SSID Tracker and WiFi Signal Strength Mapper

CS-331 Computer Networks

Course Project Group-1

Github Link

Srijahnavi Chinthalapudi (22110063)
Sneha Gautham (22110255)
Gangannagudem Siri (22110083)
Rutuja Swami (22110267)

***Abstract*—This project presents a comprehensive WiFi network analysis and mapping system designed to visualize wireless network coverage across campus environments. The WiFi Signal Strength Mapper is a dynamic. This tool continuously monitors WiFi signals at configurable intervals, recording signal strength measurements, authentication types, channel information, and signal-to-noise ratios. The user's location is first determined using browser-based geolocation, after which data are scanned and collected accordingly, and then visualized on an interactive map interface. The program collects data and updates on the map at the specified interval by the user.**

I. Introduction

RELIABLE wireless network connectivity has become fundamental infrastructure in modern academic environments, supporting everything from classroom activities and research to administrative functions and student life. As campus networks grow in complexity with increasing user density and bandwidth demands, understanding the spatial distribution of WiFi signal quality becomes critical for ensuring consistent connectivity experiences.

WiFi networks at different locations across campus vary in their signal strengths for various reasons including: diverse building materials affecting signal propagation, varying user densities across spaces, potential interference sources, and the need to support multiple network types with different security protocols. These factors often lead to connectivity dead zones and inconsistent performance.

This project develops a systematic approach to dynamically mapping and visualizing WiFi networks across campus environments. Unlike static or manual collection tools, our system continuously captures WiFi signal data in real time as the user moves at every interval, enabling seamless tracking of SSID details and associated signal metrics. These metrics, including signal strength, signal-to-noise ratio, authentication methods, and channel information, are visualized live on an interactive map to provide an evolving and comprehensive understanding of wireless network performance across campus. If the user is in the same location it updates the Wifi details if there are any changes in that location

II. Tools and Techinques used

- **Programming language:** Python, HTML and JavaScript(for visualizing)
- **Data storage:** JSON Format for structured data storage and retrieval
- **Flask Web framework:** For building the visualization component.
- **Data collection techniques:** Windows - command: `netsh wlan show networks mode=bssid`. Linux - commad : `iwlist scanning and iw dev [interface] scan`.
- **HTML and JS libraries:** For visualization of the web based Interface.
- **Libraries and Dependencies:** Browser geolocation API-Primary method for location detection ,Webbrowser-For launching the browser-based interface, socket-For network communication and port management

III. Implementation of the Project

A. *Data Format and Collection*

In our project, we collected the WiFi data in a JSON file. The data at every interval is stored in the json file. If after an interval, the program is in the same location, it just updates the data of that particular location if any changes in the wifi signal happens, else the data is appended in the json file.

- **Data Aggregation:** Multiple samples are taken and averaged to ensure accuracy.
- **Storage:** Saves all collected information in a structured JSON format for reuse. Each location data gets appended to the json file.

The wifi data and all it's parameters are collected via the commands netsh in windows or iwlist in linux. The following figure shows how the above commands display the output:

```
C:\Users\sirig>netsh wlan show networks mode=bssid

Interface name : Wi-Fi
There are 1 networks currently visible.

SSID 1 : IITGN-SSO
    Network type            : Infrastructure
    Authentication          : WPA2-Enterprise
    Encryption              : CCMP
```

Fig. 1. Output of WiFi scan in windows

These details is stored in a json file. The overall structure of the data is: The top-level object is a dictionary with a key "locations". It maps to another dictionary, where each key

```
C:\Users\sirig>netsh wlan show interfaces

There is 1 interface on the system:

    Name                   : Wi-Fi
    Description            : MediaTek MT7921 Wi-Fi 6 802.11ax PCIe Adapter
    GUID                   : b5be3fc5-7e0d-4bed-b906-7519fd83d6b9
    Physical address       : 2c:3b:70:d6:fb:0d
    Interface type         : Primary
    State                  : connected
    SSID                   : IITGN-SSO
    AP BSSID               : 54:a2:74:be:92:81
    Band                   : 2.4 GHz
    Channel                : 6
```

