
Difficulty	Medium–High
Key idea	CT clamp current sensing, kWh tracking, phantom-load detection

Protocol / stack	GPIO digital + relay/valve actuation
Budget	US$ 60–270
Difficulty	Medium
Key idea	Layered defense: sensor → local alert → siren → valve shutoff

Protocol / stack	I2C (SCD40/41)
Budget	US$ 35–115
Difficulty	Low
Key idea	Best entry point for learning I2C properly

Protocol / stack	I2C (temp/humidity) + GPIO (door/vibration)
Budget	US$ 25–115
Difficulty	Low–Medium
Key idea	Combine with UPS NUT integration and smart-plug power monitoring

Protocol / stack	1-Wire (DS18B20) + GPIO (door)
Budget	US$ 20–60
Difficulty	Low
Key idea	Best true first project — no mains contact at all

Protocol / stack	GPIO/ultrasonic + relay for pump control
Budget	US$ 35–150

Difficulty	Medium–High
Key idea	Pump/contactor sizing matters — check inrush current

Protocol / stack	GPIO digital (reed switches, wired bus)
Budget	US$ 30–120
Difficulty	Low
Key idea	Prioritize reliability: no battery, no radio, no cloud

Protocol / stack	I2C (SCD4x + BME280/680 + SGP40) + UART (PM sensor)
Budget	US$ 70–220
Difficulty	Low–Medium
Key idea	Combines every I2C/UART skill from Ch. 4 in one build

Protocol / stack	1-Wire (dual DS18B20) + optional CT clamp
Budget	US$ 30–120
Difficulty	Medium
Key idea	Delta-T between supply/return reveals filter and efficiency issues

Protocol / stack	UART (PZEM) or vendor local API (Shelly EM/3EM)
Budget	US$ 80–300
Difficulty	Medium–High
Key idea	Direct extension of Project 1, per-circuit granularity

