

PROGETTO "PARTIAMO DA NOI" · PIANO ESTATE

Modulo Smart Tech Lab

Fondi Strutturali Europei – Programma Nazionale "Scuola e competenze" 2021-2027 – Fondo sociale europeo plus (FSE+)
– Priorità 1 – Obiettivo specifico ESO4.6, sotto-azione ESO4.6.A.4.A

Docente esperto

prof. Alessio Atzei

Docente tutor

prof. Lorenzo Podda

Durata

10 lezioni da 3 ore

Il progetto "PARTIAMO DA NOI"

Percorsi educativi e formativi per il potenziamento delle competenze, l'inclusione e la socialità.

Finalità

Potenziare le competenze degli studenti con attività laboratoriali, percorsi innovativi e interventi mirati allo sviluppo delle abilità personali, digitali e disciplinari.

Quadro di finanziamento

Avviso pubblico prot. n. 81652 del 23/05/2025 (seconda finestra), decreti ministeriali 11 aprile 2024 n. 72 e 22 maggio 2025 n. 96.

Destinatari

Studenti e studentesse dell'istituto, con il coinvolgimento di docenti, famiglie e personale ATA.

Riconoscimento delle ore

Le ore svolte sono riconosciute come attività di Formazione Scuola-Lavoro, a condizione di una frequenza minima pari al 75% del monte ore previsto.

Calendario del modulo Smart Tech Lab

Dieci incontri il mercoledì dalle 14:30 alle 17:30, dal 7 ottobre al 9 dicembre 2026.

Lezione 1	Mercoledì 7 ottobre 2026	14:30 – 17:30	Lezione 6	Mercoledì 11 novembre 2026	14:30 – 17:30
Lezione 2	Mercoledì 14 ottobre 2026	14:30 – 17:30	Lezione 7	Mercoledì 18 novembre 2026	14:30 – 17:30
Lezione 3	Mercoledì 21 ottobre 2026	14:30 – 17:30	Lezione 8	Mercoledì 25 novembre 2026	14:30 – 17:30
Lezione 4	Mercoledì 28 ottobre 2026	14:30 – 17:30	Lezione 9	Mercoledì 2 dicembre 2026	14:30 – 17:30
Lezione 5	Mercoledì 4 novembre 2026	14:30 – 17:30	Lezione 10	Mercoledì 9 dicembre 2026	14:30 – 17:30

Docenti del modulo

Docente esperto: prof. Alessio Atzei · Docente tutor: prof. Lorenzo Podda

CORSO DI DOMOTICA · LEZIONE 2

Comandare la luce a 230 V con un pulsante a 5V

Collegiamo un pulsante all' ESP32 e pilotiamo la luce



Il percorso della lezione

Il punto di partenza di oggi: pulsante a 5 V, ESP32, relè e lampada a 230 V.

01

Due mondi elettrici

230 V in alternata per la lampada, 3,3–5 V in continua per l'elettronica: non si toccano mai.

02

Il relè fa da ponte

Un segnale a 5 V chiude il contatto COM–NO: isolamento galvanico fra comando e potenza.

03

L'ESP32 è il cervello

Programmato da Arduino IDE via USB (driver CP2102), pilota il relè dal GPIO 23.

04

Il pulsante a toggle

Su GPIO 18 con INPUT_PULLUP: sul fronte di discesa inverte lo stato, con debounce di 200 ms.

Lo schema: dal pulsante a 5 V alla lampada a 230 V

Il segnale viaggia da sinistra a destra e cambia di mondo solo dentro al relè.



La logica in tre righe

Premo il pulsante → il GPIO passa da alto a basso → l'ESP32 inverte lo stato memorizzato → scrive l'uscita verso il pin IN del relè → il contatto commuta e la lampada si accende.

Il debounce software (circa 20 ms) evita che un solo tocco venga letto come più pressioni.

Collegamenti e codice

Il relè si accende con il pulsante premuto

Il cablaggio del lato comando e lo sketch minimo per far funzionare il circuito.

Cablaggio lato 5 V

Pulsante

Un terminale al GPIO 18, l'altro a GND

Relè · VCC

Pin 5 V (VIN) dell'ESP32

Relè · GND

GND dell'ESP32, massa in comune

Relè · IN

GPIO 23 dell'ESP32

Lato 230 V

Fase su COM (sempre morsetto centrale), uscita da NO alla lampada

Sketch essenziale (Arduino IDE)

```
#define RELAY 23
#define BUTTON 18
void setup() {
  pinMode(RELAY, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);
  Serial.begin(115200);
}
void loop() {
  int stato = digitalRead(BUTTON);

  if(stato == LOW){ //pulsante premuto
    digitalWrite(RELAY, HIGH);
    Serial.println("Relè acceso");
  }
  else{
    digitalWrite(RELAY, LOW);
    Serial.println("Relè spento");
  }
  delay(20); //piccolo debounce
}
```

Il pull-up: perché il pulsante ha bisogno di aiuto

Senza pull-up l'ingresso resta "a mezz'aria" e l'ESP32 legge valori casuali.