Fig. 2. Output of WiFi interfaces in windows

```
    State                  : connected
    SSID                   : IITGN-SSO
    AP BSSID               : 54:a2:74:be:92:81
    Band                   : 2.4 GHz
    Channel                : 6
    Network type           : Infrastructure
    Radio type             : 802.11n
    Authentication         : WPA2-Enterprise
    Cipher                 : CCMP
    Connection mode        : Auto Connect
    Receive rate (Mbps)    : 144.4
    Transmit rate (Mbps)   : 144.4
    Signal                 : 80%
    Profile                : IITGN-SSO
    QoS MSCS Configured          : 0
    QoS Map Configured           : 0
    QoS Map Allowed by Policy    : 0

    Hosted network status  : Not available


C:\Users\sirig>
```

Fig. 3. Output of WiFi interfaces in windows

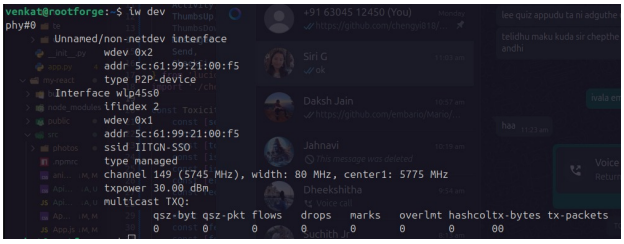

Fig. 4. Output of WiFi interfaces in Linux


```
{
  "locations": {
    "H_hostel_entrance_2025-04-09T22:51:05.143805": {
      "name": "H_hostel_entrance",
      "latitude": 23.20993867100029,
      "longitude": 72.68369603428854,
      "timestamp": "2025-04-09T22:51:05.143805",
      "networks": [
        {
          "ssid": "IITGN-SSO",
          "signal": -91,
          "signal_percent": 42,
          "auth": "WPA2-Enterprise",
          "channel": 48,
          "noise_floor": -95,
          "snr": 4,
          "samples": 3,
          "signal_variance": 10
        },
        {
          "ssid": "eduroam",
          "signal": -97,
          "signal_percent": 5,
          "auth": "WPA2-Enterprise",
          "channel": 11,
```



```
"locations": {
  "H_hostel_entrance_2025-04-09T22:51:05.143805": {
    "networks": [
      {
        "ssid": "IITGN-GUEST",
        "signal": -97,
        "signal_percent": 5,
        "auth": "Open",
        "channel": 161,
        "noise_floor": -95,
        "snr": -2,
        "samples": 3
      },
      {
        "ssid": "IITGN-EXAM",
        "signal": -97,
        "signal_percent": 5,
        "auth": "WPA2-Personal",
        "channel": 149,
        "noise_floor": -95,
        "snr": -2,
        "samples": 3
      }
    ],
    "note": "Auto-detected location: Browser-based geolocation"
```


Fig. 5. Data in json format

represents a unique scan entry. The key name is comprised of the location name and timestamp. Each location entry consists of name, latitude, longitude, timestamp and networks.

Each entry in the networks list represents a detected Wi-Fi network and includes its **ssid, signal, signal-percent, authorization, channel, noise_floor, snr and samples**.

B. Dynamic Wifi Data mapping

For real time capturing, we map the geolocation of the system and track the SSIDs and their corresponding signal metrics using dynamic.py file. The Dynamic WiFi Tracker is a real-time, location-based WiFi monitoring system designed to collect, store, and visualize WiFi signal data across various geographic locations. It is implemented using Python with the Flask framework to provide an interactive web interface.

Key Components

Flask Web Server:
The server is initiated via the `start_webapp()` function. It provides the following endpoints:

- View the interactive map interface.
- **/get_wifi** — Fetch WiFi data for a specific location.

