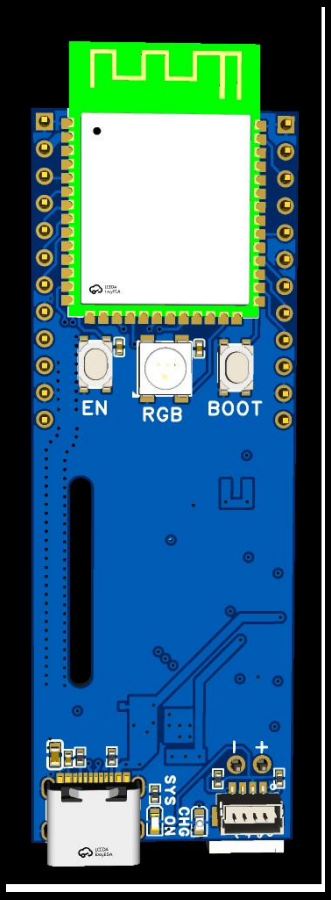
OFFICIAL DOCUMENTATION

ESP32-S3 Dev Board

User Guide

A Wi-Fi + Bluetooth LE development board with a built-in 0.96″ OLED display, USB-C, LiPo battery charging, and a Qwiic connector — running a live weather clock out of the box.

ESP32-S3-WROOM-1-N8 · 8 MB flash · 70.1 × 25.4 mm



HW DESIGNS
docs.hwdesigns.us

UG-01 · Rev A · July 2026
Applies to board revision 2026-06-28

WELCOME

Thanks for picking this board up

This guide covers everything on the board: getting it online, programming it with the Arduino IDE, every pin on the headers, the power and battery system, and the built-in display, LED, and Qwiic connector. Keep it next to the **pinout reference** (the last page of this PDF, and a separate download at docs.hwdesigns.us).

At a glance

Module	ESP32-S3-WROOM-1-N8 — dual-core Xtensa LX7 @ 240 MHz
Wireless	Wi-Fi 4 (2.4 GHz) + Bluetooth 5 LE
Memory	8 MB flash · 512 KB SRAM
Display	0.96″ 128 × 64 OLED (SSD1306, SPI), factory-fitted ribbon
Power	USB-C · 1-cell LiPo with charging + protection · fused 5 V input pin
Expansion	2 × 12 header pins (breadboard friendly) · Qwiic / STEMMA QT (I²C)
Status	Addressable RGB LED (SK6812) · CHG and SYS ON indicators
Size	70.1 × 25.4 mm board · ≈ 72.4 mm including the antenna overhang

In this guide

No drivers, no soldering
The board uses the ESP32-S3's native USB — Windows 10/11, macOS, and Linux recognize it out of the box. Everything in this guide works with a data-capable USB-C cable and the free Arduino IDE.

SECTION 1

Safety & handling

Battery

Use a **single-cell 3.7 V LiPo** with a JST-XH plug. Battery lead polarity is **not standardized** between vendors — before plugging in any new battery, match its red (+) and black (–) leads to the + / – marks printed next to the connector on the back of the board. Reverse-polarity protection is fitted, but checking first is the habit that keeps packs healthy.

The charger delivers a fixed ≈ 200 mA, so use cells of **400 mAh or larger** to stay at or below a gentle 0.5 C charge rate. Never charge a swollen, punctured, or damaged cell, and don't leave a charging battery unattended for long periods.

Electrical

All I/O runs at **3.3 V logic** and the pins are **not 5 V tolerant**. Connecting 5 V signals or supplies to a GPIO, 3V3, or the Qwiic connector can permanently damage the module. The only pin that accepts 5 V is the one marked 5V_IN (J5 pin 12).

Handle the board by its edges, especially in dry environments — like any exposed electronics it is sensitive to static discharge.

