Local Network Load Monitoring IoT Device with ESP32

Gloria Patricia Marsel Acosta Heredia

Universidad Politécnica de Victoria

April 15, 2024

Index

Abstract

This project focuses on the creation of an IoT device using ESP32 to perform efficient load monitoring in a local network. The device collects real-time data about network traffic, analyzes the load, and provides visual and audible feedback based on weighted levels.

Features such as constant real-time monitoring, alerts constant real-time monitoring, visible and audible alerts, for representation, custom configuration of load thresholds, IoT connectivity for remote data access, and resource optimization to improve network performance.

Keywords: IoT device,ESP32,Efficient monitoring,Visual and audible feedback,IoT connectivity.

Introduction

- Network flow monitoring provides the information network administrators need to determine whether it is functioning properly or is saturated.
- The network structure at Universidad Politecnica de Victoria has undergone an expansion of its architecture, currently being five buildings, also increasing the number of students and staff bringing traffic that has come to saturate the academic network for lapses. This problem was previously addressed using pfSense and telegraf but was unsuccessful, leading to the development of this project.

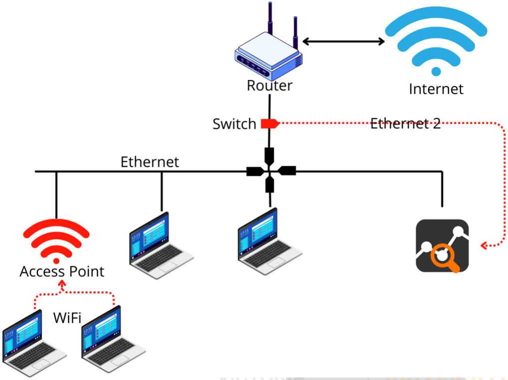
Objectives

- General objective
  - Develop and implement a network usage monitoring device using an ESP for local network for further use in the systems department of the Universidad Politecnica de Victoria.
- Specific objectives
  1. Real Time Traffic Monitoring.
  2. Remote Access to NTopng Information.
  3. Identification of Problematic Users.
  4. Automatic Alerts Generation.

Theoretical framework

Network traffic monitoring is a fundamental network management practice that consists of observing and analyzing the flow of data circulating through a computer network [1]. This activity is performed using specialized tools that collect data on network traffic in real time and provide detailed information on its usage and performance.

Figure: Operation of Ntopng



Theoretical framework

In the case of our selected network monitoring system NTopng:

Table: Characteristics of Ntopng.

Real-Time Monitoring
Deep Traffic Analysis
Intuitive Web Interface
User and Device Detection
Report Generation

NTopng (Network Top Next Generation): An open source network monitoring tool that provides a detailed, real-time view of network traffic.
It is an evolution of its predecessor, NTop, and has been designed to offer enhanced features and a more modern user interface[2].

Theoretical framework

Other software used in the project was:

Table: Characteristics of pfSense.

Firewall.
Routing
Intuitive Web Interface.
VPN
Proxy.
Load Balancing.

pfSense is an open source software distribution based on FreeBSD that is used as a firewall and network router. It provides a robust and customizable platform for managing and protecting computer networks[3].

Theoretical framework.

Other hardware and software used within the project was:

- **The Raspberry Pi 3 Model B v2** is a single-board mini-processor that aims to give everyone the power to explore computing.
- **InfluxDB** is a time series database specially designed to store and query time-varying data such as metrics, events and sensor data [4].
- **Grafana** is a visualization and analytics platform that allows you to create interactive dashboards and graphs from data stored in various sources, including InfluxDB [5].
- **The ESP32** is a low-cost, high-performance microcontroller designed by Espressif Systems.

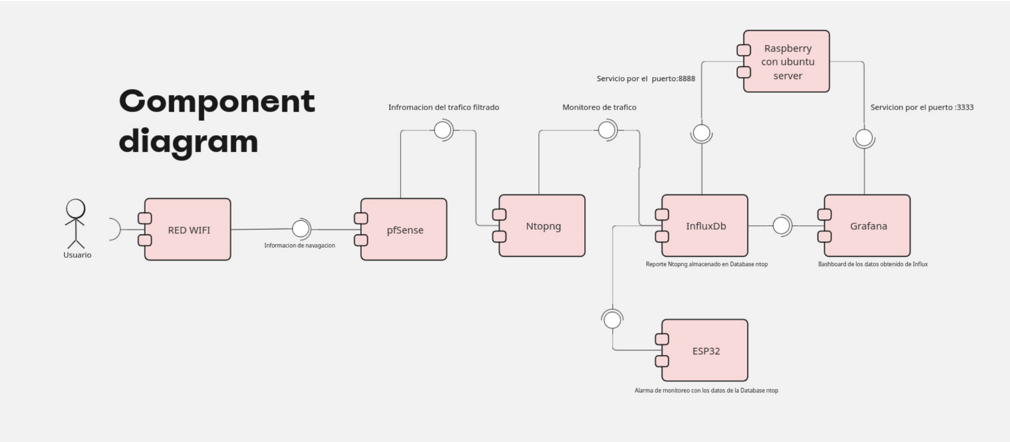
Proposed System: Tools and libraries

Table: Tools and technologies used during development.

