

Valentino Board - Configuration Guide

Jumpers, headers, status LEDs and shield limits · Build 3

Board	Valentino: ESP32 + BQ24650 solar/battery charger + LoRa + sensors, 110 x 120 mm
Source	KiCad schematic (2026-09-18) and the corrected PCB (2026-09-23, silkscreen "Valentino Build 3"). Schematic and PCB netlists were compared and match (0 differences).
Datasheets used	TI BQ24650 (SLUSA75B), TI TPL5010 (SNAS651A)
Language	The schematic and PCB notes are in Spanish; they are translated here and the originals are quoted in Appendix A.
Prepared	2026-09-24. Values marked "netlist" are calculated from the schematic; nothing here has been bench-measured yet.

Read this first: things to check

- **Silkscreen legends.** The earlier Build 3 PCB file printed wrong legends for J7 (position E-K as 0.5 A, when it is 3.0 A) and for JP7/JP8 (JP8 alone as 15 V, when it is 14.2 V). The corrected PCB fixes both and this guide agrees with it. If your boards were made from the earlier file, use the tables in this guide and ignore the printed legends. See sections 4 and 5.
- **Setting a battery.** Battery type sets JP3 and JP7/JP8 together; battery size sets J7. The settings table is in section 4.2.
- **Chemistry polarity in firmware.** Build 1 firmware treats GPIO39 LOW as LiFePO4. On Build 3, JP3 fitted (LOW) means **lead-acid**, so the polarity is inverted. See section 12.
- Check the charge voltage and current with a meter before connecting a battery.

1. What can be configured

Ref	Silkscreen	Configures	Default (no jumper)
JP3	LiFePo4 Jumper OFF / Lead Acid Jumper ON	Battery chemistry: charge termination on/off. Also readable by the ESP32 (GPIO39).	OFF = LiFePO4 (termination on)
JP7, JP8	Battery Charging Levels	Battery charge (regulation) voltage: 13.6 / 14.2 / 14.9 V	Both open = 13.6 V
J7	Battery Charging Level J7 (Charge / Panel / Battery)	Fast-charge current, 0.5 A to 8.0 A, by adding sense resistors in parallel	No jumper = 0.5 A
JP1	Jumper ON Watchdog ENABLED / OFF DISABLED	Connects the TPL5010 watchdog reset output to the ESP32 EN pin	OFF = watchdog disabled
J8	Time (L-Q ... P-U)	Watchdog timeout: 1 s / 1 min / 10 min / 30 min / 1 h	Fit exactly one jumper
J5	THERMISTOR	10 k NTC for battery temperature. Charging is suspended if it is missing.	Fit a 10 k NTC
D8, D10	COMPLETE, IN PROGRESS	Charger status LEDs (read by the ESP32 through the ADS1115)	n/a

Not jumper-selectable (fixed by components): solar-input regulation point 17.8 V, input range 10-24 V, sensor I²C addresses. See section 9.

2. Board map

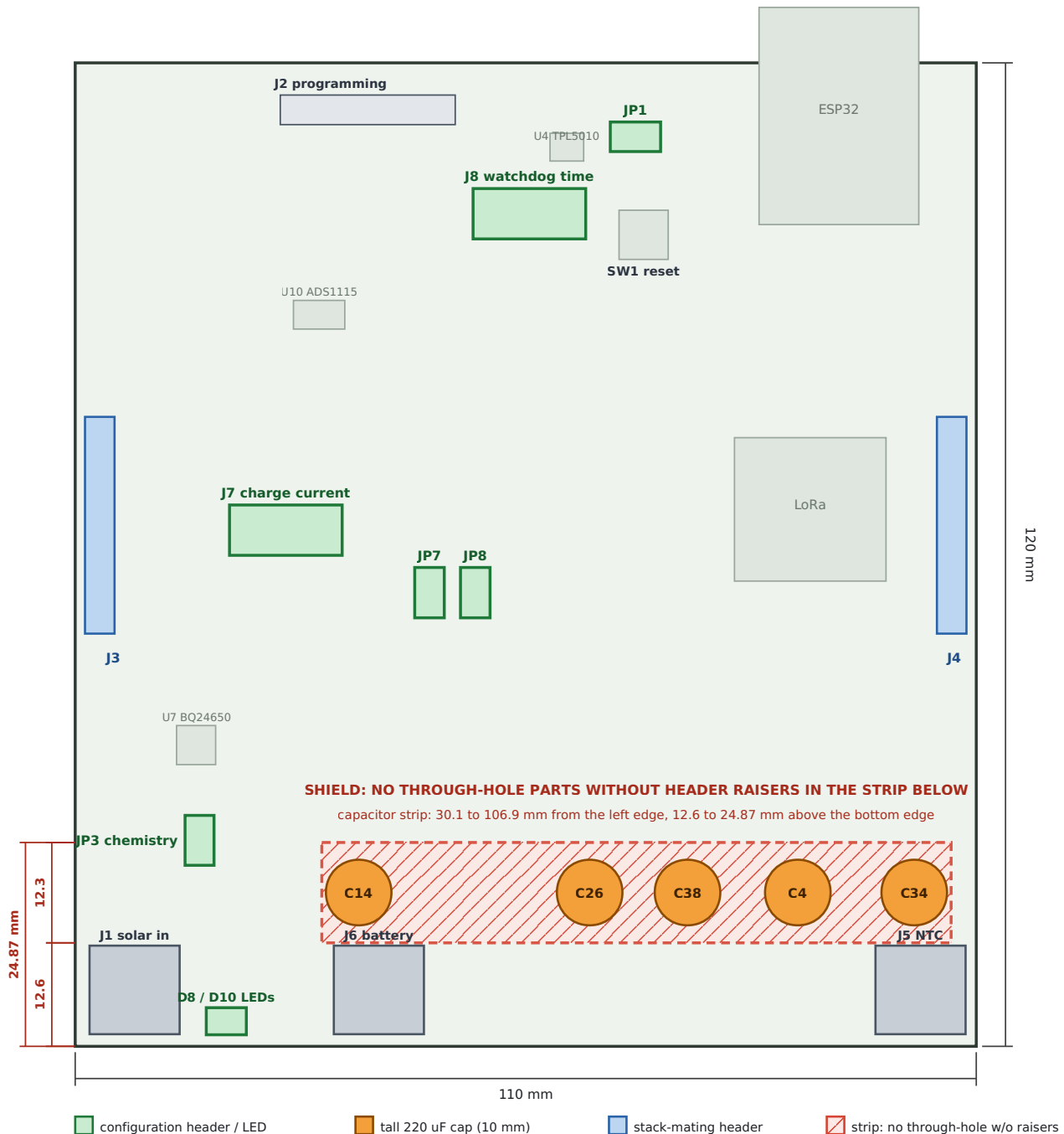


Figure 1. Top view (component side), to scale, from the PCB file. Green = configuration headers and LEDs, orange = the five 220 uF, 10 mm tall electrolytic capacitors, blue = the J3/J4 side headers, hatched = the strip the five capacitors occupy (12.6 to 24.87 mm above the bottom edge, 30.1 to 106.9 mm from the left edge), where a shield must carry no through-hole components without header raisers (section 10). The ESP32 antenna overhangs the top edge by 6.9 mm.

3. Battery chemistry - JP3

JP3 does two things: it sets the BQ24650 **TERM_EN** pin (charge termination on/off) and it presents the same level to ESP32 **GPIO39** (SENSOR_VN) so firmware can read the setting. GPIO39 is input-only; the pull-up is the external R16 (10 k to 3.3 V), and R18 (100 Ω) sits between that node and TERM_EN.

JP3	Chemistry	TERM_EN / GPIO39	Charger behaviour
OFF (no jumper)	LiFePO4 (lithium)	High (3.3 V)	Charging terminates when the current falls to 10% of the set current. STAT2 ("COMPLETE") lights. A new cycle starts when the battery falls about 0.3 V (50 mV at VFB) below the set voltage.
ON (jumper fitted)	Lead-acid	Low (GND)	Termination disabled : the charger holds the set voltage indefinitely (float). "COMPLETE" is not expected to light.

JP3 does not choose the voltage. Set the charge voltage with JP7/JP8 (section 4) to suit the battery; section 4.2 gives the JP3 and JP7/JP8 settings together. In lead-acid mode the selected voltage is held permanently, so use 13.6 V for a standby battery (kept full and rarely discharged). A battery that is discharged and recharged every day, such as solar on a post, needs 14.2 V (section 4.2); check that its datasheet allows continuous charging at that voltage. The ESP32 has no control over the charger on this board; it can only read it back.

4. Charge voltage - JP7 / JP8

The charge voltage is set by the feedback divider on the BQ24650 VFB pin: $V = 2.1 \text{ V} \times (1 + R_A / R_B)$, with $R_A = R48 = 499 \text{ k}\Omega$. R_B is a chain $R1 \ 82 \text{ k}\Omega + R5 \ 4.7 \text{ k}\Omega + R11 \ 4.7 \text{ k}\Omega$. **JP7 shorts R11 and JP8 shorts R5**, so each jumper removes 4.7 kΩ from R_B . The 2.1 V reference is the datasheet typical value.

