WEB Server con Display OLED

PREMESSA

Ormai da tempo questo tipo di display sono presenti in diversi dispositivi e rappresentano una valida alternativa ai precedenti e più complessi display TFT a matrice attiva.

I display TFT (Thin Film Transistor cioè transistor a pellicola sottile), sono essenzialmente dei display a cristalli liquidi con una retroilluminazione (luce bianca). I singoli pixel sono costituiti da 3 transistor che controllano la luce che viene fatta passare su 3 filtri RGB. In pratica il singolo colore viene generato dalla combinazione dei 3 colori RGB che si ottengono facendo passare quantità di luce bianca differente su ogni filtro colore.
Il principio di emissione della luce è comunque basato su una fonte di luce bianca, chiamata retroilluminazione.

I display OLED (Organic Light Emitting Diode) invece sono formati da una matrice di pixel che è in grado di generare autonomamente una luce colorata, non c'è perciò bisogno della retroilluminazione.
La luce ed il colore vengono generati mediante l'interazione di molecole a base di carbonio sotto tensione, da qui la parola Organic.
Il principio utilizzato è quello dell'elettroluminescenza, cioè la capacità di alcuni materiali di emettere luce quando sono attraversati da corrente elettrica. Il colore della luce emessa può essere modificato introducendo degli elementi droganti, che consentono di generare i 3 colori fondamentali Rosso, Verde e Blu.
Anche nei display OLED ci saranno per ogni pixel 3 elementi che generano i 3 colori, ma in questo caso lo fanno in maniera autonoma senza alcuna sorgente di luce.

Ovviamente le due tecnologie presentano pro e contro, i TFT hanno una maggiore durata a discapito però della qualità dell'immagine, gli OLED invece presentano colori più nitidi e riescono a creare un nero assoluto e di conseguenza contrasti più elevati.

Nel mondo dell'elettronica di consumo, si trovano oggi entrambi i tipi di display, anche se gli OLED stanno pian piano conquistando una maggiore fetta di mercato.

Gli OLED sono anche disponibili in un formato economico con piccoli schermi monocolore, utili alla realizzazione di svariati progetti con i vari microcontrollori.
In questo tutorial vedremo un comunissimo display OLED che monta un driver SSD1306 e che viene fornito in due formati dalla Adafruit, un'azienda americana operante nel settore dell'elettronica open source.
Due formati molto comuni sono i seguenti:

0,91 inch
128x32 pixel

0,96 inch
128x64 pixel

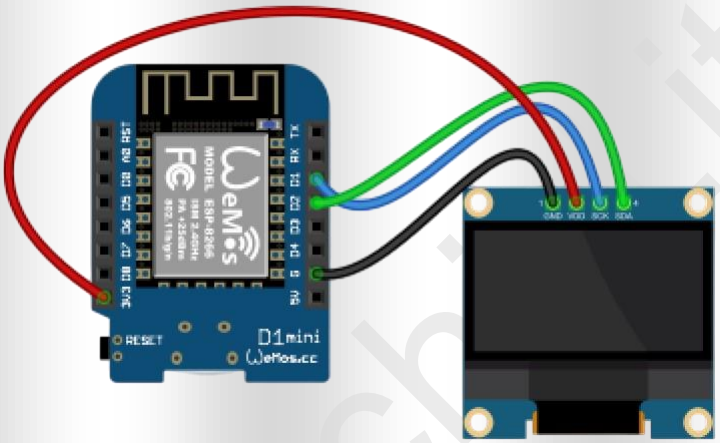
Sono entrambi display monocolore funzionanti su bus I2C ed hanno un costo molto basso dell'ordine dei 2 euro.

COLLEGAMENTO

Per realizzare un WEB Server dotato di un piccolo display possiamo utilizzare la scheda Wemos D1Mini già vista in altri tutorial, che monta a bordo un modulo ESP8266, in grado di connettersi ad una rete WiFi o di offrire una propria rete.

