Republic of Tunisia
Ministry of Higher Education
and Scientific Research

University of Sfax

National School of Electronics
and Telecommunications of Sfax

Engineer in :
Data Engineering
and Decision Systems

Option :
Data Science

Internship Report

Presennted at

National School of Electronics and Telecommunications of Sfax

by

Aziz HACHICHA

AI Real-time Home Gateway Testing Tool

Host company :

Company supervisor : MR. Abdallah GUIDARA

ACKNOWLEDGMENTS

First and foremost, I thank Almighty God for granting me the strength, wisdom, and opportunity to successfully complete this internship.

I would like to express my sincere gratitude to my supervisor at Sofrecom, MR Abdallah Guidara, for his invaluable guidance, constant support, and patience throughout my internship. His insights and feedback were instrumental in the success of this project.

My thanks also go to the entire Sofrecom team for their warm welcome and collaborative spirit. I am grateful for the knowledge I gained from each team member.

Finally, I extend my deepest appreciation to my parents for their unwavering encouragement and support during this period.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI Artificial Intelligence

API Application Programming Interface

AP Access Point

AUC Area Under the Curve

CSS Cascading Style Sheets

CSV Comma-Separated Values

dB Decibel

dBm Decibels-milliwatts

DNN Deep Neural Network

ESP32 A series of low-cost, low-power system-on-a-chip microcontrollers

HTML HyperText Markup Language

HTTP HyperText Transfer Protocol

IDE Integrated Development Environment

JS JavaScript

JSON JavaScript Object Notation

KPI Key Performance Indicator

LAN Local Area Network

ML Machine Learning

MSE Mean Squared Error

NaN Not a Number

RAM Random Access Memory

ReLU Rectified Linear Unit

RF Radio Frequency

ROC Receiver Operating Characteristic

RSSI Received Signal Strength Indicator

SNR Signal-to-Noise Ratio

STA Station (as in WiFi Station Mode)

TFLite TensorFlow Lite

UI User Interface

WAN Wide Area Network

GENERAL INTRODUCTION

In the digital era, the reliability and stability of wireless networks have become paramount for individuals and organizations alike. From smart homes to corporate offices, a consistent and robust WiFi connection is the backbone of modern operations. However, wireless networks are inherently susceptible to a variety of environmental and technical factors that can lead to intermittent connectivity, slow speeds, and complete outages. These issues are often difficult to diagnose and resolve without specialized knowledge and tools.

This project addresses this challenge by introducing an AI-driven solution for monitoring and predicting WiFi network stability. By leveraging advancements in embedded systems and machine learning, this system offers a proactive approach to network management, moving beyond traditional reactive troubleshooting. The core of this system lies in its ability to collect real-time network performance data, analyze it with a sophisticated artificial intelligence model, and provide actionable insights to the user. This report details the design, development, and evaluation of this intelligent monitoring system, highlighting its potential to transform how we understand and maintain the health of our wireless networks. This report details the design, development, and evaluation of this intelligent monitoring system. The subsequent chapters are structured as follows. Chapter 1 System Architecture provides a high-level overview of the hardware and software components, illustrating the data flow from the ESP32 to the web dashboard. Chapter 2 Data Collection and Preprocessing explains the key WiFi metrics gathered by the system and the methods used to prepare this data for analysis. Chapter 3 AI Model Development details the design, training, and deployment of the hybrid AI model, which combines a rule-based system with a TensorFlow Lite neural network. Chapter 4 Web Dashboard describes the user interface and data visualization components that present the real-time and historical network data. Chapter 5 Testing and Evaluation presents the methods used to assess the system's

performance, including unit tests and overall accuracy. Finally, Chapter 6 Conclusion and Future Work summarizes the project's achievements and discusses potential enhancements for future iterations.

Chapitre

1 Company Presentation

Sommaire

1.1 Overview

Sofrecom, an Orange subsidiary, is a consulting and engineering firm. Sofrecom advises and supports the development and transformation of telecom operators, governments and international institutions. The company has established itself as a global leader in telecommunications consulting, operating in more than 100 countries worldwide.

FIGURE 1.1 – Orange Logo

FIGURE 1.2 – Sofrecom Logo

1.2 Corporate Heritage and Background

Sofrecom, an Orange subsidiary, has developed over 50 years a unique know-how about operator businesses, making it a world leading specialist in telecommunications consultancy and engineering. The company was founded with the vision of becoming "The Know-How Network" in the telecommunications sector, leveraging decades of experience in both mature and emerging markets.

This extensive experience has positioned the company as an indispensable partner for operators, governments, and international investors.

1.3 Mission and Core Values

As a subsidiary of the Orange Group, Sofrecom advises, guides and supports the development and the digital transformation of the main telecommunications players (operators, governments

and international institutions). The company's mission centers on supporting the transformation and development of telecom actors to create a connected world.

The company focuses on digitalization, innovation, and sustainable development, offering expertise in connectivity, networks, and operational solutions. Sofrecom is committed to helping clients overcome complex challenges through tailored consulting services and technical expertise.

1.4 Services and Expertise

Sofrecom's experts provide Business, Network and IT consulting to improve long-term operational performance with innovative approach. The company's comprehensive service portfolio encompasses :

- **Strategic Consulting** : Advising on technology and development decisions
- **Network Engineering** : Specialized expertise in telecommunications infrastructure
- **Digital Transformation** : Supporting clients in their digitalization journey
- **Operational Performance** : Improving long-term operational efficiency through innovative approaches

1.5 Global Presence and Diversity

With a workforce representing more than 34 nationalities, Sofrecom has established itself as an international company with a diverse talent pool. This multinational composition enables the company to deliver culturally aware and operationally nuanced consulting services to clients across different markets.

The company's global reach is supported by strategic offices worldwide, with Sofrecom Tunisia serving as a key hub for operations in North Africa and the broader Mediterranean region.

1.6 Sofrecom Tunisia Specifics

Sofrecom Tunisia represents the company's commitment to the North African market, providing localized expertise while maintaining the global standards and methodologies that have made Sofrecom a world leader in telecommunications consulting.

The Tunisian office benefits from the group's extensive experience and proven methodologies while adapting them to the specific needs and characteristics of the local and regional telecommunications market.

