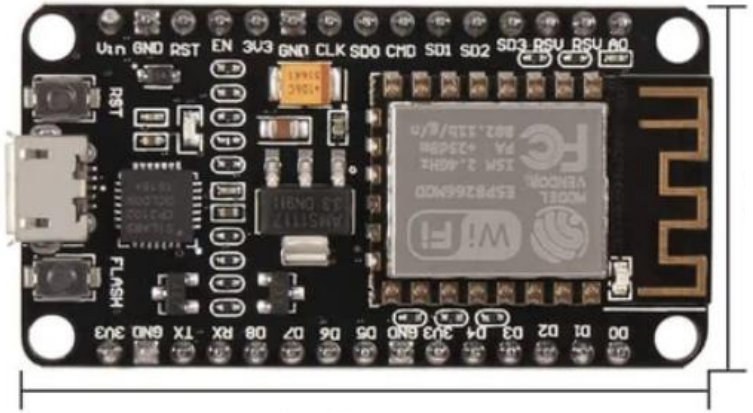
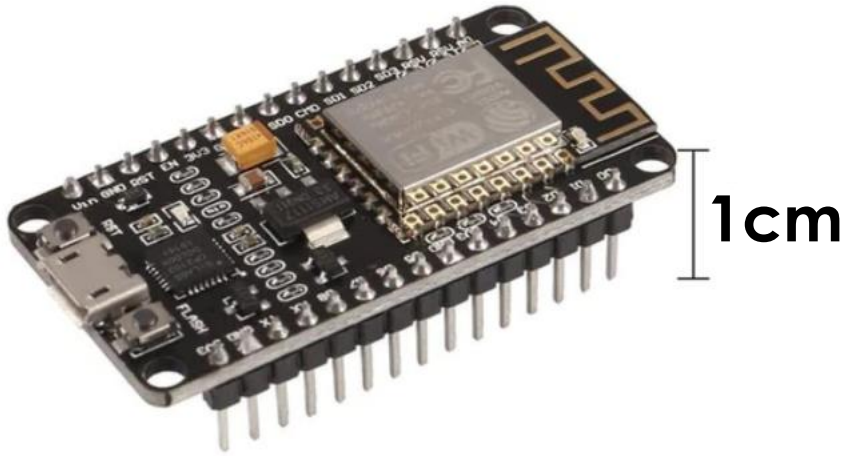
ESP8266 NodeMCU v3