Antenna

The green area at the top of the module is the 2.4 GHz antenna, and it slightly overhangs the board edge on purpose. Keep metal, batteries, and fingers away from it; for best range, let that end hang over the edge of your breadboard or enclosure.

If something looks wrong
Stop and email support@hwdesigns.us with a photo of your setup and the code printed along the back edge of the board — it tells us your exact revision.

SECTION 2

Getting started

The board arrives ready to use. Steps 1–2 get the built-in demo online; steps 3–4 set you up to write your own code.

Step 1 · Plug it in

Connect the board to any USB-C charger or computer port. It powers up and starts the built-in Weather Clock on the OLED display.

Step 2 · Put it on your Wi-Fi

On first boot the board opens its own setup hotspot. Join it from your phone, and a setup page appears — pick your home network and enter its password. The board restarts and shows your local time and weather, with the RGB LED tinted by the temperature outside.

Note: the board uses 2.4 GHz Wi-Fi. If your router broadcasts separate 2.4 GHz and 5 GHz names, choose the 2.4 GHz one.

Step 3 · Set up the Arduino IDE

Install the free Arduino IDE from arduino.cc/en/software, open **File → Preferences**, and paste this into Additional boards manager URLs:

```
https://espressif.github.io/arduino-esp32/package_esp32_index.json
```

Open **Tools → Board → Boards Manager**, search for `esp32`, and install esp32 by Espressif Systems. Then match these settings under Tools:

Setting	Value
Board	`ESP32S3 Dev Module`
USB CDC On Boot	`Enabled`
Flash Size	`8MB (64Mb)`
Partition Scheme	`8M with spiffs (use “RainMaker 8MB” only for the RainMaker example)`

No driver install is needed — the ESP32-S3’s native USB is recognized by Windows 10/11, macOS, and Linux out of the box.

Step 4 · Upload your first sketch

Select the board’s port under **Tools → Port**, paste this in, and click Upload. The onboard LED cycles red, green, blue:

```
#define RGB_PIN 48   // onboard RGB LED

void setup() {}

void loop() {
  neopixelWrite(RGB_PIN, 40, 0, 0);  delay(400);  // red
  neopixelWrite(RGB_PIN, 0, 40, 0);  delay(400);  // green
  neopixelWrite(RGB_PIN, 0, 0, 40);  delay(400);  // blue
}
```

If the port doesn't appear or the upload won't start: use a data-capable USB-C cable, or hold the **BOOT** button while plugging the board in, then try again.

Step 5 · Keep going

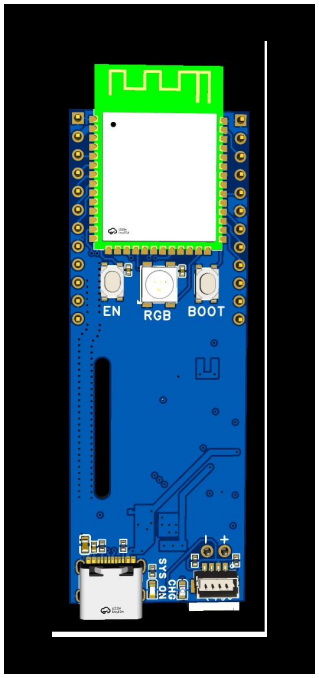
Take the board tour in the next section, keep the pin reference (Section 4 and Appendix A) on the bench, and then build something from the example projects at docs.hwdesigns.us — including the full source of the Weather Clock your board shipped with, a five-minute Qwiic sensor project, and a smart-home starter using ESP RainMaker.

