

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 3

Comandare la luce dal browser con l'ESP32

Nella lezione 2 abbiamo letto l'IP. Oggi creiamo il web server e la pagina con ACCENDI e SPEGNI, poi proviamo due sfide: pulsante unico e CSS.



Il percorso della lezione 3

Punto di partenza: IP letto sul monitor seriale. Traguardo: due collegamenti funzionanti.

01

Recap della lezione 2

Wi-Fi a 2,4 GHz, credenziali corrette e lettura dell'IP a 115200 baud.

02

Il browser incontra il server

Indirizzo IP, porta 80, richiesta HTTP e risposta HTML.

03

Il server e le route

La libreria WebServer collega /, /on e /off alle funzioni C++.

04

La pagina con due collegamenti

ACCENDI e SPEGNI comandano la luce e mostrano lo stato dopo ogni clic.

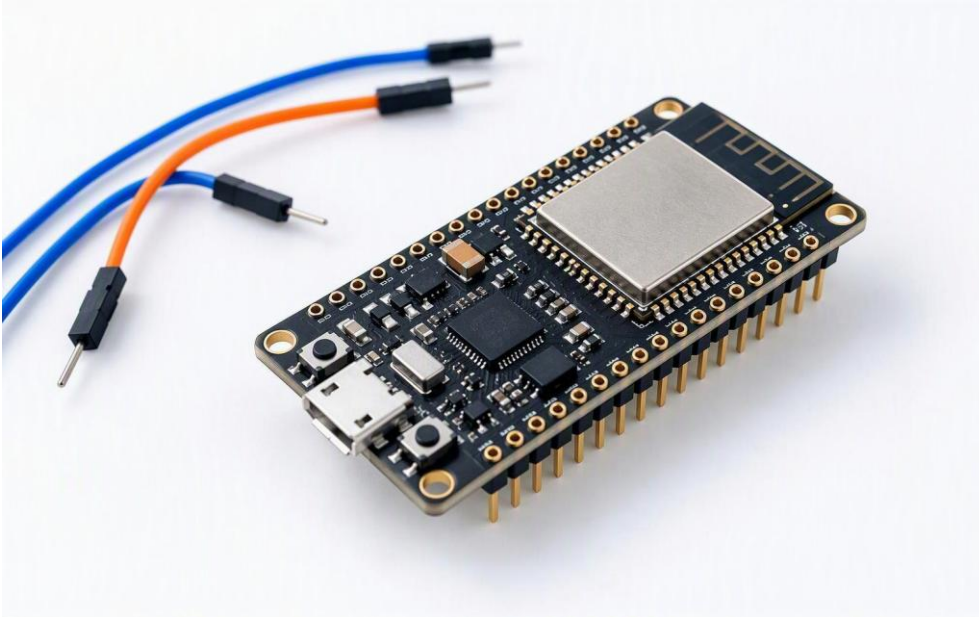
05

Le sfide agli studenti

Prima un solo pulsante web. Poi CSS per una pagina chiara e personale.

Recap veloce: Wi-Fi, IP ed errori comuni

Nella lezione 2 siamo arrivati alla lettura dell'IP. Ripartiamo da questi controlli.



Il punto raggiunto

WiFi.localIP() ci dà l'indirizzo della scheda. Per aprire una pagina serve anche avviare il web server: lo facciamo oggi.

Rete a 2,4 GHz, non a 5 GHz

La scheda ESP32 del laboratorio usa i 2,4 GHz. Controlla bene la banda anche sull'hotspot.

SSID e password esatti

Controlla maiuscole, spazi e nome della rete. In STA la scheda si collega al router.

Monitor seriale a 115200 baud

Se il testo è illeggibile, verifica la velocità. Dopo il caricamento riavvia la scheda.

Dispositivi sulla stessa rete locale

Evita reti ospiti isolate e dati mobili. PC o telefono possono usare anche la banda 5 GHz.

IP attuale, non quello dell'esempio

Rileggi l'IP dopo il riavvio: il router può cambiarlo. Annota quello mostrato dal monitor.

Recap: lo sketch che legge l'IP

Questo è il codice già raggiunto nella lezione 2. Lo teniamo come base per il server.

