Indice

1 Panoramica 3
1.1 Caratteristiche 3
2 Alimentazione 3
2.1 Pannello interruttori 4
2.2 Inversione di polarità 4
2.3 Protezione ESD 4
2.4 Filtri EMI 4
2.5 Buck - Step Down Converter 5
3 Soft Latch Power On 5
3.1 Principio di funzionamento 6
3.2 Schema semplificato 6
3.3 Componenti su scheda 7
3.4 Tempo di decadimento 7
3.5 Note Firmware 8
4 USB-C Connector 8
4.1 Protezione ESD 8
4.2 Schema semplificato 9
4.3 Componenti su scheda 9
5 Elettrovalvola 10
5.1 Alimentazione 10
5.2 Logica di controllo 10
5.3 Schema semplificato 10
5.4 Componenti su scheda 12
5.5 Raccomandazioni di layout 12
5.6 Note Firmware 12
6 MCU 13
6.1 Alimentazione e filtraggio 14
6.2 Interfaccia USB-C e programmazione 14
6.3 GPIO mapping 14
6.4 Circuito di reset e boot 16
6.4.1 Pin EN — Reset della MCU 16
6.4.2 Pin GPIO0 — Modalità di boot 17
6.4.3 Schema semplificato reset/boot 17
6.4.4 Sequenze di boot e programmazione 18
6.4.5 Note firmware - Controllo reset e boot 19
6.5 Raccomandazioni di layout 20
6.6 Firmware OTA — Aggiornamento via Wi-Fi 21
6.6.1 Partition table su ESP-IDF 22
6.6.2 Partition table su PlatformIO 22
6.6.3 Partition table su Arduino IDE 22
6.6.4 Esempio di aggiornamento OTA via HTTPS 22
6.7 Flash integrata 26
6.7.1 Partition table per 4 MB di flash 27
6.7.2 Partition table per 8 MB di flash 27
6.7.3 Partition table per 16 MB di flash 28
7 Display 28
7.1 Caratteristiche principali 28
7.2 Pin e segnali su scheda 29
7.3 Alimentazione e retroilluminazione 29
7.4 Note firmware 29

8 Interfaccia utente 34
8.1 Pulsante di accensione del core di sistema 35
8.2 Pulsanti di navigazione 35
8.3 LED di stato 35
9 Real Time Clock - DS3231 36
9.1 Schema semplificato 36
10 Sensori 37
11 Schema 37
12 PCB 41
13 Distinta materiali (BOM) 42
14 Immagini 3D scheda 45

Alina Vista

1 Panoramica

Alina Vista è una centralina elettronica autonoma progettata per la gestione intelligente dell'irrigazione in piccoli impianti, balconi, serre e coltivazioni urbane. È pensata per funzionare in modo completamente **offline**, senza app o connessione di rete, offrendo invece un'interfaccia fisica immediata e affidabile basata su:

- un display grafico Nokia 5110 (PCD8544),
- un pannello di comandi dedicato con pulsanti e interruttori,
- un sistema di menu intuitivo.

La centralina è in grado di leggere sensori ambientali, controllare un'elettrovalvola a 12V e visualizzare informazioni in tempo reale, mantenendo un approccio semplice e robusto.

Alina Vista è composta di una scheda di controllo stand-alone basata su ESP32-S3-WROOM-1, progettata per applicazioni di monitoraggio e automazione locale. Opera principalmente tramite un display integrato e sensori ambientali, ed è alimentata direttamente da una linea a 12 V (con protezioni e conversione integrata a 3.3 V). Prevede il controllo di attuatori come elettrovalvole e il supporto per display grafici a basso consumo (Nokia 5110).

Pur operando in modo totalmente autonomo, la scheda è predisposta per funzionalità aggiuntive come il Wi-Fi o l'aggiornamento firmware OTA, attivabili in versioni future del firmware.

La scheda è stata pensata sia per utenti finali sia per sviluppatori che desiderano estendere o adattare il firmware grazie a una struttura modulare e documentata.

1.1 Caratteristiche

- Alimentazione 12V con protezione anti-inversione e filtraggio.
- Micro-controller ESP32-S3 WROOM-1 alimentato tramite buck sincrono.
- Controllo per elettrovalvola esterna a 12V
- Supporto per display LCD PCD8544.
- Connettività dei sensori ambientali: AM2302/DHT22, sensore capacitivo di umidità del suolo, DS3231.
- Porta USB-C solo dati per programmazione e debug (alimentazione tramite 12V).
- Soft-latch su linea EN per attivazione tramite pulsante e automantenimento via firmware.
- Possibilità di utilizzo connettività WiFi/BLE e aggiornamento firmware OTA.

2 Alimentazione

La scheda **Alina Vista** prevede un connettore jack **J1** per alimentazione da sorgente esterna a 12V. Per motivi di sicurezza e affidabilità la sezione di alimentazione integra:

- Protezione da inversione di polarità.
- Protezione da sovratensioni transitorie ed ESD di potenza.
- Conversione step-down 12 V → 3.3 V ad alta efficienza.
- Filtri EMI per la soppressione del rumore.

2.1 Pannello interruttori

La scheda prevede un connettore **J8** che espone 8 terminali per un pannello esterno o interrutti cablati:

- **GEN_1** e **GEN_2**: interruttore generale (ON/OFF) in serie alla linea 12 V di ingresso.
- **FUSE_1** e **FUSE_2**: portafusibile in serie a monte della scheda. Consigliato **2A** *(max. 3A)*.
- **GND** e **VALVE_ON**: pulsante momentaneo (push) per attivazione manuale valvola *(opzionale)*.
- **VALVE_1** e **VALVE_2**: interruttore generale valvola 12 V. In OFF disabilita la valvola a prescindere da firmware e pulsante.

2.2 Inversione di polarità

La protezione da inversione di polarità è fornita alla scheda tramite l'implementazione del MOSFET P-channel **AO4407A (U5)** *(in configurazione diodo ideale)*, montato a monte della linea 12V protetta *(12V_PROT)* e a valle del fusibile esterno del pannello degli interruttori. L'utilizzo del mosfet permette di proteggere la scheda da inversioni di polarità sul connettore jack 12V di ingresso alimentazione alla scheda, salvando potenzialmente anche il fusibile, ed evitando cadute significative di tensione . Il gate del mosfet è protetto dal diodo **1N4742 (D5**) che regola tensione G-S. L'uscita di questo circuito di protezione, fornisce alimentazione in ingresso sia al buck converter che all'elettrovalvola, pertanto sono presenti due condensatori, **C1** da *(si consiglia)* **100μF (polarizzato smd)** e **C2** da *(si consiglia)* **100nF**. Nella realizzazione del pcb è fortemente consigliato posizionare i due condensatori **molto vicino** sia al mosfet che alla **diramazione**, in modo da ottenere un serbatoio lento e da contrastare gli spike di linea.

2.3 Protezione ESD

Al **drain** del mosfet *(diodo ideale)* **AO4407A (U5)** è collegato un **TVS** unidirezionale **SMBJ15A** *(consigliato)* (catodo a +12V_PROT, anodo a GND), il quale, grazie anche alla sua velocità di azione, protegge tutta l'elettronica a valle da scariche ad alta energia, ESD e spike.

2.4 Filtri EMI

La protezione contro le interferenze elettromagnetiche è realizzata in due diffenti punti della scheda tramite semplici filtri LC:

1. Ingresso buck (12 V → VIN AP63203). Questo filtro in serie, si trova all'ingresso del buck sincrono, sulla tensione in uscita dalla diramazione del circuito di protezione contro inversione di polarità *(12V_PROT)*. Il filtro è realizzato con:

- **FB1**: ferrite bead da **120Ω@100MHz** 4A *(consigliato)*
- **C3**: 22μF
- **C4**: 100nF

2. Rail 3V3 → MCU. Questo filtro si trova vicino all'ingresso dell'alimentazione della MCU, dopo l'uscita del buck sincrono sul **rail 3V3**. Il filtro è così composto:

   - **FB2**: ferrite bead da **120Ω@100MHz** 4A *(consigliato)*
   - **C5**: 100μF *(elettrolitico - smd)*
   - **C6**: 10μF *(elettrolitico - smd)*
   - **C7**: 100nF

2.5 Buck - Step Down Converter

Per poter alimentare la MCU è necessario convertire la tensione di ingresso alla scheda a 3.3V. Questa tensione è ottenuta grazie al **AP63203WU** che permette di avere una buona efficienza e quindi bassa dispersione di calore. Il AP63203WU è un **buck sincrono** che trasforma la tensione di ingresso in una tensione regolata **fissata** a **3.3V** (fSW ∼ 1.1 MHz). Il circuito è composto da:

- I condensatori in paralallelo all'ingresso del buck, da posizionare in prossimità dei pin IN e GND dell'IC:
  - **C3**: 22μF X7R *(low ESR)*
  - **C4**: 100nF
- L'induttanza **L1** da **4.7μH** ***(4.5A - min. 2A)*** *(consigliato)* che collega il pin **SW** all'uscita 3V3 e al pin **FB**.
- Un condensatore di bootstrap **C16** da **100nF** tra i pin **BST** e **SW**.
- I condensatori di uscita, sempre in parallelo, tra GND ed uscita 3V3 (uscita nodo L1):
  - **C17**: 22μF X7R *(low ESR)*
  - **C18**: 22μF X7R *(low ESR)*

> Il circuito prevede anche la gestione del pin **EN** *(Enable)* tramite un circuito di soft latch che vedremo nella sezione seguente *(Vedi 3 Soft Latch Power On)*.

L'**AP63203WU**, da datasheet, richiede un layout di questo tipo, con condensatori molto vicini al circuito integrato:

3 Soft Latch Power On

La scheda **Alina Vista** adotta un circuito di accensione controllata ("**soft-latch**") per gestire l'alimentazione del regolatore **AP63203WU** tramite il pin **EN** (Enable). Questo permette di accendere la centralina con un semplice **pulsante momentaneo** e di mantenere l'alimentazione stabile tramite un **GPIO** della MCU, eliminando la necessità di un interruttore meccanico a tenuta.

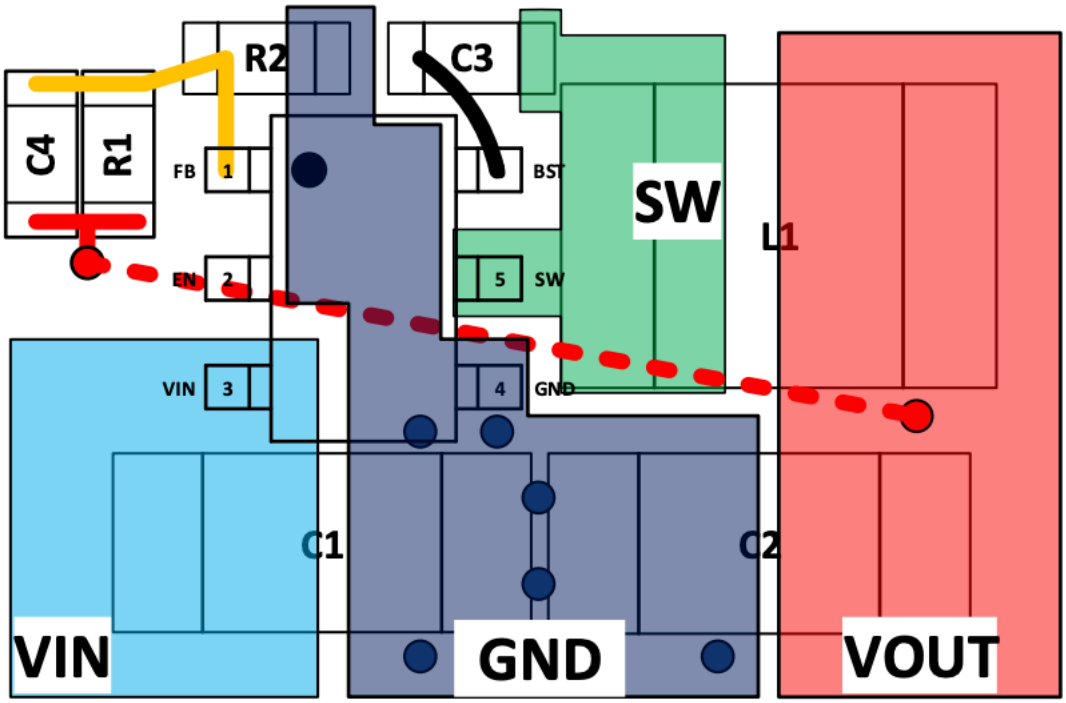

Figura 1: Layout AP63203WU

3.1 Principio di funzionamento

All'accensione, la pressione del pulsante fornisce un impulso al pin **EN** del buck, portandolo alto (HIGH) per un intervallo sufficiente a consentire l'avvio dell'ESP32. Il firmware, una volta inizializzato, deve impostare il **GPIO38** in stato **alto** per mantenere il pin EN attivo anche dopo il rilascio del pulsante. Quando il firmware imposta il GPIO a basso, il buck si disattiva e l'intera scheda si spegne ordinatamente.

3.2 Schema semplificato

Come si vede dall'immagine in Figura 2 il pin **EN** del buck è controllato dal pin **GPIO38** della MCU.