Il problema: pin flottante

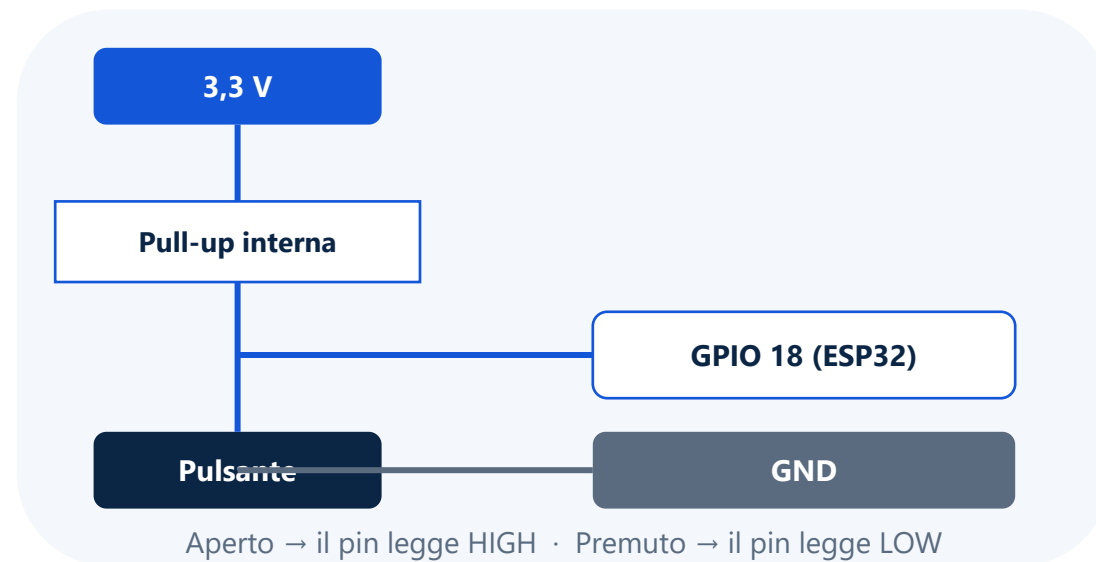
Con il pulsante aperto il GPIO non è collegato a nulla: capta disturbi e legge a caso HIGH o LOW.

La soluzione: pull-up interna

L'ESP32 attiva al suo interno un collegamento verso i 3,3 V che tiene l'ingresso stabile a HIGH quando il pulsante è aperto.

Premendo il pulsante

Il pin viene collegato a GND: la lettura diventa LOW. Logica invertita: premuto = 0.



In pratica sull'ESP32

L'ESP32 ha pull-up interne: basta `pinMode(18, INPUT_PULLUP)` e non serve alcuna resistenza esterna. Nel codice il pulsante risulta premuto quando `digitalRead(18) == LOW`.

Il pulsante a toggle: una pressione accende, la successiva spegne

Il relè non segue il pulsante: cambia stato solo sul fronte di discesa, cioè nell'istante della pressione.

Come ragiona il codice

Stato memorizzato

relayState conserva acceso/spento; lastButton ricorda la lettura precedente del pulsante

Fronte di discesa

Con INPUT_PULLUP il pin è HIGH a riposo: HIGH → LOW significa "appena premuto"

Inversione

relayState = !relayState inverte lo stato e digitalWrite lo porta subito sul relè

Debounce

delay(200) ignora i rimbalzi meccanici del contatto, che darebbero più commutazioni

Aggiornamento

lastButton = currentButton chiude il ciclo: senza questa riga il toggle non funziona