DIMENSIONI





LINUX DAY

bit01

ESP8266 NodeMCU v3

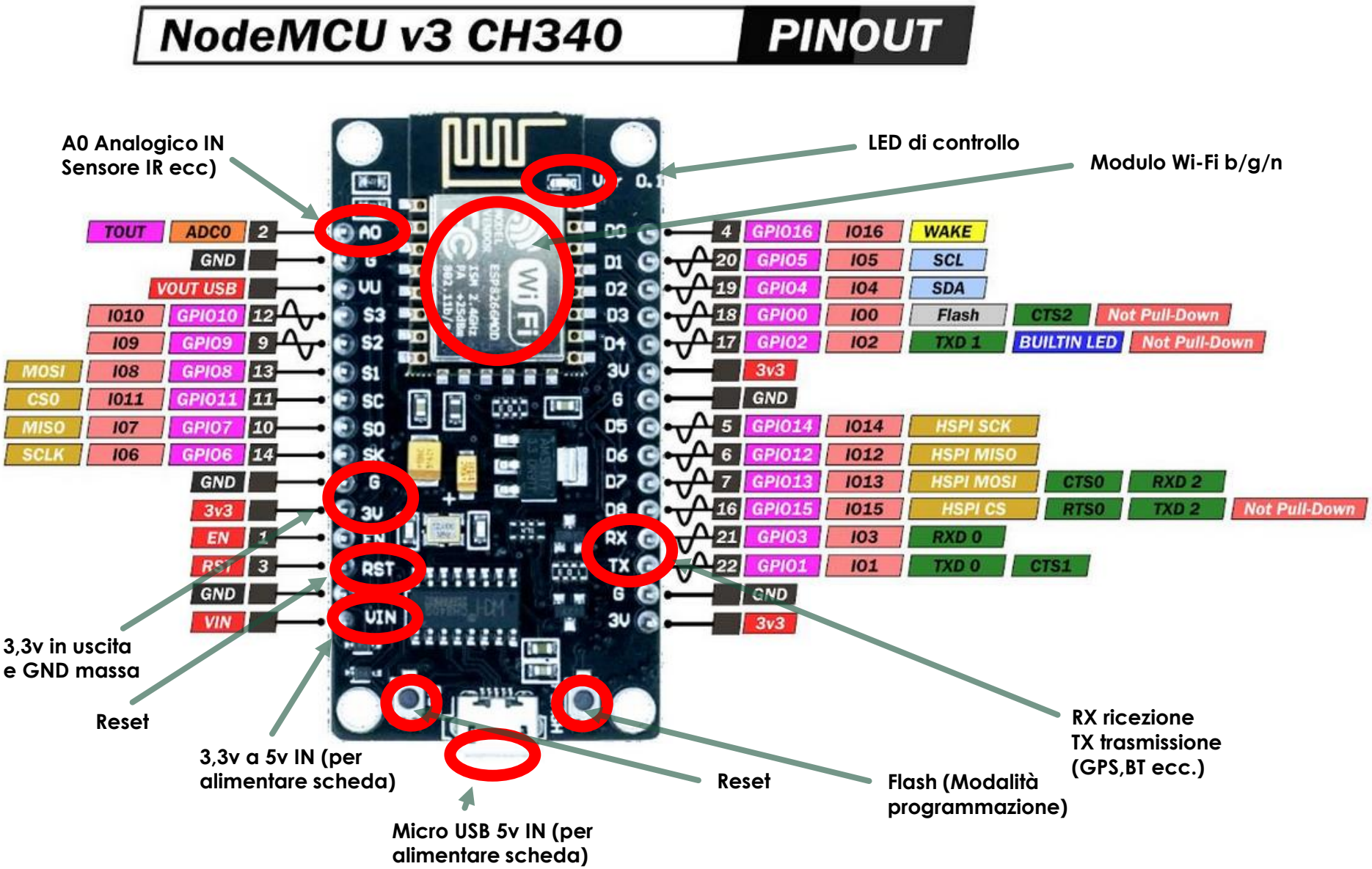
CARATTERISTICHE

- CPU ESP8266 32bit 80mhz
- Memoria tipo flash 4MB
- Memoria RAM 32KB
- Modulo Wi-Fi b/g/n Client/AP
- GPIO
- UART (pin Seriali per com. gps, BT ecc)

LINUX DAY

bit01

ESP8266 NodeMCU v3



ESP8266 NodeMCU v3 VS Arduino

- CPU 32 bit	- CPU 8 bit
- CONNETTIVITA' Wi-Fi	- NO
- MEMORIA 4MB	- MEMORIA 32KB
- RAM 32KB	- RAM 2KB
- GPIO 18 (1 ANALOG.)	- GPIO 20 (6 ANALOG.)
- COSTO € 4	- COSTO € 25

Se hai bisogno di connettività Wi-Fi e una maggiore potenza di elaborazione.