4.1 The three voltages

JP7	JP8	RB (netlist)	Charge voltage (netlist)	Silkscreen says
open	open	91.4 kΩ	13.56 V	13.6 V
closed	open	86.7 kΩ	14.19 V	14.2 V
open	closed	86.7 kΩ	14.19 V	not printed (same as JP7 closed alone)
closed	closed	82.0 kΩ	14.88 V	14.9 V

Each jumper removes the same 4.7 kΩ, so **JP7 closed alone and JP8 closed alone give the same 14.2 V**. The silkscreen prints the JP7 case. Only one jumper closed means 14.2 V, both closed means 14.9 V, neither means 13.6 V.

Earlier PCB file. The earlier Build 3 file printed 13.7 / 14.3 V and "JP7 OPEN + JP8 CLOSED = 15 V". The last one is wrong: that setting gives 14.2 V, and 14.9 V needs both jumpers closed. The corrected PCB prints 13.6 / 14.2 / 14.9 V.

4.2 Which JP7 / JP8 setting for which battery

The charge voltage follows the **battery type**, the same choice that sets JP3. It does **not** depend on the battery size or on J7: J7 (charge current) follows the battery size, section 5. So set **JP3 and JP7/JP8 together from the battery type**, and J7 from the battery size.

Standby or cycling? **Standby** means the battery is kept full and only rarely discharged, like the battery in an alarm panel; it sits at the float voltage. **Daily cycling** means it is discharged at night or in cloud and recharged the next day, which is what a solar device on a post does even though it stays installed for years. A solar device is a cycling case, so it takes the higher voltage.

Battery (12 V class)	Use	JP3	JP7	JP8	Charge voltage
LiFePO4, 4 cells (12.8 V)	any	OFF (no jumper)	closed	open	14.2 V (3.55 V per cell)
Lead-acid, sealed (AGM or gel)	standby: kept full, rarely discharged (float)	ON (fitted)	open	open	13.6 V
Lead-acid, sealed (AGM or gel)	daily cycling, for example solar on a post	ON (fitted)	closed	open	14.2 V
Lead-acid, flooded	standby: kept full, rarely discharged (float)	ON (fitted)	open	open	13.6 V
Lead-acid, flooded	daily cycling, watched	ON (fitted)	closed	closed	14.9 V
LiFePO4, 4 cells	any	OFF	closed	closed	14.9 V: do not use (3.72 V per cell, above the 3.65 V limit)
LiFePO4, 4 cells	any	OFF	open	open	13.6 V: works, but the pack only charges partly (3.4 V per cell)

JP8 closed with JP7 open gives the same 14.2 V as JP7 closed with JP8 open. The lead-acid rows keep JP3 fitted, so charge termination is off and the chosen voltage is held indefinitely. Lead-acid and LiFePO4 values are typical ones; check the battery datasheet before relying on them.

Worked examples (battery size then sets J7, from the table in section 5):

Battery	JP3	JP7	JP8	J7	Panel
12 Ah LiFePO4, 4 cells	OFF	closed	open	E-K (3.0 A)	40 W
4 Ah LiFePO4, 4 cells, small 10 W panel	OFF	closed	open	none (0.5 A)	10 W
12 Ah sealed lead-acid, standby (float)	ON	open	open	E-K (3.0 A)	40 W
24 Ah flooded lead-acid, daily cycling, watched	ON	closed	closed	B-G + D-I + E-K (6.0 A)	80 W
9 Ah sealed lead-acid, solar on a post (daily cycling)	ON	closed	open	C-H (2.0 A)	30 W
6 Ah sealed lead-acid, daily cycling	ON	closed	open	B-G (1.5 A)	20 W

9 Ah has no printed row on the silkscreen. C-H (2.0 A) is the largest single setting not above the C/4 rule (2.25 A); the 30 W panel is the rule of thumb of charge current x about 14 V. If the 9 Ah battery is only kept as a standby (rarely discharged), leave both JP7 and JP8 open (13.6 V).

Both charge voltages are for 12 V-class packs: 4 LiFePO4 cells or 6 lead-acid cells. They are wrong for any other cell count.

Firmware safety net. The ESP32 cannot read JP7 or JP8. ValentinoTest therefore warns in its report when the battery voltage is above **14.6 V on LiFePO4** (or 15.0 V on lead-acid). That catches the dangerous mistake of both jumpers closed on a LiFePO4 pack, which charges it to 14.9 V. It does not say which jumper is wrong, and it only warns once the battery has reached that voltage.

4.3 Lead-acid or LiFePO4?

JP3 supports both. For a compact device on a post, cycled daily by a small solar panel, the practical differences are these (typical values; check the datasheet of the battery you fit):

Item	Sealed lead-acid	LiFePO ₄ , 4 cells (12.8 V)
Size and weight	Bulky	About a third to a half of the size and weight for the same usable energy
Life with daily cycling	About 300 - 500 cycles at 50% depth	Thousands of cycles, so years of daily solar cycling
Price	Low	Two to three times higher up front
Charging in the cold	Tolerates it	Must not be charged below 0 °C
Protection	Robust, none needed	Needs a battery management system (BMS); most packs include one
How this board charges it	JP3 ON: the chosen voltage (13.6 or 14.2 V) is held constant indefinitely	JP3 OFF: charges to 14.2 V, stops at 10% of the set current, restarts about 0.3 V lower
Voltage as a charge gauge	Reasonable: voltage falls steadily as it empties	Poor: the curve is very flat, so voltage says little about the charge left

The board's thermistor stops charging below about 0.5 °C and above about 41 °C for both chemistries (section 7). That suits LiFePO₄. For lead-acid it is an extra limit that leaves the battery uncharged on frosty mornings.

Compact solar device on a post. LiFePO₄ is the better fit when size and life matter: a 3 - 5 Ah, 4-cell pack holds roughly the usable energy of a 9 Ah sealed lead-acid battery in less than half the space, and it lasts far longer under daily cycling. Sealed lead-acid is the better fit when price matters most or when frost is common. For LiFePO₄ the settings are JP3 off, JP7 closed, JP8 open; for sealed lead-acid on a post they are JP3 on, JP7 closed, JP8 open.

5. Charge current - J7

The BQ24650 regulates the charge current so that the voltage across its sense resistor is 40 mV: $I = 40 \text{ mV} / R_{\text{sense}}$. R49 (80 mΩ) is always in circuit and gives **0.5 A**. Each J7 jumper puts one more resistor in parallel with it. Each jumper bridges one vertical pair of J7 (a top-row letter A-E with the letter below it F-K); the top-row pins are all tied to the sense node, so a jumper simply connects that resistor in.

Position	Pins	Resistor added	Adds	Total with only this jumper
A-F	1-2	R51 (80 mΩ)	+0.50 A	1.00 A
B-G	3-4	R52 (40 mΩ)	+1.00 A	1.50 A
C-H	5-6	R53 (27 mΩ)	+1.48 A	1.98 A
D-I	7-8	R54 (20 mΩ)	+2.00 A	2.50 A
E-K	9-10	R55 (16 mΩ)	+2.50 A	3.00 A
none	-	R49 only (80 mΩ)	-	0.50 A

Rule: total current = 0.50 A + the sum of the “Adds” of every fitted jumper (exact: 40 mV x sum of 1/R). Any combination is allowed; all 32 are listed below, sorted by current. Amber rows exceed the 6 A average charge current in the board's own design note.

Current	Jumpers fitted	Current	Jumpers fitted
0.50 A	none	4.48 A	A-F + C-H + D-I
1.00 A	A-F	4.48 A	C-H + E-K
1.50 A	B-G	4.50 A	A-F + B-G + E-K
1.98 A	C-H	4.98 A	A-F + C-H + E-K
2.00 A	A-F + B-G	4.98 A	B-G + C-H + D-I
2.48 A	A-F + C-H	5.00 A	D-I + E-K
2.50 A	D-I	5.48 A	A-F + B-G + C-H + D-I
2.98 A	B-G + C-H	5.48 A	B-G + C-H + E-K
3.00 A	A-F + D-I	5.50 A	A-F + D-I + E-K
3.00 A	E-K	5.98 A	A-F + B-G + C-H + E-K
3.48 A	A-F + B-G + C-H	6.00 A	B-G + D-I + E-K
3.50 A	A-F + E-K	6.48 A	C-H + D-I + E-K
3.50 A	B-G + D-I	6.50 A	A-F + B-G + D-I + E-K
3.98 A	C-H + D-I	6.98 A	A-F + C-H + D-I + E-K
4.00 A	A-F + B-G + D-I	7.48 A	B-G + C-H + D-I + E-K
4.00 A	B-G + E-K	7.98 A	A-F + B-G + C-H + D-I + E-K

Limits

- **Per jumper:** each jumper carries 40 mV / R of its own resistor: 0.5, 1.0, 1.5, 2.0 and 2.5 A. The schematic note says the jumper limit is 3 A and to use 2.5 A maximum, so every single position is within it.
- **Input side (schematic note):** input 10 - 24 V DC; at least 0.5 A input to run the regulators; **6 A average** battery charge current maximum; 9 A maximum with the regulators running.
- **Precharge:** below the low-voltage threshold the charger uses 10% of the set current; the 10% figure is also the termination threshold in lithium mode.
- **Smaller panel:** the charger regulates the panel to 17.8 V (section 9). If the panel cannot supply the set current, the charge current simply drops; it does not collapse the panel.