SECTION 3

Board tour

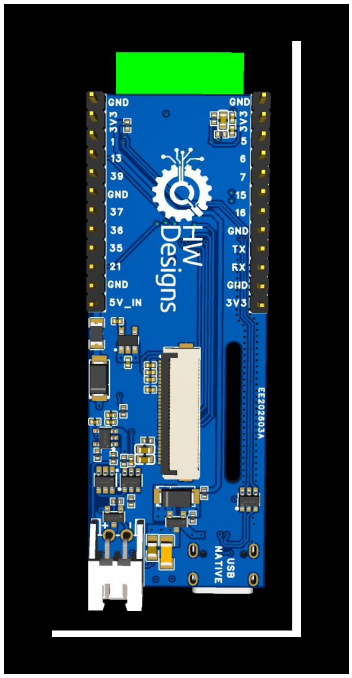
Both sides, and what everything does. Renders show the bare board — the OLED display mounts on the front, over the ribbon slot.

Front



- ESP32-S3 module up top — the green area is the Wi-Fi antenna; keep it clear of metal.
- EN (reset) on the left and BOOT on the right, with the RGB status LED between them.
- Ribbon slot for the factory-fitted OLED, left of center.
- USB-C at the bottom for power, charging, and programming.
- CHG (amber) and SYS ON (green) indicator LEDs beside the USB port.
- Qwiic / STEMMA QT connector on the bottom-right edge for solder-free I²C sensors.

Back



- Every header pin labeled on the silkscreen — power (3V3, GND, 5V_IN), GPIO numbers, TX/RX.
- 30-pin ribbon connector for the OLED display, center of the board.
- LiPo battery connector (JST-XH) with polarity marked + / –, near the USB end.
- Board revision code printed along the edge — include it when you contact support.

70.1 × 25.4 mm · 0.1″ header pitch on 0.9″ row centers — it plugs straight into a standard breadboard. Designing an enclosure? Grab the 3D model (STEP) at docs.hwdesigns.us.

SECTION 4

Pin reference

Two 12-pin headers, **pin 1 nearest the antenna** on both. J4 is the left row and J5 the right row when you look at the front of the board with the antenna up; the labels repeat on the back silkscreen. The full-page diagram is in Appendix A.

J4 — left header

Pin	Label	GPIO	Also on this pin
1	**GND**	—	Ground
2	**3V3**	—	3.3 V regulator output
3	**IO5**	GPIO5	ADC1_CH4 · TOUCH5 · RTC (deep-sleep wake)
4	**IO6**	GPIO6	ADC1_CH5 · TOUCH6 · RTC
5	**IO7**	GPIO7	ADC1_CH6 · TOUCH7 · RTC
6	**IO15**	GPIO15	ADC2_CH4 · RTC · U0RTS
7	**IO16**	GPIO16	ADC2_CH5 · RTC · U0CTS
8	**GND**	—	Ground
9	**TX**	GPIO43	U0TXD — serial data out of the board
10	**RX**	GPIO44	U0RXD — serial data into the board
11	**GND**	—	Ground
12	**3V3**	—	3.3 V regulator output

J5 — right header

Pin	Label	GPIO	Also on this pin
1	**GND**	—	Ground
2	**3V3**	—	3.3 V regulator output
3	**IO1**	GPIO1	ADC1_CH0 · TOUCH1 · RTC
4	**IO13**	GPIO13	ADC2_CH2 · TOUCH13 · RTC
5	**IO39**	GPIO39	JTAG MTCK by default — still works as a normal GPIO
6	**GND**	—	Ground
7	**IO37**	GPIO37	Free — no PSRAM on this module
8	**IO36**	GPIO36	Free — no PSRAM on this module
9	**IO35**	GPIO35	Free — no PSRAM on this module
10	**IO21**	GPIO21	RTC (deep-sleep wake)
11	**GND**	—	Ground
12	**5V_IN**	—	5 V power input · 0.5 A fuse · input only — never outputs 5 V

Used on the board (not on the headers)

These GPIOs are wired to on-board hardware and are not brought out, so nothing on the headers ever conflicts with the display, LED, USB, or Qwiic port.