- Il diodo $\mathbf{D_{IS}}$ è necessario per evitare che la tensione da 12 V entri nel pin digitale della MCU durante la pressione del pulsante **Push** momentaneo rappresentato in figura.
- Il condensatore $\mathbf{C_{EN}}$ è il ***condensatore di temporizzazione*** ed è necessario per permettere alla MCU di accendersi ed impostare il **GPIO38** alto. Si consiglia un valore compreso tra **4.7μF** e **10μF**. Per il calcolo vedi 3.4 Tempo di decadimento.
- La resistenza da $\mathbf{R_{LIM}}$, non è strettamente obbligatoria, ma protegge il GPIO limitando la corrente di carica del condensatore di temporizzazione quando è scarico e il GPIO38 viene impostato alto.
- La resistenza $\mathbf{R_{PD}}$ è la resistenza di pull-down, che mantiene il buck spento ed evita che il pin EN resti "flottante".
- Il condensatore $\mathbf{C_F}$ ha il compito di filtrare i disturbi *(opzionale)*.

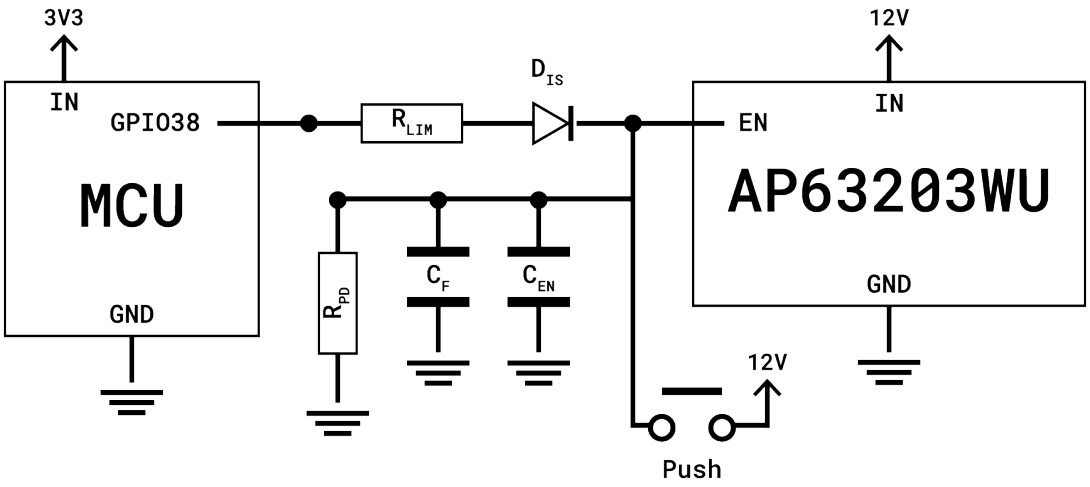

Figura 2: Schema circuito di Soft Latch

> Il pin **EN** ha un intervallo di tensione ampio (-0.3V $\rightarrow$ 35V), e ha una ***Threshold (ON)*** di **1.18V** e una ***Threshold (OFF)*** di **1.1V** pertanto può accettare, ed attivarsi, sia con la tensione di 12V (alla pressione del pulsante) che con la tensione di 3.3V del GPIO.

3.3 Componenti su scheda

Riferimento	Scheda	Valore consigliato
R_{LIM}	**R20**	1k
D_{IS}	**D6**	1N4148
R_{PD}	**R2**	100k
C_F	**C14**	100nF
C_{EN}	**C15**	4.7μF

> **Nota bene** Sulla scheda non è previsto alcuno switch ***"Push"***, poichè la scheda fornisce due terminali di uscita riservati al collegamento di uno switch esterno per il soft latch. Per maggiori informazioni vedi Interfaccia comandi.

3.4 Tempo di decadimento

Il tempo di decadimento è l'intervallo durante il quale la tensione sul pin EN rimane sopra la soglia di accensione dopo che hai rilasciato il pulsante. In pratica, indica quanto a lungo il condensatore C_{EN} riesce a mantenere il pin EN alto prima che si scarichi attraverso la resistenza R_{PD}.

Quando viene rilasciato il pulsante, il condensatore inizia a scaricarsi con costante di tempo:

La tensione residua su EN nel tempo è:

Dove:

- $\mathbf{V_{IN}}$ è la tensione di alimentazione ($\approx$ 12 V)
- **T** è la costante di tempo in secondi

Il tempo di decadimento utile (fino alla soglia di spegnimento, $\approx$ 1.18 V) si calcola con:

Dove V_{TH} rappresente la tensione di **Threshold ON**.

Per R_{PD} = 100 kΩ, C_{EN} = 4.7 μF e V_{IN} = 12 V, il **tempo di decadimento** sarà:

In questo caso che si avrà circa 1 secondo di margine perché l'MCU completi il boot e imposti il GPIO alto. Per diminuire il tempo di decadimento è necessario impostare un valore di C_{EN} minore; Per aumentarlo, invece serve aumentare C_{EN} o R_{PD}.

3.5 Note Firmware

Come intuito dalle sezioni precedenti, è strettamente necessario impostare dal firmware il **GPIO38** con stato **HIGH** il prima possibile nel firmware in modo da mantenere il dispositivo acceso non appena viene premuto il pulsante momentaneo di accensione *(esterno alla scheda)*

```
/*Arduino (core ESP32)*/
const int EN_HOLD = 38;

void setup() {
  pinMode(EN_HOLD, OUTPUT);
  digitalWrite(EN_HOLD, HIGH);
}
```

4 USB-C Connector

La scheda espone un **connettore USB-C** (***USB type C 2.0 16P Receptacle***). Il connettore non ha funzione di alimentazione che deve essere comunque fornita tramite il **jack 12V** di ingresso alimentazione presente sulla scheda, che fornisce alimentazione a tutto il circuito. L'unico scopo del connettore **USB-C** è quello di sfruttare le ***linee dati*** per poter ***programmare la MCU***. Lato firmware è possibile implementare anche una la modalità **update firmware OTA (Over-the-air)**, ma per la prima programmazione della scheda resta necessario utilizzare il connettore **USB-C**, salvo poi anche poter smontare il connettore in uno step successivo.

4.1 Protezione ESD

La porta **USB-C** della scheda **Alina Vista** è protetta da scariche elettrostatiche (ESD) e disturbi transitori tramite un array di diodi di protezione **SRV05-4** (**U3**). Questo componente è un **Transient Voltage Suppressor (TVS)** a 4 linee, specificamente progettato per interfacce ad alta velocità come

USB 2.0, in grado di deviare scariche elettrostatiche fino a ± 15 kV (IEC 61000-4-2) con capacità parassita molto bassa (tip. 0.8 pF).

4.2 Schema semplificato

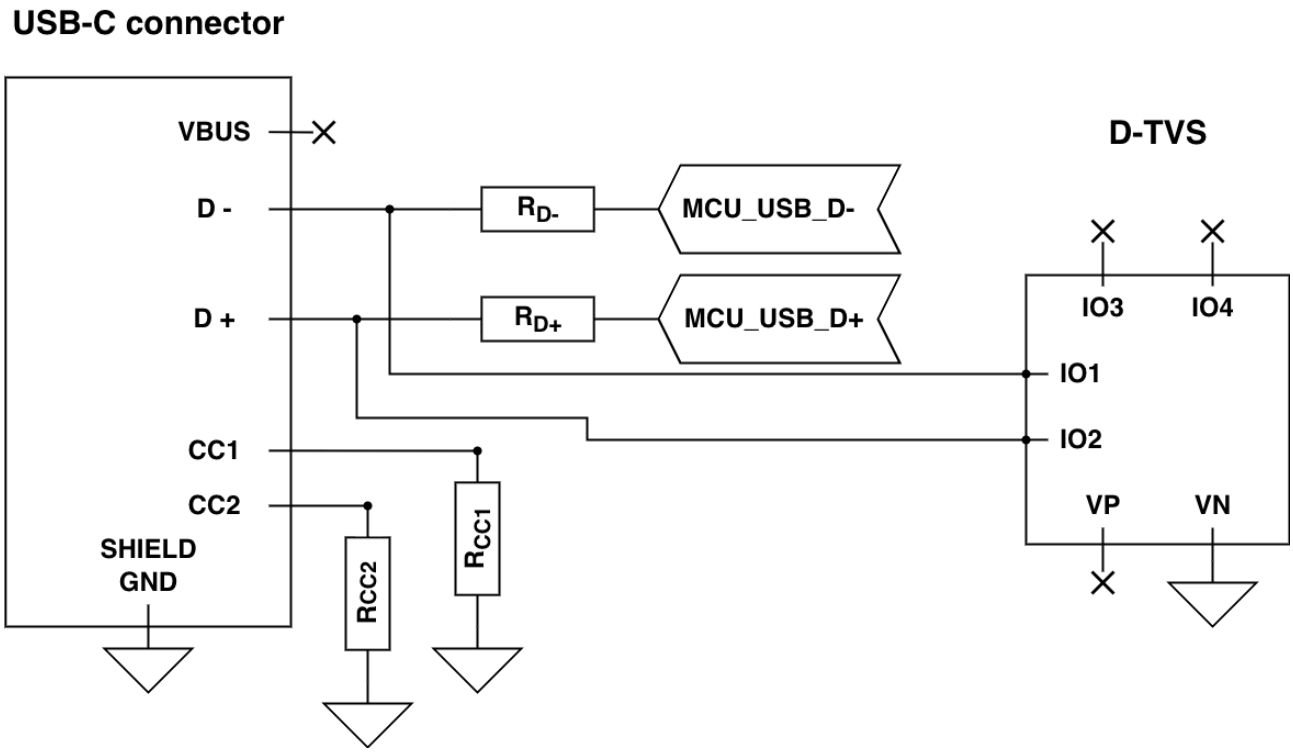

Figura 3: Schema circuito USB

Come si vede dal circuito in figura, il connettore USB-C ha il pin **VBUS non connesso**. Pertanto anche l'array **D-TVS** è connesso in configurazione con i pin **VN** a **GND** e **VP NC**. Mentre per quanto riguarda i pin "**IO**" del **D-TVS** sono utilizzati solo **IO1** e **IO2** collegati direttamente ai dati del connettore, rispettivamente **D-** e **D+**.

- **R_{D-}** e **R_{D+}** sono le resistenze in serie sulle linee dati **D-** e **D+** che vanno ai corrispondenti pin **USB_D-/+** della **MCU**. Questi due resistori vanno posizionati fisicamente vicino ai pin della MCU.
- **R_{CC1}** e **R_{CC2}** sono le resistenze di configurazione ed hanno il compito di portare i canali di configurazione del connettore usb-c a GND (tipicamente 5.1 KΩ)

> Nel layout del pcb il **D-TVS** deve trovarsi il più **vicino** possibile **al connettore**.

4.3 Componenti su scheda

Riferimento	Scheda	Valore consigliato
D-TVS	**U3**	SRV05-4
R_{CC1}	**R5**	5.1 kΩ

Riferimento	Scheda	Valore consigliato
R_{CC2}	**R6**	5.1 kΩ
R_{D-}	**R7**	22 Ω
R_{D+}	**R8**	22 Ω

5 Elettrovalvola

La scheda **Alina Vista** è progettata per pilotare un'elettrovalvola a solenoide *(bobina)* a **12 V DC** con corrente nominale fino a **0.5 A**. Il circuito di comando è stato realizzato per garantire isolamento, protezione contro le sovratensioni induttive e compatibilità diretta con le uscite logiche a 3.3 V della MCU.

5.1 Alimentazione

L'elettrovalvola viene alimentata a **12 V**, tramite il **rail** denominato **12V_PROT** *(vedi sez. 2)*, tramite la sua diramazione dedicata alla sezione della valvola. L'**alimentazione generale 12V** dell'elettrovalvola può essere staccata o attaccata grazie al connettore J8 che prevede tra gli altri i terminali per la gestione dell'elettrovalvola (vedi Pannello Interruttori).

5.2 Logica di controllo

1. Mettere su ON l'interruttore generale di alimentazione della valvola.
2. Per l'attivazione manuale della valvola si utilizza il pulsante momentaneo collegato ai pin **5** e **6** del connettore **J8**.
3. Per controllare la valvola tramite firmware (digitalmente), bisogna agire sul **GPIO4** (HIGH/LOW).