Solar panel and battery size legend (silkscreen “Solar Charge Panel Battery Size”)

The legend on the board pairs each charge current with a recommended panel and battery size. The rule behind it is about **battery = 4 x charge current** (charge rate about C/4) and **panel watts = charge current x about 14 V**, rounded. The corrected PCB prints these rows, and the last column shows that each agrees with the netlist. Use the row for the battery size, fit the jumpers shown, and set JP3 and JP7/JP8 for the battery type (section 4.2).

J7 jumpers fitted	Printed charge	Panel	Battery	Netlist current
none	0.5 A	10 W	2 Ah	0.50 A
A-F	1.0 A	-	-	1.00 A
B-G	1.5 A	20 W	6 Ah	1.50 A
C-H	2.0 A	-	-	1.98 A
D-I	2.5 A	-	-	2.50 A
E-K	3.0 A	40 W	12 Ah	3.00 A
C-H + E-K	4.5 A	60 W	18 Ah	4.48 A
B-G + D-I + E-K	6.0 A	80 W	24 Ah	6.00 A

Highlighted rows carry a panel and battery size on the silkscreen. The 6.0 A row is at the 6 A average limit in the design note. Position letters: the top row of J7 is A B C D E and the bottom row is F G H I K (the letter J is skipped on the silkscreen).

Small panel (10 W). Leave J7 with **no jumpers (0.5 A)**, the silkscreen row for 10 W. A 10 W panel cannot put more than about 0.6 A into the battery (10 W at about 14 V, less conversion losses), so a higher setting would never be reached. Over a day it supplies roughly 2 - 3 Ah at 12 V (about 4 peak sun hours; an estimate that depends on the site), so keep the device's average current well under about 100 mA. A bigger battery only adds reserve for cloudy days, and recharges slowly.

If your boards carry the earlier legend. The earlier Build 3 PCB file printed E-K = 0.5 A (really 3.00 A), D-I = 1.5 A (really 2.50 A), C-H = 3.0 A (really 1.98 A), B-G = 4.5 A (really 1.50 A), A-F = 6.0 A (really 1.00 A) and A-B = 7.5 A (A and B are on the same net, so it does nothing). On such boards use the table above, not the printed text.

6. Watchdog - JP1 and J8

U4 is a TI TPL5010 timer used as a watchdog. It emits a **WAKE** pulse (to ESP32 GPIO25) at the end of every interval; the firmware must answer with a **DONE** pulse (from GPIO26). If it does not, the TPL5010 pulls **RSTn** low, which resets the ESP32 through JP1 and the EN pin.

Jumper	Setting	Effect
JP1	ON	TPL5010 RSTn is connected to ESP32 EN: a missed DONE resets the ESP32 (watchdog enabled)
JP1	OFF	RSTn is not connected: the timer still runs and pulses WAKE, but nothing is reset (watchdog disabled)

J8 position	Pins	Resistors to GND	REXT	Silkscreen	Calculated interval
L-Q	1-2	R23 124 k	124.00 kΩ	1 h	59.0 min
M-R	3-4	R35 182 k R45 187 k	92.23 kΩ	30 min	29.9 min
N-S	5-6	R44 124 k R21 107 k	57.44 kΩ	10 min	10.0 min
O-T	7-8	R12 22 k	22.00 kΩ	1 min	59.8 s
P-U	9-10	R22 5.1 k + R37 100 R	5.20 kΩ	1 s	1.0 s

- **Fit exactly one J8 jumper.** TI specifies REXT between 500 Ω and 170 kΩ; with no jumper the pin is open, and with two jumpers the resistors combine in parallel and give a shorter, unintended interval.
- **Power-cycle after moving a J8 jumper.** The TPL5010 reads REXT only once, at power-up.
- **Firmware contract:** after each WAKE rising edge, send a DONE low-to-high transition (at least 100 ns wide) at least 20 ms before the next WAKE. A missed DONE produces a reset pulse of up to 320 ms. A hung ESP32 is normally reset within one to two intervals.
- The J8 legend on the silkscreen matches the circuit in every position. The resistor values follow the examples in TI's Table 3; the 1 h setting (124 kΩ) is about 1 minute short of an hour.

7. Battery temperature - J5 thermistor

J5 (silkscreen "THERMISTOR") takes a **10 kΩ NTC** on the battery. That node feeds the BQ24650 TS pin (through R28, 100 Ω) and the ADS1115 channel AIN0, so the ESP32 can also read the temperature. It sits in a divider: R26 5.23 kΩ to 3.3 V and R27 30.1 kΩ to GND. The schematic recommends the LCSC parts **C5200575** and **C394023**.

Situation	TS voltage	Result
No thermistor fitted	85% of VREF (2.81 V)	Above the 73.5% cold threshold: charging is suspended (both status LEDs off)
10 k NTC at 25 °C	59% of VREF	Inside the 47.5% - 73.5% window: charging allowed
Fixed 10 k resistor instead of an NTC	59% of VREF	Always inside the window: charging always allowed, temperature protection is lost

With a 10 k NTC of beta about 3435 K (103AT-2 type) the charging window works out to roughly **0.5 °C to 41 °C**. Charging stops above about 41 °C battery temperature, which matters for outdoor installations in hot climates. This is a calculation from the datasheet thresholds and an assumed beta, so treat it as approximate.

8. Charger status - STAT LEDs and ESP32 read-out

STAT1 (U7 pin 3) and STAT2 (U7 pin 5) are open-drain outputs, active low. Each drives one LED (CT-1608UGC-P4) that is fed from the charger's 3.3 V VREF output through a 470 Ω resistor, roughly 2 to 3 mA when lit. **D10 ("IN PROGRESS") is on STAT1 and D8 ("COMPLETE") is on STAT2.**

Status table (BQ24650 datasheet, Table 2)

Charge state	STAT1 / D10 IN PROGRESS	STAT2 / D8 COMPLETE
Charge in progress (precharge, fast charge, constant voltage)	ON (lit)	off
Charge complete	off	ON (lit)
Charge suspend, over-voltage, sleep (no valid input), battery absent, faults such as the 30-minute precharge timeout or thermistor out of range	off	off

- There is **no blinking fault code** on the BQ24650: every fault or suspend condition shows as both LEDs off.
- In lead-acid mode (JP3 on) termination is disabled, so "COMPLETE" should not light; infer a full battery from the battery current instead (INA219 at 0x41).
- Both LEDs off in daylight means the charger is suspended: panel below the 17.8 V set point, battery absent, thermistor missing or out of range (section 7), or a precharge timeout.

Reading the status from the ESP32 (already wired in this revision)

The schematic connects the LED-anode nodes to the ADS1115 (U10, I²C address 0x48, powered from 3.3 V): **AIN2 = stat1** (node between R14 and D10) and **AIN3 = stat2** (node between R13 and D8). The nodes never exceed VREF (3.3 V), so they are within the ADS1115 input limit (VDD + 0.3 V).

Voltage on AIN2 / AIN3	Meaning
about 0 V	Charger has no input power (its VREF output is off); status unknown
about 2 V (LED forward voltage + STAT low level)	STAT asserted (LED lit)
about 3.3 V	STAT released (LED off), charger powered

AIN2 (stat1)	AIN3 (stat2)	Decoded state
asserted	released	Charging
released	asserted	Charge complete
released	released	Suspended / fault / battery absent / sleep with input present
about 0 V	about 0 V	No charger input power (night, panel disconnected)

Firmware notes. (1) Configure the ADS1115 for ±4.096 V full scale; the default ±2.048 V saturates at these levels. (2) Suggested thresholds are about 0.5 V (unpowered) and about 2.8 V (released), but the asserted level depends on the LED's forward voltage, so measure it on a real board and tune. If the LED's forward voltage is close to 3 V, the asserted and released levels would be hard to separate; check the LED datasheet. The two nodes are read as single-ended inputs and can be sampled slowly.