Function	GPIO	Notes
OLED display (SPI)	`CS 10 · DC 9 · RST 14 · SCLK 12 · MOSI 11`	Factory-fitted 0.96″ SSD1306
Qwiic I²C	`SDA 8 · SCL 18`	4.7 kΩ pull-ups to 3.3 V fitted on board
RGB status LED	`GPIO48`	SK6812, addressable
BOOT button	`GPIO0`	Boot strap at reset; user button afterwards (LOW = pressed)
USB D– / D+	`GPIO19 / GPIO20`	Native USB, ESD-protected
Battery voltage sense	`GPIO4`	Reads half the battery voltage (2 × 100 k divider)
USB voltage sense	`GPIO2`	Reads half the USB voltage — detects USB power

Choosing pins

Analog + Wi-Fi: ADC2 shares hardware with the radio, so IO13, IO15, and IO16 can't do analog reads while Wi-Fi is active. Use the ADC1 pins — IO1, IO5, IO6, IO7 — for analog input in a connected project.

Deep-sleep wake: any pin marked RTC (IO1, IO5–IO7, IO13, IO15, IO16, IO21) can wake the chip from deep sleep.

Peripheral routing: the ESP32-S3's GPIO matrix lets you place UART, I²C, SPI, and PWM on any header GPIO — there are no fixed "SPI pins" to hunt for.

Serial header: TX (GPIO43) carries data out of the board and RX (GPIO44) carries data in — in Arduino these are `Serial0`, while `Serial` is the USB port.

Limits: 3.3 V logic only, and keep each pin under about 20 mA.

SECTION 5

Power & battery

Three ways to power it

Source	Where	Details
USB-C	Front, bottom edge	The everyday option — powers the board, charges the battery, and programs the chip. Works with C-to-C and A-to-C cables.
LiPo battery	JST-XH, on the back	Single-cell 3.7 V. On-board protection guards against over-charge, over-discharge, over-current, and reverse polarity.
5V_IN pin	J5 pin 12	Regulated 5 V in through a 0.5 A resettable fuse. Input only — this pin never outputs 5 V, so it can't back-power your USB port or breadboard rail.

Plug in USB while a battery is connected and the board switches to USB power automatically and charges the cell — that's the normal, supported combination. Unplug USB and it falls back to the battery without a reset.

One rule for 5V_IN
When powering from the 5V_IN pin, leave the battery unplugged. 5V_IN feeds the same rail the battery discharges into, and the pack would see an unregulated voltage — the protection circuit will step in, but it isn't a supported combination. Battery charging happens over USB-C only.

Charging

The MCP73831 charger delivers a fixed **≈ 200 mA**, so a 1,000 mAh cell fills from empty in roughly six hours. The amber **CHG** LED is on while charging and turns off when the cell is full (with no battery connected it may flicker — that's normal for this charger). The green SYS ON LED simply shows the 3.3 V rail is up.

What to expect from a battery

Condition	Measured
Weather Clock running, Wi-Fi connected (average)	≈ 70 mA
Active phase (display + network activity)	≈ 110 mA
Wi-Fi transmit peaks (brief)	≈ 400 mA
Runtime on a 1,000 mAh cell	≈ 14 hours
Deep sleep floor (regulator + battery monitor)	on the order of 0.1 mA

Measured on this board revision with the shipping firmware. Small or aging cells can sag below the ESP32's brown-out level during the 400 mA transmit peaks — if the board resets when Wi-Fi starts on battery, try a healthy cell of 400 mAh or more.

Powering your add-ons

The 3V3 pins come from the on-board 600 mA regulator, which also feeds the ESP32's transmit peaks — budget about **150 mA** for your sensors and accessories. The 3V3 pins are outputs: don't feed external power into them. For long storage, unplug the battery — the always-on voltage monitor draws a small standby current.