5.3 Schema semplificato

- J_{EV}: **Connettore dell'elettrovalvola**. *Nota tecnica: In caso di cavi lunghi (> 1 m) verso l'elettrovalvola, si consiglia di montare un ulteriore diodo flyback o un piccolo TVS da 18 V direttamente sul connettore della valvola, per sopprimere eventuali spike induttivi lontani dal PCB.*
- D_{FB}: **Diodo di flyback** collegato in ***anti-parallelo*** rispetto alla tensione di alimentazione dell'elettrovalvola, per sopprimere la tensione induttiva generata all'apertura del MOSFET. Questo componente garantisce protezione efficace contro gli spike di ritorno tipici di un carico induttivo, evitando danni sia al MOSFET che al regolatore 12 V.
- C_S: **Condensatore di stabilizzazione**, necessario per stabilizzare la tensione e ridurre il ripple. È collegato in parallelo alla valvola.
- C_D: **Condensatore di disaccoppiamento** in parallelo alla valvola.
- **Q**: **MOSFET N-Channel di potenza logica**. Il gate è controllato dal segnale del pin digitale **GPIO4** della **MCU**. Il segnale HIGH (3.3 V) porta il gate del MOSFET in conduzione, chiudendo il circuito di massa e attivando la

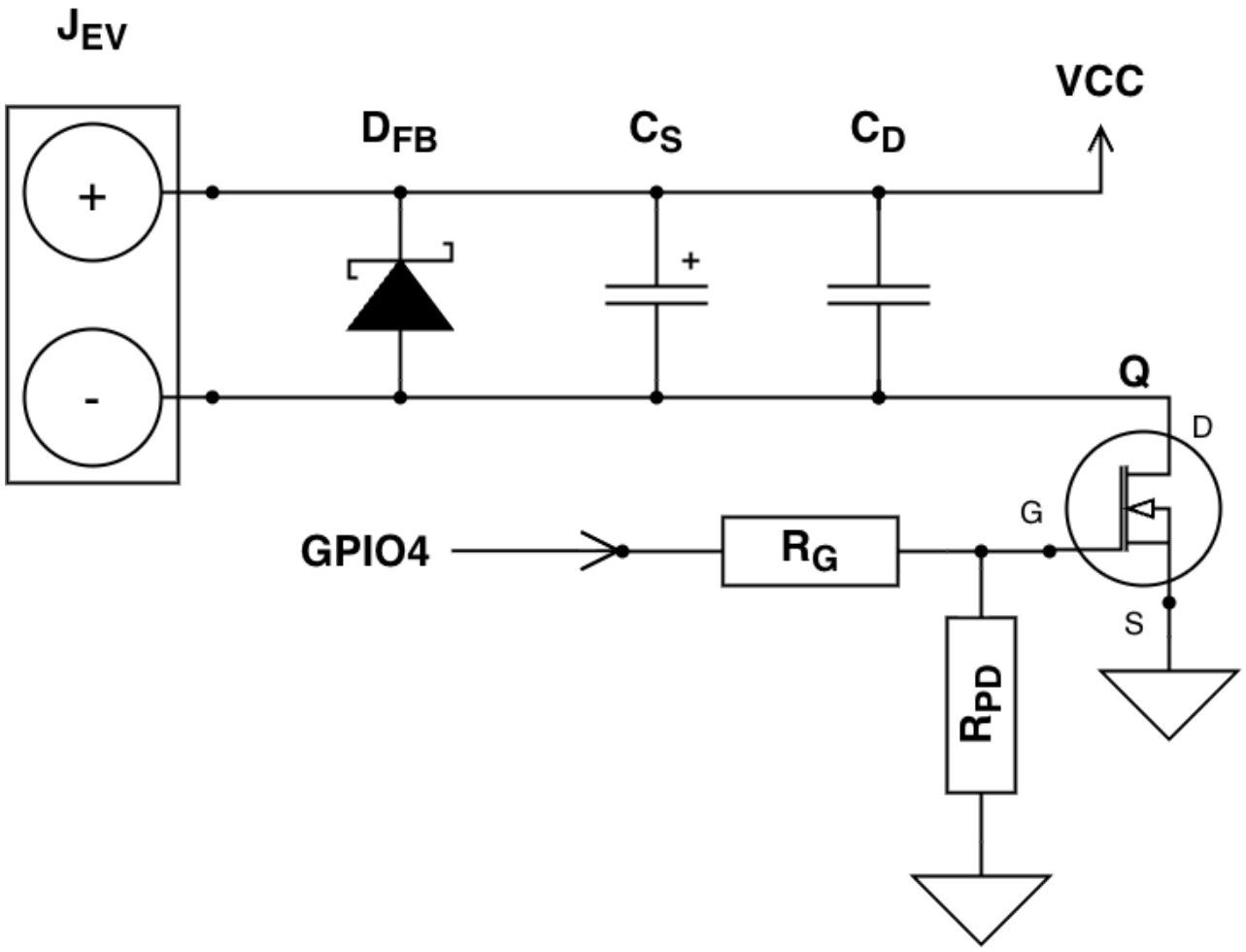

Figura 4: Schema circuito Valvola

valvola. Il segnale LOW la disattiva. *Nota tecnica:* Il MOSFET scelto deve avere una $V_{GS(th)} \leq 2$ V, per garantire piena conduzione anche con logica 3.3 V, e a bassa $R_{DS(on)}$.
- **R_G: Resistenza di gate.** Limite corrente di carica del gate.
- **R_{PD}: Resistenza di pulldown.** Mantiene il gate a 0V in idle.

5.4 Componenti su scheda

Riferimento	Scheda	Valore consigliato
J_{EV}	**J3**	Conn. 01x02
D_{FB}	**D2**	SS34
C_D	**C9**	100 nF
C_S	**C10**	100 uF
Q	**Q1**	IRLZ44N
R_{PD}	**R9**	100 kΩ
R_G	**R10**	220 Ω

5.5 Raccomandazioni di layout

- Posizionare **Q1** e **D2 vicino al connettore dell'elettrovalvola** per minimizzare induttanze parassite.
- Tenere separate le masse di potenza e di segnale, unendole in un punto comune ("stella") vicino al MOSFET.
- Usare piste da almeno **1.5 mm** per la corrente della valvola (0.5 A @ 12 V).
- Posizionare i condensatori **C9** e **C10** molto vicini al connettore **J3** o al drain del MOSFET **Q1**.
- Inserire una **ferrite bead** opzionale sul filo positivo della valvola se si riscontrano interferenze EMI.

> *Nota: Il diodo Schottky SS34 è adatto per correnti fino a 3 A e presenta tempi di recupero molto rapidi (< 50 ns), risultando sufficiente per elettrovalvole da 500 mA, anche con cablaggi di qualche decina di centimetri.*

5.6 Note Firmware

Per gestire la valvola via firmware basta gestire il pin digitale **GPIO4**. È possibile anche implementare un timeout per gestirle lo spegnimento della valvola.

```
/*Arduino (core ESP32)*/

/*
 *  Questo codice prevede l'utilizzo di una valvola a solenoide NO (normalmente aper
 *  L'esempio utilizza FreeRTOS, ma è possibile utilizzare anche soltanto la libreri
 *  senza utilizzare task.
```

```
 */

#include <Arduino.h>
#include <atomic>
#include "freertos/FreeRTOS.h"
#include "freertos/timers.h"

constexpr int VALVE_PIN = 4;
constexpr unsigned long valveTimerMillis = 10000;   // timer valvola ON in ms
std::atomic<bool> valveOn{false}; // flag valvola esterno al task

void valveTask (void*) {
    pinMode(VALVE, OUTPUT);
    digitalWrite(VALVE, LOW);
    bool busy = false;  // flag per evitare sovrapposizione dei cicli.

    for (;;) {
        if (!busy && valveOn.load(std::memory_order_acquire)) {
            busy = true;
            digitalWrite(VALVE_PIN, HIGH);
            vTaskDelay(pdMS_TO_TICKS(valveTimerMillis));
            digitalWrite(VALVE_PIN, LOW);
            busy = false;
        }
        vTaskDelay(pdMS_TO_TICKS(5)); // libera la CPU per circa 5 ms (esegue altri
    }
}

void setup() {
  xTaskCreatePinnedToCore(
    valveTask,  // funzione che implementa il task
    "Valve",
    3072,   // dimensione stack del task in words
    nullptr,    // parametro di input del task
    2,  // priorità
    nullptr,    // handle del task
    APP_CPU_NUM // core app (tip. 1)
  );
}
```

6 MCU

Il cuore della scheda **Alina Vista** è costituito dal modulo **ESP32-S3-WROOM-1** prodotto da Espressif Systems. Si tratta di un microcontrollore a **dual-core Xtensa LX7** con architettura **RISC**, clock fino a 240 MHz, memoria Flash integrata (fino a 16 MB) e PSRAM opzionale. Offre connettività **Wi-Fi 2.4 GHz** e **Bluetooth LE 5.0**, numerose periferiche di I/O digitali e analogiche, e supporto nativo per interfacce come **UART, SPI, I²C, PWM, ADC** e **USB OTG Full-Speed**.

La MCU è montata in formato **WROOM-1**, un modulo certificato che integra l'antenna PCB, l'oscillatore principale a 40 MHz e il sistema di alimentazione interno, semplificando la progettazione del PCB e garantendo stabilità RF.

6.1 Alimentazione e filtraggio

Il modulo ESP32-S3-WROOM-1 opera a una tensione nominale di **3.3 V ± 5%**, fornita dal convertitore step-down (buck) descritto nella sezione 2.5. L'assorbimento massimo previsto è di circa **600 mA** con Wi-Fi attivo e periferiche abilitate. Per garantire stabilità e immunità ai disturbi, la linea **3V3** è filtrata tramite uno stadio LC, vedi sezione 2.4.

> *Nota di **layout**:* È consigliabile che il **GND** del modulo sia connesso direttamente al piano di massa, con un'ampia superficie per favorire il ritorno delle correnti di switching.

6.2 Interfaccia USB-C e programmazione

L'interfaccia di comunicazione USB è gestita direttamente dal microcontrollore **ESP32-S3-WROOM-1**, che integra un controller **USB OTG Full-Speed** compatibile con la funzione di **programmazione e debug nativo**. Non è richiesto alcun convertitore esterno, poiché la connessione avviene direttamente tramite i pin **D+** e **D-** del modulo. La sezione hardware del connettore è descritta nel capitolo dedicato "4 - USB-C Connector".

Il pin **VBUS** del connettore non è connesso all'alimentazione della scheda: l'intera MCU è alimentata dal rail **3V3** generato dal buck converter. Tale configurazione permette la comunicazione USB pur mantenendo l'alimentazione completamente isolata dal bus del computer.

Durante la programmazione, la MCU entra automaticamente in modalità **USB-Serial/JTAG** interna, consentendo il caricamento del firmware tramite la porta USB-C e l'IDE Arduino o ESP-IDF.

6.3 GPIO mapping

La sezione seguente elenca i pin GPIO utilizzati dal modulo ESP32-S3-WROOM-1 sulla scheda Alina Vista, con la relativa funzione hardware, direzione del segnale e note firmware associate.

Pin MCU	Etichetta	Direzione	Funzione	Componente associato	Note
GPIO0	D_BOOT	I	Boot mode / programmazione	SW2	Pulsante di boot — collega il pin a GND per avvio in modalità programmazione

Pin MCU	Etichetta	Direzione	Funzione	Componente associato	Note
GPIO2	SOIL	I (ADC)	Sensore umidità suolo	J7	Legge il segnale analogico **AOUT** del sensore capacitivo di umidità
GPIO4	VALVE	O	Comando MOSFET elettroval-vola	Q1	Attiva/disattiva l'elettro-valvola
GPIO5	DC_LDC	O	Data/Commanc display	J5	Seleziona modalità dati o comando per display PCD8544
GPIO6	RST_LDC	O	Reset display	J5	Resetta il controller PCD8544
GPIO7	CS_LDC	O	Chip select display	J5	Selezione SPI display
GPIO8	D_BOOT_CTR	O	Controllo remoto boot	Q2	Permette digitalmen-te l'aggiorna-mento firmware (collega GPIO0 a GND via transi-stor)
GPIO9	D_EN_CTR	O	Controllo remoto enable	Q3	Permette digitalmen-te il re-set/enable del modulo (collega EN a GND via transi-stor)

Pin MCU	Etichetta	Direzione	Funzione	Componente associato	Note
GPIO10	BL_LDC	O	Retroilluminazione display	J5	Controlla i LED di backlight del display
GPIO11	MOSI_LDC	O	SPI Data	J5	Dati verso il display PCD8544
GPIO12	SCK_LDC	O	SPI Clock	J5	Clock SPI del display
GPIO13	AM2302	I	Sensore temperatura/umidità	J6	Legge il pin **DATA** del sensore AM2302
GPIO15	SCL	O	I²C Clock	U4	Bus I²C per RTC DS3231
GPIO16	SDA	I/O	I²C Data	U4	Bus I²C per RTC DS3231
GPIO38	HOLD_ON	O	Soft latch power hold	U2	Mantenuto HIGH dopo il boot per tenere alimentata la scheda

6.4 Circuito di reset e boot

Il modulo **ESP32-S3-WROOM-1** utilizza due pin principali per la gestione dell'avvio e della programmazione:

- **EN** (Enable / Reset): abilita o tiene in reset il core della MCU.
- **GPIO0**: pin di *strapping* che determina la modalità di boot (esecuzione da Flash o bootloader ROM).

La scheda **Alina Vista** implementa sia il **reset manuale** tramite pulsanti, sia il **controllo "digitale"** di EN e GPIO0 tramite due GPIO dedicati (D_BOOT_CTR e D_EN_CTR), per consentire il riavvio in modalità programmazione senza intervento fisico in modo da poter implementare via firmware una procedura automatica di **firmware update**.

6.4.1 Pin EN — Reset della MCU Il pin **EN** dell'ESP32-S3 funziona come ingresso di abilitazione / reset:

- EN = **HIGH** → MCU attiva, esecuzione normale.
- EN = **LOW** → MCU in reset, tutte le periferiche disabilitate.

Sulla scheda **Alina Vista** il pin EN è tipicamente collegato a:

- una resistenza di **pull-up** verso 3.3 V (mantiene la MCU attiva di default),
- un eventuale **pulsante di reset** (se previsto) che porta EN a GND,
- un transistor di controllo (**Q3**) comandato dal pin **GPIO9 (D_EN_CTR)** per la gestione "software" del reset.

In questo modo la MCU può essere resettata:

- **manualmente**, premendo il pulsante di reset (EN → GND),
- **da firmware**, portando D_EN_CTR in uno stato che attiva Q3, forzando EN a livello basso.

> Una sequenza tipica di "reset controllato" prevede di portare D_EN_CTR in stato attivo per alcuni millisecondi (EN = LOW), quindi rilasciarlo (EN = HIGH), provocando un riavvio pulito della MCU.

6.4.2 Pin GPIO0 — Modalità di boot Il pin **GPIO0** è un ingresso di *boot strapping*:

- se, al momento del reset, **GPIO0 è HIGH** → l'MCU avvia normalmente il firmware dalla Flash interna;
- se, al momento del reset, **GPIO0 è LOW** → l'MCU entra in **bootloader seriale/USB** per la programmazione.