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 l'IP

WiFi.localIP() stampa sul monitor seriale (115200 baud) l'indirizzo da usare nel browser

Sketch: connessione Wi-Fi

```
#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 1 · Che cos'è un web server

Il browser è il client. L'ESP32 riceve una richiesta e restituisce una risposta.

1 · Indirizzo e percorso

In `http://192.168.1.50/on`, l'IP identifica la scheda e `/on` indica il comando. Usa il tuo IP.

2 · Il server ascolta

`WebServer server(80)` prepara il server sulla porta 80. `server.begin()` avvia l'ascolto HTTP.

3 · La route esegue l'azione

La route `/` mostra la pagina. Le route `/on` e `/off` cambiano lo stato della luce.

4 · La risposta HTML

Il server invia HTML con `ACCESA` o `SPENTA` e due collegamenti. Il browser lo visualizza.

L'idea chiave

Il solo IP non basta: servono il server avviato e `handleClient()` nel loop. Per questa prova basta la rete locale, senza accesso a Internet.

Passo 2 · Il server e le route in C++

Integriamo questi blocchi nello sketch Wi-Fi: un solo setup() e un solo loop().

Come unire il codice

Oggetti e funzioni fuori da setup()

Aggiungi WebServer.h, server, luce e le funzioni.

La variabile luce

Il bool memorizza il comando: true accende, false spegne. Il relè del laboratorio è attivo basso.

Sketch: il server e le route

```
#include <WiFi.h>
#include <WebServer.h>

const int RELAY = 23;
WebServer server(80);      // porta HTTP
bool luce = false;
const char* ssid          = "NOME_RETE";
const char* password      = "PASSWORD";

void aggiorna() {
    digitalWrite(RELAY, luce ? HIGH : LOW);
}

void accendi() { luce = true;  aggiorna(); pagina(); }
void spegni()  { luce = false; aggiorna(); pagina(); }
```

Passo 2 · Il server e le route in C++

Integriamo questi blocchi nello sketch Wi-Fi: un solo setup() e un solo loop().

Come unire il codice

Il setup() già esistente

Inizializza il relè, conserva il blocco Wi-Fi e registra le route dopo la connessione.

Le route e l'avvio

server.on() associa i percorsi alle funzioni. server.begin() avvia il server.

server.handleClient()

Nel loop richiama spesso handleClient(): controlla le richieste ed esegue la route.

Sketch: il server e le route

```
void setup() {  
  pinMode(RELAY, OUTPUT);  
  aggiorna();  
  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());  
  server.on("/", pagina);  
  server.on("/on", accendi);  
  server.on("/off", spegni);  
  server.begin();  
}
```

Passo 3 · La pagina con due collegamenti

Aggiungi pagina() fuori da setup() e loop(). Poi carica lo sketch completo sull'ESP32.

Come è fatta la pagina

Una stringa di HTML

La pagina è semplice testo: si compone in una String e si invia, non serve nessun file

Lo stato in chiaro

Il titolo legge luce e mostra ACCESA o SPENTA al momento della risposta HTTP.

Due collegamenti, ancora senza CSS

`<a href='/on'>ACCENDI</a>` accende. `<a href='/off'>SPEGNI</a>` spegne.

Il meta viewport

Adatta la larghezza della pagina allo schermo del telefono. Lo stile arriverà nella sfida CSS.

server.send(...)

Invia codice 200 (OK), il tipo text/html e il contenuto: il browser disegna la pagina

Sketch: la funzione pagina()

```
void loop() {
  server.handleClient();
}

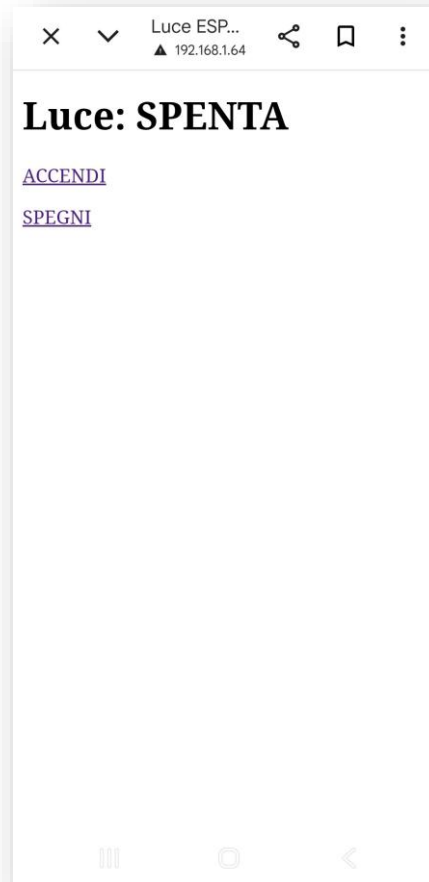
void pagina() {
  String stato = luce ? "ACCESA" : "SPENTA";