SECTION 6

Using the peripherals

OLED display (SPI · SSD1306)

The 0.96″ display is driven over SPI. With the **U8g2** library (Library Manager → “U8g2”):

```
#include <U8g2lib.h>
#include <SPI.h>

// board wiring: CS 10, DC 9, RST 14, SCLK 12, MOSI 11
U8G2_SSD1306_128X64_NONAME_F_4W_HW_SPI
    u8g2(U8G2_R0, /*cs=*/10, /*dc=*/9, /*rst=*/14);

void setup() {
  SPI.begin(/*sck=*/12, /*miso=*/-1, /*mosi=*/11, /*ss=*/10);
  u8g2.begin();
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_ncenB10_tr);
  u8g2.drawStr(10, 38, "Hello, world!");
  u8g2.sendBuffer();
}
void loop() {}
```

RGB status LED (SK6812 · GPIO48)

One addressable LED on GPIO48. The Arduino core’s built-in `neopixelWrite(48, r, g, b)` drives it with no library at all (see the Step 4 sketch), or treat it as a 1-pixel strip with Adafruit NeoPixel / FastLED. It idles dark until your code lights it.

Qwiic / STEMMA QT (I²C)

Plug any Qwiic or STEMMA QT sensor into the side connector — no soldering, no pull-ups needed (4.7 kΩ resistors are already on the board). The bus lives on SDA = GPIO8, SCL = GPIO18, at 3.3 V:

```
#include <Wire.h>

void setup() {
  Serial.begin(115200);
  Wire.begin(/*SDA*/8, /*SCL*/18);
  for (byte a = 1; a < 127; a++) {          // quick bus scan
    Wire.beginTransmission(a);
    if (Wire.endTransmission() == 0)
      Serial.printf("Found device at 0x%02X\n", a);
  }
}
void loop() {}
```

Buttons

EN resets the chip. **BOOT** (GPIO0) selects download mode when held during a reset — and after boot it’s all yours as a user button: `pinMode(0, INPUT_PULLUP)`, reads LOW while pressed.

Battery & USB monitoring

Two on-board dividers let your sketch watch the power situation:

```
#define PIN_VBAT_SENSE 4    // battery voltage / 2
#define PIN_VBUS_SENSE 2    // USB voltage / 2

void loop() {
  float vbat = analogReadMilliVolts(PIN_VBAT_SENSE) * 2 / 1000.0;
  bool onUsb = analogReadMilliVolts(PIN_VBUS_SENSE) > 1500;
  Serial.printf("Battery %.2f V, USB %s\n", vbat, onUsb ? "yes" : "no");
  delay(1000);
}
```

SECTION 7

Troubleshooting

Symptom	What to try
Computer doesn't see the board	Use a data-capable USB-C cable — many bundled cables are charge-only. If the port still doesn't appear, hold BOOT while plugging the board in, release it, and check again.
Upload starts, then fails	Close any open serial monitors, try the BOOT-while-plugging trick, and check that the Tools settings match the table in Step 3.
Nothing in the Serial Monitor	Set "USB CDC On Boot: Enabled" under Tools, re-upload, and select the board's port.
Wi-Fi won't connect	The board is 2.4 GHz only — pick the 2.4 GHz network name. To change networks, run the setup step again from the documentation site.
Board resets when Wi-Fi starts on battery	Transmit peaks reach ≈ 400 mA. Use a healthy cell of 400 mAh or more and check the connector seating.
Display stays dark after your own sketch	Your code owns the screen — nothing shows until a sketch draws to it. Use the Section 6 snippet, and re-flash the Weather Clock any time from the documentation site.
Analog reads fail on IO13 / IO15 / IO16	Those are ADC2 pins, which can't convert while Wi-Fi is active. Move analog inputs to IO1, IO5, IO6, or IO7 (ADC1).
Charging LED flickers with no battery	Normal for this charger IC when no cell is attached — it stops once a battery is plugged in.

Still stuck? Email support@hwdesigns.us — tell us what you tried and what happened. A photo of your setup helps, and the code printed on the back edge of the board tells us your exact revision. We read every message.

