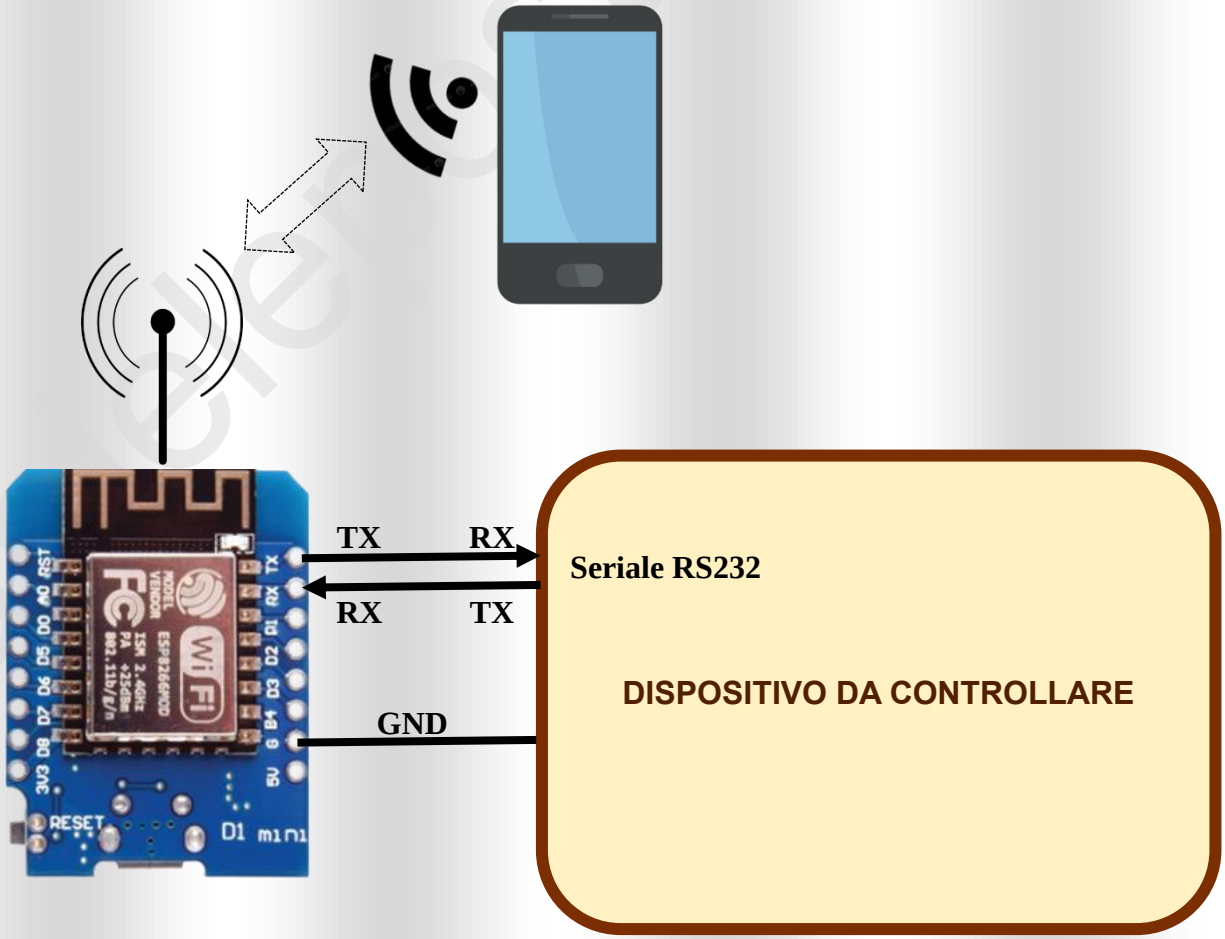
WEB Server con ESP8266 con funzione AP

Una delle applicazioni più comuni è quella di poter gestire da smartphone, PC o tablet, un dispositivo elettronico dotato di connessione WIFI. Ci sono diverse modalità per realizzare questa funzione, una delle più economiche e facilmente implementabile è quella che utilizza il modulo ESP8266. Nel seguente esempio utilizzeremo la scheda D1Mini che al suo interno ha un ESP8266mod, e soprattutto non utilizzeremo alcuna rete wifi esterna, in quanto il modulo ESP8266 ha anche la funzione AP che in pratica gli consente di fornire una rete Wifi a cui i vari client possono connettersi.
Al seguente link è presente un tutorial sempre per realizzare un Webserver con la scheda D1 mini, pertanto la parte iniziale che consente la configurazione della scheda sull'IDE di Arduino rimane la stessa:
https://danielepostacchini.it/wp-content/uploads/2023/02/ESP8266_WebServer.pdf

Un volta che l'IDE di Arduino è stata già predisposta per lavorare con la scheda vediamo come utilizzarla per inviare e ricevere comandi.
Immaginiamo di avere un nostro dispositivo che può ricevere comandi da un'interfaccia seriale RS232, il nostro Webserver potrebbe semplicemente aggiungersi a quanto è già esistente inviando i comandi ricevuti da un client sulla seriale.

La struttura del sistema sarà la seguente:



danielepostacchini.it

Supponiamo che il nostro dispositivo debba ricevere i seguenti 5 comandi:

- *START*
- *STOP*
- *CARICA*
- *AZZERA*
- *VALORE*

Supponiamo che i primi 4 siano comandi semplici e l'ultimo "VALORE" sia un comando a cui dovrà seguire un valore numerico da inviare al dispositivo.

L'idea è quella di realizzare un unico programma a bordo della scheda D1 Mini, il programma invierà al client connesso alla rete Wifi creata dalla scheda, una pagina HTML con i form necessari per inviare i comandi ed il valore e con uno spazio dove leggere le stringhe di risposta.