  String html =
    "<!DOCTYPE html><html><head>"
    "<meta charset='utf-8'>"
    "<meta name='viewport'"
    "  content='width=device-width'>"
    "<title>Luce ESP32</title></head><body>"
    "<h1>Luce: " + stato + "</h1>"
    "<p><a href='/on'>ACCENDI</a></p>"
    "<p><a href='/off'>SPEGNI</a></p>"
    "</body></html>";

  server.send(200, "text/html", html);
}
```

Passo 3 · La pagina con due collegamenti

Come è fatta la pagina



Prova pratica: ACCENDI e SPEGNI

Obiettivo base raggiunto quando entrambi i collegamenti funzionano dal browser.

01

Apri l'indirizzo IP

Apri `http://` seguito dall'IP letto sul monitor seriale. Devi vedere lo stato e i due link.

02

Clicchi ACCENDI

Il browser invia all'ESP32 una richiesta HTTP per il percorso `/on`.

03

Il server esegue la route

`handleClient()` gestisce `/on` e chiama `accendi()`: luce diventa `true`, poi `aggiorna()` pilota il relè.

04

Il relè commuta

Nel nostro modulo attivo alto, `aggiorna()` scrive `HIGH` sul GPIO 23 per accendere.

05

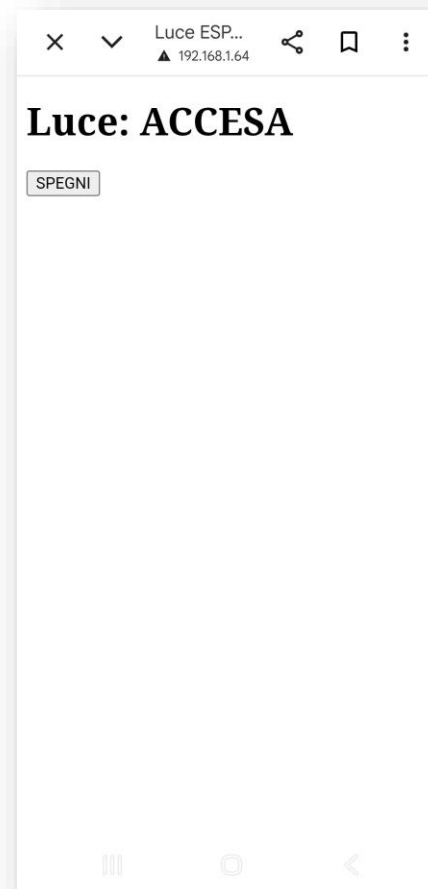
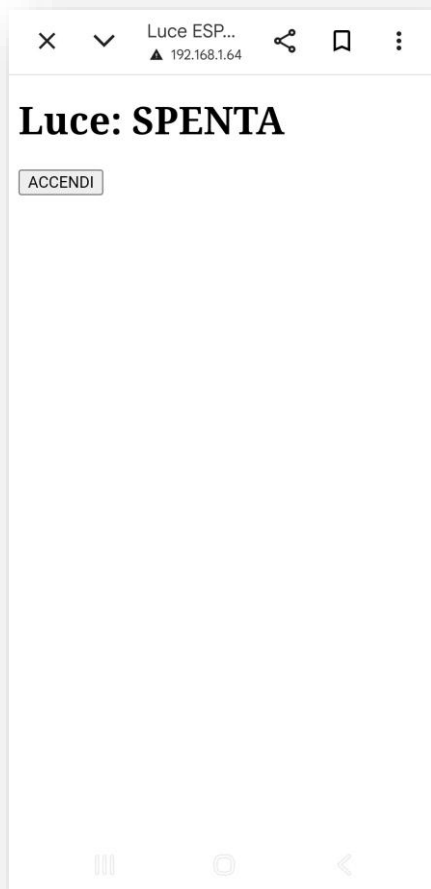
Provi anche SPEGNI

Clicca SPEGNI: luce diventa `false`, il GPIO torna `LOW` e la nuova pagina mostra SPENTA.

Sfida 1 · Un solo pulsante nella pagina

Due collegamenti diventano un unico comando web. Il circuito resta invariato.

Risultato finale



Sfida 1 · Un solo pulsante nella pagina

Due collegamenti diventano un unico comando web. Il circuito resta invariato.

La consegna

Una sola azione: invertire

Crea /toggle: inverte luce, chiama aggiorna() e risponde al browser con pagina().

Un vero pulsante HTML

Sostituisci i due link un <button>. Usa POST verso /toggle.

Testo che cambia

A luce spenta il pulsante dice ACCENDI. A luce accesa dice SPEGNI.

Prova di funzionamento

Con tre pressioni consecutive devi alternare lo stato tre volte, una volta per pressione.

Modifiche al codice