1.7 Market Position

Sofrecom, an Orange Group company, is a world leading specialist in telecommunications consultancy and engineering, with a proven track record of delivering transformation and innovation services to operators, governments, and regulators worldwide. This year it will celebrate its 50 years of transformation and innovation services, marking a significant milestone in the company's journey of excellence in the telecommunications sector.

The company's unique position as part of the Orange Group provides it with unparalleled insights into operator challenges and opportunities, combined with the agility and specialized focus of a dedicated consulting firm.

Chapitre

2 System Architecture

Sommaire

2.1 INTRODUCTION

The AI WiFi Stability Monitor is an embedded system that integrates multiple components to achieve its goal of intelligent network analysis. This chapter provides a comprehensive overview of the project's architecture, detailing the hardware and software components that work together to collect data, perform AI-driven analysis, and present actionable information to the user. The high-level design illustrates the flow of data from the sensor to the user interface, while subsequent sections provide a breakdown of each key component and its role within the system.

2.2 High-Level Design

The AI WiFi Stability Monitor is designed as a self-contained, embedded system that integrates data collection, AI-powered analysis, and user-facing visualization into a single device. The architecture is centered around the ESP32 microcontroller, which acts as the core processing unit. Below is a high-level diagram illustrating the system's architecture and data flow :

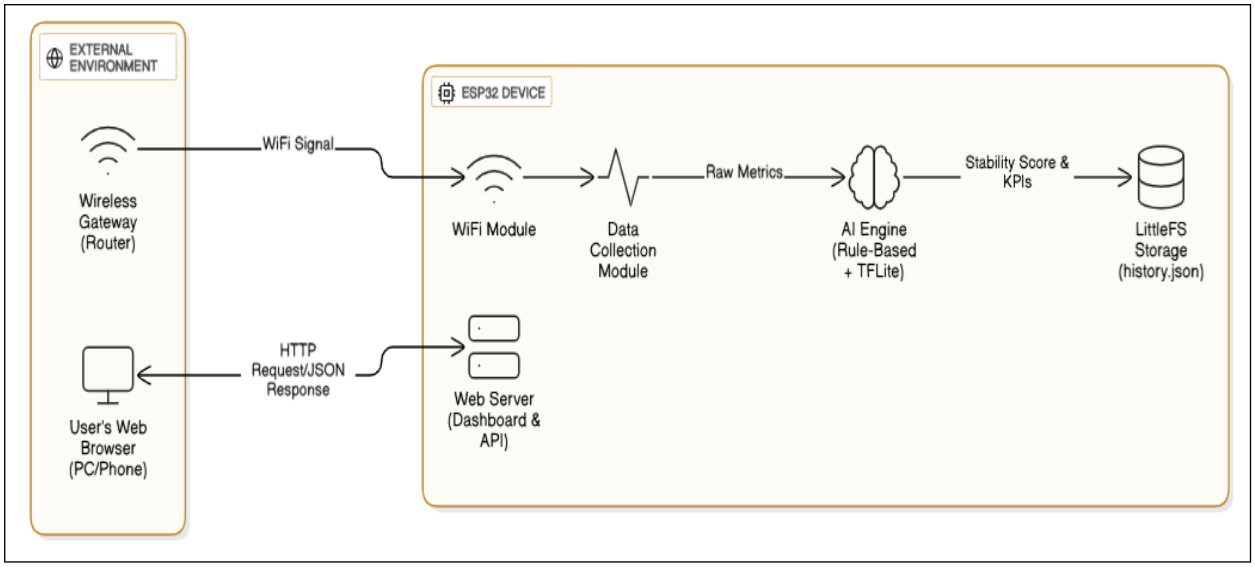

FIGURE 2.1 – Architecture overview

The data flow is orchestrated in a continuous cycle (every 10 seconds) :

1. Data Collection : The ESP32's integrated WiFi module actively scans the wireless environment

to capture key performance indicators (KPIs) from the Wireless Gateway.

2. AI Analysis : These raw metrics are fed into the onboard AI Engine. This engine employs a hybrid approach : a rule-based AI performs heuristic analysis and feature engineering, while a TensorFlow Lite model provides a data-driven stability prediction.

3. Prediction and Storage : The AI Engine generates a combined stability score and intelligent alerts. This complete data point (raw metrics + AI analysis) is timestamped and stored locally as a JSON object in the ESP32's LittleFS file system.

4. Data Visualization : The ESP32 hosts a web server that serves a web-based dashboard. A user can connect to this dashboard from any device on the same network. The dashboard's JavaScript periodically fetches both real-time status and historical data from the ESP32's API endpoints, visualizing the information in the form of charts and status cards.

2.3 Hardware Components

The hardware for this project is intentionally minimalist to ensure low cost and ease of deployment. It primarily consists of the ESP32 itself, which monitors the signals from an existing wireless gateway.

. ESP32 (Microcontroller) : The ESP32 is a series of low-cost, low-power system-on-a-chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. In this project, it serves as the all-in-one brain and communication hub. Its powerful dual-core processor is capable of running the AI models, its sufficient RAM handles the web server and data processing, and its built-in WiFi module allows it to both collect network data and serve the user dashboard without requiring any external components.

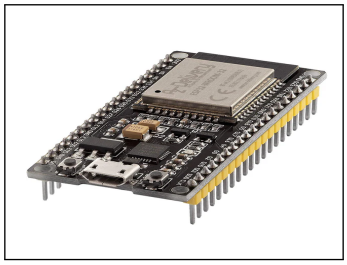

FIGURE 2.2 – ESP32

. Wireless Gateway (Router) : A gateway is a network device that connects one network to another ; in a typical home or office, this is the WiFi router that connects the local area network (LAN) to the internet (WAN). The AI WiFi Stability Monitor does not include a gateway but operates by connecting to an existing one. It continuously measures the quality and characteristics of the signal broadcast by the gateway to assess the network's overall stability.

FIGURE 2.3 – Gateway

2.4 Software Components

The software architecture is comprised of several key components working in concert :

- **Firmware (Arduino/C++) :** The core logic of the device is implemented in C++ using the Arduino framework, managed by the PlatformIO IDE. The firmware is responsible for all primary tasks, including WiFi network management, data collection scheduling, interfacing with the AI models, managing the file system, and running the web server.
- **Web Dashboard (HTML, CSS, JavaScript) :** A responsive and interactive user interface is built using standard web technologies. These static files are stored on the ESP32's file system and served directly to the user's browser, eliminating the need for any external hosting. The JavaScript component is crucial for fetching data from the ESP32's API and dynamically updating the charts and status information.
- **AI Model (TensorFlow Lite & Rule-Based AI) :**
    - A TensorFlow Lite model, a lightweight version of TensorFlow, is used for efficient, on-device machine learning inference. This pre-trained neural network provides a data-driven prediction of network stability.