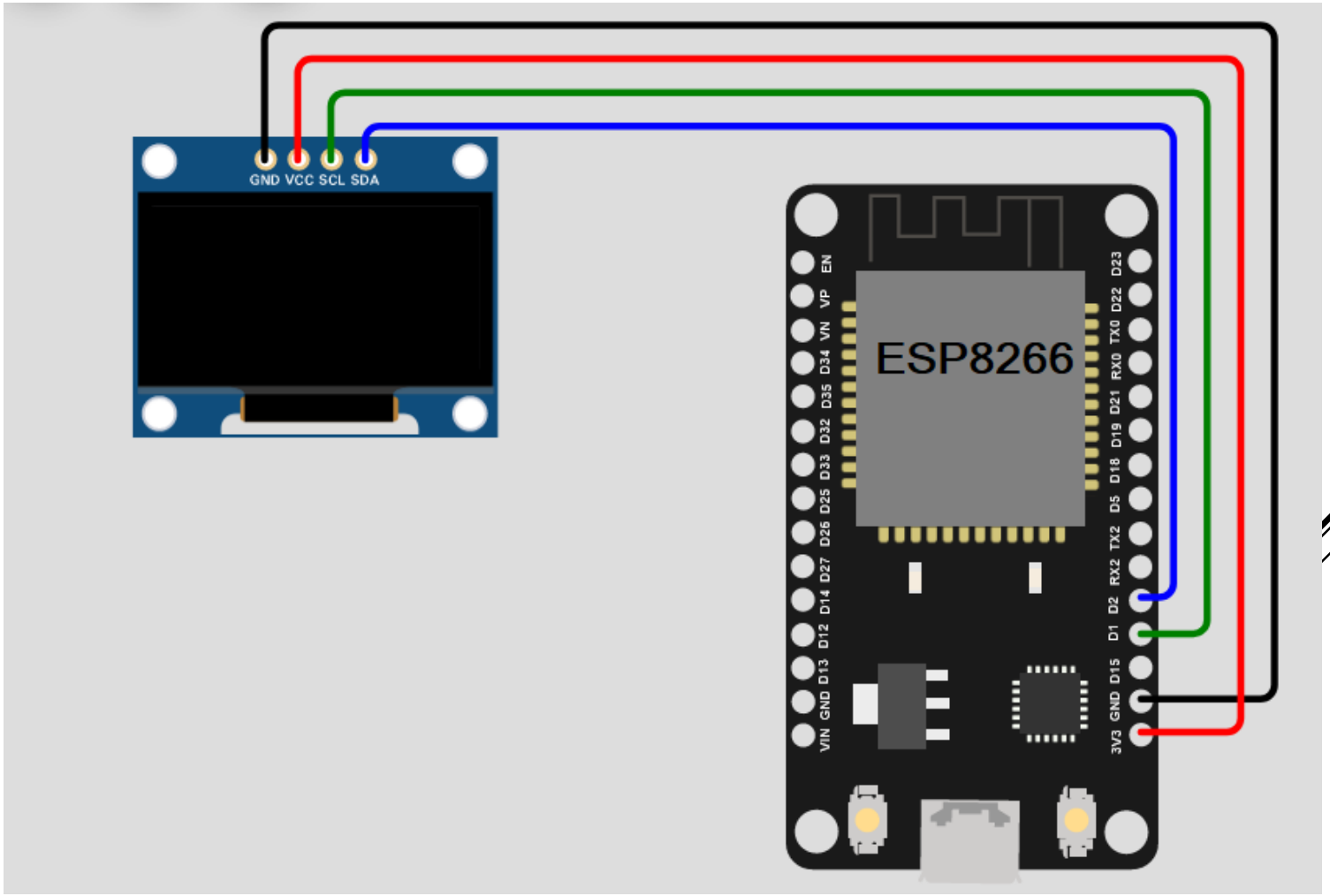
Se hai bisogno di un'ampia comunità di sviluppatori e una facilità di programmazione.

LINUX DAY

bit01

ESP8266 NodeMCU v3

Collegamenti - Generatore password



LINUX DAY

bit01

ESP8266 NodeMCU v3

Codice - Generatore password


```
// Importa le librerie
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Arduino.h>

// direttiva utilizzata per creare macro o costanti
#define SCREEN_WIDTH 128 // Larghezza del display OLED in pixel
#define SCREEN_HEIGHT 64 // Altezza del display OLED in pixel

// Imposta l'indirizzo I2C (seriale) a 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
0x3C);

void setup() { // Funzione principale obbligatoria
  // Inizializza il display OLED
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("Errore nell'inizializzazione del display OLED"));
    while(true);
  }
```