```
void pagina() {
  String stato = luce ? "ACCESA" : "SPENTA";
  String comando = luce ? "SPEGNI" : "ACCENDI";

  String html =
    "<!DOCTYPE html>"
    "<html lang='it'>"
    "<head>"
      "<meta charset='utf-8'>"
      "<meta name='viewport' content='width=device-width, initial-scale=1'>"
      "<title>Luce ESP32</title>"
    "</head>"
    "<body>"
      "<h1>Luce: " + stato + "</h1>"
      "<p>"
        "<a href='/toggle'>"
          "<button> " + comando + "</button>"
        "</a>"
      "</p>"
    "</body>"
    "</html>";

  server.send(200, "text/html; charset=utf-8", html);
}
```

```
void toggle() {
  luce = !luce;
  aggiorna();
  pagina();
}
```

Sfida 2 · Una pagina personale con CSS

Parti dal pulsante unico funzionante. Cambia l'aspetto senza rompere i comandi.

La consegna

CSS dentro <head>

Inserisci `<style>...</style>` nella String HTML, prima di `</head>`. Non servono file esterni.

Una pagina leggibile

Scegli sfondo, colore del testo e font. Lascia spazio intorno al titolo e al comando.

Un pulsante comodo

Personalizza padding, font-size e border-radius. Deve essere facile da premere sul telefono.

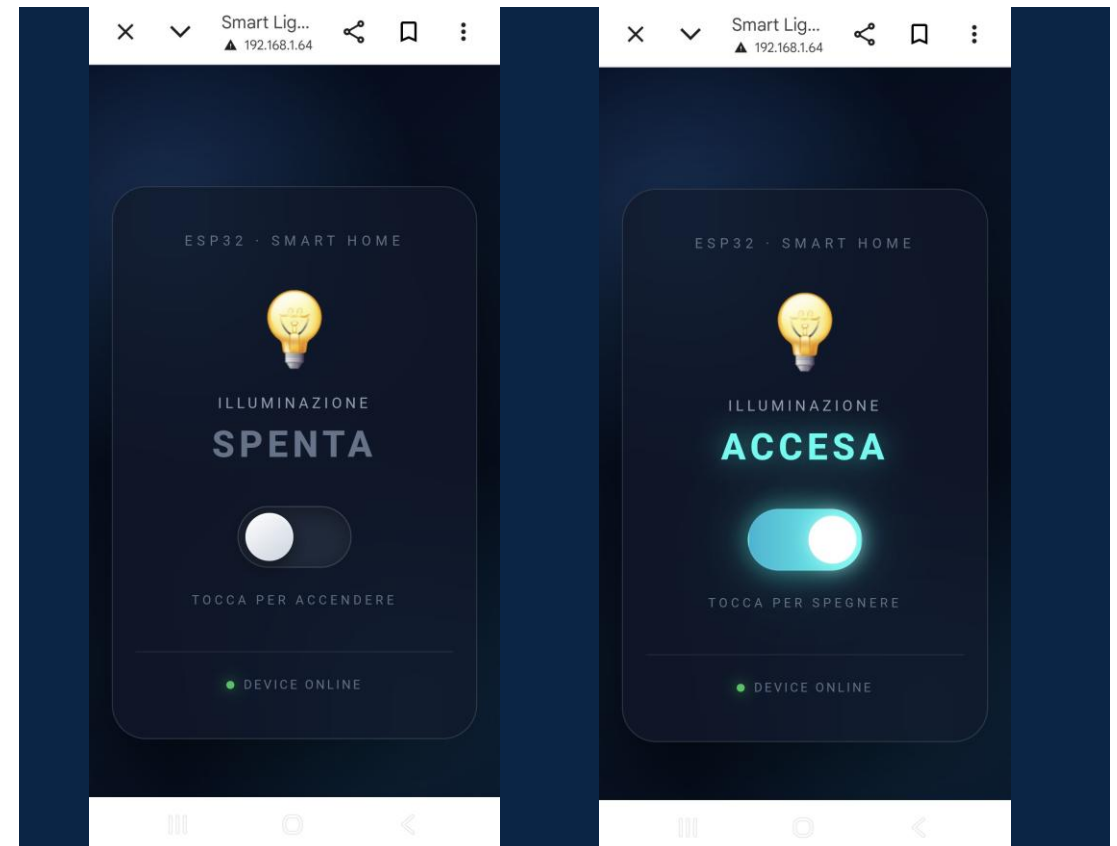
Stato sempre riconoscibile

Mantieni ACCESA o SPENTA in testo. Il colore da solo non basta a spiegare lo stato.

Criteri di riuscita

La pagina funziona da PC e telefono, il testo si legge e il pulsante continua a commutare.

Idee di realizzazione



Approfondimento · Anche il pulsante fisico

Estensione facoltativa: il pulsante della lezione 1 condivide la variabile luce con il browser.

La regola d'oro: una sola variabile

Stato condiviso

Sia il browser sia il pulsante modificano la stessa variabile luce: non esistono due verità diverse

Pulsante su GPIO 18

pinMode(BTN, INPUT_PULLUP): a riposo legge HIGH, premuto legge LOW, senza resistenze esterne

Fronte di discesa

Il cambio HIGH → LOW è l'istante della pressione: solo lì si fa luce = !luce

Debounce semplificato

Il delay(200) evita rimbalzi, ma sospende il server. Con millis() puoi evitare questa pausa.

Stato della pagina

aggiorna() pilota il relè. Dopo un clic fisico ricarica la pagina per leggere il nuovo stato.

Sketch: toggle + server insieme