Il collegamento col display OLED è molto semplice in quanto utilizza solamente il bus I2C, presende sui pin D1 e D2 della scheda D1 mini.

Scheda D1Mini	***Display OLED***
3v3	*VCC*
GND	*GND*
D1	*SCL – CLOCK*
D2	*SDA - DATI*



Tutto il sistema potrà essere alimentato direttamente dal connettore USB della scheda Di Mini, o mediante un'alimentazione a 5Volt esterna da portare sul pin 5V della scheda D1 Mini.

PROGRAMMAZIONE ED AMBIENTE DI SVILUPPO

Per la programmazione del sistema utilizzeremo l'IDE di Arduino, dove andrà installata la scheda D1 Mini e le librerie necessarie.

Per preparare l'IDE di Arduino occorre seguire le indicazioni in questi due tutorial:

https://danielepostacchini.it/wp-content/uploads/2024/02/Esercizio-WEMOS-D1mini-Oled.pdf

pag.3 del seguente tutorial https://danielepostacchini.it/wp-content/uploads/2023/02/ESP8266_WebServer.pdf

Una volta preparata la piattaforma possiamo procedere a scrivere il programma.

L'idea è quella di realizzare un sistema che visualizzi sul display un messaggio scritto su una pagina HTML ed esegua dei comandi come ad esempio disegnare un rettangolo o un cerchio.

Per fare questo abbiamo bisogno di leggere il seguente tutorial:

https://danielepostacchini.it/wp-content/uploads/2025/08/WEB-Server-con-AP-e-ESP8266.pdf

Inoltre potrebbe servire conoscere la libreria del display OLED, tutte le informazioni sono disponibili ai seguenti link: *https://learn.adafruit.com/adafruit-gfx-graphics-library*

Classi e metodi: *https://adafruit.github.io/Adafruit-GFX-Library/html/class_adafruit___g_f_x.html*

https://adafruit.github.io/Adafruit_SSD1306/html/class_adafruit___s_s_d1306.html

Guida in PDF: *https://cdn-learn.adafruit.com/downloads/pdf/adafruit-gfx.pdf*

Librerie da installare: *https://github.com/adafruit/Adafruit-GFX-Library*
https://github.com/adafruit/Adafruit_SSD1306

PROGRAMMA

Come detto sopra vogliamo fare in modo che digitando l'indirizzo IP del server presente sulla scheda D1 Mini, si apra una pagina HTML dove poter inviare dei comandi o inviare una scritta da visualizzare.
Ovviamente il programma è solo un esempio senza alcuna utilità, ma una volta compreso li funzionamento si potrebbe adattare ad ogni esigenza.
Il programma di fatto è un mix tra i programmi descritti nei tutorial consigliati sopra, pertanto la descrizione delle singole parti non sarà molto approfondita.

In questa prima parte si includono tutte le librerie necessarie per gestire il display OLED, per gestire la connessione WIFI ed il Server. Successivamente si definiscono delle costanti per la gestione del display, per i dati della rete e si creano i due oggetti "display" e "server" gestiti poi dai metodi delle librerie.

```
#include <Wire.h>
#include <Adafruit_GFX.h>     //libreria grafica di base indipendente dal tipo di display
#include <Adafruit_SSD1306.h> //libreria specifica per il controller SSD1306
#include <ESP8266WiFi.h>      //libreria per gestire la connessione WiFi
#include <ESP8266WebServer.h> //libreria per gestire le funzioni del server web

#define SCREEN_WIDTH 128 // OLED larghezza, in pixels
#define SCREEN_HEIGHT 64 // OLED altezza, in pixels
#define OLED_RESET     -1

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

//dati della rete WiFi
const char* ssid = "D1Mini_AP";
const char* password = "12345678";

ESP8266WebServer server(80);
```

In questa parte dichiariamo le uniche due variabili globali che utilizzeremo, una per i messaggi che vengono ricevuti ed inviati dal server, e due variabili per le dimensioni del display.

```
// Buffer messaggi
String logMessages = "";
int larghezza,altezza; //conterranno le dimensioni del display
```

