Codice di Riferimento

Scansione WiFi - Base

```
#include "WiFi.h"
const int pinLED = 18;

void setup() {
    Serial.begin(115200);
    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    delay(100);
    pinMode(pinLED, OUTPUT);
}

void loop() {
    int n = WiFi.scanNetworks();
    if (n == 0) {
        Serial.println("Nessuna rete trovata");
    } else {
        Serial.printf("%d reti trovate\n", n);
        Serial.println("Nr | SSID | RSSI | CH | Encryption | MAC");

        for (int i = 0; i < n; ++i) {
            Serial.printf("%2d", i + 1);
            Serial.print(" | ");
            Serial.printf("%-32.32s", WiFi.SSID(i).c_str());
            Serial.print(" | ");
            Serial.printf("%4ld", WiFi.RSSI(i));
            Serial.print(" | ");
            Serial.printf("%2ld", WiFi.channel(i));
            Serial.print(" | ");

            switch (WiFi.encryptionType(i)) {
                case WIFI_AUTH_OPEN: Serial.print("open"); break;
                case WIFI_AUTH_WEP: Serial.print("WEP"); break;
                case WIFI_AUTH_WPA_PSK: Serial.print("WPA"); break;
                case WIFI_AUTH_WPA2_PSK: Serial.print("WPA2"); break;
                case WIFI_AUTH_WPA_WPA2_PSK: Serial.print("WPA+WPA2");
break;
                default: Serial.print("unknown");
            }

            Serial.print(" | ");
            Serial.println(WiFi.BSSIDstr(i));
            delay(10);
```

```
        }
    }

    WiFi.scanDelete();
    delay(5000);
}
```

Scansione WiFi - Asincrona

```
#include "WiFi.h"
const int pinLED = 18;

void setup() {
    Serial.begin(115200);
    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    delay(100);
    WiFi.scanNetworks(true);
}

void loop() {
    int16_t scanStatus = WiFi.scanComplete();
    if (scanStatus >= 0) {
        for (int i = 0; i < scanStatus; ++i) {
            // Stessa logica di stampa del codice precedente
        }
        WiFi.scanDelete();
        WiFi.scanNetworks(true);
    } else if (scanStatus == WIFI_SCAN_FAILED) {
        Serial.println("Scansione fallita. Riavvio...");
        WiFi.scanNetworks(true);
    }
}
```

Client WiFi con Gestione Stati

```
#include <WiFi.h>

const char *ssid = "tessarolo";
const char *password = "password";

void setup() {
    Serial.begin(115200);
    delay(10);

    Serial.printf("[WiFi] Connessione a %s\n", ssid);
    WiFi.begin(ssid, password);
```

```
    int tryDelay = 500;
    int numberOfTries = 20;

    while (true) {
        switch (WiFi.status()) {
            case WL_NO_SSID_AVAIL:
                Serial.println("[WiFi] SSID non trovato");
                break;
            case WL_CONNECT_FAILED:
                Serial.println("[WiFi] Connessione fallita");
                return;
            case WL_CONNECTION_LOST:
                Serial.println("[WiFi] Connessione persa");
                break;
            case WL_DISCONNECTED:
                Serial.println("[WiFi] Disconnesso");
                break;
            case WL_CONNECTED:
                Serial.println("[WiFi] Connesso!");
                Serial.printf("[WiFi] IP: %s\n",
WiFi.localIP().toString().c_str());
                return;
        }

        delay(tryDelay);
        if (numberOfTries <= 0) {
            Serial.println("[WiFi] Impossibile connettersi");
            WiFi.disconnect();
            return;
        }
        numberOfTries--;
    }
}
```

Client WiFi con IP Statico

```
#include <WiFi.h>

const char *ssid = "tessarolo";
const char *password = "password";

IPAddress local_IP(192, 168, 0, 33);
IPAddress gateway(192, 168, 0, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress dns1(8, 8, 8, 8);
IPAddress dns2(8, 8, 4, 4);
```

```
const char *host = "192.168.0.3";
const int httpPort = 80;

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

    if (!WiFi.config(local_IP, gateway, subnet, dns1, dns2)) {
        Serial.println("Configurazione IP fallita");
        return;
    }

    WiFi.begin(ssid, password);

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

    Serial.println("\nWiFi connesso!");
    Serial.printf("IP: %s\n", WiFi.localIP().toString().c_str());
    Serial.printf("MAC: %s\n", WiFi.macAddress().c_str());
    Serial.printf("Subnet: %s\n", WiFi.subnetMask().toString().c_str());
    Serial.printf("Gateway: %s\n", WiFi.gatewayIP().toString().c_str());
    Serial.printf("DNS: %s\n", WiFi.dnsIP().toString().c_str());
}

void loop() {
    WiFiClient client;
    if (client.connect(host, httpPort)) {
        client.println("Messaggio di test");
        client.stop();
    }
    delay(1000);
}
```

Esercizi Pratici

Esercizio 1: Monitor di Rete

Consegna: Implementare un sistema che monitora una rete WiFi specifica ("tessarolo") e controlla un LED in base alla potenza del segnale:

- LED fisso: segnale ottimo (> -15dBm)
- LED lampeggio veloce: segnale buono (-20dBm)
- LED lampeggio medio: segnale discreto (-45dBm)
- LED lampeggio lento: segnale debole (-60dBm)
- LED lampeggio molto lento: segnale critico (-80dBm)