```
const int BTN = 18;
bool lastBtn = HIGH;

void setup() {
  pinMode(BTN, INPUT_PULLUP);
  // + relè, Wi-Fi e server come prima
}

void loop() {
  server.handleClient(); // comandi dal browser

  bool btn = digitalRead(BTN);
  if (lastBtn == HIGH && btn == LOW) {
    luce = !luce;           // stesso stato del web
    aggiorna();
    delay(200);             // debounce
  }
  lastBtn = btn;
}
```

Problemi comuni con Wi-Fi e server

Se la luce non risponde, quasi sempre il motivo è uno di questi.

Resta su "Connessione..."

Controlla SSID, password e banda 2,4 GHz della scheda o dell'hotspot.

L'IP cambia da solo

È il DHCP del router: riserva l'indirizzo al MAC della scheda o imposta un IP statico

Nessun IP sul monitor

Verifica Serial.begin(115200), velocità del monitor e porta USB. Poi riavvia la scheda.

La luce è invertita

Controlla il tipo di modulo con il docente: nel nostro esempio LOW accende e HIGH spegne.

La pagina non si apre

Usa http:// e l'IP attuale. Verifica stessa LAN, server.begin() e assenza di isolamento.

Il server si blocca

Nel loop ci sono delay troppo lunghi: handleClient() deve essere richiamata spesso

Regola di sicurezza

Il lato 230 V resta esattamente quello della lezione 1: si cabla sempre a impianto scollegato e con i morsetti chiusi in scatola. Le prove di rete si fanno con la lampada già montata e il coperchio chiuso.

Riepilogo della lezione 3

Dal punto raggiunto nella lezione 2 alla pagina web, poi alle due sfide.

Il punto di partenza

Connessione Wi-Fi a 2,4 GHz e IP letto sul monitor seriale: questo era il traguardo della lezione 2.

Obiettivo base di oggi

Aprire la pagina dell'ESP32 e comandare la luce con i collegamenti ACCENDI e SPEGNI.

Sfida 1: un solo pulsante web

Una route /toggle inverte lo stato. Il pulsante propone l'azione corretta: ACCENDI oppure SPEGNI.

Sfida 2: personalizzazione con CSS

Una pagina leggibile, un pulsante comodo e colori personali, senza perdere il funzionamento.

Condivisione finale

Ogni gruppo mostra la pagina e spiega il percorso: clic, richiesta HTTP, funzione C++, comando al relè.