Concetti Fondamentali

Prima di addentrarci nel codice, è importante comprendere alcuni concetti chiave:

- **Socket**: Combinazione di indirizzo IP e porta che identifica univocamente una connessione
- **Server**: Dispositivo che fornisce servizi, identificato da IP e porta
- **Client**: Dispositivo che richiede servizi al server
- **Porta**: Numero che identifica un servizio specifico (es. 80 per HTTP)

1. Client Base (ClientSender)

Questo è l'esempio più semplice di client che si connette a una rete WiFi e invia dati.

```
#include <WiFi.h>

const char *ssid = "NomeRete";
const char *password = "12345678";

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

    // Connessione alla rete WiFi
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("\nConnesso a: ");
    Serial.println(WiFi.localIP());
}

void loop() {
    NetworkClient client;

    // Tentativo di connessione al server
    if (client.connect(host, port)) {
        client.write('A');  // Invia un carattere
        delay(500);

        // Legge eventuali risposte dal server
        while (client.available() > 0) {
            int c = client.read();
            Serial.println(c);
```

```
        }

        client.stop();  // Chiude la connessione
    }
}
```

2. Server Base (ServerCodebreak)

Questo server accetta connessioni e risponde ai client in base a un codice di stato.

```
#include <WiFi.h>

const char *ssid = "NomeRete";
const char *password = "12345678";
NetworkServer server(5000);  // Server sulla porta 5000
int codice = 50;  // Codice di stato atteso

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

    WiFi.begin(ssid, password);
    WiFi.setHostname("Cristiano");  // Nome host personalizzato

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

    server.begin();
    Serial.println("IP: " + WiFi.localIP().toString());
}

void loop() {
    NetworkClient client = server.accept();

    if (client) {
        while (client.connected() && client.available()) {
            int stato = client.read();
            if (stato == codice) {
                client.write(1337);  // Risposta di successo
            }
        }
    }
}
```

3. Client Avanzato (ClientES)

Client con gestione della connessione più robusta e comunicazione continua.

```
#include <WiFi.h>

const char* ssid = "Nome";
const char* password = "12345678";
const char* host = "192.168.1.12";
const int port = 3458;

void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);

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

    Serial.println(WiFi.localIP());
}

void loop() {
    NetworkClient client;

    while (client.connect(host, port)) {
        client.write(200);  // Invia stato

        while (client.available() > 0) {
            Serial.print("Client connesso");
        }

        client.stop();
        delay(5000);
    }
}
```

4. Server Avanzato (ServerES)

Server con gestione degli stati e verifica dei client.

```
#include <WiFi.h>

const char* ssid = "Nome";
const char* password = "12345678";
NetworkServer server(3459);

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

    WiFi.setHostname("Server");
    WiFi.begin(ssid, password);
```

```
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
    }

    server.begin();
    delay(2000);
    IPAddress myIP = WiFi.localIP();
}

void loop() {
    NetworkClient client = server.accept();

    if (client) {
        while (client.connected() && client.available()) {
            char stato = client.read();
            if (stato != 'g') {
                Serial.println("Non va bene");
            }
        }
    }
}
```

Punti Chiave per la Verifica

1. **Inizializzazione WiFi**:
   - Sempre iniziare con `WiFi.begin(ssid, password)`
   - Attendere la connessione con `WiFi.status()`
2. **Client**:
   - Creare con `NetworkClient client`
   - Connettere con `client.connect(host, port)`
   - Leggere dati con `client.read()`
   - Chiudere con `client.stop()`
3. **Server**:
   - Creare con `NetworkServer server(porta)`
   - Iniziare con `server.begin()`
   - Accettare client con `server.accept()`
4. **Controlli Importanti**:
   - Verificare sempre lo stato della connessione WiFi
   - Controllare se il client è connesso prima di comunicare
   - Verificare la disponibilità dei dati con `available()`

Note sulla Comunicazione

- Il client inizia sempre la comunicazione

- Il server resta in ascolto e risponde quando contattato
- Utilizzare `delay()` con attenzione per non bloccare la comunicazione
- Chiudere sempre le connessioni dopo l'uso

Suggerimenti per il Debug

1. Usare `Serial.println()` per monitorare lo stato
2. Controllare gli indirizzi IP di client e server
3. Verificare che le porte utilizzate siano corrette
4. Monitorare lo stato della connessione WiFi