Sul client sarà sufficiente aprire il browser e digitare l'indirizzo http://192.168.4.1, in questo modo si aprirà la pagina HTML presente a bordo della scheda D1 mini con tutti i comandi previsti.

Per agevolare l'accesso al dispositivo si potrebbe anche creare un QRCode con le applicazioni disponibili che dovrà semplicemente contenere l'URL ***http://192.168.4.1****.*

Si potrebbe anche realizzare un APP ma sarebbe totalmente inutile in quanto dovrebbe solo contenere l'indirizzo della scheda D1 mini.

Vediamo ora come strutturare il programma per l'IDE di Arduino.

La prima parte è l'inclusione delle librerie necessarie.

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
```

La prima libreria ESP8266WIFI.h consente di gestire il WIFI sia modalità client che AP come in questo caso.
La seconda libreria ESP8266WebServer.h consente di gestire un server WEB che risponde alle richieste http.

Successivamente vanno indicate le credenziali della rete Wifi che verrà creata in due costanti:

```
const char* ssid = "Nome-Rete";
const char* password = "12345678";
```

Poi va stanziato l'oggetto ***server*** *della classe* ***ESP8266WebServer*** *che lavorerà sulla porta* ***80****, in pratica è come se si attivasse il server. Il nome potrebbe anche essere differente, si potrebbe anche chiamare* ***mioserver*** *o in altro modo.*

```
ESP8266WebServer server(80);
```

Ora va creata una stringa ***logMessage*** *che servirà per contenere tutti i dati inviati e ricevuti, inizialmente la stringa sarà vuota.*

```
// Buffer messaggi
String logMessages = "";
```

Il prossimo passo è quello di creare una funzione che restituisce una stringa che conterrà la pagina in HTML.