- **/get_all_wifi** — Retrieve a summary of all available SSIDs and their signal strengths.
- **/get_data_for_download** — Download the complete stored data.
- **/get_all_locations** — Get a list of all known scanning locations.

Data Storage

WiFi data is stored in a JSON file named dynamic_data.json.

New scans automatically supplement our comprehensive baseline dataset Location-aware updates: For previously surveyed locations, data is intelligently updated rather than duplicated Real-time processing: As users navigate the campus with our browser-based tool, the system continues to gather and process fresh signal data

Dynamic Collection Engine

- The `dynamic_wifi_collection()` function continuously scans WiFi networks in the environment at regular intervals.
- It collects:
  - GPS location using `get_current_location()`
  - WiFi network data using `get_wifi_networks()`
- The collected data is matched with existing entries:
  - If the current location is within a 10-meter threshold of a previously scanned location, the existing data is updated.
  - Otherwise, a new location entry is created.
- Each data entry includes:
  - Timestamp
  - A note indicating whether the scan was a new entry or an update

Location Matching Logic:

- Locations are considered identical if they lie within a 0.5-meter radius.
- The distance between two GPS points is calculated using the Haversine formula in the `calculate_distance()` function.

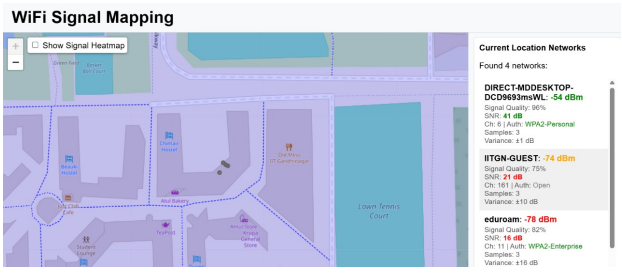

Fig. 6. Updated data points

Web-Based Visualization

The interactive web interface (`map.html`) provides users with the following features:

- **Click on the map** to view WiFi data from nearby scanned locations.
- **Updates the wifi data** on the map at every interval scanned.
- **Download collected data** in JSON format for further offline analysis or storage.
- **Automatic browser launch** — the browser window opens automatically when the Flask server starts, improving user convenience.

Threaded Execution:

The Flask server runs in a separate daemon thread to allow the data collection loop and web server to operate simultaneously without blocking each other.

Cleanup:

On termination, the system ensures that all collected data is saved and that any temporary data is cleared through the `cleanup_and_transfer_data()` function.

Overall Workflow:

1) The user starts the tracker, which spins up a local web server.
2) The tracker begins periodic WiFi scanning and location detection.
3) The data is either stored as a new entry or used to update a nearby location.
4) All scanned data can be visualized on a web interface with interactive map features.
5) Users can retrieve and download specific data based on geographic queries.

Use Case for Campus:

- Helps visualize SSID coverage across hostels, academic blocks, and open areas.
- Assists in identifying WiFi blind spots and areas with weak signal strength.

C. Static Wifi Data Mapping

Survey Methodology:

We conducted a systematic survey of the campus wireless environment by collecting signal data from approximately 100 strategic locations in both residential and academic areas, including:

- All hostel blocks (A through L)
- Academic buildings (AB1-AB13)
- Common areas (Central Arcade, Sports Complex)
- Campus gathering points (mess areas, student lounges)

At each location, we collected detailed signal metrics including:

- Signal strength (in dBm)
- Signal-to-noise ratio (SNR)
- Channel information
- Authentication types for all visible wireless networks

Browser-based Geolocation: A temporary HTTP server is started that serves an HTML page requesting location permission.

WiFi Scanning: Uses platform-dependent commands to extract SSID and RSSI values:

- `iwlist` for Linux
- `netsh` for Windows

Multiple samples are taken and averaged to ensure accuracy. The collected information is saved in a structured JSON format for reuse. Each location data gets appended to the JSON file.