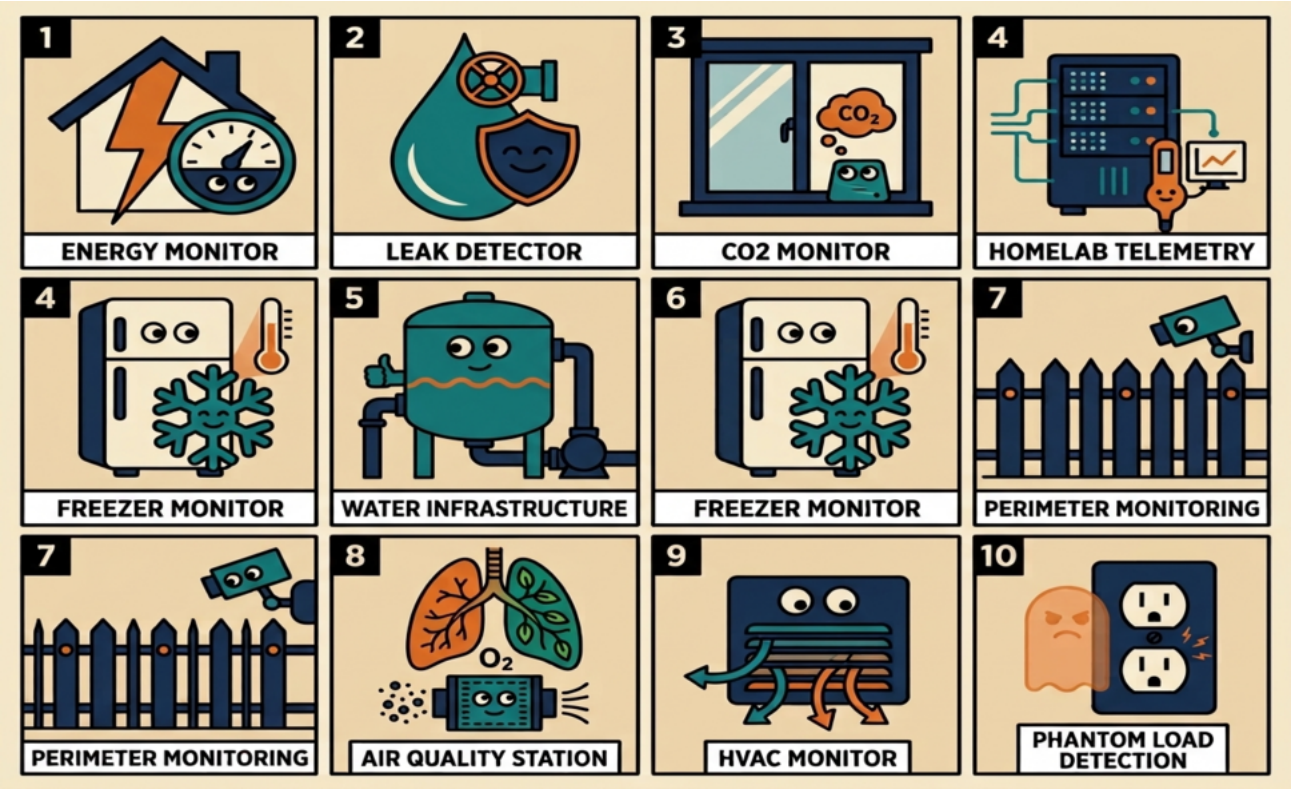
10. Troubleshooting Cheatsheet

Symptom	Likely Cause	Fix
I2C sensor not detected	Wrong SDA/SCL pins, missing pull-ups, address collision	Run an I2C scanner sketch/ESPHome i2c: scan: true; check datasheet address
UART sensor gives garbage data	Wrong baud rate, TX/RX swapped	Confirm baud rate in datasheet; swap TX/RX and retest
DS18B20 reads -127°C	Missing pull-up resistor, bad wiring	Add 4.7k pull-up between data and VCC; check probe address in logs
ESP32 resets/brownouts under load	Weak USB power supply, sensor current spike	Use a proper 5V/2A supply, add a capacitor near sensor VCC
Relay chatters or won't hold	Wrong trigger voltage/logic level for the relay module	Check active-HIGH vs active-LOW; confirm 3.3V vs 5V trigger compatibility
Home Assistant doesn't discover device	Different VLAN/subnet, mDNS blocked	Add manually by IP; check router/AP isolation and mDNS/Bonjour is allowed
Water leak sensor false triggers	Condensation, humidity bridging contacts	Elevate contacts, use a sensor with hysteresis/debounce delay
Soil moisture sensor drifts over time	Capacitive sensor degradation, corrosion	Recalibrate every season; budget for replacement every 1–2 years
Wi-Fi keeps dropping on ESP32	Power-saving mode enabled	Disable Wi-Fi power_save_mode in ESPHome (set to none)

11. Suggested Learning Path

A realistic 2–3 week path from zero to being able to execute any project in this handbook with real understanding, not copy-pasted tutorials.

When	Milestone	What it teaches
Week 1, Days 1–2	Electronics fundamentals (Ch. 3)	Ohm's law, voltage levels, GPIO/PWM/ADC
Week 1, Days 3–5	Build the Freezer Monitor (Project 5)	1-Wire + GPIO, zero electrical risk
Week 1, Days 6–7	ESPHome + Home Assistant basics (Ch. 5–6)	Flash firmware, get entities into HA, write first automation
Week 2, Days 1–3	Build the CO2 Monitor (Project 3)	First real I2C project
Week 2, Days 4–6	Build the Leak Detector (Project 2)	Introduces relay/valve actuation safely (low voltage)
Week 2, Day 7	Electrical safety deep-dive (Ch. 7)	Read before touching anything mains-adjacent
Week 3+	Move to mains-adjacent projects (1, 6, 9, 10)	With electrician consultation where the chapter flags it



12. Glossary

ADC — Analog-to-Digital Converter — reads a continuous voltage as a discrete number.

Binary sensor — Home Assistant/ESPHome term for a sensor with only two states (on/off, open/closed).

CT clamp — Current transformer clamp — measures current by clamping around a wire, no cutting required.

Debounce — Filtering out rapid false triggers from a mechanical or noisy sensor signal.

Galvanic isolation — Electrically separating two circuits so no direct current path exists between them — critical for mains safety.

GPIO — General Purpose Input/Output — a microcontroller pin that can be configured as digital in or out.

I2C — Inter-Integrated Circuit — two-wire bus protocol (SDA/SCL) for connecting multiple addressable sensors.

Local-first — An architecture where core functionality works fully on local hardware/network, without depending on a cloud service.

MQTT — A lightweight local publish/subscribe messaging protocol commonly used by Shelly and other smart devices.

NDIR — Non-Dispersive Infrared — the sensing method used by accurate CO2 sensors like the SCD4x line.

Pull-up / pull-down resistor — A resistor that holds a digital input at a known default state (HIGH or LOW) when idle.

UART — Universal Asynchronous Receiver-Transmitter — serial protocol using TX/RX lines.

1-Wire — A protocol allowing multiple addressable devices to share a single data line, used by the DS18B20.

LocalFirstHome.com — Building smart homes that work without the cloud.