```
String getPage() {
  String html = R"rawliteral(
    .
    PAGINA HTML
    .
)rawliteral";
  return html;
}
```

La pagina potrebbe essere scritta in HTML o si potrebbe aggiungere anche il CSS per abbellirla e colorare i pulsanti. Nel nostro caso utilizziamo la soluzione più semplice con una pagina in HTML.
All'inizio della funzione viene creata una variabile stringa chiamata ***html,*** *questa variabile conterrà la pagina ed il costrutto* ***R"rawliteral*** *serve a dichiarare che il contenuto tra le parentesi è una stringa grezza (RAW) pertanto non ci saranno caratteri speciali come fine linea o carriage return.*

La pagina HTML che andremo a realizzare conterrà il titolo, i 5 pulsanti allineati, lo spazio per digitare ed inviare il valore, e lo spazio per i messaggi ricevuti.

La pagina HTML comincia col TAG DOCTYPE e con il tag html che verrà chiuso alla fine, e con il titolo ***ESP8266 Control*** *tra i tag HEAD e TITLE aperti e chiusi.*
Poi aprendo il TAG BODY si va a digitare il titolo da visualizzare, ***Controllo ESP8266****.*

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>ESP8266 Control</title>
</head>
<body>
  <h2>Controllo ESP8266</h2>
```

Poi vengono inseriti i 5 pulsanti dentro al TAG ***form****, che utilizzerà nei 5 casi il metodo GET.*
In questo caso ogni volta che si fa clic su un pulsante verrà inviata una richiesta di tipo GET contenente una campo del form con nome ACTION ed un valore che è differente per ogni pulsante, più avanti capiremo meglio perché ACTION è comune.
Perciò dentro a value va inserito il comando che si vuole inviare, dopo il simbolo maggiore invece ciò che si vuole visualizzare sul browser.
Ad esempio cliccando su START la richiesta del browser sarà ***http://192.168.4.1/cmd?action=START***

```
<form action="/cmd" method="GET">
  <button name="action" value="START">START</button>
  <button name="action" value="STOP">STOP</button>
  <button name="action" value="CICLO">CICLO</button>
  <button name="action" value="CARICA">CARICA</button>
  <button name="action" value="AZZERA">AZZERA</button>
</form>
```

Dopo la scritta "Invia Valore" va inserito il form per inviare il VALORE.
Anche in questo caso si usa il metodo GET, l'input sarà di tipo testo ed il nome inviato sarà ***valore*** *seguito dal numero inserito.*
Ad esempio con il valore 123 la richiesta del browser sarà: ***http://192.168.4.1/valore?valore=123***

```
<h3>Invia Valore</h3>
<form action="/valore" method="GET">
  <input type="text" name="valore" placeholder="Inserisci valore">
  <button type="submit">Invia</button>
</form>
```

Subito sotto al campo dove inserire il valore, c'è il pulsante per cancellare i messaggi di log
Il relativo form contiene solo un pulsante che se Cliccato Invierà sul browser la seguente stringa: ***http://192.168.4.1/clear?***

```
<form action="/clear" method="GET">
  <button type="submit">Cancella Log</button>
</form>
```

Al termine prima dei tag di chiusura abbiamo la sezione messaggi, con la scritta ***Log Messaggi****.*
Con il TAG div, andiamo ad identificare un contenitore dove inserire e visualizzare i messaggi ricevuti.
*L'identificativo dei messaggi di Log che vengono visualizzati è "****log****" ed inizialmente vale la stringa "Nessun messaggio" compresa tra i tag.*

```
  <h3>Log Messaggi</h3>
  <div id="log">Nessun messaggio</div>

  <script>
    async function updateLog(){
      let res = await fetch('/log');
      let text = await res.text();
      document.getElementById('log').innerText = text;
    }
    setInterval(updateLog, 1000);
  </script>
</body>
</html>
```