Come visto nel tutorial del WebServer con Access Point, dobbiamo ora creare una funzione che restituirà la pagina HTML al client. Su questa pagina sono presenti i pulsanti per l'invio dei comandi e del valore.

```
//*******************************************************
// Pagina HTML semplice
String getPage() {
  String html = R"rawliteral(

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

  <form action="/cmd" method="GET">
    <button name="action" value="CANCELLA">CANCELLA</button>
    <button name="action" value="RETTANGOLO">RETTANGOLO</button>
    <button name="action" value="CERCHIO">CERCHIO</button>
  </form>

  <h3>Invia Valore</h3>
  <form action="/messaggio" method="GET">
    <input type="text" name="messaggio" placeholder="Inserisci messaggio">
    <button type="submit">Invia</button>
  </form>
  <form action="/clear" method="GET">
    <button type="submit">Cancella Log</button>
  </form>

  <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>
)rawliteral";
  return html;
}
```

```
void setup() {
  Serial.begin(115200);
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    while(1){} //programma bloccato
  }
  display.clearDisplay();
  display.display();
  //LEGGO LARGHEZZA ED ALTEZZA DEL DISPLAY
  larghezza=display.width();
  altezza=display.height();
  // 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("/messaggio", handleValore);
  server.on("/log", handleLog);
  server.on("/clear", handleClear);
  server.begin();
}//setup
```

Nel setup inizializzeremo la comunicazione seriale attendendo la sua attivazione.

Successivamente cancelleremo il contenuto del display e leggeremo le sue dimensioni.

Poi andrà avviata la rete WiFi a cui si connetterà poi il client.

Ed infine vanno abbinati gli handle agli endpoin ricevuti cliccando sui pulsanti della pagina HTML.

Con server.begin() si attiva il web server sulla porta 80.

*Gli handle sono delle funzioni che vengono eseguite in corrispondenza di una richiesta proveniente dalla pagina HTML a cui è abbinato un endpoint. Es. http://192.168.4.1/cmd in questo caso l'endpoin è **/cmd** e verrà eseguito il relativo handle (funzione).*

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

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

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

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

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

*La funzione **processCommand** invece si occupa di individuare il comando inviato ed eseguire la relativa funzione.*

```
// Gestione comandi
void processCommand(String cmd, String value = "") {
  if (cmd == "MESSAGGIO") {
        display.clearDisplay();
        display.setTextSize(2);
        display.setTextColor(WHITE);
        display.setCursor(12,12);
        display.println(value);
        display.display();
        logMessages += "→ " + cmd + " " + value + "\n";
  }//if
  if (cmd == "CANCELLA") {
        display.clearDisplay();
        display.display();
        logMessages += "→ Display cancellato\n";
  }//if
  if (cmd == "RETTANGOLO") {
        display.clearDisplay();
        int coordX=0;
        int coordY=0;
        int raggio=2;
        display.drawRoundRect(coordX, coordY, larghezza-2, altezza-2, raggio, WHITE);
        display.display();
        logMessages += "→ Rettangolo disegnato\n";
  }//if
  if (cmd == "CERCHIO") {
        display.clearDisplay();
        int centroX=larghezza/2;
        int centroY=altezza/2;
        int diametro=30;
        display.drawCircle(centroX, centroY, diametro, WHITE);
        display.display();
        logMessages += "→ Cerchio disegnato\n";
  }//if
}//processCommand
```

```
void loop() {

  server.handleClient();

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

L'ultima parte è il ciclo di loop, al cui interno c'è la funzione per il controllo della presenza del client, e le righe che consentono di inviare alla pagina HTML eventuali messaggi provenienti dalla Seriale.

Come detto inizialmente la descrizione del programma è molto sintetica, ma leggendo i tutorial indicati si possono comprendere meglio tutte le righe del codice di esempio scritto.