    - A rule-based AI, implemented in C++, complements the neural network by performing advanced feature engineering (e.g., calculating trends, variance) and generating specific diagnostic alerts based on a set of predefined heuristics.

- **Data Storage (LittleFS) :** Historical KPI data is persisted on the ESP32's onboard flash memory using the LittleFS file system. Data is stored in a single `history.json` file.
- **Communication Protocols (HTTP, JSON) :** Communication between the web dashboard (client) and the ESP32 (server) is handled via HTTP. The ESP32 exposes API endpoints that provide data in JSON format.

2.5 CONCLUSION

In summary, the architecture of the AI WiFi Stability Monitor is built upon a foundation of efficiency, integration, and autonomy. By selecting the ESP32 microcontroller as the central hardware component, the system is able to consolidate data collection, advanced AI processing, and web-based user interaction into a single, low-cost device.

Chapitre

3 Data Collection and Preprocessing

Sommaire

3.1 INTRODUCTION

The effectiveness of any AI system is fundamentally dependent on the quality and relevance of the data it processes. This chapter details the crucial first steps in the AI WiFi Stability Monitor's workflow : defining the key metrics to be collected, the mechanism for their regular collection and storage, and the preprocessing techniques employed on the ESP32 to prepare the data for intelligent analysis.

3.2 WiFi Metrics and Interpretation

. RSSI (Received Signal Strength Indicator) : This metric measures the power level of the signal received by the ESP32 from the wireless router. It is measured in decibels-milliwatts (dBm). a negative value where a number closer to 0 indicates a stronger signal :

- **Excellent :** -30 dBm to -55 dBm
- **Good :** -56 dBm to -70 dBm
- **Weak :** -71 dBm to -85 dBm
- **Unusable :** Below -85 dBm

. Noise : This represents the background radio frequency (RF) noise in the environment that can interfere with the WiFi signal. It is also measured in dBm. A lower (more negative) value is better, as it indicates a quieter environment with less interference :

- **Excellent :** -90 dBm or lower
- **Good :** -80 dBm to -89 dBm
- **High Interference :** -79 dBm or higher

. SNR (Signal-to-Noise Ratio) : SNR is arguably the most critical metric for determining signal quality. It is the difference, in decibels (dB), between the received signal (RSSI) and the background noise level. A higher SNR means the signal is clearer and more distinct from the noise, leading to more reliable data transmission :

- **Excellent :** 30 dB or higher

- **Good :** 20 dB to 29 dB
- **Poor :** 10 dB to 19 dB
- **Very Poor :** Below 10 dB

. Channel Utilization : This metric, expressed as a percentage, estimates how "busy" the current WiFi channel is. High utilization indicates network congestion, as many devices (in your own or neighboring networks) are competing for the same transmission medium. This can lead to slowdowns and increased latency, even with a strong signal :

- **Excellent :** 0% to 30%
- **Good :** 31% to 60%
- **High Congestion :** 61% to 100%

3.3 Data Collection Mechanism

Data is collected in a consistent and automated manner to ensure a steady stream of information for both real-time monitoring and historical analysis.

- **Collection Frequency :** The system is programmed to collect a new set of the four KPIs mentioned above **every 10 seconds**. This interval is short enough to capture meaningful fluctuations in network performance without overwhelming the ESP32's processor or storage.
- **Role of `saveKPI()` function :** This core function, located in the main firmware file (`main.cpp`), orchestrates the entire data collection process. Triggered by a timer (`Ticker`), `saveKPI()` performs the following sequence of actions every 10 seconds :

  1. It queries the ESP32's WiFi hardware to get the current KPIs.
  2. It calculates the SNR from the RSSI and Noise values.
  3. It passes these metrics to the AI prediction function to get a stability score.
  4. Finally, it packages all this information (the four raw metrics plus the AI stability score and a timestamp) into a single record and saves it.

3.4 Data Storage

To enable historical trend analysis, the collected data is stored locally on the device. This is beneficial especially for real-time monitoring. Thanks to this method, data are treated locally, immediately which means no latency.

. LittleFS for history.json : The ESP32's onboard flash memory is formatted with the LittleFS file system, which is specifically designed for endurance and reliability on flash-based storage. All historical data is appended to a single text file named history.json.

. Data Retention Policy : The system enforces a 5-day data retention policy to prevent the flash memory from filling up. During each execution of the saveKPI() function, a cleanup routine checks for and removes any records from history.json that are older than five days, ensuring the file contains a rolling window of recent performance data.

3.5 Data Preprocessing and Feature Engineering

Before the collected data can be used by the AI models, it undergoes crucial preprocessing steps directly on the ESP32. This on-device processing transforms raw, disjointed data points into a rich, context-aware feature set.

3.5.1 Normalization for TensorFlow Lite Model Input

The TensorFlow Lite model requires its input data to be scaled to a uniform range to ensure accurate and stable predictions. This process, known as *min-max normalization*, is applied to all four features using formulas derived from their expected operational ranges.

General Formula

$$\text{Normalized Value} = \frac{\text{Current Value} - \text{Min Value}}{\text{Max Value} - \text{Min Value}} \tag{3.1}$$

Specific Implementations in the Firmware

— **RSSI :**

$$\text{Normalized RSSI} = \frac{\text{RSSI} - (-90)}{(-30) - (-90)} \tag{3.2}$$

This scales the typical RSSI range of $-90\,\mathrm{dBm}$ (min) to $-30\,\mathrm{dBm}$ (max), as shown in Equation 3.2.

— **Noise :**

$$\text{Normalized Noise} = \frac{\text{Noise} - (-100)}{(-50) - (-100)} \tag{3.3}$$

This scales a common noise range of $-100\,\mathrm{dBm}$ (min) to $-50\,\mathrm{dBm}$ (max), as defined in Equation 3.3.

— **SNR :**

$$\text{Normalized SNR} = \frac{\text{SNR} - (-10)}{50 - (-10)} \tag{3.4}$$

This maps the practical SNR range of $-10\,\mathrm{dB}$ (min) to $50\,\mathrm{dB}$ (max), according to Equation 3.4.