Per questo, sulla scheda **Alina Vista** GPIO0 è collegato a:

- una resistenza di **pull-up** a 3.3 V (modalità normale di default),
- il pulsante **SW2**, che collega il pin a GND quando premuto,
- un transistor di controllo (es. **Q2**) comandato da **GPIO8 (D_BOOT_CTR)**, che permette di portare GPIO0 a GND in modo automatico.

6.4.3 Schema semplificato reset/boot

Riferimento	Scheda	Funzione	Valore	Descrizione
SW_{EN}	**SW1**	Pulsante di reset su scheda	SW_Push_THT_6mm	-
SW_B	**SW2**	SW_Push_THT_6m	Pulsante di boot su scheda	-
C_{EN}	**C8**	Condensatore di disaccoppiamento (reset)	100 nF	Mantiene il pin EN alto per alcuni millisecondi dopo la pressione del pulsante

Riferimento	Scheda	Funzione	Valore	Descrizione
R_{EN}	**R3**	Resistenza di pull-up (reset)	10 kΩ	Mantiene il pin EN (reset) normalmente alto
R_B	**R4**	Resistenza di pull-up (boot)	10 kΩ	Mantiene il pin GPIO0 (boot) normalmente alto
R_{EC}	**R13**	Resistenza di base (reset control)	10 kΩ	-
R_{BC}	**R12**	Resistenza di base (boot control)	10 kΩ	-
D_B	**D3**	Diodo di blocco (boot control)	1N4148	Impedisce che il GPIO0 (boot), quando viene tenuto alto, faccia retro-alimentare il GPIO8 (D_BOOT_CTR)
Q_{EN}	**Q3**	Controllo digitale reset (EN)	2N3904	Controllato da GPIO9 (D_EN_CTR)
Q_B	**Q2**	Controllo digitale boot (GPIO0)	2N3904	Controllato da GPIO8 (D_BOOT_CTR)
R_{EPD}	**R11**	Resistenza di pull-down (reset control)	100 kΩ	-
R_{BPD}	**R14**	Resistenza di pull-down (boot control)	100 kΩ	-

6.4.4 Sequenze di boot e programmazione Le principali modalità di avvio sono:

A) **Boot normale (esecuzione firmware)**

1. EN = HIGH (pull-up attivo, nessun reset premuto).
2. GPIO0 = HIGH (pull-up, nessun pulsante premuto, Q2 inattivo).
3. La MCU esegue il firmware presente in Flash.

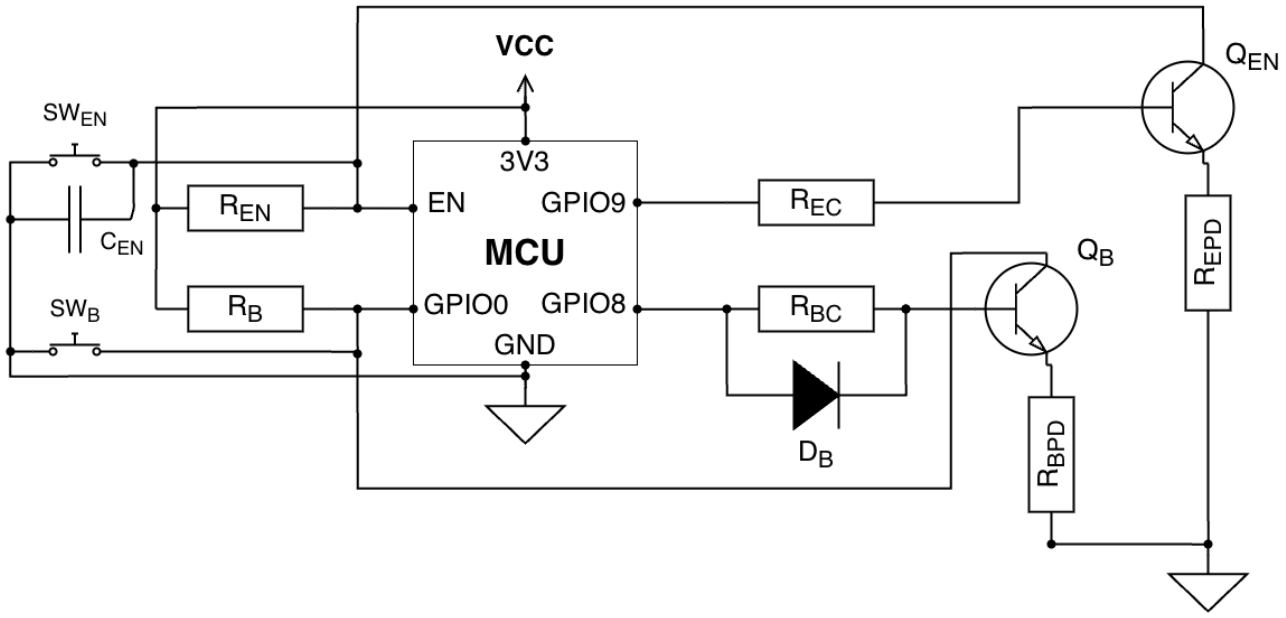

Figura 5: Schema reset/boot

B) **Bootloader per programmazione manuale**

1. Premere SW2 (GPIO0 forzato LOW).
2. Generare un reset (pulsante EN o reset del sistema).
3. Rilasciare il reset mantenendo SW2 premuto.
4. L'ESP32-S3 entra in bootloader ROM, pronto per ricevere il nuovo firmware via USB.

C) **Bootloader per programmazione controllata da firmware**

1. Attivare via firmware **D_BOOT_CTR (GPIO8)** per far condurre Q2 → GPIO0 = LOW.
2. Attivare **D_EN_CTR (GPIO9)** per tenere EN = LOW per alcuni millisecondi.
3. Rilasciare D_EN_CTR → EN torna HIGH mentre GPIO0 è ancora LOW.
4. La MCU si avvia in bootloader senza intervento fisico sui pulsanti.

> Questa sequenza può essere usata per implementare funzioni di "firmware update" remoto: la MCU può riavviarsi in modalità bootloader in risposta a un comando (ad es. da USB, Wi-Fi o menu utente), evitando di intervenire fisicamente sui pulsanti.

6.4.5 Note firmware - Controllo reset e boot

1. Realizzare un **riavvio software**

```
const int PIN_EN_CTRL = 9;   // D_EN_CTR $\rightarrow$ controlla Q_EN

void setup() {
    pinMode(PIN_EN_CTRL, OUTPUT);
    digitalWrite(PIN_EN_CTRL, LOW);   // mantiene EN non interferito
```

```
}

// Reset software
void softResetBoard() {
    // Attiva il reset portando EN a LOW tramite Q_EN
    digitalWrite(PIN_EN_CTRL, HIGH);
    delay(50);  // Il delay può variare tra 10 e 100 ms; 50 ms è sicuro.

    // Rilascia il reset riportando EN a HIGH
    digitalWrite(PIN_EN_CTRL, LOW);
}
```

2. Entrare in Bootloader (**aggiornamento firmware**)

```
const int PIN_BOOT_CTRL = 8;    // GPIO di controllo del BOOT (GPIO0)
const int PIN_EN_CTRL = 9;  // GPIO di controllo del RESET (EN)

void setup() {
    pinMode(PIN_BOOT_CTRL, OUTPUT);
    pinMode(PIN_EN_CTRL,   OUTPUT);

    digitalWrite(PIN_BOOT_CTRL, LOW);   // transistor spento
    digitalWrite(PIN_EN_CTRL,   LOW);   // transistor spento
}

// Entra in modalità bootloader via controllo digitale
void enterBoot() {
    // 1. Porta GPIO0 a LOW tramite Q_B
    digitalWrite(PIN_BOOT_CTRL, HIGH);
    delay(10);

    // 2. Reset: porta EN a LOW tramite Q_EN
    digitalWrite(PIN_EN_CTRL, HIGH);
    delay(50);

    // 3. Rilascia EN
    digitalWrite(PIN_EN_CTRL, LOW);
    delay(30);

    // 4. Rilascia GPIO0
    digitalWrite(PIN_BOOT_CTRL, LOW);
}
```

6.5 Raccomandazioni di layout

- L'antenna del modulo deve rimanere **completamente libera** da piani di massa, piste o componenti sotto e ai suoi lati per almeno **15–20 mm**.
- Il bordo della scheda non deve creare riflessioni: mantenere la parte dell'antenna **sporgente** dal PCB o comunque libera.
- Evitare di collocare vicino all'antenna:
  - cavi di alimentazione 12 V,
  - switching del buck (AP63203),

  - la bobina dell'elettrovalvola.

- Il contenitore deve essere in **plastica**; evitare parti metalliche nella regione vicina all'antenna.

6.6 Firmware OTA — Aggiornamento via Wi-Fi

La scheda Alina Vista supporta la possibilità di aggiornare il firmware tramite connessione WiFi, senza necessità di collegare un computer via connettore USB-C. Questa funzionalità è opzionale, ma può essere attivata in qualsiasi momento tramite firmware, poichè l'hardware non richiede componenti aggiuntivi.

Il sistema OTA permette di:

- caricare un nuovo firmware tramite rete locale
- effettuare aggiornamenti remoti
- implementare sistemi di configurazione web
- aggiornare la scheda senza accesso fisico ai pulsanti BOOT/RESET

Uno dei migliori metodi per implementare l'update OTA è quello di utilizzare un **server HTTPS** che espone un file .bin con la nuova versione. In questo modo la scheda scarica il firmware, lo scrive in una partizione OTA libera e si riavvia eseguendo la nuova immagine.

Per utilizzare l'OTA è necessario che la memoria Flash dell'ESP32-S3 sia suddivisa in una tabella di partizioni che preveda le seguenti partizioni:

Partizione	Contenuto
NVS	Configurazioni
OTADATA	Metadati OTA
APP0	Firmware
APP1	Firmware
FS	File system (es. SPIFFS)

Come si vede dalla tabella è necessario prevedere **due slot firmware**, ovvero partizioni **app0** e **app1** che contengono due copie del firmware: una è quella in esecuzione, l'altra è quella che viene riscritta durante l'aggiornamento OTA.

Un esempio di tabella OTA in formato CSV (standard Espressif) è il seguente:
Partition Table

```
# Name,   Type,   SubType,  Offset,    Size
nvs,      data,   nvs,      0x9000,    0x5000
otadata,  data,   ota,      0xE000,    0x2000
app0,     app,    ota_0,    0x10000,   0x190000
app1,     app,    ota_1,    0x1A0000,  0x190000
```

L'ESP32-S3 gestisce automaticamente il passaggio tra app0 e app1 tramite esp_ota_set_boot_partition().

L'uso di una **partition table personalizzata** si realizza in modo differente a seconda dell'IDE che si utilizza.

6.6.1 Partition table su ESP-IDF Per usare una tabella personalizzata in ESP-IDF: 1. Creare un file partitions.csv nella root di progetto. 2. In idf.py menuconfig selezionare: **Partition Table** → **Partition Table** → Custom partition table CSV **Custom partition CSV file** → partitions.csv 3. Compilare e flashare normalmente: bash idf.py -p /dev/ttyUSB0 flash ESP-IDF si occuperà di scrivere la tabella partizioni e poi le app negli slot corretti.

6.6.2 Partition table su PlatformIO In PlatformIO è possibile usare una partition table personalizzata aggiungendo un file CSV e riferendolo nel file platformio.ini.

1. Creare un file partitions.csv nella cartella del progetto con il contenuto desiderato.
2. Aggiungere in platformio.ini: ini board_build.partitions = partitions.csv
3. Compilare e caricare il firmware: bash pio run -t upload PlatformIO userà il file partitions.csv per generare e flashare la tabella di partizioni insieme al firmware.

6.6.3 Partition table su Arduino IDE In Arduino IDE si può soltanto selezionare uno schema tra quelli predefiniti: 1. Aprire il menu **Tools** → **Partition Scheme**. 2. Selezionare uno schema di tipo **"Default OTA"** oppure equivalente (OTA con due applicazioni). 3. Caricare normalmente lo sketch.

Questo carica una partition table predefinita che contiene già **app0**, **app1**, **otadata** e **nvs** in modo compatibile con l'OTA.