9. Fixed parameters (not jumper-selectable)

Item	Value	Where
Solar input range	10 - 24 V DC (schematic note)	J1 "+SOLAR INPUT-"
Solar regulation (MPPT) point	17.8 V = 1.2 V x (1 + 499 k / 36 k). Charger disabled if MPPSET < 75 mV, enabled above 175 mV. The panel's maximum-power voltage should be at or above about 17.8 V (typical 12 V / 36-cell panel).	R36, R47 on U7 MPPSET
Battery connection	J6 "+BATTERY-"; a diode stops the battery discharging into the panel when the panel voltage is lower	charger_batt sheet, D6
Power bus	Battery and panel are joined through diodes; the higher voltage supplies the system (VDD). VDD feeds two TPS5430 buck regulators: 5.0 V (V50) and 3.3 V (3V3).	reg5v0, reg3v3 sheets
Input current sensor	INA219 U8, address 0x40 , 2 mΩ shunt R9	lin sheet
Battery current sensor	INA219 U9, address 0x41 , 2 mΩ shunt R10	bateria sheet
ADC	ADS1115 U10, address 0x48 : AIN0 thermistor, AIN1 RTC coin-cell voltage (R41 1 MΩ / R46 2 MΩ), AIN2 stat1, AIN3 stat2	Valentino sheet
Other devices	PCF8563 RTC + CR1220 holder; DS18B20 on GPIO27 (R40 4.7 k pull-up); LoRa RA-02 on SPI (SCK 14, MISO 12, MOSI 13, NSS 15, RST 4, DIO 35); I ² C on GPIO21 (SDA) / GPIO22 (SCL)	rtc, tempera, lora sheets
Reset / programming	SW1 resets (EN); J2 pins: V_PROG (5 V, via D2 to V50), EN, TX0, RX0, IO0, GND x3	Valentino sheet

Side headers J3 / J4 (1 x 10 female)

Pin	J3 (left edge)	J4 (right edge)
1	GPIO32	GPIO5 (CS)
2	GPIO33	GPIO18 (CLK)
3	GPIO16 (UART2 RX)	GPIO23 (MOSI)
4	GPIO17 (UART2 TX)	GPIO19 (MISO)
5	3V3	SDA (GPIO21)
6	not connected	SCL (GPIO22)
7	VDD (battery/panel bus)	3V3
8	VDD (battery/panel bus)	V50 (5 V)
9	GND	GND
10	GND	GND

10. Shield rule: no through-hole parts above the tall capacitors

Five 220 uF electrolytic capacitors (C14, C26, C38, C4, C34) stand **10 mm** tall in a row along the bottom of the board. The PCB carries this note on its user layer, quoted exactly:

Separacion de PCB de 8.0+1.6 = 9.6mm
Altura de componentes de 10mm, capacitores
No poner otro PCB en esta parte

Translation: PCB separation 8.0 + 1.6 = 9.6 mm. Component height 10 mm, capacitors. Do not put another PCB in this part.

The rule

On a shield stacked on Valentino, **over the strip the five tall capacitors occupy there must be no through-hole components without header raisers**. The strip is 12.6 to 24.87 mm above the bottom edge of the PCB and 30.1 to 106.9 mm from the left edge (12.3 mm x 76.8 mm). Their lead tips protrude below the shield and would touch the 10 mm capacitors underneath; header raisers are the option that lifts a through-hole part clear.

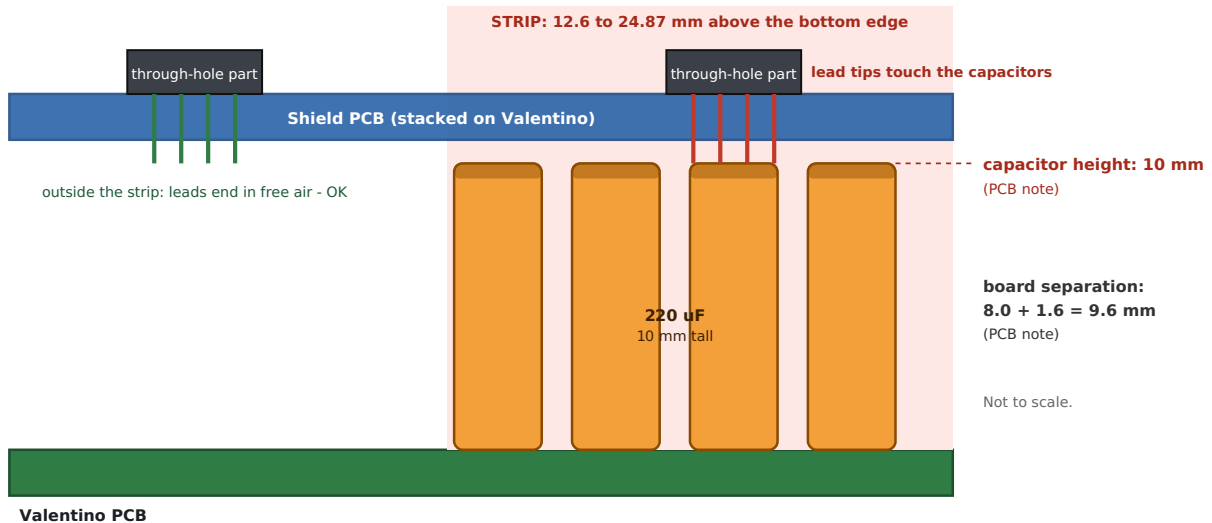


Figure 2. Side view of the stack, schematic and not to scale. Outside the strip a through-hole part is fine because its leads end in free air. Over the strip the leads land on the capacitors.

- **The strip** is the area the five capacitors occupy (the hatched area in Figure 1): **12.6 to 24.87 mm above the bottom edge** (12.3 mm tall) and **30.1 to 106.9 mm from the left edge** (76.8 mm long), taken from the capacitor outlines. 24.87 mm is also the PCB's own dimension mark for the top of the capacitor row.
- **Clearance:** the note gives a board separation of 9.6 mm (8.0 mm standoff + 1.6 mm PCB) against capacitors 10 mm tall, so there is no clearance under the shield over the strip. Anything that projects below the shield here lands on the capacitors: through-hole leads, and any tall part.
- The bottom 12.6 mm of the board is not part of the strip, so through-hole parts there, such as terminal blocks, do not touch the capacitors.
- The side headers J3 and J4 (left and right edges, well above the strip) are the natural mating points for a shield.

Ref	Part	Centre, from left edge (mm)	Centre, from bottom edge (mm)
C14	220 uF electrolytic, 10 mm tall	34.6	18.8
C26	220 uF electrolytic, 10 mm tall	62.8	18.8
C38	220 uF electrolytic, 10 mm tall	74.7	18.8
C4	220 uF electrolytic, 10 mm tall	88.2	18.8
C34	220 uF electrolytic, 10 mm tall	102.4	18.8

The strip is measured from the bottom-left corner with the board seen from the component side (as in Figure 1). Positions are calculated from the PCB file.

11. Shield outline, notches and capacitor strip

A shield is a second board that plugs onto Valentino through J3 and J4 and covers its lower part. Valentino's tall capacitors and its bottom-edge terminal blocks fix the shield's shape. Figure 3 shows the shield outline (blue) with Valentino's parts underneath it (grey).

