ESP-8266 Based Wi-Fi Packet Sniffer – Code Implementation Manual

# Part 1: Prerequisites

Before pasting any code, ensure the following:

## Software Requirements:

1. Arduino IDE  
 - Version: 1.8+ or Arduino IDE 2.x  
 - Install from: https://www.arduino.cc/en/software  
  
2. Python (Recommended 3.10+)  
 - Install from: https://www.python.org/downloads/  
 - Use pip for installing required libraries (shown later)

## Hardware Requirements:

- ESP8266 (NodeMCU preferred)  
- Micro USB cable for flashing  
- Laptop with Wi-Fi adapter  
- Smartphone (optional, for mobile hotspot)

# Part 2: Configuration Steps

## Step 1: Setup Arduino IDE for ESP8266

1. Open Arduino IDE.  
2. Go to File > Preferences.  
3. In the Additional Board Manager URLs, paste:  
 http://arduino.esp8266.com/stable/package\_esp8266com\_index.json  
4. Go to **Tools** > **Board** > **Boards Manager**, search **ESP8266** and click Install.

# Part 3: ESP8266 Code (Sniffer + UDP Sender)

## Where to paste:

Paste into Arduino IDE, save the file as WiFiSniffer.ino.

## How to upload:

1. Connect ESP8266 via USB.  
2. Select board: Tools > Board: NodeMCU 1.0 (ESP-12E)  
3. Choose correct port: Tools > Port  
4. Click Upload (Right Arrow icon).

## Esp 8266 Code

#include <ESP8266WiFi.h>

#include <WiFiUdp.h>

#define SNIFFING\_DURATION 10000     // 10 seconds sniffing

#define TRANSMISSION\_DURATION 5000  // 5 seconds transmission

const char\* txSSID = "DESKTOP-LHC063N 9595";

const char\* txPassword = "6$84kB95";

IPAddress laptopIP(192, 168, 137, 1);

WiFiUDP udp;

unsigned long modeSwitchTime = 0;

bool isSniffing = true;

String capturedData = "";

struct sniffer\_buf {

  uint16\_t len;

  uint8\_t buf[112];

  uint16\_t cnt;

  int8\_t rssi;

};

void setup() {

  // put your setup code here, to run once:

  Serial.begin(115200);

  delay(1000);

  startSniffingMode();

}

void loop() {

  // put your main code here, to run repeatedly:

  if (isSniffing && millis() - modeSwitchTime > SNIFFING\_DURATION) {

    stopSniffingMode();

    startTransmissionMode();

  }

  else if (!isSniffing && millis() - modeSwitchTime > TRANSMISSION\_DURATION) {

    stopTransmissionMode();

    startSniffingMode();

  }

}

void sniffer\_callback(uint8\_t \*buf, uint16\_t len) {

  processPacket(buf, len);

}

void startSniffingMode() {

  Serial.println("\n[SNIFF] Starting sniffing mode...");

  WiFi.disconnect();

  wifi\_set\_opmode(STATION\_MODE);

  wifi\_set\_channel(6);

  wifi\_promiscuous\_enable(0);

  wifi\_set\_promiscuous\_rx\_cb(sniffer\_callback);

  wifi\_promiscuous\_enable(1);

  modeSwitchTime = millis();

  isSniffing = true;

  capturedData = ""; // Reset buffer

}

void processPacket(uint8\_t \*buf, uint16\_t len) {

  struct sniffer\_buf \*sniffer = (struct sniffer\_buf\*) buf;

  String packetInfo = String(millis()) + ",";       // Timestamp

  packetInfo += String(len) + ",";                  // Length

  packetInfo += String(6) + ",";                    // Channel

  packetInfo += String(sniffer->rssi) + "\n";       // RSSI strength

  capturedData += packetInfo;

  Serial.print(".");

}

void stopSniffingMode() {

  wifi\_promiscuous\_enable(0);

  Serial.println("\n[SNIFF] Captured " + String(capturedData.length()) + " bytes");

}

void startTransmissionMode() {

  Serial.println("[TX] Connecting to WiFi...");

  WiFi.begin(txSSID, txPassword);

  unsigned long startAttemptTime = millis();

  while (WiFi.status() != WL\_CONNECTED && millis() - startAttemptTime < 10000) {

    delay(500);

    Serial.print(".");

  }

  if (WiFi.status() != WL\_CONNECTED) {

    Serial.println("\n[TX] Failed to connect. Skipping transmission.");

    return;

  }

  Serial.println("\n[TX] Connected! Sending data...");

  int chunkSize = 1200;

  for (int i = 0; i < capturedData.length(); i += chunkSize) {

    String chunk = capturedData.substring(i, i + chunkSize);

    udp.beginPacket(laptopIP, 1234);

    udp.write(chunk.c\_str());

    udp.endPacket();

    delay(50);

  }

  modeSwitchTime = millis();

  isSniffing = false;

  capturedData = "";