6.6.4 Esempio di aggiornamento OTA via HTTPS

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <HTTPClient.h>
#include <Update.h>
#include <ArduinoJSON.h>

#define VERSION "1.0.0" // versione firmware attualmente installato

const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";

// Ceritificato CA root
const char* root_ca = R"EOF(
-----BEGIN CERTIFICATE-----
... CERTIFICATO DELLA CA ...
-----END CERTIFICATE-----
)EOF";

const char* versionsUrl = "https://alina-firmware.vercel.app/firmware/versions.jso

static const int MAX_VERSIONS = 20; // Numero massimo di versioni disponibili da tr

String availableVersions[MAX_VERSIONS]; // Array delle versioni (numero di versione
```

```
String availableUrls[MAX_VERSIONS];  // Array delle versioni (url di versione)
int versionsCount = 0;

bool fetchAvaibleVersions () {
    versionCount = 0;
    WifiClientSecure client;
    client.setCACert(root_ca);  // Imposta il certificato

    HTTPClient https;
    if (!https(client, versionsUrl)) {
        Serial.println("versions.json: https.begin() failed");
        return false;
    }

    int httpCode = https.GET();
    if (httpCode != HTTP_CODE_OK) {
        Serial.printf("versions.json: HTTP ERROR - %d\n", httpCode);
        https.end();
        return false;
    }

    JsonDocument doc;
    DeserializationError error = deserializeJson(doc, https.getStream());
    https.end();

    if (err) {
        Serial.print("versions.json: parse JSON failed: ");
        Serial.println(err.c_str());
        return false;
    }

    if (!doc.containsKey("versions")) {
        Serial.println("versions.json: 'versions' field missing");
        return false;
    }

    JsonArray arr = doc["versions"].as<JsonArray>();

    for (JsonObject v : arr) {
        if (!v.containsKey("version") || !v.containsKey("url")) continue;
        if (versionCount >= MAX_VERSIONS) break;

        availableVersions[versionsCount] = v["version"].<as>String();
        availableUrls[versionsCount] = v["url"].<as>String();

        versionCount++;
    }

    return versionCount > 0;
}

bool runHttpsOtaUpdate (const char* url) {
    WifiClientSecure client;
```

```
    client.setCACert(root_ca);  // Imposta il certificato

    HTTPClient https;

    if (!https.begin(client, url)) {
        Serial.println("OTA: https.begin() failed");
        return false;
    }

    int httpCode = https.GET();

    if (httpCode != HTTP_CODE_OK) {
        Serial.printf("OTA: HTTP ERROR - %d\n", httpCode);
        https.end();
        return false;
    }

    int contentLength = https.getSize();
    WiFiClient* stream = https.getStreamPtr();

    if (contentLength <= 0) {
        Serial.println("OTA: Content dimension is not valid");
        https.end();
        return false;
    }

    Serial.printf("OTA: firmware dimension = %d bytes\n", contentLength);

    if (!Update.begin(contentLength)) {
        Serial.println("OTA: Update.begin() failed");
        Update.printError(Serial);
        https.end();
        return false;
    }

    size_t written = Update.writeStream(*stream);
    if (written != (size_t)contentLength) {
        Serial.printf("OTA: Error writing firmware.\n\tWritten bytes: %d\n\tExpecte
        Update.end();
        https.end()
        return false;
    }

    if (!Update.end()) {
        Serial.println("OTA: Update.end() failed");
        Update.printError(Serial);
        https.end();
        return false;
    }

    if (Update.isFinished()) {
        Serial.println("OTA: Update not complete");
        https.end();
```

```
        return false;
    }

    Serial.println("OTA: update completed. Restarting system...");
    https.end();

    esp_restart();
    return true;
}

void setup () {
    Serial.begin(115200);

    WiFi.begin(ssid, password);
    Serial.print("Connecting Wi-Fi");
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(500);
    }
    Serial.println("\nWi-Fi connected.");
}
```

Questo codice di esempio prevede l'esistenza di un sito web su un server che esponga una rotta per il download di un file json, che contenga tutte le versioni disponibili del firmware e che debbano essere ordinate in ordine decrescente *(dalla più recente alla più vecchia)*, e che abbia una struttura di questo tipo:

```
/*
 *  File versions.json
 *  In questo file vengono elencate tutte le versioni disponibili del firmware
 *  in ordine decrescente.
 */
{
    "versions": [
        {
            "version": "1.1.1",
            "url": "https://alina-firmware.vercel.app/firmware/alina-1.1.1.bin",
            "note": "..."
        },
        {
            "version": "1.1.0",
            "url": "https://alina-firmware.vercel.app/firmware/alina-1.1.0.bin",
            "note": "..."
        },
        {
            "version": "1.0.0",
            "url": "https://alina-firmware.vercel.app/firmware/alina-1.0.0.bin",
            "note": "..."
        }
    ]
}
```

6.7 Flash integrata

L'ESP32-S3 utilizza una memoria **SPI Flash esterna** collegata internamente alla MCU tramite un bus flash dedicato. Questa memoria contiene:

- Bootloader,
- Tabella partizioni,
- Firmware (app0/app1),
- NVS (Non-Volatile Storage),
- Eventuali file system (SPIFFS/LittleFS),
- Dati utente persistenti.

La gestione della Flash non avviene tramite accesso diretto a indirizzi di memoria, ma tramite il sistema di partizioni definito da Espressif, come già illustrato nella Tabella partzioni, che garantisce stabilità, **sicurezza OTA** e protezione della memoria.

La scheda **Alina Vista** richiede l'utilizzo di un modulo MCU ESP32-S3 WROOM-1 con una **memoria flash integrata** di almeno **4 MB**, ma si consiglia di utilizzare un modulo che abbia una flash integrata di **8 MB** per non avere problemi di spazio specialmente se si utilizzano:

- Doppia partizione firmware
- Firmware OTA
- Sensori
- Interfaccia display
- File system

8 MB di memoria flash forniscono, infatti, spazio sufficiente per:

- Firmware principali (fino a $\sim$ 1.8 MB cad.)
- Doppia partizione
- NVS
- Eventuale SPIFFS

Di seguito è riportato il codice di esempio di utilizzo della flash per leggere/scrivere dalla **NVS *(Non-Volatile-Storage)***:

```
#include <Preferences.h>

Preferences prefs;

void setup() {
    Serial.begin(115200);

    if (!prefs.begin("alina", false)) {
        Serial.println("Errore NVS");
    }

    // Scrivere valori
    prefs.putInt("irrig_on", 1);
    prefs.putFloat("soil_thr", 45.5);
    prefs.putString("ver", "1.0.3");

    // Leggere valori e salvarli in RAM
    int irrig = prefs.getInt("irrig_on", 0);
    float thr = prefs.getFloat("soil_thr", 30.0);
```

```
    String ver = prefs.getString("ver", "0.0.0");

    // Cancellare una chiave
    prefs.remove("irrig_on");

    // Cancellare tutto lo spazio NVS
    prefs.clear();
}
```

> *Nota bene: NVS scrive su flash in modo sicuro e con wear-leveling, NON usare accesso raw alla flash per salvare configurazioni.*

Come visto precedentemente è possibile anche implementare un file system che "giri" in flash memory, ma non è l'obiettivo di questa documentazione mostrarne l'implementazione.

A seconda quindi della dimensione della memoria flash del modulo che si installa sulla scheda è possibile creare una **partition table** custom di diverso tipo.

6.7.1 Partition table per 4 MB di flash

```
# Name,   Type, SubType, Offset,   Size
nvs,      data, nvs,     0x9000,   0x5000
otadata,  data, ota,     0xE000,   0x2000
app0,     app,  ota_0,   0x10000,  0x180000
app1,     app,  ota_1,   0x190000, 0x180000
spiffs,   data, spiffs,  0x310000, 0x0F0000
```

Partzione	Spazio
NVS	20 KB
OTADATA	8 KB
APP0	1.5 MB
APP1	1.5 MB
SPIFFS	∼ 960 KB

6.7.2 Partition table per 8 MB di flash

```
# Name,   Type, SubType, Offset,   Size
nvs,      data, nvs,     0x9000,   0x5000
otadata,  data, ota,     0xE000,   0x2000
app0,     app,  ota_0,   0x10000,  0x300000
app1,     app,  ota_1,   0x310000, 0x300000
spiffs,   data, spiffs,  0x610000, 0x1F0000
```

Partzione	Spazio
NVS	20 KB
OTADATA	8 KB
APP0	3 MB

Partzione	Spazio
APP1	3 MB
SPIFFS	∼ 1.94 MB

6.7.3 Partition table per 16 MB di flash

```
# Name,   Type, SubType, Offset,   Size
nvs,      data, nvs,     0x9000,   0x5000
otadata,  data, ota,     0xE000,   0x2000
app0,     app,  ota_0,   0x10000,  0x500000
app1,     app,  ota_1,   0x510000, 0x500000
spiffs,   data, spiffs,  0xA10000, 0x5F0000
```

Partzione	Spazio
NVS	20 KB
OTADATA	8 KB
APP0	5 MB
APP1	5 MB
SPIFFS	∼ 5.94 MB

7 Display

Il display utilizzato dalla scheda **Alina Vista** è un modulo grafico Nokia 5110 basato sul controllore **PCD8544**. È un display monocromatico a bassa potenza con risoluzione **84×48 pixel**, ideale per visualizzare testo, icone e semplici interfacce.

Il modulo richiede solo 6 segnali digitali e un piccolo assorbimento di corrente, ed è pienamente compatibile con l'ambiente ESP32-S3.

7.1 Caratteristiche principali

- Driver: **PCD8544** (driver grafico a 6 pagine × 84 colonne)
- Risoluzione: **84 × 48 pixel**
- Colore: monocromatico (1 bit/pixel)
- Retroilluminazione: LED bianchi (gestita da pin dedicato)
- Tensione operativa: **3.3V** (compatibile con ESP32-S3)
- Interfaccia: SPI-like (non è SPI standard, ma compatibile)
- Assorbimento:
  - **Display**: ∼ 200 μA
  - **Backlight**: 5–20 mA in base alla resistenza

7.2 Pin e segnali su scheda

Il display è collegato all'ESP32-S3 tramite il connettore **J5**:

Pin J5	Segnale Display	Pin MCU
1	VCC	3V3
2	GND	GND
3	SCE	GPIO7
4	RST	GPIO6
5	D/C	GPIO5
6	MOSI	GPIO11
7	SCK	GPIO12
8	BL	GPIO10

> *Nota:* Il PCD8544 non usa MISO. L'interfaccia è unidirezionale verso il display.

7.3 Alimentazione e retroilluminazione

Il display opera a **3.3V** ed è alimentato direttamente dal rail **3V3** della scheda.

La retroilluminazione è controllata dal pin **GPIO10 (BL_LDC)**. È presente una resistenza **R15** in serie sulla scheda sul per limitare la corrente dei LED, salvaguardando così il GPIO, la MCU e gli stessi LED del display. Si consiglia di usare un valore per la **resistenza di backlight** compreso tra i **470** Ω e **1.2 k**Ω.

7.4 Note firmware

La gestione software del display Nokia 5110 (PCD8544) può essere implementata in diversi modi. Il driver è molto semplice e richiede solo due categorie di operazioni:

1. l'invio dei comandi di configurazione iniziale (contrast, bias, modalità display);
2. l'aggiornamento del contenuto tramite la trasmissione sequenziale di 504 byte.

Esistono varie librerie disponibili per ESP32, dalla più minimalista a quelle con framebuffer integrato. Per la scheda **Alina Vista** si consiglia di utilizzare la libreria PCD8544_lib che:

- garantisce la massima **compatibilità cross-platform** (ESP32/Arduino);
- non richiede un **framebuffer** interno (utile per ridurre l'uso di RAM);
- fornisce un'interfaccia semplice e diretta per scrivere testo e grafica;
- integra una libreria di **menu** e gestione input, in modo da integrare facilmente il menu e la gestione dei comandi di interfaccio della scheda Alina Vista;

- è facilmente estensibile per nuovi font o metodi di disegno.

Si riporta di seguito l'esempio di utilizzo della libreria e del relativo driver PCD8544 per la creazione e l'utilizzo di un menu gestito da pulsanti.

```
/*
 * Questo file di esempio costruisce e stampa sui moduli display, comunemente chiama
 * che usano driver PCD8544 un menu di base per le impostazioni di visibilità dello
 * L'esempio è pensato per l'uso su arduino. Qualora si volesse utilizzare l'esempio
 * un esp32 bisognerà adattare alcune parti del codice, in particolare la nominazion
 * rendere lo sketch compatibile con i gpio del modello esp che si utilizza.
 */
#include <Arduino.h>
#include <SPI.h>
#include <PCD8544.h>
#include <font/mono_5x8px/data.h>
#include <font/mono_5x8px/meta.h>
#include <menu/menu.h>

#define SCK 13
#define MOSI 11
#define LCD_CS 10
#define LCD_DC 9
#define LCD_RST 8
#define LCD_BL 5 // scegliere canale PWM

#define BACK_BTN A0
#define SELECT_BTN A1
#define FORWARD_BTN A2

// ottiene il numero di cifre di un numero
uint8_t getDigits (int num) {
    if (num == 0) return 1;
    uint8_t i = 0;
    while (num != 0) {
        num /= 10;
        i++;
    }
    return i;
};

// Calcola la x di partenza per centrare il testo
uint8_t calcAbscissaToCenter (uint8_t strLen, uint8_t maxLen) {
    return (uint8_t)((uint16_t)(maxLen - (uint16_t)(strLen * (GLYPH_WIDTH + GLYPH_S
}

PCD8544 lcd(SPI, {SCK, MOSI, LCD_CS, LCD_DC, LCD_RST, LCD_BL}, 1000000, SPI_MODE0)
MenuController menu({
    BACK_BTN,
    FORWARD_BTN,
    SELECT_BTN,
    true, // active low = true (pulsante chiude a GND, pullup a VCC) | false (pulsan
    false, // resistenza interna (pullup/pulldown) = true (resistenza interna) | fal
```

```
    150 // debounce ms per pulsanti
});


void onBack();
const MenuItem BackItem("Indietro", onBack, nullptr, 0);    // Voce: Indietro

// Impostazioni    ------------------------------------------------------------------

void onContrast();
void onBrightness();
void onBias();
void onTC();
bool inContrast = false;
bool inBrightness = false;

const MenuItem ContrastItem ("Contrasto", onContrast, nullptr, 0);  // Voce: Contra
const MenuItem BrightnessItem ("Luminosita", onBrightness, nullptr, 0); // Voce: Lu
const MenuItem BiasItem ("Bias", onBias, nullptr, 0);   // Voce: Bias
const MenuItem TCItem ("TC", onTC, nullptr, 0); // Voce: TC

// array delle voci di "Impostazioni"
const MenuItem settingsItems[] = {
    BackItem,
    ContrastItem,
    BrightnessItem,
    BiasItem,
    TCItem,
};