— **Channel Utilization :**

$$\text{Normalized Channel Util} = \frac{\text{Channel Utilization}}{100} \tag{3.5}$$

As channel utilization is already a percentage $(0\text{–}100)$, Equation 3.5 simply converts it to a decimal format.

3.5.2 Mathematical Foundations of Feature Engineering

The rule-based AI component performs sophisticated on-the-fly feature engineering. It maintains a short-term historical buffer (a "moving window") of the last 10 data points to calculate temporal features. The mathematical principles are as follows :

1. Variance (Signal Stability)

To measure the stability of a signal, the system calculates the statistical variance for metrics like RSSI and Noise. A low variance indicates a stable, consistent signal, while a high variance points to erratic fluctuations.

The variance σ^2 is calculated as :

$$\sigma^2 = \frac{\sum(x_i - \mu)^2}{N} \tag{3.6}$$

where :

— x_i is an individual data point in the buffer,

— μ is the average (mean) of all data points in the buffer,

— N is the number of data points (10 in this case).

2. Trend Analysis (Signal Degradation or Improvement)

To determine if a signal is improving or degrading over time, the system performs a simple linear regression on the 10-point historical buffer. The most important output of this is the slope (m) of the trendline, which indicates the rate and direction of change.

The formula for the slope is :

$$m = \frac{N\sum(xy) - \sum x \sum y}{N\sum(x^2) - (\sum x)^2} \tag{3.7}$$

where :

— N is the number of data points (10),

— x is the time index $(0, 1, 2, \ldots, 9)$,

— y is the value of the WiFi metric at that time index.

The interpretation of the slope is direct :

— A positive slope ⇒ improving trend (e.g., SNR is increasing).

— A negative slope ⇒ degrading trend (e.g., RSSI is getting weaker).

— A slope near zero ⇒ stable trend.

3. Outlier Detection (Sudden Events)

An outlier is a data point that deviates significantly from previous observations, indicating a sudden network event. The `advanced_ai.h` implementation detects outliers using a simplified statistical approach.

A new data point is considered an outlier if its value falls outside a defined threshold based on the historical mean and variance :

$$\text{is\_outlier} = \text{TRUE if } \ |\text{new\_value} - \mu| > k \cdot \sigma \tag{3.8}$$

where :

— μ is the historical average,

— σ is the historical standard deviation (the square root of variance),

— k is a multiplier constant (e.g., 2 or 3) that defines the sensitivity.

By applying these mathematical transformations on-device, the system creates derived, temporal, and outlier features that provide a far more nuanced understanding of network behavior than instantaneous raw metrics alone. This is essential for the high-quality predictions generated by the AI engine.

3.6 CONCLUSION

In conclusion, this chapter has laid out the critical data pipeline that serves as the foundation for the entire system. By systematically collecting key WiFi metrics, storing them efficiently on-device, and applying a series of robust preprocessing and feature engineering techniques, the raw, instantaneous network data is transformed into a rich and contextually-aware feature set. This rigorous preparation is not merely a preliminary step but a fundamental prerequisite for enabling the sophisticated analyses to be performed by the AI models.

Chapitre

4 AI Model Development and Evaluation

Sommaire

4.1 INTRODUCTION

This chapter provides a comprehensive examination of the project's intelligent core, the design, training, deployment, and validation of the AI systems. The project employs a powerful hybrid or "ensemble" strategy, combining a data-driven neural network with an expert-driven rule-based system to leverage the strengths of both approaches. This section covers the complete lifecycle of the AI components, from the mathematical foundations to final on-device performance metrics.