Soluzione:

```
#include "WiFi.h"
const int pinLED = 18;

void setup() {
    Serial.begin(115200);
    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    pinMode(pinLED, OUTPUT);
}

void loop() {
    int n = WiFi.scanNetworks();
    int targetIndex = -1;

    for (int i = 0; i < n; i++) {
        if (WiFi.SSID(i) == "tessarolo") {
            targetIndex = i;
            break;
        }
    }

    if (targetIndex != -1) {
        int rssi = WiFi.RSSI(targetIndex);

        if (rssi > -15) {
            digitalWrite(pinLED, HIGH);
        }
        else if (rssi > -20) {
            digitalWrite(pinLED, HIGH);
            delay(150);
            digitalWrite(pinLED, LOW);
            delay(150);
        }
        else if (rssi > -45) {
            digitalWrite(pinLED, HIGH);
            delay(250);
            digitalWrite(pinLED, LOW);
            delay(250);
        }
        else if (rssi > -60) {
            digitalWrite(pinLED, HIGH);
            delay(500);
            digitalWrite(pinLED, LOW);
            delay(500);
        }
        else if (rssi > -80) {
            digitalWrite(pinLED, HIGH);
```

```
            delay(1000);
            digitalWrite(pinLED, LOW);
            delay(1000);
        }
    }

    WiFi.scanDelete();
    delay(2000);
}
```

Esercizio 2: Client con Retry Automatico

Consegna: Creare un client WiFi che:

- Si connette alla rete specificata
- Ritenta la connessione automaticamente se fallisce
- Invia un messaggio personalizzato al server ogni 30 secondi
- Mostra tutti i dettagli di rete quando connesso

Soluzione:

```
#include <WiFi.h>

const char *ssid = "tessarolo";
const char *password = "password";
const char *serverIP = "192.168.0.3";
const int serverPort = 80;

void setup() {
    Serial.begin(115200);
    WiFi.mode(WIFI_STA);
}

void connectToWiFi() {
    Serial.printf("\nConnessione a %s", ssid);

    WiFi.begin(ssid, password);
    int attempts = 0;

    while (WiFi.status() != WL_CONNECTED && attempts < 20) {
        delay(500);
        Serial.print(".");
        attempts++;
    }

    if (WiFi.status() == WL_CONNECTED) {
        Serial.println("\nConnesso!");
        Serial.printf("IP: %s\n", WiFi.localIP().toString().c_str());
```

```
        Serial.printf("Subnet: %s\n", WiFi.subnetMask().toString().c_str());
        Serial.printf("Gateway: %s\n", WiFi.gatewayIP().toString().c_str());
        Serial.printf("RSSI: %d dBm\n", WiFi.RSSI());
    } else {
        Serial.println("\nConnessione fallita");
    }
}

void loop() {
    if (WiFi.status() != WL_CONNECTED) {
        connectToWiFi();
    } else {
        WiFiClient client;
        if (client.connect(serverIP, serverPort)) {
            client.println("ESP32 Client Test");
            client.stop();
            Serial.println("Messaggio inviato");
        }
    }

    delay(30000);
}
```

Esercizio 3: Scanner Reti Multiple

Consegna: Creare un sistema che:

- Scansiona le reti in modo asincrono
- Monitora più reti specifiche contemporaneamente (es: "tessarolo", "lab_wifi")
- Mostra un confronto della potenza del segnale tra le reti
- Utilizza il Serial Monitor per visualizzare una tabella comparativa aggiornata ogni 10 secondi

Soluzione:

```
#include "WiFi.h"

const String NETWORKS[] = {"tessarolo", "lab_wifi"};
const int NETWORK_COUNT = 2;

void setup() {
    Serial.begin(115200);
    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    WiFi.scanNetworks(true);
}

void loop() {
```

```
    int16_t scanStatus = WiFi.scanComplete();
    if (scanStatus >= 0) {
        Serial.println("\n=== Monitoraggio Reti ===");
        Serial.println("SSID | RSSI | Canale | Sicurezza");

        for (int i = 0; i < NETWORK_COUNT; i++) {
            bool found = false;
            for (int j = 0; j < scanStatus; j++) {
                if (WiFi.SSID(j) == NETWORKS[i]) {
                    found = true;
                    Serial.printf("%-10.10s | %4d | %7d | ",
                        NETWORKS[i].c_str(),
                        WiFi.RSSI(j),
                        WiFi.channel(j));

                    switch (WiFi.encryptionType(j)) {
                        case WIFI_AUTH_OPEN: Serial.println("Aperta");
                            break;
                        case WIFI_AUTH_WPA2_PSK: Serial.println("WPA2");
                            break;
                        default: Serial.println("Altro");
                    }
                    break;
                }
            }
            if (!found) {
                Serial.printf("%-10.10s | Rete non trovata\n",
                    NETWORKS[i].c_str());
            }
        }

        WiFi.scanDelete();
        WiFi.scanNetworks(true);
    }

    delay(10000);
}
```

Questi esercizi sono stati progettati per coprire progressivamente tutti gli aspetti principali della programmazione WiFi su ESP32, dalla gestione base della connessione fino a implementazioni più complesse con gestione degli stati e funzionalità asincrone.