// ---------------------------------------------------------------------------------

// Main Menu     ----------------------------------------------------------------

const MenuItem SettingsSubMenu("Impostazioni", nullptr, settingsItems, sizeof(sett

// array delle voci di "Main Menu"
const MenuItem mainMenuItems[] = {
    BackItem,
    SettingsSubMenu,
};

// ---------------------------------------------------------------------------------
// Dichiarazione della root del menu ("Main Menu")
const MenuItem rootMenu("Main Menu", nullptr, mainMenuItems, sizeof(mainMenuItems)



/* COMMON SETTING ACTIONS */
// Render setting
```

```
void renderSetting (const char* label, uint16_t curVal) {
    lcd.clear();
    delay(25);
    lcd.printStringCentered(label, 0);
    uint8_t valL = getDigits(curVal);
    uint8_t xval = calcAbscissaToCenter((valL + 4), COLUMNS);
    lcd.setCursor(xval, 2);
    lcd.print("- ");
    lcd.print(curVal);
    lcd.print(" +");
}

// Exit action
inline void exitAction () { menu.exitAction(); lcd.softRefresh(); }
/* ------------------------------------- */


/* CONTRAST ACTION */
void contrastLeft () {
    lcd.setContrast(lcd.getContrast() - 1);
    delay(10);
}
void contrastRight () {
    lcd.setContrast(lcd.getContrast() + 1);
    delay(10);
}
void contrastRender () {
    renderSetting("Contrasto", lcd.getContrast());
}

void onContrast() {
    MenuController::Action a;
    a.onLeft = contrastLeft;
    a.onRight = contrastRight;
    a.onSelect = exitAction;
    a.onRender = contrastRender;

    menu.enterAction(a);
}

/* ------------------------------------- */

/* BRIGHTNESS ACTION */
void brightnessLeft () {
    lcd.backlightLevel(lcd.getBrightness() - 1);
}
void brightnessRight () {
    lcd.backlightLevel(lcd.getBrightness() + 1);
}
void brightnessRender () {
    renderSetting("Luminosita", lcd.getBrightness());
}
void onBrightness() {
```

```
    MenuController::Action a;
    a.onLeft = brightnessLeft;
    a.onRight = brightnessRight;
    a.onSelect = exitAction;
    a.onRender = brightnessRender;

    menu.enterAction(a);
}
/* ------------------------------------- */

/* BIAS ACTION */
void biasLeft () {
    lcd.setBias(lcd.getBias() - 1);
}
void biasRight () {
    lcd.setBias(lcd.getBias() + 1);
}
void biasRender () {
    renderSetting("Bias", lcd.getBias());
}
void onBias() {
    MenuController::Action a;
    a.onLeft = biasLeft;
    a.onRight = biasRight;
    a.onSelect = exitAction;
    a.onRender = biasRender;

    menu.enterAction(a);
}
/* ------------------------------------- */

/* TC ACTION */
void tcLeft () {
    lcd.setTC(lcd.getTempCoeff() - 1);
}
void tcRight () {
    lcd.setTC(lcd.getTempCoeff() + 1);
}
void tcRender () {
    renderSetting("TC", lcd.getTempCoeff());
}
void onTC() {
    MenuController::Action a;
    a.onLeft = tcLeft;
    a.onRight = tcRight;
    a.onSelect = exitAction;
    a.onRender = tcRender;

    menu.enterAction(a);
}
/* ------------------------------------- */

/* Torna indietro */
```

```
void onBack () {
    menu.exit();
    menu.displayMenu();
}
/* ------------------------------------- */

void setup() {
    Serial.begin(115200);
    delay(2000);
    lcd.begin(30, 50, 4, 2);
    delay(500);
    lcd.setFont(MONO_5x7);
    menu.attachDisplay(&lcd);
    menu.createMenu(&rootMenu);