Static Data Plotting: Another program, `app.py`, is specifically used for plotting the static data that is already stored. It maps them on a map for visualization.

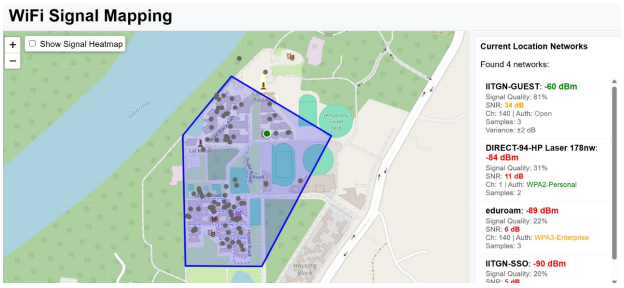

Fig. 7. Data in json format

D. *Visualization Web Interface*

Map Display: The web application uses Leaflet.js to display the map of IITGN on an interactive map. The system is designed to visualize data dynamically as users move across the campus, displaying data points and wifi details after every 10 seconds of automatic update. The map outlines the IIT Gandhinagar campus boundary and updates in real time as new WiFi scans are performed.

Location-Based Data Retrieval: When running `dynamic.py`, as the user moves or clicks within the campus boundary, the system:

- Retrieves and displays all WiFi networks detected at that position.
- After every 10 sec, it updates the current location and displays wifi networks as per new location.
- Shows detailed network parameters including SSID, signal strength, signal-to-noise ratio, authentication method, and channel.
- Details of previous data points are stored in json file and get cleared upon terminating the program.

Signal Visualization: Networks are displayed in a real-time updated list with color-coded signal strength indicators:

- Excellent (-30 to -65 dBm): Green
- Good (-65 to -75 dBm): Yellow
- Fair (-75 to -85 dBm): Orange
- Poor (-85 to -100 dBm): Red

Heatmap Generation: The system supports both real-time and static heatmap generation for WiFi coverage analysis across campus:

- Real-time updates as the user moves, with signal strength overlaid on the map
- Static heatmaps using spatial interpolation for offline documentation
- Color gradient visualization from red (poor) to green (excellent)
- Network-specific filtering options (e.g., IITGN-SSO, IITGN-GUEST)

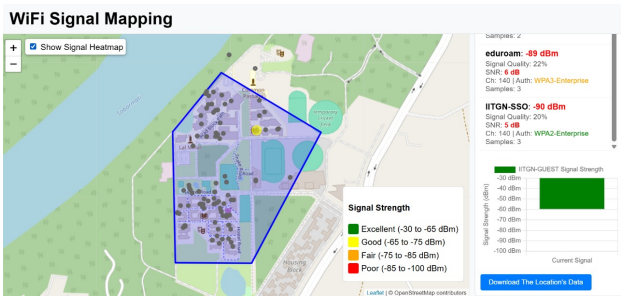

Fig. 8. Color coding for various strengths

Data Management: The application supports efficient handling of real-time and historical data:

- Location-specific data download for any scanned point.
- Full dataset retrieval through a built-in API endpoint.
- Continuous and persistent storage of WiFi scan results in structured JSON format.
- For offline working, already stored wifi network details in the json file for around 90-100 datapoints are used by static_app.py file.

E. *Techincal Implementation*

- **Nearest Point Algorithm:** Implementation of a proximity algorithm to find the closest data point to user clicks.
- **Progressive Enhancement:** System works with existing data but allows for continuous addition of new measurements.
- **Responsive Web Design:** Interface adapts to different screen sizes and devices.

IV. Results and Discussion

A JSON database of scanned SSIDs and their signal strengths was created for multiple known and detected locations. The Flask server successfully rendered a map interface showing all SSIDs collected at various coordinates. Users could select any point on the map and visualize SSIDs along with signal quality levels ranging from Excellent to Poor.