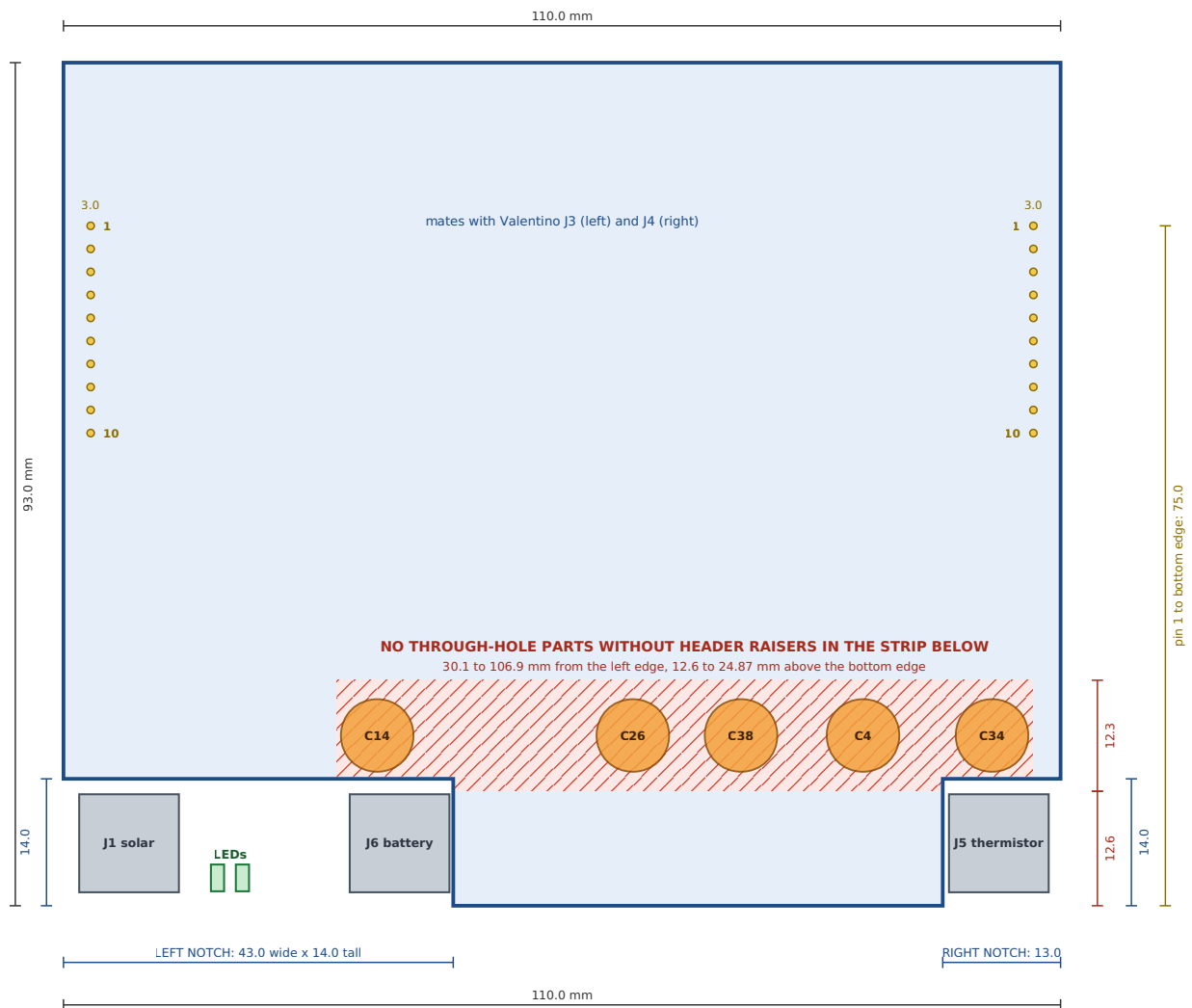


Figure 3. Shield outline seen from above, to scale, in millimetres. The shield covers the lower 93 mm of Valentino with the bottom and side edges flush. Hatched = the strip over the tall capacitors, where the shield must carry no through-hole components without header raisers.

Item	Dimension	Why
Outline	110.0 mm x 93.0 mm rectangle	Same width as Valentino (110 mm). It covers the lower 93 mm of Valentino's 120 mm, so the top 27.0 mm stays open (ESP32 and antenna, J2, JP1, J8, U4, SW1).
Position	Bottom and side edges flush with Valentino	Pin 1 of the mating headers is 75.0 mm above the bottom edge on both boards.
Mating headers	2 x 1x10, 2.54 mm pitch, 3.0 mm in from each side edge; pin 1 at 18.0 mm from the shield top (75.0 mm above the bottom edge), pin 10 at 40.9 mm from the top	Plugs into Valentino J3 (left) and J4 (right); pin 1 at the top on both.
Left notch	43.0 mm wide x 14.0 mm tall , bottom-left corner	Exposes the solar terminal J1 (1.7 - 12.7 mm from the left edge) and the battery terminal J6 (31.6 - 42.6 mm from the left edge), leaving only 0.4 mm beyond J6. It also exposes the status LEDs D8 and D10 (16.3 - 20.5 mm from the left edge). Minimum width to clear J6 is 42.6 mm, so do not make this notch narrower.
Right notch	13.0 mm wide x 14.0 mm tall , bottom-right corner	Exposes the thermistor terminal J5 (1.3 - 12.3 mm from the right edge), leaving 0.7 mm. Minimum size to clear J5 is 12.3 mm each way.
Notch depth	14.0 mm from the bottom edge	The three terminal blocks occupy 1.5 - 12.3 mm above the bottom edge, so 14.0 mm leaves 1.7 mm.
No through-hole strip	12.6 to 24.87 mm above the bottom edge (68.13 - 80.4 mm from the shield top) and 30.1 to 106.9 mm from the left edge	Lead tips of through-hole parts would touch the 10 mm capacitors (section 10); header raisers are the exception.

The 43.0 mm left notch, the 13.0 mm right notch and the 14.0 mm depth are the values in the ValentinoShieldTemplate KiCad project. Measured against the terminal outlines in the Valentino PCB file they clear J6 by 0.4 mm, J5 by 0.7 mm and the tops of the terminal blocks by 1.7 mm. The outline is 110 mm wide like Valentino and 93 mm tall.

Access with the shield fitted

- **Covered by the shield:** J7 (charge current), JP7, JP8 (charge voltage) and JP3 (chemistry). Set them **before** fitting the shield, or cut windows in it.
- **Still reachable:** JP1 and J8 (watchdog), the J2 programming header and SW1 reset (all in the top 27 mm), the LEDs D8/D10 and terminals J1, J6, J5 (through the notches).

12. Firmware starting point: ValentinoTest

ValentinoTest is the starting point for every new device built on a Valentino base board and a shield. It contains only the Valentino hardware: RTC, both current monitors, the ADS1115 (thermistor, coin cell, charger status), the watchdog, the chemistry jumper, the board temperature sensor and LoRa. Anything shield-specific goes into three empty hook functions.

Item	Value
Sketch	<code>~/Data/DigitalStables/Projects/ValentinoTest/ValentinoTest.ino</code> (single file, about 440 lines)
Repository	https://github.com/arifainchtein/ValentinoTest
Board	<code>esp32:esp32:esp32da</code> , ESP32 core 3.3.7, sketchbook <code>ProjectsSupport/Esp32ArduinoLib</code>
Build check	Compiles with <code>arduino-cli 0.35.3</code> : 1,020,414 bytes of flash (77%) and 49,796 bytes of RAM (15%). It has not been run on a board yet.
Serial	115200 baud. Commands: Help, Status, Scan, SetTime, WdtStop, LoraTest. A report prints every 5 s.

12.1 Libraries

Library	Used for
DigitalStablesEsp32Lib	PCF8563TimeManager (read and set the RTC, 1 Hz output), TimeUtils (timezone), GeneralFunctions (state-of-charge curves), SolarInfo and BatteryPowerManager (chemistry, state of charge, sleep and comma voltages)
Adafruit_INA219 1.2.3	Solar input monitor (0x40) and battery monitor (0x41)
ADS1X15 0.5.2	ADS1115 at 0x48
OneWire 2.3.8, Dallas	DS18B20 board temperature
arduino-LoRa 0.8.0	RA-02 (SX1278) at 433 MHz

12.2 What changed since Build 1

Firmware written for Valentino Build 1 needs these changes on Build 3:

Item	Build 1	Build 3 (ValentinoTest)
Chemistry pin, GPIO39	LOW = LiFePO4, HIGH = lead-acid	JP3 open = HIGH = LiFePO4 ; JP3 fitted = LOW = lead-acid. Polarity inverted.
RTC coin-cell voltage	GPIO36 (SENSOR_VP) with analogRead through the R41/R46 divider	ADS1115 AIN1; GPIO36 is no longer connected
Charger status	not readable	STAT1 / STAT2 on ADS1115 AIN2 / AIN3
Battery temperature	no thermistor input	Thermistor on ADS1115 AIN0 (and BQ24650 TS)
RTC 1 Hz output, GPIO34	no pull-up on the board	R15 10 k pull-up to 3.3 V added
Charger IC	LM25085 (non-synchronous buck)	BQ24650. Input current is still pulsed, so the 20-sample average is kept.
ADS1115	none on the base board	U10 at 0x48 on the base board; a shield ADC at 0x49 does not clash

12.3 Pins and addresses a shield can use

Used by Valentino	Free on the side headers
GPIO21 / 22: I ² C (shared with the shield) GPIO25 / 26: watchdog WAKE / DONE GPIO27: DS18B20 GPIO34: RTC 1 Hz GPIO39: JP3 chemistry GPIO14, 12, 13, 15, 4, 35: LoRa (SCK, MISO, MOSI, SS, RESET, DIO0) GPIO0, TX0, RX0, EN: programming header J2	J3: GPIO32, GPIO33, GPIO16 (UART2 RX), GPIO17 (UART2 TX) J4: GPIO5, GPIO18, GPIO23, GPIO19 (the default SPI pins, free because LoRa uses its own) Power on the headers: 3V3, V50 (5 V), VDD (battery/panel bus), GND I²C addresses in use: 0x40, 0x41, 0x48, 0x51

12.4 The code, block by block

The listing below is the sketch file in order, cut at the `// ==== SECTION ====` markers, with a short note before each block.