    menu.displayMenu();
}

void loop() {
    menu.update();
}
```

8 Interfaccia utente

L'obiettivo dell'interfaccia utente di **Alina Vista** è offrire all'utente un'interazione semplice, stabile e completamente locale, adatta all'utilizzo in ambienti esterni o senza connettività.

L'interfaccia utente di **Alina Vista** si basa su:

- Display grafico Nokia 5110 (PCD8544) (vedi sezione Display).
- Menu system leggero, integrato nel firmware.
- Controller fisico dedicato.

Il **controller** è una scheda separata che si connette alla scheda principale di **Alina Vista**, tramite il il connettore dedicato **J4** posizionato sulla main board.

Il connettore **J4** è un connettore FPC/FFC (passo 1.0) che espone 8 pin:

Tabella collegamenti J4

N. Pin	Collegamento su main board	Collegamento su controller	Funzione
1	U2(IN)	SW3 (A)	Pulsante powe
2	U2(EN)	SW3 (B)	Pulsante powe
3	R19	D4	Collegamento anodo
4	MCU(GPIO21)	SW6	Pulsante nav. f
5	MCU(GPIO18)	SW5	Pulsante nav.
6	MCU(GPIO17)	SW4	Pulsante nav.
7	MCU(GPIO1)	D4	Status LED *(acti*
8	GND	GND	-

Il **controller** si compone di:

- Pulsante di accensione del core di sistema.
- Pulsanti di navigazione.
- Led indicazione accensione.

8.1 Pulsante di accensione del core di sistema

Il pulsante di accensione del controller gestisce l'avvio del core logico tramite circuito di **soft-latch** (vedi sez. Soft latch power on). Alla pressione del pulsante:

1. viene alimentata la MCU,
2. Il core del sistema entra in funzione.
3. Si accende il led di stato del controller

Il soft-latch permette di accendere/spegnere il sistema in modo controllato senza interruttori meccanici diretti sulla linea 12V.

8.2 Pulsanti di navigazione

Il controller include tre pulsanti dedicati alla navigazione dell'interfaccia grafica:

- **BACK** → spostamento o decremento
- **FORWARD** → spostamento o incremento
- **SELECT** → conferma/ingresso nei menu

La gestione firmware include:

- debounce automatico
- invio degli eventi al sistema di menu interno.

I tre pulsanti sono ***active LOW***, cioè il i pulsanti chiudono a **GND** ed hanno ***pull-up esterno a VCC*** (3.3 V), da 10 kΩ *(per ogni pulsante)*.

Riferimenti scheda pulsanti di navigazione

Pulsante	Riferimento	Pull-up	GPIO
BACK	SW4	R21	GPIO17
FORWARD	SW6	R23	GPIO21
SELECT	SW5	R22	GPIO18

8.3 LED di stato

Il dispositivo integra un LED di stato collegato al rail 3.3 V tramite una resistenza limitatrice (installata sulla scheda principale), mentre il catodo è collegato al pin GPIO1 della MCU. La logica è quindi active-LOW: il LED si accende portando il pin a livello basso.

Il LED fornisce un'indicazione visiva immediata sullo stato operativo della centralina, utile sia durante l'utilizzo normale sia nelle fasi diagnostiche. L'implementazione del comportamento del LED è gestita interamente via firmware, attraverso pattern di lampeggio o di "breathing".

> *Nota tecnica: Si raccomanda una corrente di funzionamento ridotta (0.5–1.5 mA), sufficiente a garantire visibilità in condizioni standard ma senza introdurre consumi eccessivi o disturbi sull'alimentazione. La luminosità effettiva dipende dal tipo di LED SMD scelto.*

Il LED supporta quattro modalità operative, ognuna associata a una condizione distinta del sistema. Questo permette una diagnostica immediata senza interfaccia grafica.

Stato	Descrizione	Comportamento LED
BOOT	La MCU sta inizializzando il sistema	**Lampeggio lento**
OK	Sistema operativo stabile, nessun errore rilevato	**Acceso fisso**
ERROR/FAULT	Errore critico o condizione anomala	**Lampeggio rapido continuo**
OTA/Configuration	Modalità configurazione / aggiornamento firmware in corso	**Effetto respiro (breathing)**

9 Real Time Clock - DS3231

Alina Vista integra un circuito dedicato al Real-Time Clock, basato sul chip DS3231, utilizzato per garantire la misurazione accurata dell'orario e della data anche in condizioni di assenza temporanea di alimentazione principale. Il chip opera tramite interfaccia I²C ed include un cristallo di riferimento ad alta stabilità e un sistema di compensazione termica.

Il circuito RTC assicura la disponibilità continua di:

- data e ora correnti,
- timestamp affidabili per eventi e log,
- sincronizzazione temporale per operazioni pianificate (es. irrigazione, lettura sensori),
- mantenimento dell'orario anche in caso di reset o power loss.

9.1 Schema semplificato

Riferimento	Scheda	Valore consigliato	Descrizione
C_{IN}	**C12**	100 nF	Condensatore di disaccoppiamento alimentazione chip

Riferimento	Scheda	Valore consigliato	Descrizione
C_{BAT}	**C13**	100 nF	Condensatore di disaccoppiamento alimentazione batteria
BAT	**BT1**	CR2032	Batteria (3 V)
R_{SDA}	**R16**	4.7 kΩ	Resistenza di pull-up per segnale I^2C (SDA)
R_{SCL}	**R17**	4.7 kΩ	Resistenza di pull-up per segnale I^2C (SCL)

10 Sensori

Alina Vista integra due tipi di sensori ambientali: 1. un sensore di temperatura/umidità digitale **AM2302** (DHT22). 2. un **sensore capacitivo** di umidità del suolo con uscita analogica.

Per ridurre i consumi, aumentare l'affidabilità e prolungarne la vita operativa, entrambi i sensori vengono alimentati selettivamente tramite un unico stadio di commutazione posto sul loro GND comune, controllato dal **GPIO14** della MCU tramite il collegamento alla base del transistor NPN **Q4** ***2N3904*** *(consigliato)* con in serie resistenza da 10 kΩ.

La logica on/off del sistema è quindi: |GPIO14|Risultato| |–|–| |LOW|Sensori spenti (nessuna lettura)| |HIGH|Sensori accesi (lettura in corso)|

Gli obiettivi del circuito dei sensori sono:

- Misurazione dell'umidità del terreno per irrigazione automatica.
- Acquisizione della temperatura e dell'umidità relativa dell'aria.