A. *Coverage Analysis*

- **Network Distribution:** Analysis of the distribution of different SSIDs across campus.
- **Signal Quality Map:** Identification of areas with strong and weak coverage.
- **Network Density:** Visualization of the number of available networks by location.

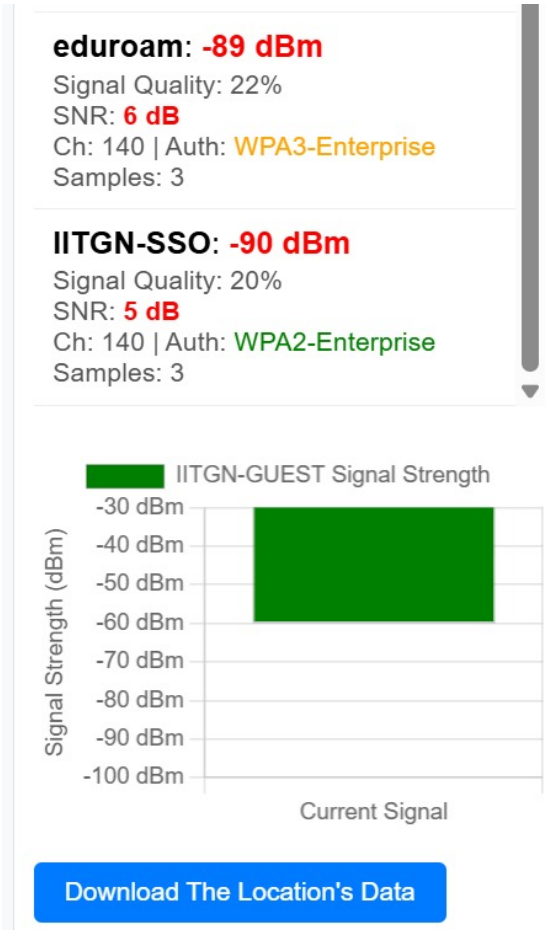

Fig. 9. Download option for a location

B. Performance Insights

- **Signal vs. Location:** Correlation between building structures and signal strength.
- **Network Congestion:** Identification of potentially overused channels in specific areas.
- **Live Data Handling:** Dynamic scans update without reloading the page, ensuring a smooth user experience.
- **Historical Access:** All stored locations and scans are available for export and review.

C. System Limitations

- **Temporal Variations:** Measurements represent point-in-time data that may vary based on time of day.
- **Interpolation Accuracy:** Signal strength estimation between measurement points has inherent limitations.
- **GPS Accuracy** may impact how precisely location data is matched or stored.
- **Environmental Factors:** Weather conditions and mobile obstructions can impact actual signal performance.

V. Future Work

- **Automated Collection:** Development of a mobile app to streamline and automate data collection.
- **3D Signal Mapping:** Extension to multistory buildings with Z-axis visualization.
- **Predictive Coverage Modeling:** Use of machine learning models to predict signal strength in unmeasured areas.
- **Network Recommendation:** Suggesting optimal connection points based on the location of the user and the load of the network.
- **User Feedback Integration:** Allow users to report connectivity issues to enhance the map.

VI. Challenges

Initially, our attempt to generate a comprehensive WiFi signal heatmap from one location proved ineffective as we cannot get a specific location's data unless our program is physically at that location.

To address this, we revised our strategy by collecting data from approximately 100 strategic locations on the IIT Gandhinagar campus. This required manually executing the scanning script at each point and aggregating the outputs into a unified JSON dataset. But this is not dynamic.

Another challenge was ensuring the accurate geolocation of the collected data. Mapping each signal reading to its exact physical coordinate on the campus map required precise calibration and validation. Furthermore, generating a meaningful heatmap involved complex interpolation techniques and a sufficient density of data points. The quality of the heatmap depended on the density and precision of the collected data points, making the process sensitive to both sampling strategy and data processing techniques.