Includes and pins

Pin names follow the Valentino net names. GPIO39 is JP3, not the Build 1 power-mux sense pin, and the RA-02 is on its own SPI pins.

```
// ValentinoTest - starting point for any device built on a Valentino base board plus a shield.
// Board: esp32:esp32:esp32da          Valentino: Build 3 (see Valentino_Board_Configuration_Guide.pdf)
// Serial: 115200 baud, send "Help" for the command list.

#include <Arduino.h>
#include <Wire.h>
#include <SPI.h>
#include <LoRa.h>
#include <OneWire.h>
```

```

#include <DallasTemperature.h>
#include <Adafruit_INA219.h>
#include "ADS1X15.h"
#include <PCF8563TimeManager.h>
#include <TimeUtils.h>
#include <GeneralFunctions.h>
#include <SolarInfo.h>
#include <BatteryPowerManager.h>

#define SDA_PIN 21
#define SCL_PIN 22

#define RTC_CLK_OUT 34      // PCF8563 1 Hz output, input-only pin; R15 10k pull-up to 3V3 (Build 1 had none)
#define TEMPERATURE 27     // DS18B20 board temperature, OneWire, R40 4.7k pull-up
#define WATCHDOG_WAKE 25  // TPL5010 -> ESP32, pulse at the end of every watchdog interval
#define WATCHDOG_DONE 26  // ESP32 -> TPL5010, rising edge = "I am alive"
#define CHEMISTRY_PIN 39   // JP3 (SENSOR_VN): R16 10k pull-up to 3V3, jumper to GND

#define LORA_SCK 14         // RA-02 is on its own SPI pins, not the default VSPI pins
#define LORA_MISO 12
#define LORA_MOSI 13
#define LORA_SS 15
#define LORA_RESET 4
#define LORA_DIO0 35

```

I²C addresses and ADS1115 channels

The four ADS1115 channels are fixed by the Build 3 schematic. The gain must be set to ± 4.096 V: the default ± 2.048 V saturates on the 3.3 V status signals.

```

#define INA219_SOLAR_ADDR 0x40    // U8, solar input current (R9 2 mOhm)
#define INA219_BATTERY_ADDR 0x41 // U9, battery current (R10 2 mOhm)
#define ADS1115_ADDR 0x48        // U10, ADDR pin to GND
#define PCF8563_ADDR 0x51        // U3, real-time clock

#define ADS_CH_THERMISTOR 0       // J5 NTC divider, also feeds the BQ24650 TS pin
#define ADS_CH_RTC_BATT 1        // CR1220 coin cell through the R41/R46 divider
#define ADS_CH_STAT1 2           // BQ24650 STAT1, LED D10 "IN PROGRESS"
#define ADS_CH_STAT2 3           // BQ24650 STAT2, LED D8 "COMPLETE"
#define ADS_GAIN_4V096 1         // full scale  $\pm 4.096$  V; the default  $\pm 2.048$  V would clip 3.3 V signals

```

Calibration constants

The INA219 library assumes a 0.1 Ω shunt, so current is calculated from the shunt voltage and the real 2 m Ω resistors. The thermistor constants come from the divider on the schematic (section 7); `NTC_BETA` is an assumption until the thermistor datasheet is checked. The two status thresholds are estimates to be tuned on a real board (section 8). The two battery voltage limits drive the over-voltage warning (section 4.2). The battery capacity, site position and timezone are placeholders.

```

#define SHUNT_OHMS 0.002f        // R9 and R10; mA = mV / ohms
#define SOLAR_SAMPLES 20         // the buck converter draws pulsed input current, average several samples

#define RAIL_3V3 3.3f
#define NTC_PULLUP_OHMS 5230.0f // R26 from 3V3 to the thermistor node
#define NTC_PARALLEL_OHMS 30100.0f // R27 from the thermistor node to GND
#define NTC_NOMINAL_OHMS 10000.0f // thermistor resistance at 25 C
#define NTC_BETA 3435.0f         // ASSUMED (103AT-2 type): take it from the datasheet of the thermistor you fit
#define NTC_MISSING_VOLTS 2.75f // no thermistor in J5 reads about 2.81 V

#define RTC_BATT_R1 1000000.0f   // R41
#define RTC_BATT_R2 2000000.0f   // R46

#define STAT_UNPOWERED_MAX_VOLTS 0.5f // VREF is off: the charger has no input power
#define STAT_ASSERTED_MAX_VOLTS 2.8f  // LED lit: LED forward voltage plus STAT low level; measure and tune

#define LIFEP04_MAX_VOLTS 14.6f    // 4 cells at 3.65 V; JP7 and JP8 both closed charge to 14.9 V
#define LEAD_ACID_MAX_VOLTS 15.0f  // just above the highest JP7/JP8 setting (14.9 V)

#define BATTERY_CAPACITY_MAH 7000.0f // PLACEHOLDER: usable capacity of the battery you fit
#define LATITUDE -37.8136           // PLACEHOLDER site position, only needed to construct SolarInfo
#define LONGITUDE 144.9631
#define ALTITUDE 30.0
#define DEFAULT_TIMEZONE "AEST-10AEDT,M10.1.0,M4.1.0/3"

#define REPORT_SECONDS 5

```

Types, objects and state

`ValentinoReadings` collects everything one report needs. The interrupt flags are `volatile`; the ADS1115 and both INA219s share the one Wire bus.

```

enum StatLevel { STAT_UNPOWERED, STAT_ASSERTED, STAT_RELEASED };
enum ChargerState { CHARGER_NO_INPUT, CHARGER_CHARGING, CHARGER_COMPLETE, CHARGER_SUSPENDED, CHARGER_INVALID };

struct ValentinoReadings {
    float solarVolts;
    float solarMa;
    float batteryVolts;
    float batteryMa;
    float rtcBatteryVolts;
    bool thermistorPresent;
    float thermistorC;
    float stat1Volts;
    float stat2Volts;
    ChargerState charger;
    float boardC;
    uint8_t stateOfCharge;
    bool batteryOverVoltage;
};

PCF8563TimeManager timeManager(Serial);
Adafruit_INA219 solarMonitor(INA219_SOLAR_ADDR);
Adafruit_INA219 batteryMonitor(INA219_BATTERY_ADDR);
ADS1115 ads(ADS1115_ADDR);
OneWire oneWire(TEMPERATURE);
DallasTemperature boardTempSensor(&oneWire);
SolarInfo solarInfo(Serial, LATITUDE, LONGITUDE, ALTITUDE);
BatteryPowerManager* batteryManager = nullptr;
BatteryChemistry currentChemistry = CHEM_LEAD_ACID;

bool foundSolar = false, foundBattery = false, foundAds = false, foundRtc = false, foundBoardTemp = false, loraActive = false;

volatile bool clockTicked = false;
volatile bool watchdogWake = false;
volatile uint32_t watchdogWakeCount = 0;
volatile uint32_t rtcTickCount = 0;
bool watchdogPetting = true;
uint32_t lastWakeMs = 0;
uint32_t watchdogIntervalMs = 0;
uint32_t lastTickMs = 0;
uint32_t seconds = 0;
uint32_t loraPacketCount = 0;
String commandBuffer;

```

Watchdog and RTC tick interrupts

The TPL5010 contract (section 6): after each WAKE pulse the firmware answers with a DONE pulse, well before the next one. The loop measures the time between WAKE pulses, so the report shows the interval that J8 actually selects. `WdtStop` stops the petting; with JP1 fitted the board then resets within one to two intervals.

```

void IRAM_ATTR clockTick() {
    clockTicked = true;
    rtcTickCount = rtcTickCount + 1;
}

void IRAM_ATTR onWatchdogWake() {
    watchdogWake = true;
    watchdogWakeCount = watchdogWakeCount + 1;
}

void petWatchdog() {
    digitalWrite(WATCHDOG_DONE, HIGH);
    delayMicroseconds(100);
    digitalWrite(WATCHDOG_DONE, LOW);
}

void serviceWatchdog() {
    if (!watchdogWake) return;
    watchdogWake = false;
    uint32_t now = millis();
    if (lastWakeMs != 0) watchdogIntervalMs = now - lastWakeMs;
    lastWakeMs = now;
    if (watchdogPetting) petWatchdog();
}

```

I²C helpers

The general-call reset clears devices left in a bad state after a brown-out. The scan lists everything on the bus and names the four Valentino devices, so anything else is the shield.

```

void resetI2CDevices() {
    Wire.beginTransmission(0x00);
    Wire.write(0x06);
}

```

```

Wire.endTransmission();
delay(10);
}

bool i2cPresent(uint8_t address) {
    Wire.beginTransmission(address);
    return Wire.endTransmission() == 0;
}

void i2cScan() {
    Serial.println("I2C scan:");
    for (uint8_t address = 1; address < 127; address++) {
        if (!i2cPresent(address)) continue;
        Serial.printf(" 0x%02X", address);
        if (address == INA219_SOLAR_ADDR) Serial.print(" INA219 solar input");
        else if (address == INA219_BATTERY_ADDR) Serial.print(" INA219 battery");
        else if (address == ADS1115_ADDR) Serial.print(" ADS1115");
        else if (address == PCF8563_ADDR) Serial.print(" PCF8563 RTC");
        else Serial.print(" (shield or other)");
        Serial.println();
    }
}