SECTION 8

Specifications

Module	ESP32-S3-WROOM-1-N8 · dual-core Xtensa LX7 @ 240 MHz
Wireless	Wi-Fi 4 (802.11 b/g/n, 2.4 GHz) · Bluetooth 5 LE
Memory	8 MB flash · 512 KB SRAM (no PSRAM — IO35–IO37 free)
USB	USB-C · native USB (serial + programming) · on-board ESD protection
Display	0.96″ 128 × 64 OLED · SSD1306 · SPI · 30-pin ribbon, factory-fitted
Indicators	SK6812 addressable RGB (GPIO48) · CHG (amber) · SYS ON (green)
Headers	2 × 12 · 0.1″ pitch · 0.9″ row spacing · 12 free GPIOs + UART0
Qwiic	JST-SH 4-pin · I²C at 3.3 V · SDA GPIO8 / SCL GPIO18 · pull-ups fitted
Battery	1-cell LiPo · JST-XH · ≈ 200 mA charge · over-charge / over-discharge / over-current / reverse protection
5V_IN	Regulated 5 V input · 0.5 A resettable fuse · input only
Regulator	3.3 V · 600 mA (AP2112K) · ≈ 150 mA budget for add-ons
Logic level	3.3 V — not 5 V tolerant
Dimensions	70.1 × 25.4 mm board · ≈ 72.4 mm overall including antenna overhang
Firmware	Ships with the Weather Clock demo · Arduino, ESP-IDF, MicroPython, and CircuitPython all supported
Document	UG-01 Rev A · July 2026 · applies to board revision 2026-06-28

Open documentation

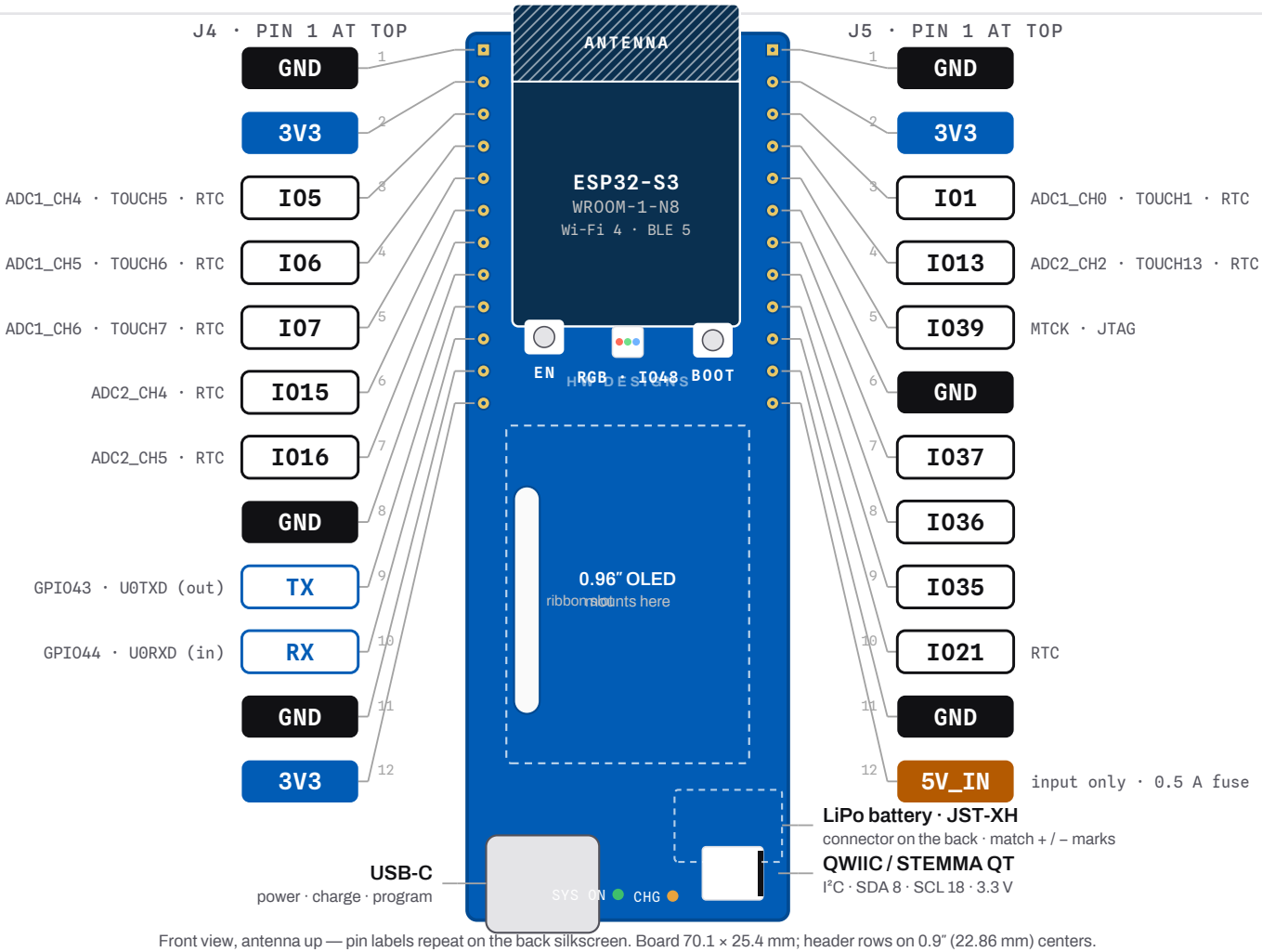
The schematic, printable pinout, 3D model (STEP), and the full source of the Weather Clock and example projects are free to download at docs.hwdesigns.us — the same page the QR code in your packaging points to.