ESP8266 NodeMCU v3

Codice - Generatore password

```
// Inizializza la comunicazione seriale
  Serial.begin(115200);

  // Avvia la generazione di password casuale
  generaPasswordCasuale();
}

void loop() { // Funzione principale obbligatoria
  // Il loop non fa nulla in questo esempio
}

void generaPasswordCasuale() { // Dichiara una funzione
  String password = "";
  String caratteri = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789@!?-_.,:";
  int lunghezzaPassword = 12; // Lunghezza desiderata della password
```

ESP8266 NodeMCU v3

Codice - Generatore password


```
for (int i = 0; i < lunghezzaPassword; ++i) { // Genera pass casuale di 12 caratteri
    int indiceCasuale = random(caratteri.length());
    char carattereCasuale = caratteri.charAt(indiceCasuale);
    password += carattereCasuale;
  }

  // Visualizza la password sul display OLED
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println("Associazione bit01");
  display.println("");
  display.println("Linux Day 2023");
  display.println("");
  display.println("Password Casuale:");
  display.println("");
  display.println(password);
  display.display();
```

ESP8266 NodeMCU v3

Codice - Generatore password


```
// Stampa la password sulla porta seriale
  Serial.println("Password Casuale:");
  Serial.println(password);
}
 // Questo è utile per il debugging o per controllare la password generata quando il dispositivo Arduino è collegato a un computer.
```

bit01

ESP8266 NodeMCU v3

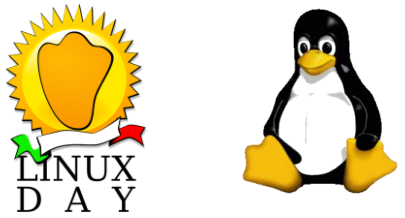
Codice – Wi-Fi Analyzer


```
// Importa le librerie
#include <ESP8266WiFi.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// direttiva utilizzata per creare macro o costanti
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
int currentNetworkIndex = 0; // Crea una variabile chiamata current….e gli assegna  valore 0

unsigned long previousMillis = 0; // dichiara la variabile previous….. E la imposta a 0
const unsigned long interval = 5000;  // Intervallo di 5 secondi tra le reti
```

bit01

ESP8266 NodeMCU v3

Codice – Wi-Fi Analyzer




```
void setup() { // Funzione principale obbligatoria
  Serial.begin(115200); // Comunicazione seriale a una velocità di 115,200 bit al secondo
  delay(100);
  WiFi.mode(WIFI_STA); // WIFI_STA", sta per "Wi-Fi Station " quindi funzionerà come station e non come AP
  WiFi.disconnect(); // Disconnette connessioni attive
  delay(100);

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Controlla se l' inizializzazione del display è riuscita o meno
    Serial.println(F("Errore nell'inizializzazione del display OLED")); // Questo gestisce un eventuale errore
    delay(1000);
    ESP.restart();
  }
```

ESP8266 NodeMCU v3

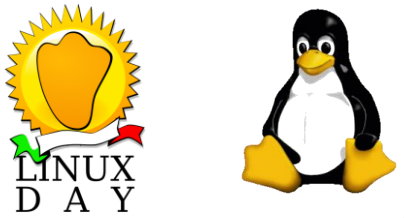
Codice – Wi-Fi Analyzer

```
display.setTextColor(SSD1306_WHITE);
  display.clearDisplay();
  Serial.println("Scansione reti Wi-Fi nelle vicinanze:"); // Stampa la frase...
  Serial.println();
  int numNetworks = WiFi.scanNetworks(); // scansione reti Wi-Fi
  if (numNetworks == 0) {
    Serial.println("Nessuna rete Wi-Fi trovata.");
    display.setTextSize(1);
    display.setCursor(0, 0);
    display.print("Nessuna rete Wi-Fi trovata.");
  } else {
    Serial.print("Numero di reti Wi-Fi trovate: ");
    Serial.println(numNetworks);
    Serial.println();
    display.setTextSize(1);
    display.setCursor(0, 0);
    display.print("Reti Wi-Fi trovate:");
  }
  display.display();
}
```