```

ADS1115 readings and status decoding

Four conversions from one chip: the thermistor node is turned into a temperature with the divider and beta equation (a node voltage above 2.75 V means no thermistor is fitted); the coin-cell node is scaled by 1.5 for the R41/R46 divider; the two STAT nodes are classified as unpowered, asserted or released and decoded with the table in section 8.

```

float readAdsVolts(uint8_t channel) {
    return ads.toVoltage(ads.readADC(channel));
}

float thermistorCelsius(float nodeVolts, bool& present) {
    present = nodeVolts < NTC_MISSING_VOLTS;
    if (!present) return NAN;
    float parallelOhms = NTC_PULLUP_OHMS * nodeVolts / (RAIL_3V3 - nodeVolts);
    float ntcOhms = 1.0f / (1.0f / parallelOhms - 1.0f / NTC_PARALLEL_OHMS);
    return 1.0f / (1.0f / 298.15f + logf(ntcOhms / NTC_NOMINAL_OHMS) / NTC_BETA) - 273.15f;
}

float rtcBatteryVolts(float nodeVolts) {
    return nodeVolts * (RTC_BATT_R1 + RTC_BATT_R2) / RTC_BATT_R2;
}

StatLevel classifyStat(float volts) {
    if (volts < STAT_UNPOWERED_MAX_VOLTS) return STAT_UNPOWERED;
    if (volts < STAT_ASSERTED_MAX_VOLTS) return STAT_ASSERTED;
    return STAT_RELEASED;
}

ChargerState decodeCharger(StatLevel stat1, StatLevel stat2) {
    if (stat1 == STAT_UNPOWERED && stat2 == STAT_UNPOWERED) return CHARGER_NO_INPUT;
    if (stat1 == STAT_ASSERTED && stat2 == STAT_RELEASED) return CHARGER_CHARGING;
    if (stat1 == STAT_RELEASED && stat2 == STAT_ASSERTED) return CHARGER_COMPLETE;
    if (stat1 == STAT_RELEASED && stat2 == STAT_RELEASED) return CHARGER_SUSPENDED;
    return CHARGER_INVALID;
}

const char* chargerText(ChargerState state) {
    switch (state) {
        case CHARGER_NO_INPUT: return "no input power";
        case CHARGER_CHARGING: return "charging";
        case CHARGER_COMPLETE: return "charge complete";
        case CHARGER_SUSPENDED: return "suspended / fault / no battery";
        default: return "invalid";
    }
}

```

Battery chemistry

Build 3 polarity, opposite to Build 1. The chemistry is read on every report and the `BatteryPowerManager` is rebuilt if the jumper changes, so its sleep and comma voltages always match the fitted battery. The ESP32 cannot read JP7 and JP8, so `batteryMaxVolts()` gives the highest voltage the detected chemistry allows and the report warns when the battery is above it.

```

// Build 3 polarity: JP3 open = LiFePO4 (pin HIGH), JP3 fitted = lead-acid (pin LOW).
// Build 1 firmware used the opposite polarity (LOW = LiFePO4); do not reuse its test on Build 3.
BatteryChemistry detectChemistry() {
    return digitalRead(CHEMISTRY_PIN) == HIGH ? CHEM_LIFEP04 : CHEM_LEAD_ACID;
}

```

```

void refreshBatteryManager() {
    BatteryChemistry chemistry = detectChemistry();
    if (batteryManager != nullptr && chemistry == currentChemistry) return;
    delete batteryManager;
    batteryManager = new BatteryPowerManager(chemistry, BATTERY_CAPACITY_MAH, solarInfo);
    currentChemistry = chemistry;
}

// JP7 and JP8 cannot be read by the ESP32, so the only way to catch a wrong setting is to compare
// the battery voltage with the highest voltage the detected chemistry allows.
float batteryMaxVolts() {
    return currentChemistry == CHEM_LIFEP04 ? LIFEP04_MAX_VOLTS : LEAD_ACID_MAX_VOLTS;
}

```

RTC integrity

The PCF8563 sets bit 7 of its seconds register when its voltage dropped or the oscillator stopped. Time read while it is set is not trustworthy, so the report says so.

```

bool rtcClockIntegrity() {
    Wire.beginTransmission(PCF8563_ADDR);
    Wire.write(0x02);
    Wire.endTransmission();
    Wire.requestFrom(PCF8563_ADDR, 1);
    return (Wire.read() & 0x80) == 0;
}

```

Reading all the sensors

The solar input current of a buck converter is pulsed, so twenty samples are averaged. The battery current sign follows the INA219 wiring, so check it on a real Build 3 board.

```

void readValentino(ValentinoReadings& r) {
    if (foundSolar) {
        float busSum = 0, shuntSum = 0;
        for (uint8_t i = 0; i < SOLAR_SAMPLES; i++) {
            busSum += solarMonitor.getBusVoltage_V();
            shuntSum += solarMonitor.getShuntVoltage_mV();
            delay(1);
        }
        r.solarVolts = busSum / SOLAR_SAMPLES;
        r.solarMa = (shuntSum / SOLAR_SAMPLES) / SHUNT_OHMS;
    }
    if (foundBattery) {
        r.batteryVolts = batteryMonitor.getBusVoltage_V();
        r.batteryMa = batteryMonitor.getShuntVoltage_mV() / SHUNT_OHMS;
    }
    if (foundAds) {
        r.thermistorC = thermistorCelsius(readAdsVolts(ADS_CH_THERMISTOR), r.thermistorPresent);
        r.rtcBatteryVolts = rtcBatteryVolts(readAdsVolts(ADS_CH_RTC_BATT));
        r.stat1Volts = readAdsVolts(ADS_CH_STAT1);
        r.stat2Volts = readAdsVolts(ADS_CH_STAT2);
        r.charger = decodeCharger(classifyStat(r.stat1Volts), classifyStat(r.stat2Volts));
    }
    if (foundBoardTemp) {
        boardTempSensor.requestTemperatures();
        r.boardC = boardTempSensor.getTempCByIndex(0);
    }
    refreshBatteryManager();
    r.stateOfCharge = batteryManager->stateOfCharge(r.batteryVolts);
    r.batteryOverVoltage = foundBattery && r.batteryVolts > batteryMaxVolts();
}

```

LoRa

Initialisation uses 433 MHz, spreading factor 9, 125 kHz, CRC, 5 dBm. The sketch transmits only when you send `LoraTest`; connect the antenna first.

```

void initLoRa() {
    SPI.begin(LORA_SCK, LORA_MISO, LORA_MOSI);
    pinMode(LORA_SS, OUTPUT);
    pinMode(LORA_RESET, OUTPUT);
    pinMode(LORA_DIO0, INPUT);
    digitalWrite(LORA_SS, HIGH);
    LoRa.setPins(LORA_SS, LORA_RESET, LORA_DIO0);
    if (!LoRa.begin(433E6)) return;
    LoRa.setSPIFrequency(1000000);
    LoRa.setTxPower(5);
    LoRa.setSpreadingFactor(9);
    LoRa.setSignalBandwidth(125E3);
    LoRa.enableCrc();
    loraActive = true;
}

```

```

}

void loraTest(const ValentinoReadings& r) {
  if (!loraActive) {
    Serial.println("LoRa not initialised");
    return;
  }
  LoRa.beginPacket();
  LoRa.print("ValentinoTest#");
  LoRa.print(++loraPacketCount);
  LoRa.print("#");
  LoRa.print(r.batteryVolts, 2);
  LoRa.endPacket();
  Serial.printf("LoRa packet %lu sent (connect the antenna before using this)\n", (unsigned long)loraPacketCount);
}

```

Shield hooks

Everything specific to a shield goes here: `shieldSetup()` once at boot, `shieldLoop()` on every pass of the loop, `shieldReport()` at the end of each report.

```

void shieldSetup() {
}

void shieldLoop() {
}

void shieldReport() {
}

```

Report and serial commands

One block of text every five seconds gives a complete picture of the base board. Commands are read a line at a time without blocking. `SetTime#DD#MM#YY#DOW#HH#MM#SS` uses the library's own format.

```

void printReport(const ValentinoReadings& r) {
  RTCInfoRecord t = timeManager.now();
  Serial.println("-----");
  Serial.printf("time      %04u-%02u-%02u %02u:%02u:%02u epoch %ld clock %s\n", t.year, t.month, t.date, t.hour,
    t.minute, t.second,
    t.epoch, rtcClockIntegrity() ? "ok" : "NOT TRUSTED (set the time)");
  Serial.printf("battery   %s  %.2f V  %.0f mA  SoC %u%%  sleep<%.1f V  comma<%.1f V\n",
    currentChemistry == CHEM_LIFEP04 ? "LiFeP04" : "lead-acid", r.batteryVolts, r.batteryMa,
    r.stateOfCharge,
    batteryManager->sleepingVoltage(), batteryManager->commaVoltage());
  if (r.batteryOverVoltage) Serial.printf("WARNING  battery is above %.1f V, the limit for %s: check JP7/JP8\n",
    batteryMaxVolts(),
    currentChemistry == CHEM_LIFEP04 ? "LiFeP04" : "lead-acid");
  Serial.printf("solar     %.2f V  %.0f mA  %.2f W\n", r.solarVolts, r.solarMa, r.solarVolts * r.solarMa / 1000.0f);
  Serial.printf("charger   %s  (STAT1 %.2f V, STAT2 %.2f V)\n", chargerText(r.charger), r.stat1Volts, r.stat2Volts);
  if (r.thermistorPresent) Serial.printf("thermistor %.1f C\n", r.thermistorC);
  else Serial.println("thermistor NOT FITTED (charging is suspended without it)");
  Serial.printf("rtc cell   %.2f V\n", r.rtcBatteryVolts);
  Serial.printf("board      %.1f C\n", r.boardC);
  Serial.printf("watchdog   interval %lu ms  wakes %lu  petting %s\n", (unsigned long)watchdogIntervalMs, (unsigned
    long)watchdogWakeCount,
    watchdogPetting ? "yes" : "NO (reset expected)");
  Serial.printf("rtc ticks %lu  lora %s\n", (unsigned long)rtcTickCount, loraActive ? "ok" : "not found");
  shieldReport();
}

void printHelp() {
  Serial.println("Help          this list");
  Serial.println("Status        print a report now");
  Serial.println("Scan          scan the I2C bus");
  Serial.println("SetTime#DD#MM#YY#DOW#HH#MM#SS  set the RTC, e.g. SetTime#24#10#26#6#17#32#00");
  Serial.println("WdtStop       stop petting the watchdog (board resets if JP1 is fitted)");
  Serial.println("LoraTest      send one LoRa packet (antenna must be connected)");
}

void handleCommand(String command) {
  command.trim();
  if (command.length() == 0) return;
  ValentinoReadings r = {};
  if (command == "Help") printHelp();
  else if (command == "Scan") i2cScan();
  else if (command == "Status") {
    readValentino(r);
    printReport(r);
  } else if (command.startsWith("SetTime#")) {
    timeManager.setTime(command);
    Serial.println();
  } else if (command == "WdtStop") {
    watchdogPetting = false;
  }
}