4.2 The Hybrid AI Architecture

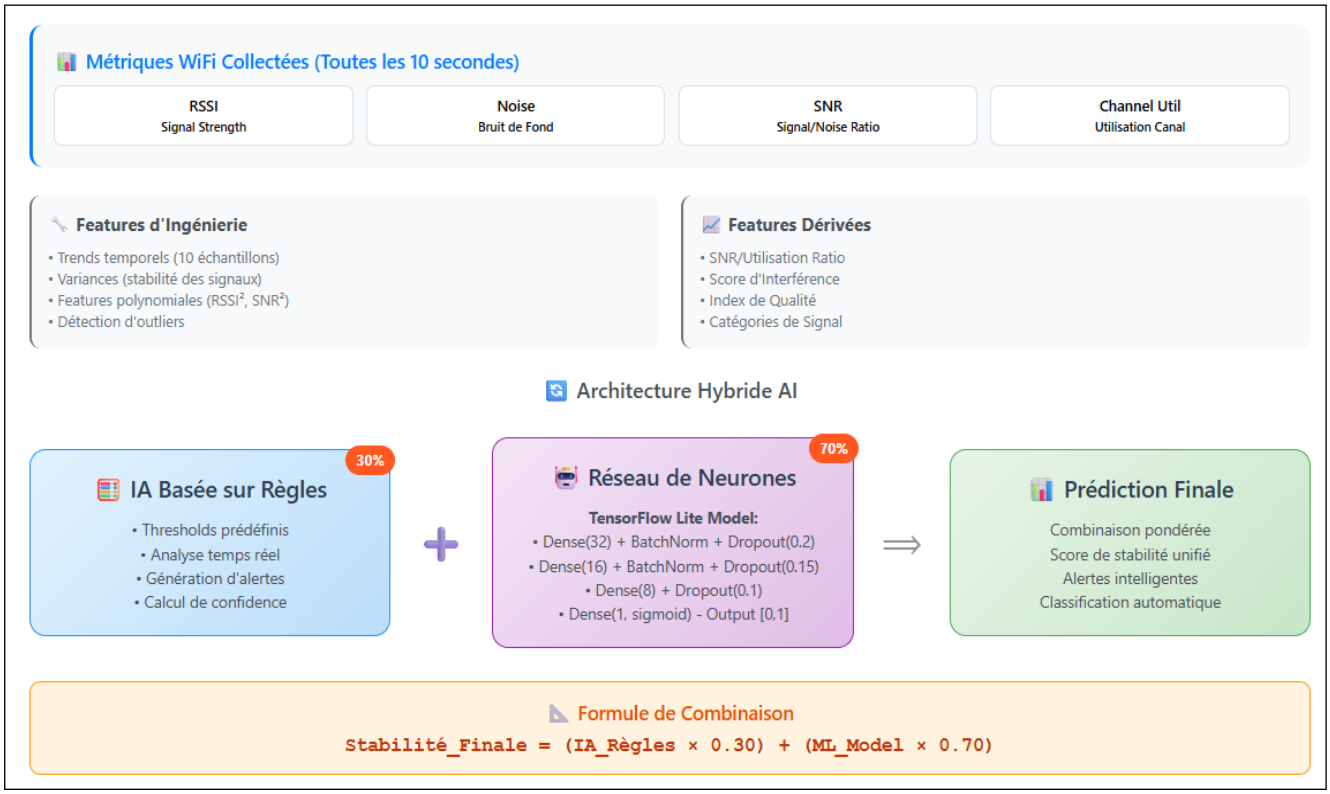

FIGURE 4.1 – AI Model Architecture overview

The above picture shows an overview of the AI model architecture. The system uses an ensemble model, which combines two different AI components to make a final, more robust prediction about WiFi stability.

Component 1 : Deep Neural Network (DNN)

A `TensorFlow Lite` model trained to recognize complex, non-linear patterns in WiFi data that correlate with stability. It provides a powerful, data-driven predictive score.

Component 2 : Advanced Rule-Based AI

A sophisticated, hand-crafted algorithm that performs temporal analysis (analyzing data over time) and applies a set of expert rules to diagnose specific, well-understood WiFi problems.

Final Prediction

The final stability score is a *weighted average* of the outputs from both components :

- The neural network's prediction contributes **70%**.
- The rule-based AI's score contributes **30%**.

This approach balances the DNN's pattern recognition capabilities with the rule-based system's clear, diagnostic logic.

4.3 Deep Neural Network (DNN) Development

The DNN is the machine learning heart of the system, trained to predict stability based on the four core WiFi features.

4.3.1 Dataset and Training Environment

Dataset

The model was trained on a custom `wifi_data.csv` dataset containing **3,600 realistic WiFi data samples**. This dataset was enriched with synthetic data to cover a wide variety of network conditions.

Framework

The model was built using **TensorFlow** with the **Keras API**.

Callbacks

To ensure optimal training, two key callbacks were used :

- **EarlyStopping :** Automatically halts the training process if the model's performance on validation data ceases to improve, preventing overfitting.
- **ReduceLROnPlateau :** Reduces the learning rate if training stagnates, allowing the model to fine-tune its weights more precisely.

Optimizer

The **Adam optimizer** was used for its efficiency in finding the optimal network weights.

Loss Function

The model was trained using **Binary Crossentropy**. This is the industry-standard loss function for binary classification problems (e.g., *"stable"* vs. *"unstable"*), measuring the distance between the predicted probability and the actual label.

The Binary Crossentropy loss is defined as :

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \Big[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i) \Big] \tag{4.1}$$

where :

- N is the number of samples,
- y_i is the true label (0 or 1),
- $\hat{y}_i$ is the predicted probability for sample i.

4.3.2 Model Architecture

FIGURE 4.2 – Architecture of the DNN Model

The above figure presents the Neural Network Architecture of the Model. The DNN is a **sequential model** specifically designed for efficiency on the ESP32 :

- **Input Layer :** Takes the 4 normalized input features.
- **Hidden Layer 1 :** 32 neurons with ReLU activation, followed by Batch Normalization and 20% Dropout.
- **Hidden Layer 2 :** 16 neurons with ReLU activation, followed by Batch Normalization and 15% Dropout.
- **Hidden Layer 3 :** 8 neurons with ReLU activation and 10% Dropout.
- **Output Layer :** A single neuron with a Sigmoid activation function. This constrains the output to a value between 0 and 1, representing the probability of the connection being stable.

4.3.3 Model Performance and Evaluation

The trained DNN was rigorously evaluated on a withheld test dataset to confirm its real-world performance and the performance metrics were outstanding.

Physical Performance

- Model File Size : 2,848 bytes (2.78 KB). This extremely small footprint is ideal for the limited storage of an ESP32.
- Inference Time : Less than 1 millisecond per prediction on the ESP32, confirming its high efficiency.