Nello script invece c'è un aggiornamento dei messaggi ricevuti ogni secondo. L'HTML ***NON*** *è un linguaggio di programmazione, pertanto per eseguire l'automatismo necessario per aggiornare i messaggi ricevuti, occorre realizzare uno script.*
All'interno dello script, viene creata una funzione asincrona che si chiama ***updateLog****.*
Una volta che il browser riceve lo script, lo caricherà nella sua memoria e lo eseguirà.
In questo caso con ***setInterval****, il browser invierà periodicamente (ogni 1000 msec) delle richieste al server.*
Lo script è scritto in Javascript, ed utilizzera l'identificativo creato prima con il nome di ***log.***
All'interno dello script ci sono 3 righe:

- ***let res = await fetch('/log');*** *con questa riga si invia al server una richiesta HTTP di tipo GET indirizzata all'endpoint /log. Perciò dal broswer partirà la stringa* ***http://192.168.4.1/log*** *la risposta sarà contenuta dall'oggetto* ***res*** *e con* ***await*** *si sospende la funzione fino all'arrivo della risposta.*
- ***let text = await res.text();*** *con questa riga si legge tutta la risposta contenuta in* ***res*** *e la si converte in stringa: messaggio1 /n - messaggio2 /n – ecc…*
  anche in questo caso con ***await*** *si sospende la funzione fino all'arrivo della risposta.*
- ***document.getElementById('log').innerText = text;*** *con questa riga si sostituisce il contenuto di* ***log*** *con il contenuto di* ***text*** *che contiene la stringa ricevuta.* ***Innertext*** *serve per interpretare il testo in maniera grezza, eventuali tag HTML non verranno considerati tali.*

Ora che abbiamo analizzato la pagina HTML, andiamo a vedere come interagisce con il resto del programma.
Analizziamo innanzitutto il SETUP.

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

  // Avvio Access Point
  WiFi.softAP(ssid, password);
  Serial.print("AP avviato. IP: ");
  Serial.println(WiFi.softAPIP());

  // Routing
  server.on("/", handleRoot);
  server.on("/cmd", handleCommand);
  server.on("/valore", handleValore);
  server.on("/log", handleLog);
  server.on("/clear", handleClear);

  server.begin();
}
```

Nel setup viene inizializzata la seriale con la velocità di 115200 bps, viene avviato l'Access Point con la funzione WiFi.softAP a cui vengono passati i dati della rete WiFi.
I dati della rete WiFi vengono poi inviati sulla seriale.

La funzione ***server.on*** *serve ad indicare una rotta all'****endpoint*** *della richiesta al server WEB.*
Ad esempio se arriva una richiesta all'endpoint CMD:
http://192.168.4.1/cmd
verrà avviata la funzione che gestirà questa richiesta.
La sintassi è la seguente:
server.on("/percorso", funzioneGestore);
Perciò si dovranno creare 5 handle (funzioni che gestiscono la richiesta) abbinati ai vari ***endpoint****.*
Se non viene indicato l'endpoint come nel primo caso, ci si riferisce alla richiesta http://192.168.4.1.
Server.begin() avvia il funzionamento del server WEB.

Analizziamo ora le 5 funzioni (handle) abbinate alle richieste http.