Appendix A on the next page is the full pinout diagram — print it and keep it on the bench.

ESP32-S3 DEV BOARD

PINOUT REFERENCE

HW DESIGNS
docs.hwdesigns.us



Front view, antenna up — pin labels repeat on the back silkscreen. Board 70.1 × 25.4 mm; header rows on 0.9″ (22.86 mm) centers.

ON-BOARD CONNECTIONS

OLED · SPI	CS 10 · DC 9 · RST 14 · SCK 12 · MOSI 11
Qwiic · I²C	SDA 8 · SCL 18 · 4.7k pull-ups
RGB LED	IO48 (SK6812)
BOOT button	IO0 — user key after boot
USB D- / D+	IO19 / IO20
VBAT sense	IO4 = battery voltage / 2
VBUS sense	IO2 = USB voltage / 2

These signals are not brought out to the headers.

GOOD TO KNOW

All I/O is 3.3 V logic — pins are not 5 V tolerant.
Keep each pin under ~20 mA.
ADC2 (IO13, IO15, IO16) is unavailable while Wi-Fi is active — use ADC1 pins (IO1, IO5–IO7) instead.
IO35–IO37 are fully free on this module (N8, no PSRAM).
IO39 is JTAG MTCK by default; still works as GPIO.
RTC pins can wake the chip from deep sleep.
Any pin can be UART / I²C / SPI / PWM (GPIO matrix).

POWER

Inputs: USB-C · 1-cell LiPo (JST-XH) · 5V_IN pin.
5V_IN is input only (0.5 A fused) — it never outputs 5 V. Unplug the battery when powering from 5V_IN.
3V3 pins are regulator outputs — budget ≈ 150 mA for your add-ons.
Battery charges from USB-C only, at ≈ 200 mA.
CHG LED: on = charging · off = full or no battery.

GND · 3V3 · 5V_IN · IOxx · UART

Build something.

Guides, example projects, schematic, and the full Weather Clock source live at our documentation site.

docs.hwdesigns.us

support@hwdesigns.us

UG-01 · Rev A · July 2026 — for board rev 2026-06-28