Performance Metrics

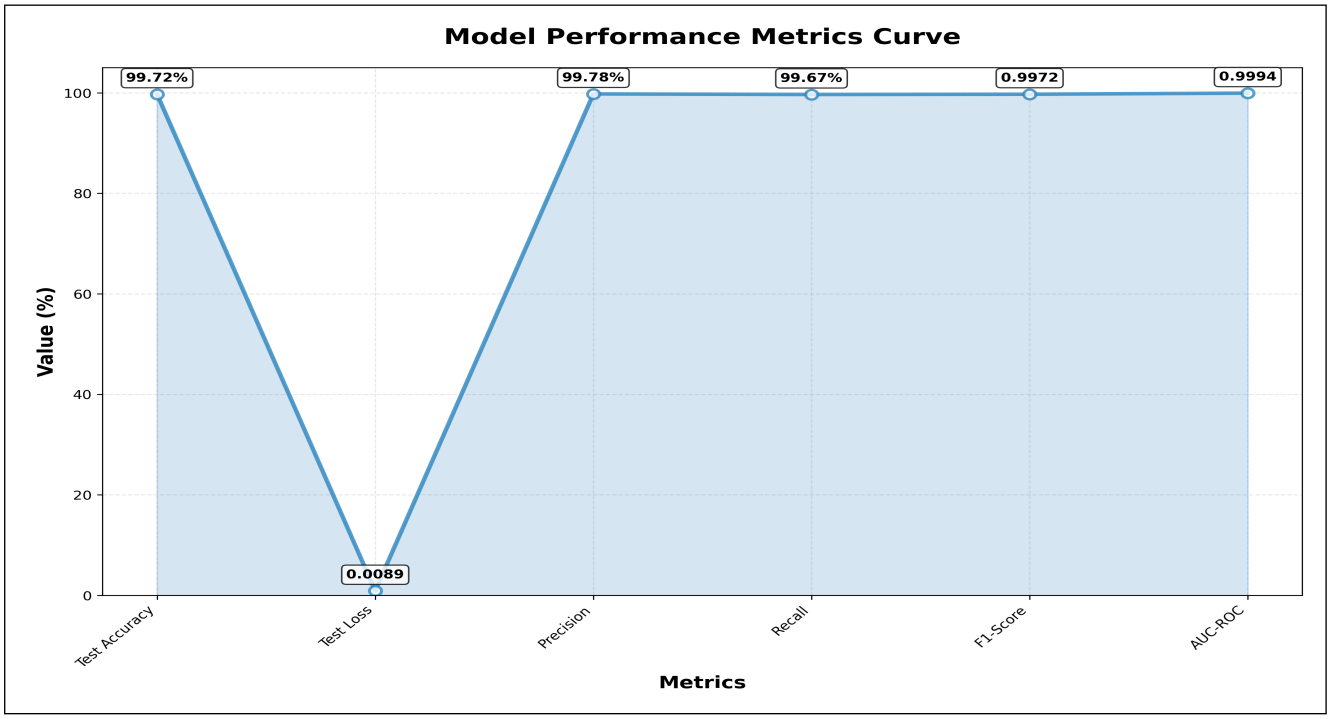

FIGURE 4.3 – Model's Performance Metrics

The above figure presents the performance metrics of the DNN model :

Let :

$$\text{TP} = \text{True Positives} \tag{4.2}$$

$$\text{TN} = \text{True Negatives} \tag{4.3}$$

$$\text{FP} = \text{False Positives} \tag{4.4}$$

$$\text{FN} = \text{False Negatives} \tag{4.5}$$

The results are as follows :

Accuracy (ACC)

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = 0.9972 \tag{4.6}$$

99.72% - The model's predictions were correct an overwhelming majority of the time.

Precision (PPV)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = 0.9978 \tag{4.7}$$

99.78% - Of all the times the model predicted "stable," it was correct 99.78% of the time.

Recall (Sensitivity)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = 0.9967 \tag{4.8}$$

99.67% - The model successfully identified 99.67% of all the instances that were actually "stable."

F1-Score

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 0.9972 \tag{4.9}$$

0.9972 - An excellent combined measure of Precision and Recall.

Test Loss (BCE)

$$\text{Test Loss} = L_{\text{test}} = 0.0089 \tag{4.10}$$

0.0089 - A very low error value, indicating predictions were extremely close to the true values.

AUC-ROC

$$\text{AUC-ROC} = 0.9994 \tag{4.11}$$

0.9994 - Demonstrates a near-perfect ability to distinguish between stable and unstable classes.

After evaluation, the model was converted into a model.tflite file and then into a C++ header (model.h) for direct integration into the firmware. The final prediction logic on the ESP32 combines both AI components.

4.4 Rule-Based AI

The system calculates the engineered features and uses them to identify the root cause of WiFi issues with a series of if/else if statements :

- If (Low RSSI AND Low SNR) → **"Weak signal strength - Move closer to router"**
- If (RSSI Trend is very negative) → **"Signal degrading - Check for obstacles"**
- If (Channel Utilization $> 90\%$) → **"Network congestion - Consider changing channel"**
- If (Noise Variance is high) → **"Unstable environment - Intermittent interference"**

Adding to that the Rule-Based AI calculates a weighted score called "Confidence" that indicates how much the model is confident from the wifi stability and it is calculated as follows :

$$\text{Confidence} = \min\left(\max\left(\text{Base Score} + \sum_{i=1}^{n}(w_i \cdot \mathbb{I}_{\text{condition}_i}), 0\right), 1\right) \tag{4.12}$$

Where :

— $\mathbb{I}_{\text{condition}_i}$ is an **indicator function** that returns `1` if the condition is true and `0` if it is false.

— w_i is the **weight** (points) assigned to that specific condition.

— The final result is **clamped** between 0.0 and 1.0.

The table below represents the weights assigned to each condition :

Category	Condition	Reward / Penalty	Logic
Signal Strength	`RSSI > -50 dBm` (Excellent)	`+0.25`	Strong signal is a clear, positive indicator.
	`RSSI > -70 dBm` (Good)	`+0.20`	Good signal adds to confidence.
	`RSSI > -80 dBm` (Fair)	`+0.15`	Fair signal is a neutral-to-positive indicator.
Signal Quality	`SNR > 30 dB` (Excellent)	`+0.20`	High SNR is a very clear, high-quality indicator.
	`SNR > 20 dB` (Good)	`+0.15`	Good SNR adds to confidence.
	`SNR > 10 dB` (Fair)	`+0.10`	Acceptable SNR adds a small amount.
Low Congestion	`Channel Utilization < 30%`	`+0.15`	Low congestion is a very clear, positive indicator.
	`Channel Utilization < 60%`	`+0.10`	Medium congestion is acceptable.
	`Channel Utilization < 80%`	`+0.05`	High congestion reduces clarity.
Positive Trends	`RSSI_trend > 0` AND `SNR_trend > 0`	`+0.15`	Improving conditions are a strong positive sign.
Stability	`RSSI_variance` AND `Noise_variance` are Low	`+0.10`	Stable signals are predictable and easy to diagnose.
Outlier Penalty	`RSSI < -95 dBm` OR other extreme value	`-0.20`	Extreme values are unusual and make diagnosis less reliable.

TABLE 4.1 – Confidence Scoring System

4.5 Conclusion

In summary, this chapter has detailed the complete engineering lifecycle of the project's intelligent core, from theoretical design to empirical validation. By adopting a hybrid ensemble model, the system successfully combines the predictive power of a data-driven Deep Neural Network with the diagnostic clarity of a feature-rich rule-based engine.

Chapitre

5 Interface and Visualization

Sommaire

5.1 INTRODUCTION

While the AI engine on the ESP32 performs the complex analysis, its value is only realized through an effective user interface. The Web Dashboard is the crucial front-end component that translates raw data and complex AI predictions into an intuitive, interactive, and human-readable visual format. Hosted directly on the ESP32, the dashboard is accessible from any web browser on the local network, requiring no external services or internet connection.

5.2 User Interface

FIGURE 5.1 – Wifi Setup Page

The above figure represents the wifi setup page that is obtained after connecting to the access point provided by the esp32 and reaching the ip address of this page provided by the esp32. This page is an intermediate page where the user will select one of the scanned wifi by the esp32, enter its password and consequently connects the esp32 to the desired network.