- Alimentazione dei sensori solo durante la lettura, per ridurre consumi, drift, corrosione dei sensori e rumori su ADC.

> **N.B.** I circuiti per l'interfacciamento dei sensori con l'MCU e per il loro funzionamento si trovano sulla scheda principale, ma i sensori veri e propri vanno montati esternamente, sono infatti esposti sulla scheda principale i connettori dedicati: * **AM2302** → J6 * **Soil Capacitivo** → J7

11 Schema

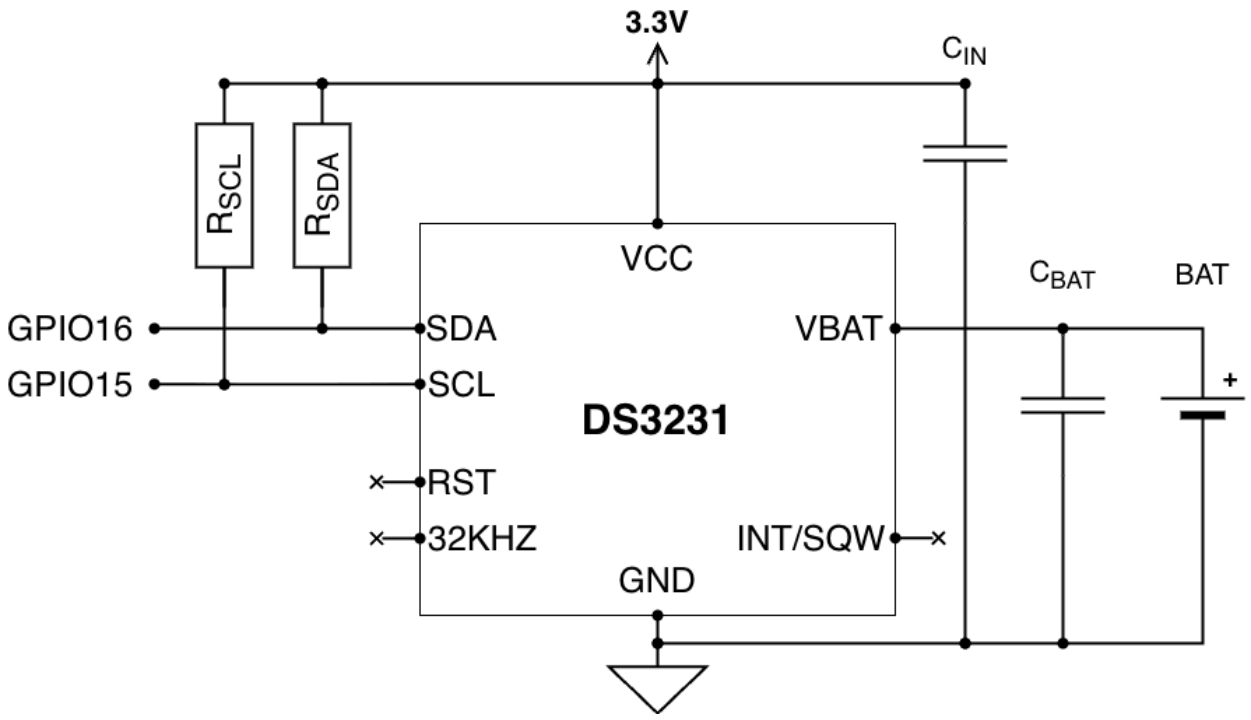

Figura 6: Schema circuito RTC

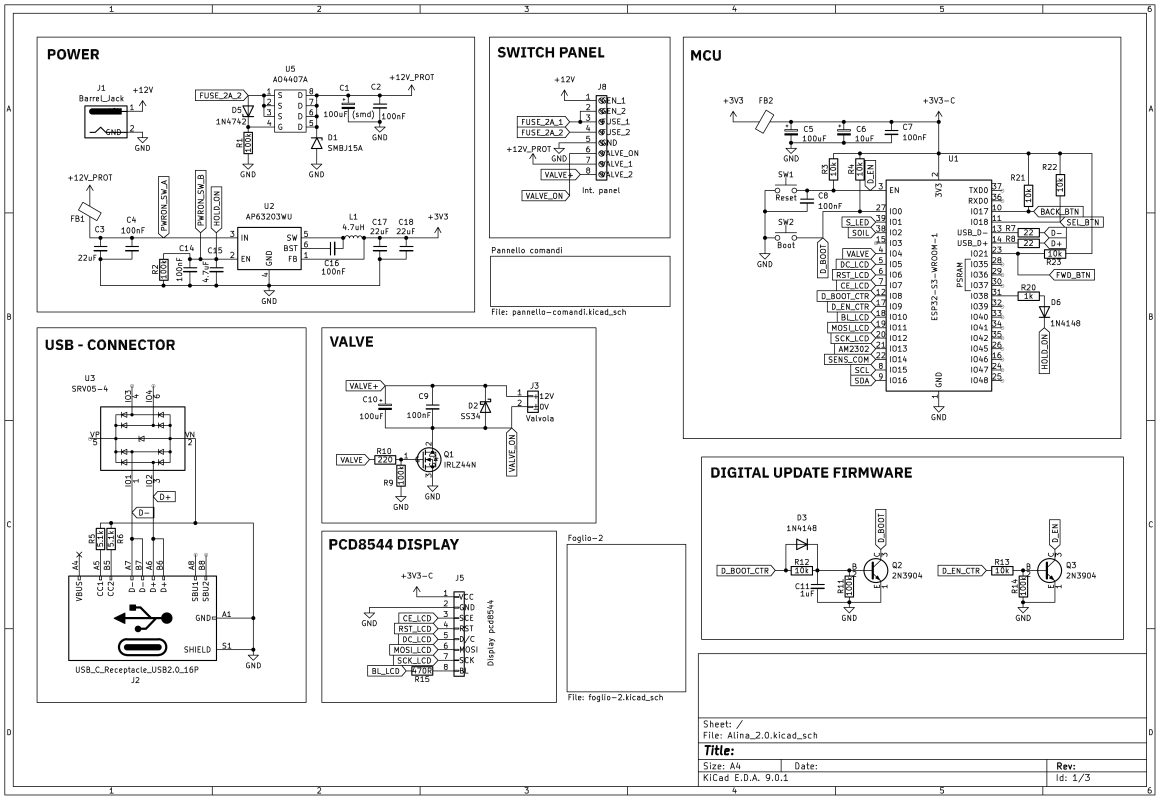

Figura 7: Schema scheda principale - 01

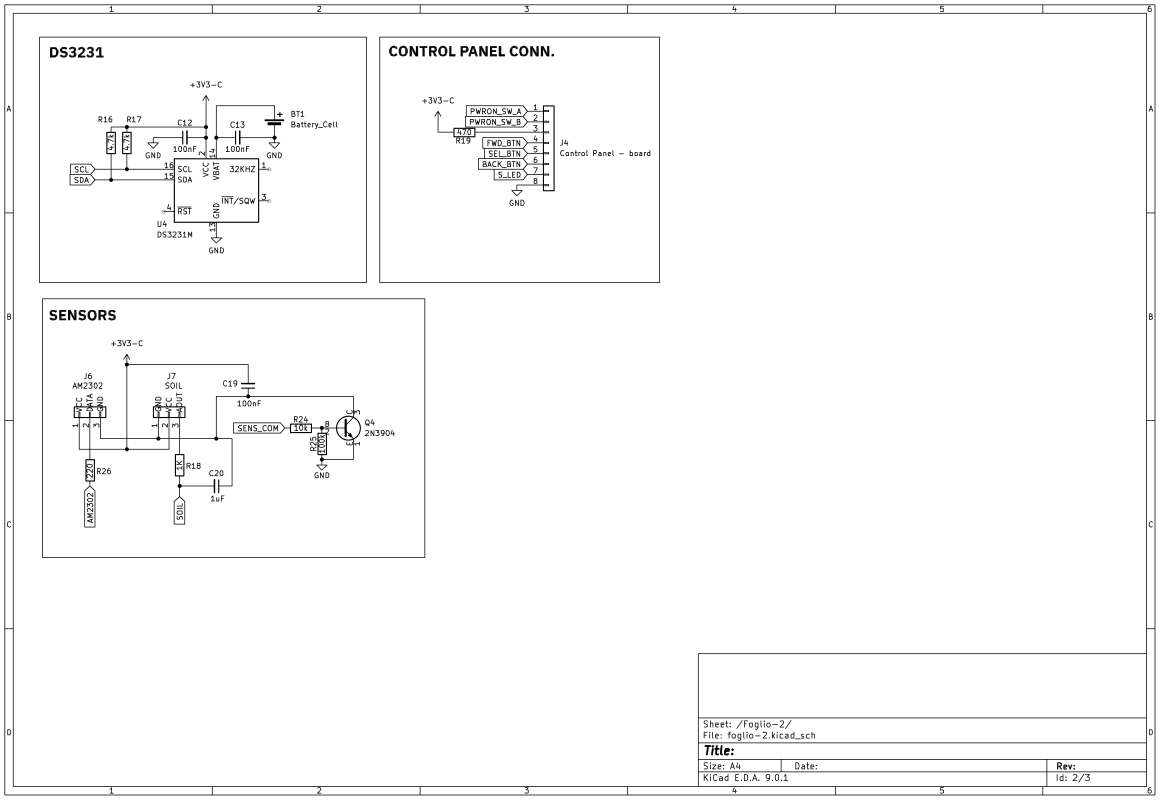

Figura 8: Schema scheda principale - 02

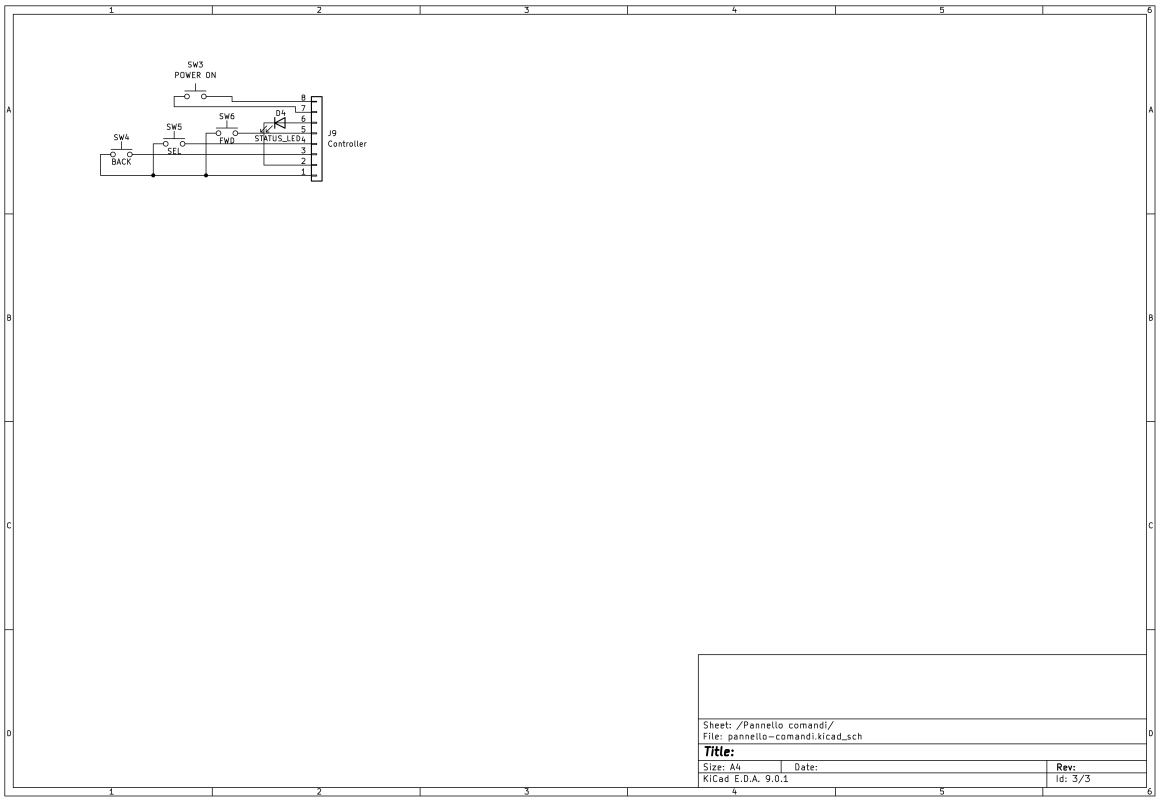

Figura 9: Schema scheda controller

12 PCB

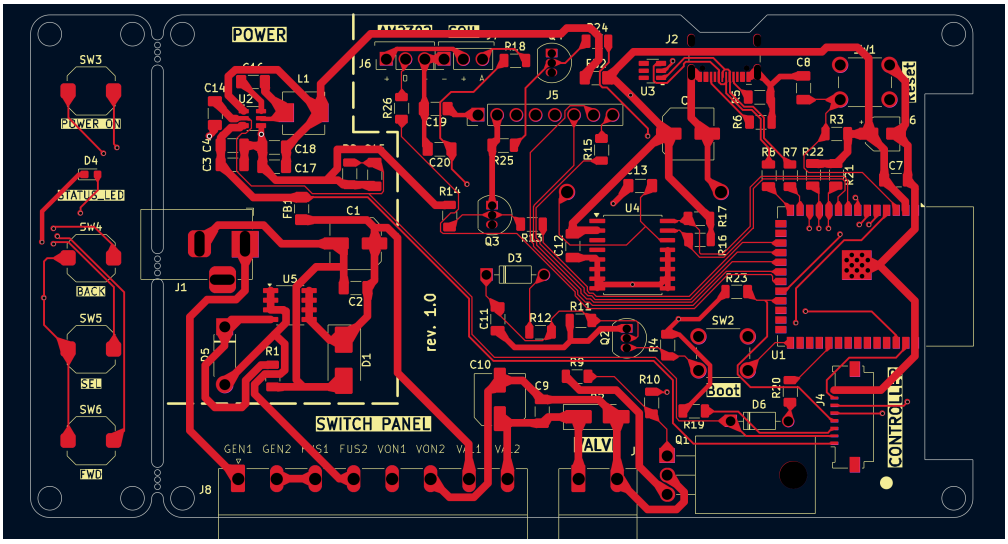

Figura 10: PCB TOP Overview

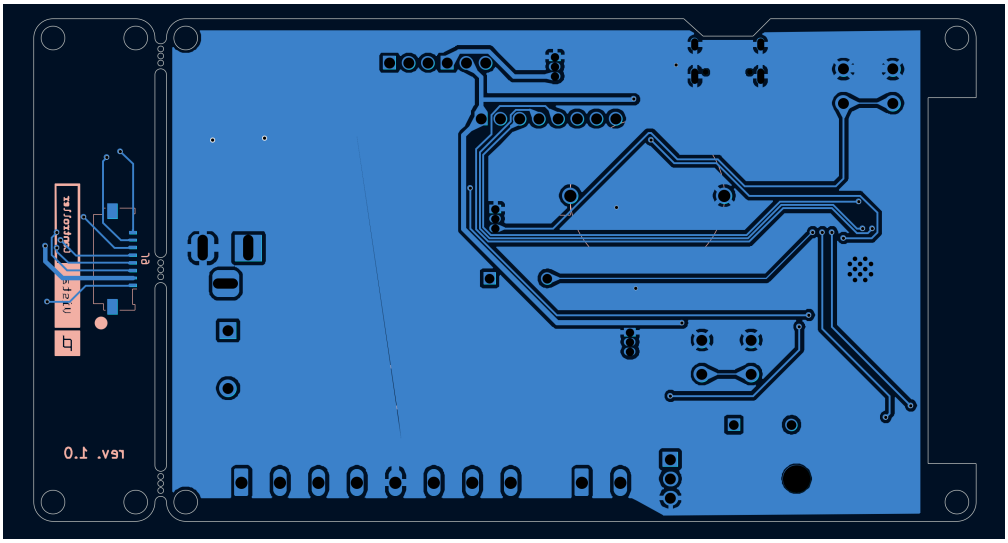

Figura 11: PCB BOTTOM Overview

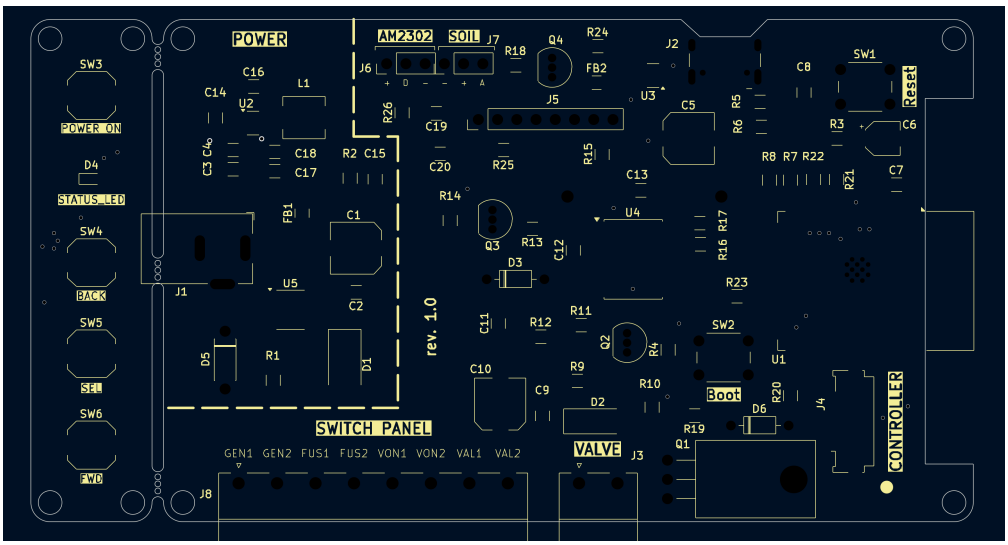

Figura 12: PCB TOP Silk

13 Distinta materiali (BOM)

Designator	Footprint	Quantity	Value
BT1	cr2032	1	Battery_Cell
C1	C_Elec_6.3x7.7	1	100uF (smd)
C10, C5	C_Elec_6.3x7.7	2	100uF
C11, C20	1206	2	1uF
C12, C13, C14, C16, C19, C2, C4, C7, C8, C9	1206	10	100nF
C15	1206	1	4.7uF
C17, C18, C3	1206	3	22uF
C6	CP_Elec_4x5.3	1	10uF
D1	D_SMB_Handsoldering	1	SMBJ15A
D2	D_SMA_Handsolderin	1	SS34
D3, D6	D_DO-35_SOD27_P7.62mm_Horizontal	2	1N4148
D4	0603	1	STATUS_LED
D5	D_DO-41_SOD81_P7.62mm_Horizontal	1	1N4742
FB1, FB2	1206	2	~
J1	BarrelJack_Horizontal	1	Barrel_Jack
J2	USB_C_Receptacle_H TYPE-C-16P-01A	1	USB_C_Receptacle_USB2.0_16
J3	PhoenixContact_MC_1,5_2-G-5.08_1x02_P5.08mm_Horizontal	1	Valvola

Designator	Footprint	Quantity	Value
J4	Molex_200528-0080_1x08-1MP_P1.00mm_Horizo	1	Control Panel - board
J5	PinHeader_1x08_P2.54mm_Vertical	1	Display pcd8544
J6	PinHeader_1x03_P2.	1	AM2302
J7	PinHeader_1x03_P2.54mm_Vertical	1	SOIL
J8	PhoenixContact_MC_G-5.08_1x08_P5.08mm_	1	Int. panel
J9	Molex_200528-0080_1x08-1MP_P1.00mm_Horizontal	1	Controller
L1	L_5.4x5.2mm	1	4.7uH
Q1	TO-220-3_Horizontal_TabDown	1	IRLZ44N
Q2, Q3, Q4	TO-92_Inline		3
R1, R11, R14, R2, R25, R9	1206	6	100k
R10, R26	1206	2	220
R12, R13, R21, R22, R23, R24, R3, R4	1206	8	10k
R15	1206	1	470R
R16, R17	1206	2	4.7k
R18, R20	1206	2	1K
R19	1206	1	470
R5, R6	1206	2	5.1k
R7, R8	1206	2	22
SW1	SW_PUSH_6mm	1	Reset
SW2	SW_PUSH_6mm	1	
SW3	SW_Push_1TS009xxxxxxxx-xxxx_6x6x5mm	1	POWER ON
SW4	SW_Push_1TS009xxxx-xxxx-xxxx_6x6x5mm	1	BACK
SW5	SW_Push_1TS009xxxxxxxx-xxxx_6x6x5mm	1	SEL
SW6	SW_Push_1TS009xxxx-xxxx-xxxx_6x6x5mm	1	FWD
U1	ESP32-S3-WROOM-1	1	ESP32-S3-WROOM-1
U2	TSOT-23-6	1	AP63203WU
U3	SOT-23-6	1	SRV05-4
U4	SOIC-16W_7.5x10.3mm_P1.27mm	1	DS3231M

Designator	Footprint	Quantity	Value
U5	SOP-8_3.76x4.96mm_P1.2	1	AO4407A

14 Immagini 3D scheda

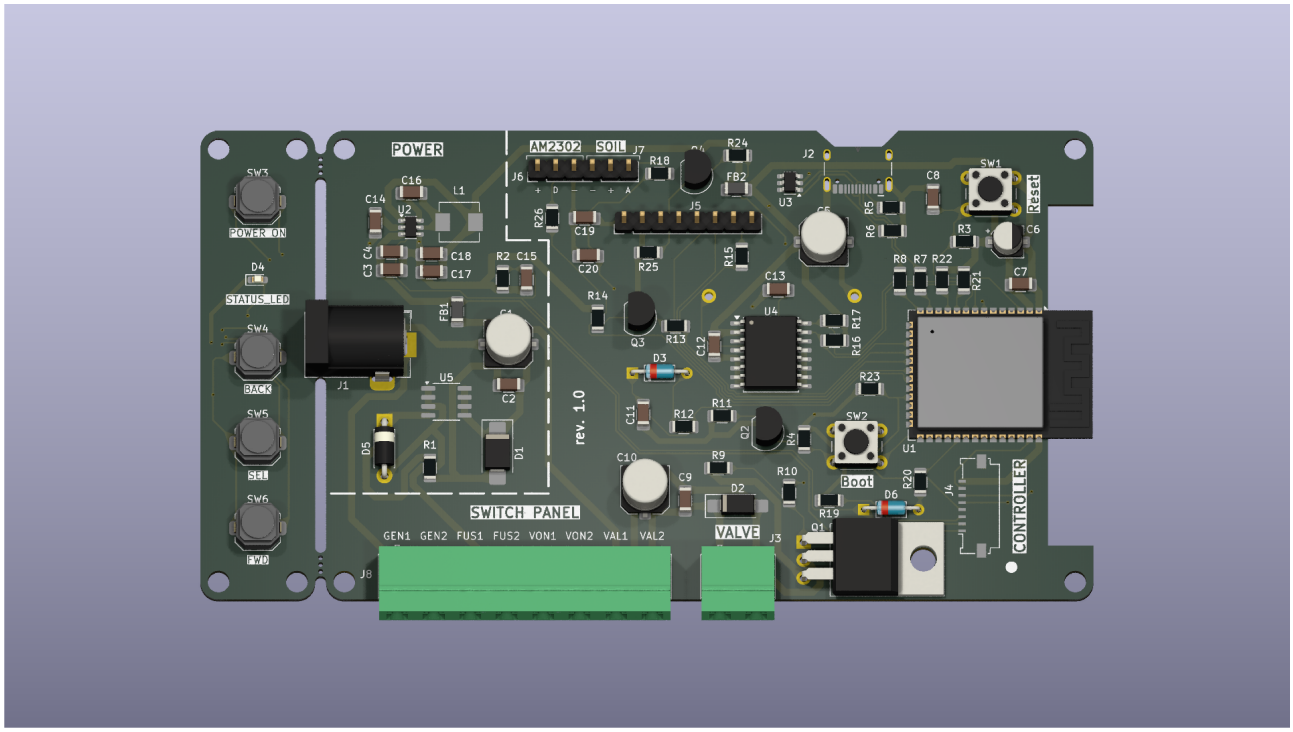

Figura 13: PCB 3D

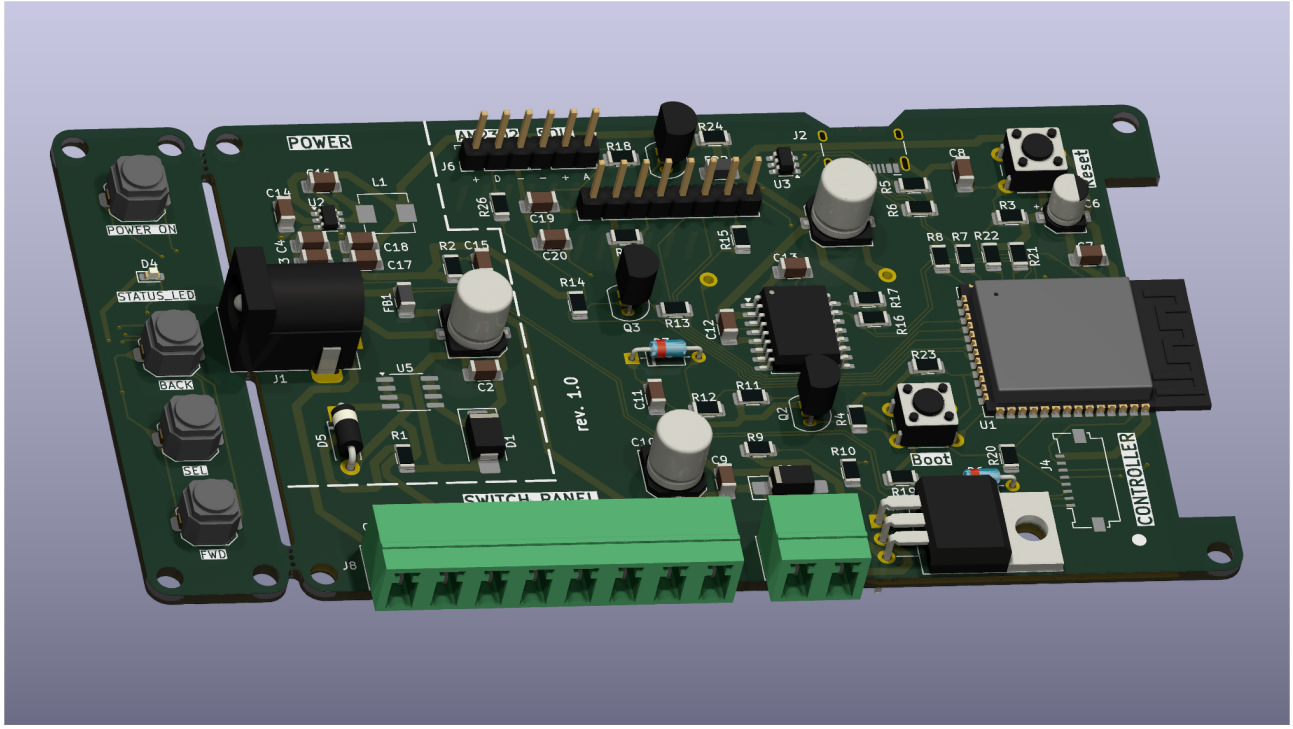

Figura 14: PCB 3D

Disclaimer

Questo prodotto hardware, comprensivo di schede elettroniche, circuiti, sensori, cablaggi, firmware e qualsiasi parte fisica o meccanica correlata (di seguito, "Dispositivo"), è fornito senza alcuna garanzia, esplicita o implicita.

Il Dispositivo:

- Non è un prodotto certificato,
- Non è destinato all'uso industriale, medico, agricolo commerciale, di monitoraggio critico o di sicurezza,
- Non è stato testato secondo normative CE, RoHS, EMC o simili,
- È destinato esclusivamente ad uso sperimentale, hobbistico o educativo.

L'autore/progettista non si assume alcuna responsabilità per danni diretti, indiretti, accidentali o consequenziali, inclusi ma non limitati a:

- guasti, malfunzionamenti, errori di misura,
- danneggiamenti a cose o persone,
- perdita di dati, perdita di raccolti, incendi o cortocircuiti,
- uso improprio, modifiche, montaggi errati o installazioni non conformi.

L'utente è l'unico responsabile:

- dell'installazione,
- dell'alimentazione del Dispositivo,
- dell'uso in ambienti umidi o esterni,
- della protezione del Dispositivo da acqua, insetti, calore, sovratensioni,
- del rispetto di tutte le normative elettriche e ambientali locali.

L’utilizzo del Dispositivo implica l’accettazione completa di questo disclaimer. Se non si accettano queste condizioni, il Dispositivo non deve essere utilizzato.

Licenza

Copyright © Lorenzo Petrecca, tutti i diritti riservati.

Il presente materiale hardware comprende (a titolo esemplificativo ma non esaustivo):

- schemi elettrici,
- layout PCB,
- file Gerber,
- liste componenti (BOM),
- render e immagini 3D,
- modelli di case o parti meccaniche,
- diagrammi, note tecniche e qualsiasi progetto correlato all'hardware della centralina "Alina Vista".

1. **Proprietà esclusiva** Tutti i contenuti hardware sono di proprietà esclusiva dell'autore. Nessun diritto viene ceduto all'utente, salvo quanto esplicitamente indicato in questa licenza.
2. **Uso consentito** È consentito:
   - consultare il materiale,
   - utilizzarlo per scopi personali, educativi o di studio.
3. **Uso vietato** È espressamente vietato:
   - copiare, modificare, redistribuire o pubblicare il materiale hardware,
   - produrre versioni derivate,
   - fabbricare, vendere o commercializzare dispositivi basati su questo progetto,
   - condividere file di progetto, PCB, case, schemi o BOM con terze parti,
   - integrare il progetto in prodotti commerciali o professionali.
4. **Nessuna garanzia** Il materiale è fornito "così com'è", senza alcuna garanzia di correttezza, sicurezza, affidabilità o idoneità per qualsiasi scopo.
5. **Limitazione di responsabilità** L'autore non è responsabile per:
   - danni materiali o personali,
   - perdite economiche,
   - uso improprio,
   - modifiche non autorizzate,
   - implementazioni errate del progetto.
6. **Accettazione** L'utilizzo del materiale implica accettazione integrale di tutte le condizioni.