bit01

ESP8266 NodeMCU v3

Codice – Wi-Fi Analyzer




```
void loop() { // Funzione principale obbligatoria
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;

    if (currentNetworkIndex < WiFi.scanComplete()) {
      String ssid = WiFi.SSID(currentNetworkIndex); // Estrae il nome della rete (SSID) e lo memorizza nella variabile ssid
      String bssid = WiFi.BSSIDstr(currentNetworkIndex); // Estrae il mac della rete (BSSID) e lo memorizza nella variabile bssid
      int rssi = WiFi.RSSI(currentNetworkIndex); // Estrae la potenza del segnale e lo memorizza nella variabile rssi
      int channel = WiFi.channel(currentNetworkIndex); // Estrae il numero del canale della rete e lo memorizza nella variabile channel
     String encryptionType; // Crea una variabile che verrà utilizzata per memorizzare il tipo di protezione della rete.
```

bit01

ESP8266 NodeMCU v3

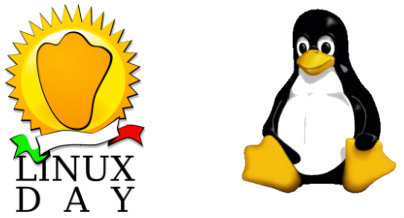
Codice – Wi-Fi Analyzer

```
switch (WiFi.encryptionType(currentNetworkIndex)) {
      case ENC_TYPE_NONE:
        encryptionType = "Senza password";
        break;
      case ENC_TYPE_WEP:
        encryptionType = "WEP";
        break;
      case ENC_TYPE_TKIP:
        encryptionType = "WPA";
        break;
      case ENC_TYPE_CCMP:
        encryptionType = "WPA2";
        break;
      case ENC_TYPE_AUTO:
        encryptionType = "Tipo automatico";
        break;
      default:
        encryptionType = "Sconosciuto";
    }
```

bit01

ESP8266 NodeMCU v3

Codice – Wi-Fi Analyzer

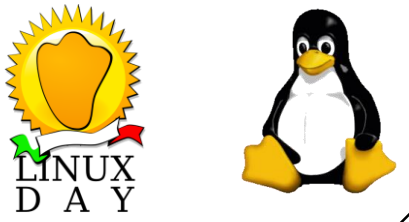



```
String networkInfo = "Rete " + String(currentNetworkIndex + 1) + ":";
    networkInfo += "\n  SSID: " + ssid; // Aggiunge il nome della rete alla stringa
    networkInfo += "\n  BSSID: " + bssid; // Aggiunge il MAC della rete alla stringa
    networkInfo += "\n  RSSI: " + String(rssi) + " dBm"; // Aggiunge la potenza del segnale alla stringa
    networkInfo += "\n  Canale: " + String(channel); // Aggiunge il numero del canale alla stringa
    networkInfo += "\n  Protezione: " + encryptionType; // Aggiunge il tipo di protezione alla stringa


Serial.println(networkInfo); // Stampa intera stringa sul monitor seriale, utile per il debugging quando il
dispositivo Arduino è collegato a un computer.

    display.clearDisplay(); // Pulisce il display
    display.setTextSize(1); // Imposta la dimensione del testo sul display OLED a 1
    display.setCursor(0, 0); // Imposta la posizione del cursore sul display OLED riga 0 colonna 0
    display.print(networkInfo); // Stampa intera stringa
    display.display(); // Aggiorna il display OLED con il nuovo contenuto
```

bit01

ESP8266 NodeMCU v3

Codice – Wi-Fi Analyzer




```
  currentNetworkIndex++; // Questo significa che la scansione di tutte le reti è stata completata.
    } else {
       WiFi.scanNetworks(true); // Scansione delle reti completata, riavvia la scansione dopo 5 secondi
      currentNetworkIndex = 0;
    }
  }
}
```