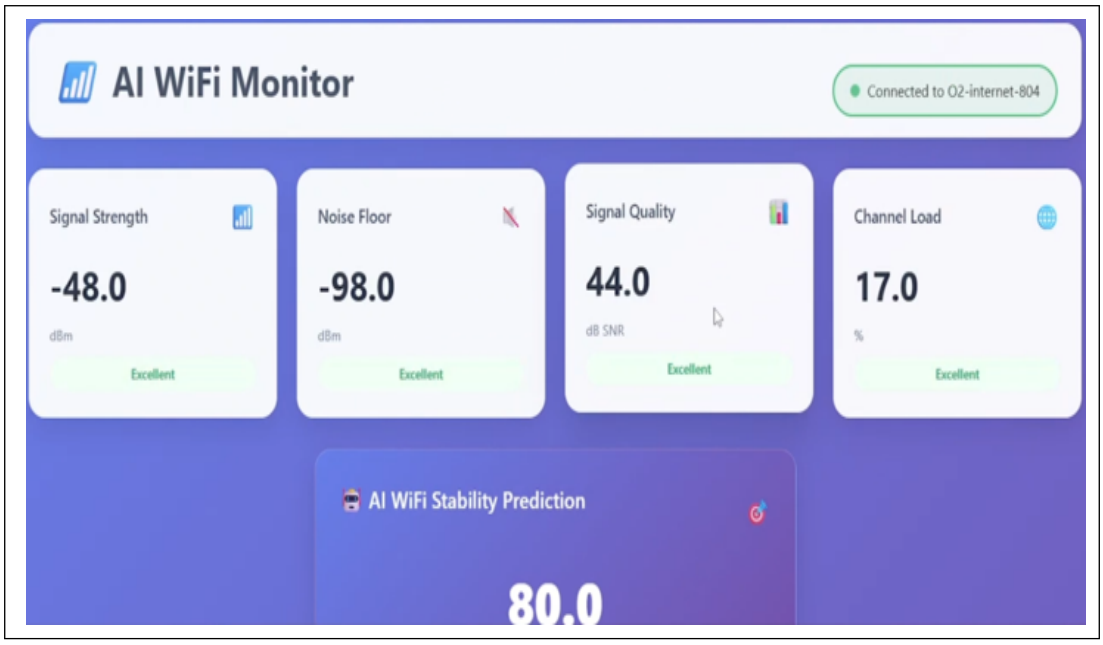

FIGURE 5.2 – KPIs Cards

The figure above represents A set of prominent cards at the top of the page that display the most current values for the four primary KPIs (RSSI, Noise, SNR, Channel Utilization) as well as the latest AI Stability Score. Moreover, the updateStatus() - Real-Time Updates : This JavaScript function is set to run on a short interval (every 10 seconds). It makes an asynchronous fetch call to the ESP32's /advanced-ai API endpoint. Upon receiving the latest JSON data, it updates the text and colors of the real-time status cards and the AI diagnostic panel, ensuring the user always has a live view of the network's current state.

FIGURE 5.3 – KPIs Chart

The figure above represents The central and largest component of the UI, dedicated to visualizing historical data trends. Also, a row of buttons that allow the user to select the time range for the historical data displayed in the chart. This chart is based on updateChart() - Historical Visualization function : This function is responsible for drawing and updating the main historical chart. When a user selects a time range, it makes a fetch call to the `/history?range=X` endpoint, where X is the number of days. The returned JSON array of historical records is then processed and formatted for the charting library. The function renders five distinct data series (curves) on a shared timeline :

- RSSI (dBm)
- Noise (dBm)
- SNR (dB)
- Channel Utilization (%)
- AI Stability (%)

So The primary control allows the user to filter the historical chart to show data for "Today," "Last 1 Day," "Last 3 Days," or "Last 5 Days." Clicking a button triggers the updateChart() function with the corresponding time parameter, seamlessly redrawing the chart with the requested data from the ESP32's local storage. Adding to that, the user disposes of a variety of visualizing options such as zooming in/out, reset chart and he can eventually download the chart in different forms.

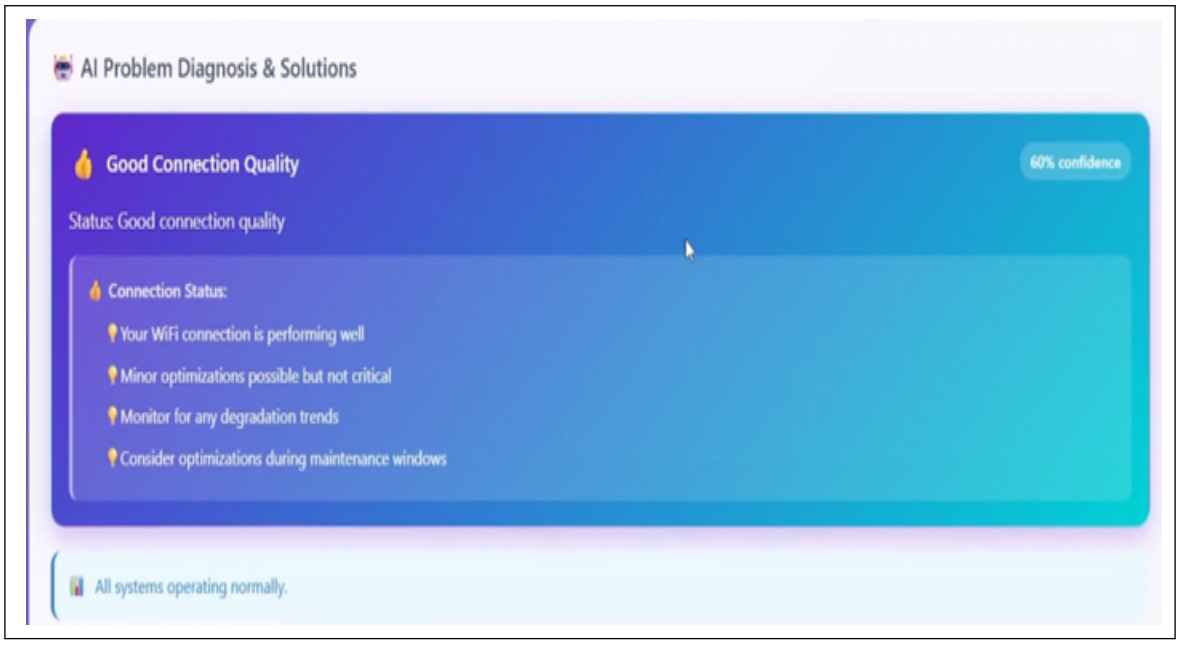

FIGURE 5.4 – KPIs Chart

The figure above represents the dedicated area that shows the most recent intelligent alert message generated by the rule-based AI, providing an immediate diagnosis of any ongoing network issues.