Lo sketch con i commenti

```
#define RELAY 23
#define BUTTON 18

bool relayState = false;
bool lastButton = HIGH;

void setup() {
  pinMode(RELAY, OUTPUT);
  pinMode(BUTTON, INPUT_PULLUP);
  Serial.begin(115200);
}

void loop() {
  bool currentButton = digitalRead(BUTTON);

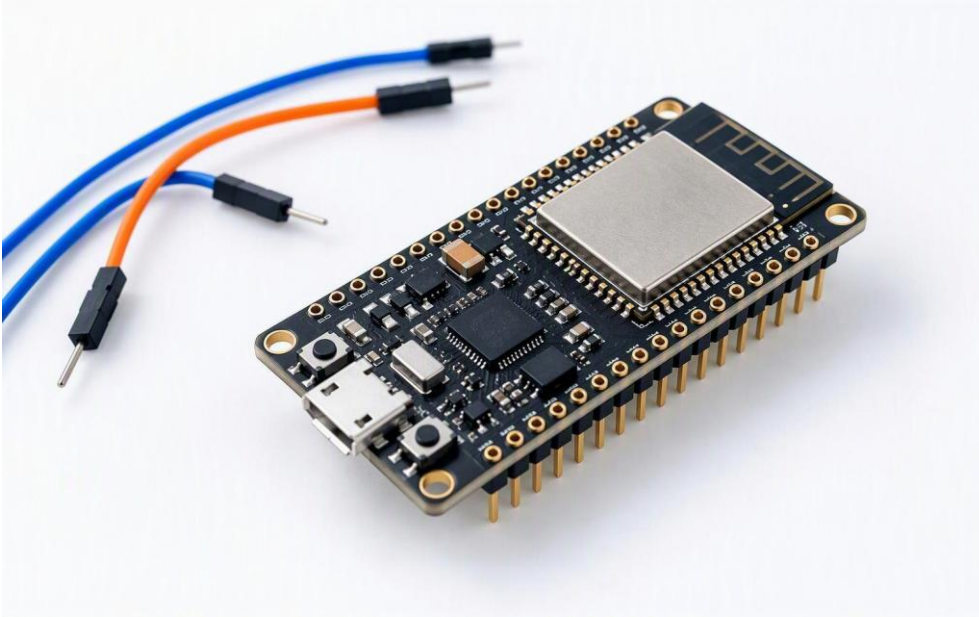
  // rileva la pressione (fronte HIGH -> LOW)
  if (currentButton == LOW && lastButton == HIGH) {
    relayState = !relayState;    // inverte stato
    digitalWrite(RELAY, relayState); // aggiorna relè

    Serial.print("Relè: ");
    Serial.println(relayState ? "ON" : "OFF");
    delay(200); // debounce software 200ms
  }

  lastButton = currentButton;
}
```

Wi-Fi e indirizzo IP: il controllo dal browser

L'IP identifica l'ESP32 nella rete locale. Lo useremo per raggiungere la pagina di comando.



Cosa faremo con l'IP

Creeremo sull'ESP32 una pagina web con i comandi Accendi e Spegni. Dal browser invieremo il comando alla scheda, che piloterà il relè.

Solo 2,4 GHz

Collegiamo la scheda del laboratorio alla rete a 2,4 GHz del router.

Modalità station (STA)

L'ESP32 entra nella rete del router come un telefono e riceve un indirizzo IP.

L'indirizzo della nostra scheda

Il router assegna l'IP via DHCP, per esempio 192.168.1.50. Leggeremo quello reale nel monitor seriale.

Telefono e scheda nella stessa rete

Per la prova colleghiamo PC o smartphone alla stessa rete Wi-Fi dell'ESP32.

Dove scriveremo l'IP

Nel browser: `http://` seguito dall'IP della scheda. Esempio: `http://192.168.1.50`

Anticipazione Lezione 3: Wi-Fi e lettura dell'IP

Primo passo: colleghiamo la scheda e annotiamo il suo IP. La pagina di comando arriverà dopo.

Come funziona lo sketch

1 • Includi la libreria

`#include <WiFi.h>` è già presente nel pacchetto esp32: non va installata nulla di esterno

2 • Metti le credenziali

ssid e password della tua rete, scritti esattamente come sono (maiuscole comprese)

3 • Avvia la connessione

`WiFi.mode(WIFI_STA)` e `WiFi.begin(ssid, password)` fanno partire l'aggancio al router

4 • Aspetta l'aggancio

Il ciclo `while` controlla `WiFi.status()`: stampa un punto ogni mezzo secondo finché non è connesso

5 • Leggi e annota l'IP

Apri il monitor seriale a 115200 baud e annota l'IP stampato da `WiFi.localIP()`.

Sketch: connessione e lettura IP

```
#include <WiFi.h>

const char* ssid    = "Nome_Rete";
const char* password = "Password";

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

    WiFi.mode(WIFI_STA);          // modalita client
    WiFi.begin(ssid, password);

    Serial.print("Connessione");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println();
    Serial.print("Indirizzo IP: ");
    Serial.println(WiFi.localIP());
}

void loop() {
    // per ora vuoto
}
```

Passo successivo: aggiungeremo il server web. Con questo sketch il browser non mostra ancora una pagina.