```
void handleRoot() { server.send(200, "text/html", getPage()); }
```

La funzione handleRoot() esegue una sola istruzione server.send che invierà la pagina html al client.
Questa è una funzione della libreria ESP8266WebServer, ed invia 3 argomenti al client:

server.send(statusCode, contentType, content);

Il primo argomento è lo ***statusCode*** *che in questo caso vale* ***200****, nel protocollo HTTP significa che la richiesta è stata gestita con successo.*

Il secondo argomento è il ***contentType*** *che indica al Browser del Clent come interpretare la risposta, ad esempio potrebbe valere:*

- ***text/html****: per i documenti HTML.*
- ***text/plain****: per i file di testo semplice.*
- ***image/png****: per le immagini in formato PNG.*
- ***application/json****: per i dati strutturati in formato JSON.*

Nel nostro caso l'argomento è ***text/html*** *ed indica che verrà inviata una pagina in HTML.*

Il terzo argomento è la stringa contenente la pagina HTML e questa stringa, come abbiamo detto in precedenza, viene restituita dalla funzione getPage() descritta all'inizio.
Perciò appena arriva la richiesta all'indirizzo http://192.168.4.1 viene restituita e visualizzata sul Browser del Client la pagina HTML.

Una volta visualizzata la pagina HTML, l'utente potrà cliccare sui pulsanti ed far partire altre richieste http al server.

Cliccando uno dei 5 pulsanti START-STOP-CICLO-CARICA-AZZERA, partirà una richiesta di tipo GET, contenente un campo chiamato action, che può contenere uno dei 5 comandi. Questa richiesta viene inviata all'endpoint /cmd. Cliccando sui 5 pulsanti partiranno rispettivamente le seguenti richieste:

- ***http://192.168.4.1/cmd?action=START***
- ***http://192.168.4.1/cmd?action=STOP***
- ***http://192.168.4.1/cmd?action=CICLO***
- ***http://192.168.4.1/cmd?action=CARICA***
- ***http://192.168.4.1/cmd?action=AZZERA***

E la funzione che gestirà questa richiesta è quella abbinata all'endpoint /cmd.

```
void handleCommand() {
  if (server.hasArg("action")) processCommand(server.arg("action"));
  server.sendHeader("Location", "/");
  server.send(303);
}
```

All'interno della funzione la prima cosa che viene fatta è quella di verificare se c'è un campo chiamato ***action****.*

if (server.hasArg("action"))

Nella pagina HTML, ogni pulsante invia lo stesso campo chiamato action, in questo modo possiamo unificare ciò che deve avvenire se si preme uno dei 5 pulsanti, ed eseguire per tutti e 5 i pulsanti la stessa funzione:

processCommand(server.arg("action"));

Viene chiamata la funzione ***processCommand*** *a cui viene passato il risultato di* ***server.arg("action")*** *che rappresenta il contenuto del campo action.*

Perciò premento un pulsante si invia alla funzione ***processCommand*** *il comando abbinato al pulsante.*

Alla funzione processCommand, viene inviato il comando, ma la funzione potrebbe ricevere un secondo parametro chiamato value, come vedremo successivamente con il pulsante VALORE.

```
// Gestione comandi
void processCommand(String cmd, String value = "") {
  if (cmd == "VALORE") {
    Serial.print(cmd);
    Serial.print(" ");
    Serial.println(value);
    logMessages += "→ " + cmd + " " + value + "\n";
  } else {
    Serial.println(cmd);
    logMessages += "→ " + cmd + "\n";
  }
}
```

Premendo uno dei 5 comandi, la funzione riceverà solamente il comando nella variabile chiamata ***cmd*** *, lo stesso comando essendo diverso da "VALORE", viene poi inviato via seriale con l'else.*

Inoltre il comando viene concatenato alla stringa ***logMessages*** *insieme al simbolo di una freccia ed al comando ASCII nuova linea. La stringa logMessages, verrà poi letta e visualizzata dal Browser, tramite lo script visto precedentemente.*

Ad esempio premendo i pulsanti START, STOP e poi CARICA in basso vedremo il log dei comandi inviati.

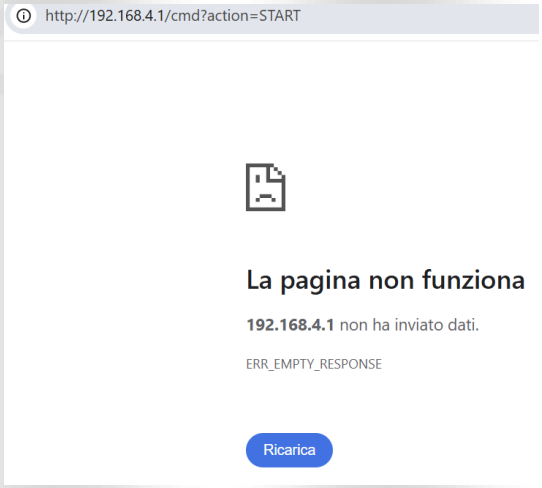
All'interno della funzione handleCommand (come anche dentro le altre) ci sono anche le seguenti righe:

- *server.sendHeader("Location", "/");*
- *server.send(303);*

La prima serve per dire al Browser di andare alla pagina home /, la seconda invia il codice 303 (redirect) per far eseguire al Browser quanto richiesto.

In pratica le due righe servono per riportare il Browser alla pagina principale.

Senza queste due righe avremmo il browser nella situazione descritta nell'immagine.



Inserendo le due righe invece il browser effettuerà una nuova chiamata della pagina HTML.

Le due righe verranno inserite in ogni funzione handle.

Se si clicca sul pulsante valore, il form utilizzato avrà un endpoint diverso che sarà /valore. Ed in questo caso partira la funzione handleValore();