```

```

    Serial.println("Watchdog petting stopped");
} else if (command == "LoraTest") {
    readValentino(r);
    loraTest(r);
} else Serial.println("Unknown command, send Help");
}

void readSerial() {
    while (Serial.available()) {
        char c = Serial.read();
        if (c == '\n') {
            handleCommand(commandBuffer);
            commandBuffer = "";
        } else if (c != '\r') commandBuffer += c;
    }
}

```

Setup and loop

Setup arms the watchdog interrupt first, then brings up I²C, the RTC, the sensors and LoRa, and prints which devices were found. The loop never blocks for long: the watchdog is serviced first, then the serial port, the shield, and finally the once-a-second RTC tick (with a millisecond fallback if the RTC is missing).

```

void setup() {
    Serial.begin(115200);
    delay(100);
    Serial.println("ValentinoTest starting");

    pinMode(WATCHDOG_DONE, OUTPUT);
    digitalWrite(WATCHDOG_DONE, LOW);
    pinMode(WATCHDOG_WAKE, INPUT);
    attachInterrupt(digitalPinToInterrupt(WATCHDOG_WAKE), onWatchdogWake, RISING);
    pinMode(CHEMISTRY_PIN, INPUT);

    Wire.begin(SDA_PIN, SCL_PIN);
    resetI2CDevices();
    i2cScan();

    TimeUtils::parseTimezone(DEFAULT_TIMEZONE);
    foundRtc = i2cPresent(PCF8563_ADDR);
    timeManager.start();
    timeManager.PCF8563osc1Hz();
    pinMode(RTC_CLK_OUT, INPUT);
    attachInterrupt(digitalPinToInterrupt(RTC_CLK_OUT), clockTick, RISING);

    foundSolar = solarMonitor.begin();
    foundBattery = batteryMonitor.begin();
    foundAds = ads.begin();
    ads.setGain(ADS_GAIN_4V096);
    Wire.setClock(400000);
    boardTempSensor.begin();
    foundBoardTemp = boardTempSensor.getDeviceCount() > 0;
    initLoRa();

    Serial.printf("RTC %s, solar INA219 %s, battery INA219 %s, ADS1115 %s, DS18B20 %s, LoRa %s\n", foundRtc ? "ok" :
"MISSING",
        foundSolar ? "ok" : "MISSING", foundBattery ? "ok" : "MISSING", foundAds ? "ok" : "MISSING",
        foundBoardTemp ? "ok" : "MISSING", loraActive ? "ok" : "MISSING");

    shieldSetup();
    printHelp();
}

void loop() {
    serviceWatchdog();
    readSerial();
    shieldLoop();

    if (clockTicked || millis() - lastTickMs >= 1100) {
        clockTicked = false;
        lastTickMs = millis();
        seconds++;
        if (seconds % REPORT_SECONDS == 0) {
            ValentinoReadings r = {};
            readValentino(r);
            printReport(r);
        }
    }
}

```

12.5 Starting a new shield project

- Copy the ValentinoTest folder to `Projects/<NewDevice>` and rename the .ino to match.

- Put shield code in `shieldSetup`, `shieldLoop` and `shieldReport`. Leave `serviceWatchdog()` at the top of the loop, and never block for longer than the watchdog interval.
- Use only the free GPIOs and I²C addresses in section 12.3, and build the shield to the outline, notches and capacitor strip in section 11.
- Replace the placeholders: battery capacity, site position and timezone.
- Before relying on it, measure the STAT levels, check the battery current sign, confirm the thermistor beta and run `WdtStop` once to prove the watchdog resets the board.

13. Open points to check

#	Item	Suggested action
1	Which PCB file the boards were made from (sections 4 and 5): the earlier Build 3 file has wrong J7 and JP7/JP8 legends	If the boards carry the earlier legends, use the tables in this guide; the corrected PCB (2026-09-23) prints them correctly
2	STAT voltage levels on AIN2 / AIN3 are estimates (section 8)	Measure the released, asserted and unpowered levels on a board and set the firmware thresholds
3	Lead-acid mode holds the chosen voltage indefinitely (sections 3 and 4.2)	Use 13.6 V for standby use; for daily solar cycling the 14.2 V setting is held indefinitely, so check the battery datasheet allows it
4	Thermistor window is about 0.5 to 41 °C (section 7)	Confirm it suits the site; the hot limit is low for outdoor use
5	The thermistor divider (R26) is fed from the 3V3 rail, while the BQ24650 thresholds are ratios of its own VREF	Both are nominally 3.3 V; a small difference shifts the window by a few degrees
6	The keep-out strip is taken from the capacitor outlines: 12.6 - 24.87 mm above the bottom edge, 30.1 - 106.9 mm from the left edge (section 10)	Confirm the extents; add a small margin at each end if the shield has wide pads
7	Jumpers J7, JP7, JP8, JP3 are under a fitted shield (section 11)	Set them before fitting, or add windows to the shield
8	ValentinoTest has been compiled but not run on a board (section 12)	Run it once; check STAT levels, battery current sign, thermistor beta and the <code>WdtStop</code> reset

Appendix A. Original Spanish notes and translations

Where	Original	Translation
PCB user layer	Separacion de PCB de $8.0+1.6 = 9.6\text{mm}$ / Altura de componentes de 10mm, capacitores / No poner otro PCB en esta parte	PCB separation $8.0 + 1.6 = 9.6\text{ mm}$ / Component height 10 mm, capacitors / Do not put another PCB in this part
Valentino sheet (JP3)	Sin Jumper, baterias con litio / Con Jumper, bateria acido plomo / Si configura al cargador	Without jumper: lithium batteries / With jumper: lead-acid battery / It configures the charger
charger_batt (J7)	La corriente limite de los jumper es 3A / Usar maximo de 2.5A por seguridad	The jumper current limit is 3 A / Use 2.5 A maximum for safety
charger_batt (LEDs)	I de led 5mA maximo	LED current 5 mA maximum
charger_batt	Diodo impide descarga de bateria cuando el voltaje del panel es menor al de la bateria	Diode prevents battery discharge when the panel voltage is lower than the battery voltage
charger_batt	VCCQ = Voltaje de salida del cargador / VCC = Voltaje de entrada del panel solar, despues de medir corriente que entra / VDIR = Voltaje de entrada del panel, para directo por diodo	VCCQ = charger output voltage / VCC = solar panel input voltage after the input current measurement / VDIR = panel input voltage, direct through a diode
lin sheet	Vin de 10Vdc a 24Vdc / Corriente minima de entrada 0.5A, para operacion de regulador principal y circuitos sin reguladores externos activos / Corriente media para carga de bateria maxima de 6 A / Corriente maxima para carga de bateria y reguladores activos 9A	Vin 10 to 24 V DC / Minimum input current 0.5 A, for the main regulator and circuits with no external regulators active / Average battery charge current maximum 6 A / Maximum current for battery charging with regulators active 9 A
Valentino sheet	Se unen voltaje de bateria y voltaje de panel solar / Se tienen diodos en cada salida / Se toma la corriente de quien tenga mas voltaje	Battery and solar-panel voltages are joined / There are diodes on each output / Current is taken from whichever has the higher voltage
bateria sheet	VDD = salida de voltaje de bateria, por diodo evita retorno de corriente	VDD = battery voltage output, through a diode that prevents reverse current
Valentino sheet	Recomendaciones para termistor	Thermistor recommendations (LCSC C5200575, C394023)