}

void stopTransmissionMode() {

  WiFi.disconnect();

  Serial.println("[TX] Transmission complete");

}

# Part 4: Python Script for Receiving UDP Packets

## Where to paste:

Open any code editor (VS Code, Notepad++, etc.). Paste the code in a file named: sniffer\_receiver.py

## Required Python Libraries:

Open Command Prompt and run:  
  
pip install matplotlib pandas

## Python code

import socket

import csv

import os

from datetime import datetime

from collections import defaultdict

import matplotlib.pyplot as plt

PORT = 1234

BUFFER\_SIZE = 4096

CSV\_FILE = "wifi\_data.csv"

DEVICE\_TIMEOUT = 60  # in seconds

device\_last\_seen = {}

rssi\_values = []

def save\_to\_csv(timestamp, data\_line):

    file\_exists = os.path.isfile(CSV\_FILE)

    with open(CSV\_FILE, mode='a', newline='') as f:

        writer = csv.writer(f)

        if not file\_exists:

            writer.writerow(['Timestamp', 'Millisecond', 'Packet Length', 'Channel', 'RSSI'])

        writer.writerow([timestamp] + data\_line.split(','))

def parse\_packet(line):

    parts = line.split(',')

    if len(parts) == 4:

        return parts

    return None

def update\_device\_seen(identifier):

    device\_last\_seen[identifier] = datetime.now()

def analyze\_devices():

    now = datetime.now()

    missing\_devices = []

    for dev, last\_seen in device\_last\_seen.items():

        delta = (now - last\_seen).total\_seconds()

        if delta > DEVICE\_TIMEOUT:

            missing\_devices.append(dev)

    return missing\_devices

def plot\_rssi():

    if not rssi\_values:

        print("No RSSI data to plot.")

        return

    plt.plot(rssi\_values)

    plt.title("RSSI over time")

    plt.xlabel("Sample Index")

    plt.ylabel("RSSI (dBm)")

    plt.grid(True)

    plt.show()

def sniffer\_mode():

    print("Starting Sniffer Mode...")

    sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

    sock.bind(("0.0.0.0", PORT))

    print(f"Listening for UDP packets on port {PORT}...")

    try:

        while True:

            data, addr = sock.recvfrom(BUFFER\_SIZE)

            decoded = data.decode(errors="ignore").strip()

            timestamp = datetime.now().isoformat()

            for line in decoded.splitlines():

                parts = parse\_packet(line)

                if parts:

                    print(f"[{timestamp}] From {addr}: {parts}")

                    save\_to\_csv(timestamp, line)

                    update\_device\_seen(addr[0])

                    rssi\_values.append(int(parts[3]))

    except KeyboardInterrupt:

        print("\nSniffer mode stopped.")

        sock.close()

def analysis\_mode():

    print("Running Analysis Mode...")

    missing = analyze\_devices()

    if missing:

        print("Missing Devices (not seen in last 60s):")

        for dev in missing:

            print(f" - {dev}")

    else:

        print("No missing devices.")

    if rssi\_values:

        print(f"Average RSSI: {sum(rssi\_values) / len(rssi\_values):.2f} dBm")

    else:

        print("No RSSI data collected yet.")

def visualization\_mode():

    print("Launching Visualization...")

    plot\_rssi()

def menu():

    while True:

        print("\n=== WiFi Packet Listener Modes ===")

        print("1. Sniffer Mode (listen and save)")

        print("2. Analysis Mode (device status, average RSSI)")

        print("3. Visualization Mode (RSSI chart)")

        print("4. Exit")

        choice = input("Enter choice: ")

        if choice == '1':

            sniffer\_mode()

        elif choice == '2':

            analysis\_mode()

        elif choice == '3':

            visualization\_mode()

        elif choice == '4':

            print("Exiting...")

            break

        else:

            print("Invalid choice.")

if \_\_name\_\_ == "\_\_main\_\_":

    menu()

# Part 5: Network Setup

Connect ESP8266 and your laptop to the same Wi-Fi (preferably your mobile hotspot).  
In ESP8266 code, update:  
  
const char\* ssid = "YourHotspotName";  
const char\* password = "YourHotspotPassword";  
IPAddress receiverIP(192, 168, X, X); // Set laptop IP here  
  
Find laptop’s IP by running ipconfig in Command Prompt.

# Part 6: Running the System

ESP8266:  
- Opens in sniffing mode (10s): collects packet info.  
- Then connects to Wi-Fi and sends packets via UDP.  
  
Python Program:  
- Run sniffer\_receiver.py in Terminal:  
  
 python sniffer\_receiver.py  
  
- You'll see real-time output:  
 - Packet logs in terminal.  
 - Data saved in sniffed\_data.csv.  
 - Graphs pop up with RSSI trends.

# Output Files & Where to Find Them

- CSV Output: Automatically saved in the script folder.  
- Graphs: Pop-up windows via matplotlib.  
- Terminal Log: Real-time packet data and analysis.