```
<form action="/valore" method="GET">
  <input type="text" name="valore" placeholder="Inserisci valore">
  <button type="submit">Invia</button>
</form>
```

```
void handleValore() {
  if (server.hasArg("valore")) processCommand("VALORE", server.arg("valore"));
  server.sendHeader("Location", "/");
  server.send(303);
}
```

*Anche in questo caso verrà lanciata la funzione **processCommand**, a cui si passeranno due valori, la stringa "VALORE" ed il valore numerico scritto nel campo valore. La funzione eseguirà quanto scritto nell'IF ed invierà sulla seriale il comando ed il valore associato, e modificherà la stringa **logMessages**.*

La funzione che verrà eseguita nel caso dell'endpoint /log è la seguente:

```
void handleLog() { server.send(200, "text/plain", logMessages); }
```

In questo caso si invierà al Browser il contenuto della stringa logMessages che verrà visualizzata nell'apposito campo.

Se invece l'endpoint è /clear, cioè nel caso di attivazione del pulsante clear, la funzione sarà la seguente:

E verrà azzerata la stringa logMessages.

```
void handleClear() {
  logMessages = "";
  server.sendHeader("Location", "/");
  server.send(303);
}
```

---

*A questo punto rimane solo da analizzare il ciclo **loop**.*

*Con la funzione **server.handleClient()** si controllano preiodicamente eventuali richieste provenienti dal Client.*
In caso di una richiesta si confronta il suo contenuto con gli endpoint registrati nel setup e si esegue la relativa funzione associata come descritto in precedenza.

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

  // Lettura da seriale
  if (Serial.available()) {
    String incoming = Serial.readStringUntil('\n');
    incoming.trim();
    if (incoming.length() > 0) {
      logMessages += "← " + incoming + "\n";
    }
  }
}
```

Nel loop vengono anche letti eventuali valori ricevuti dalla seriale (provenienti perciò dal dispositivo collegato alla scheda D1 Mini).
*I dati ricevuti dalla seriale vengono letti fino al carattere \n e successivamente pulita da caratteri speciali con la funzione **incoming.trim().***
*Al termine se la stringa ha un numero di caratteri superiore a 0, viene concatenata con la stringa **logMessages** che verrà poi letta ogni secondo dal Browser.*

Si potrebbe anche realizzare una pagina HTML graficamente più accattivante utilizzando CSS insieme all'HTML.

Al seguente link si può scaricare il programma di esempio descritto nel tutorial:
http://danielepostacchini.it/wp-content/uploads/2023/03/WebServerAP_ESP8266.zip

Al seguente link si può invece scaricare un programma analogo con una pagina realizzata utilizzando anche CSS:
http://danielepostacchini.it/wp-content/uploads/2025/08/WebServerAP_ESP8266_CSS.zip