5.3 API Endpoints

The communication between the JavaScript front-end and the ESP32 C++ back-end is facilitated by a set of simple yet powerful HTTP API endpoints.

- `/status` : Provides a single JSON object with the most recent, up-to-the-second WiFi metrics and the final combined stability score.
- `/history` : The primary endpoint for historical data. It accepts a range query parameter (e.g., `?range=3`) and returns a JSON array of historical records for the specified period.
- `/advanced-ai` : Returns a rich JSON object containing not only the current raw metrics but also the detailed output from the rule-based AI, including the calculated confidence score and the intelligent alert message.

- `/scan` & `/connect` : These endpoints are used for the initial, one-time configuration of the device, allowing the user to scan for local networks and provide credentials to connect the monitor to their home or office WiFi.

5.4 Conclusion

In essence, the Web Dashboard successfully bridges the gap between complex back-end analysis and the end-user. By leveraging standard web technologies and a dedicated charting library, it creates a responsive, intuitive, and highly functional interface for the AI WiFi Stability Monitor. The seamless communication between the JavaScript front-end and the ESP32's API endpoints ensures that real-time status updates and rich historical visualizations are delivered efficiently. This chapter has demonstrated that the dashboard is not merely a presentation layer but an integral part of the system, transforming abstract data and AI predictions into a clear, interactive, and actionable user experience.

GENERAL CONCLUSION

The AI WiFi Stability Monitor project has successfully demonstrated that it is possible to create a sophisticated, intelligent, and entirely self-contained network diagnostic tool using low-cost, readily available hardware. By architecting a hybrid ensemble of a data-driven TensorFlow Lite neural network and an expert-driven rule-based AI, the system achieves a powerful synergy of predictive accuracy and diagnostic clarity.

Throughout this report, we have detailed the complete journey from initial concept to a fully functional prototype. We have covered the meticulous process of data collection and on-device feature engineering, the rigorous training and validation of a high-performance machine learning model, and the development of an intuitive web-based dashboard for visualization. The final product is a powerful tool that transforms the often-frustrating task of WiFi troubleshooting into a proactive, data-informed process, providing clear insights into network health without compromising user privacy or requiring an internet connection.

While challenges such as hardware limitations and the need for a comprehensive training dataset were met, the project ultimately achieved its primary goal : to empower users with the ability to understand and diagnose their WiFi network's stability through the practical application of embedded artificial intelligence. This work serves as a strong foundation and proof-of-concept for future enhancements, paving the way for even more advanced and user-friendly smart-monitoring solutions.

REFERENCES

[1] **Al-Dulaimi, A., et al.** A Survey of Machine Learning for Networking : Applications, Challenges, and TFLite Deployment. **IEEE Communications Surveys & Tutorials**, 2018.

[2] **Google**. TensorFlow Lite Documentation **[en ligne]**. 2023. Disponible sur : https ://www.tensorflow.org/lite

[3] **Espressif Systems**. ESP32 Technical Reference Manual **[en ligne]**. 2023. Disponible sur : https ://www.espressif.com/en/support/documents/technical-documents

[4] **PlatformIO**. PlatformIO IDE Documentation **[en ligne]**. 2023. Disponible sur : https ://docs.platformio.org/en/latest/

[5] **Arduino**. Arduino Core for ESP32 Documentation **[en ligne]**. 2023. Disponible sur : https ://github.com/espressif/arduino-esp32

[6] **ARM Ltd.** LittleFS - A high-integrity embedded file system **[en ligne]**. 2020. Disponible sur : https ://github.com/ARMmbed/littlefs

PROJECT TITLE

Aziz HACHICHA

Résumé :

Ce rapport présente la conception et la mise en œuvre du Moniteur de Stabilité WiFi IA, un système embarqué autonome et à faible coût, conçu pour diagnostiquer et prédire de manière proactive l'instabilité des réseaux WiFi. Le système est architecturé autour d'un microcontrôleur ESP32 et emploie un modèle d'intelligence artificielle hybride, combinant un réseau de neurones profond (DNN) au format TensorFlow Lite (TFLite) pour la prédiction basée sur les données, avec un moteur de règles en C++ pour l'ingénierie des caractéristiques en temps réel et la génération d'alertes diagnostiques. Il collecte en continu les métriques WiFi clés (RSSI, SNR, Bruit, Utilisation du canal), les traite localement sur l'appareil et stocke un historique glissant sur un système de fichiers LittleFS local. Toutes les données, y compris l'état en temps réel et les tendances historiques, sont visualisées via un tableau de bord web auto-hébergé et réactif, rendant le comportement complexe du réseau compréhensible pour les utilisateurs non-experts. Le réseau de neurones implémenté a atteint une précision exceptionnelle, démontrant le succès du projet dans la création d'une solution efficace, privée et intelligente pour la surveillance des réseaux WiFi modernes.

Mots clés : ESP32, Stabilité WiFi, IA Embarquée, TensorFlow Lite, Surveillance de Réseau, Modèle Hybride, IdO.

Abstract :

This report details the design and implementation of the AI WiFi Stability Monitor, a low-cost, autonomous embedded system developed to proactively diagnose and predict WiFi network instability. The system is built on an ESP32 microcontroller and employs a hybrid artificial intelligence model, combining a TensorFlow Lite (TFLite) deep neural network (DNN) for data-driven prediction with a C++ rule-based engine for real-time feature engineering and diagnostic alerting. It continuously collects key WiFi metrics (RSSI, SNR, Noise, Channel Utilization), processes them entirely on-device and stores a rolling historical log on a local LittleFS file system. All data, including real-time status and historical trends, is visualized through a responsive, self-hosted web dashboard, making complex network behavior understandable to non-expert users. The implemented neural network achieved an outstanding accuracy, demonstrating the project's success in creating an efficient, private, and intelligent solution for modern WiFi network monitoring.

Key-words : ESP32, WiFi Stability, Embedded AI, TensorFlow Lite, Network Monitoring, Hybrid Model, IoT.