VII. Conclusion

The WiFi Signal Strength Mapper successfully transforms network scan data into actionable geospatial insights specific to IIT Gandhinagar's campus layout and infrastructure. Our analysis revealed several key findings, including signal strength disparities between academic and residential areas, significant signal degradation in certain hostels, and optimal coverage zones for critical SSIDs like IITGN-SSO and eduroam.

By mapping the relationship between physical campus structures and wireless performance, this tool bridges the gap between technical network diagnostics and practical user experience. It's ability to continuously update through dynamic scanning ensures that the visualization remains relevant as network infrastructure evolves. Signal strength fluctuated significantly even within a small area, necessitating multi-sample averaging for reliability.

This mapping solution provides the foundation for strategic network enhancement, helping to eliminate dead zones, optimize access point placement, and ultimately support the institute's academic and research mission through improved connectivity.

VIII. Work Contribution

- **Siri:** WiFi details collection at data points, report editing, Data storage, distribution.
- **Srijahnvi:** WiFi details collection at data points, report editing, dynamic mapping.
- **Rutuja:** WiFi details collection at data points, report editing, static mapping.
- **Sneha Gautam:** WiFi details collection at data points, report editing, leaflet mapping.

IX. Key Takeaways

- **Signal Propagation Analysis:** Demonstrates how WiFi signals degrade with distance and obstacles in campus environments, with a common pattern around ˜-79 dBm signal strength.
- **Channel Utilization Patterns:** Reveals channel congestion in the 2.4 GHz band (notably channels 1, 6, and 11) in contrast to more distributed usage in the 5 GHz band.
- **Authentication Security Distribution:** Maps the adoption of various security protocols across networks:
  - WPA2-Enterprise: 62%
  - Open Networks: 26%
  - WPA2-Personal: 12%
- **Multi-SSID Management:** Highlights enterprise network segmentation strategies through multiple SSIDs such as `IITGN-SSO`, `IITGN-GUEST`, and `eduroam`.
- **Signal-to-Noise Ratio (SNR):** Emphasizes the importance of SNR—not just signal strength—as a more accurate metric for assessing connection quality.

HTTP Request Monitoring: Enables passive observation of actual network traffic patterns, revealing client-server communication dynamics through logged HTTP requests to the Flask server.

References

[1] S. Zvanovec, P. Pechac, and M. Klepal, "Wireless LAN Networks Design: Site Survey or Propagation Modeling?" *Radioengineering*, vol. 12, no. 4, pp. 42–49, 2003. [Online]. Available: http://www.radioeng.cz/fulltexts/2003/03_04_42_49.pdf

[2] J. Machaj and P. Brida, "Optimization of rank based fingerprinting localization algorithm," in *Proc. Int. Conf. Indoor Positioning and Indoor Navigation (IPIN)*, 2016, pp. 1–7. 10.1109/IPIN.2016.7743608

[3] C. Esposito, M. Ficco, and A. Napolitano, "WiFi Scanners for Network Forensics: Design, Implementation and Security Issues," *J. Netw. Comput. Appl.*, vol. 83, pp. 66–79, 2017. 10.1016/j.jnca.2017.01.029

[4] X. Wang, L. Gao, S. Mao, and S. Pandey, "DeepFi: Deep Learning for Indoor Fingerprinting Using Channel State Information," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2015, pp. 1666–1671. 10.1109/WCNC.2015.7127718

[5] J. Tang et al., "Fast Fingerprint Database Maintenance for Indoor Positioning Based on UGV SLAM," *Sensors*, vol. 17, no. 6, p. 1196, 2017. 10.3390/s17061196

Acknowledgment

We would like to express our sincere gratitude to **Prof. Sameer G. Kulkarni** for his valuable guidance, insightful lectures, and continuous support throughout the course of this project. His expertise in Computer Networks greatly contributed to our understanding and execution of the project.

We are also thankful to Teaching Assistant **Mallika Chauhan**, for her prompt assistance, constructive feedback, and encouragement during the development and debugging phases.

Lastly, we acknowledge the collaborative effort of our team in making this project a success.