Hardware	Software	Libraries
Raspberry Pi 3 Model B	Ubuntu Server 23.10	InfluxDbClient
ESP 32	Arduino IDE	InfluxDbCloud
Micro SD Adata v30 A2	Grafana 22.4	WiFiMulti
Access Point	InfluxDB v1	ESP8266WiFiMulti
Monitor and keyboard for raspberry	Ntopng Comunity	
hp laptop with ubuntu system	pfSense 2.7.2	

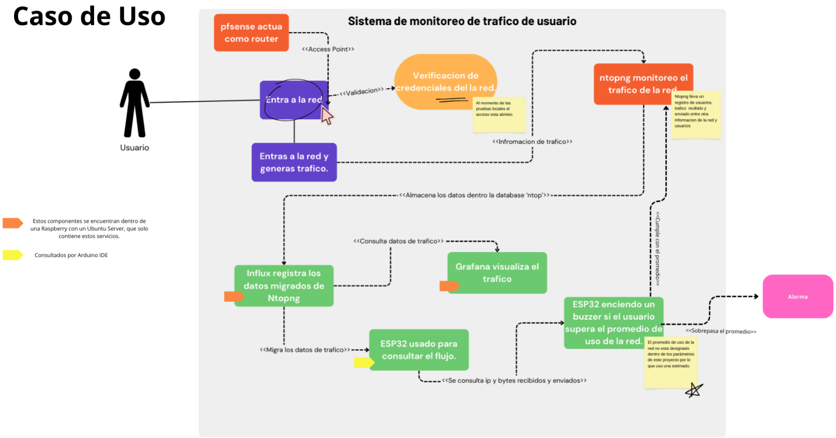
Proposed System: Component diagram

Figure: Component diagram



Proposed System: Use Case

Figure: Use Case



Proposed system.

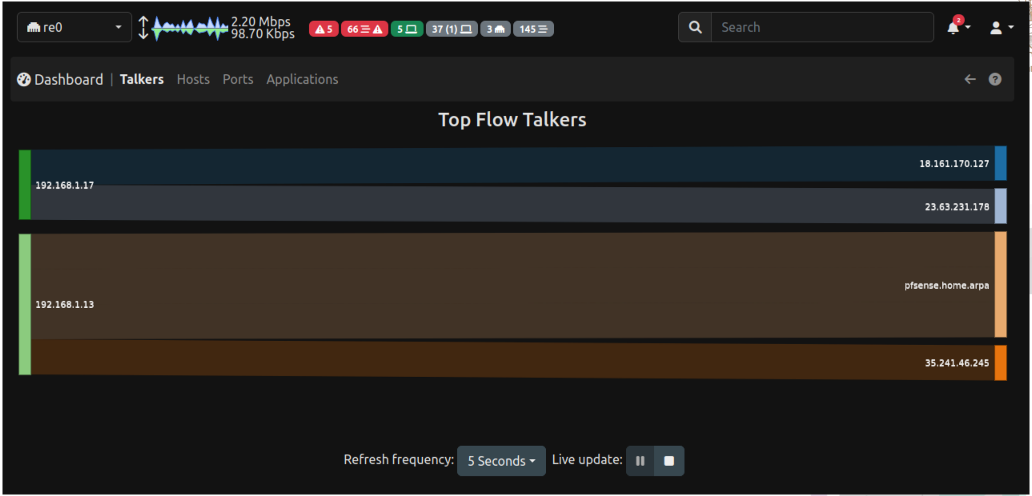


Figure: NTopng: Start of project monitoring

Proposed system:Traffic reports.

The HTTPClient and ArduinoJson libraries were used to query the Ntopng traffic report, however these tests were unsuccessful. So it was decided to continue with a Raspberry.

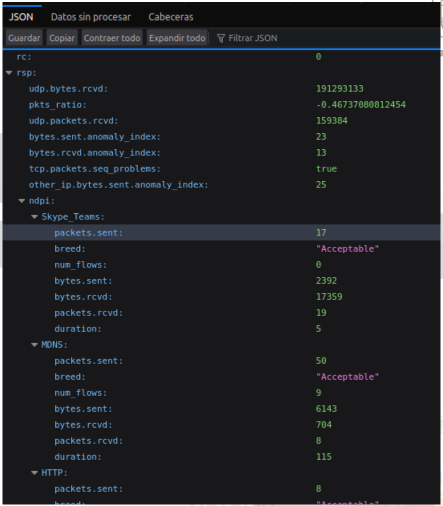


Figure: NTopng: Flow report information in .json format

Proposed system:Raspberry.

This leads us to design the proposed system using the raspberry as a second server for Ntop to communicate with InfluxDB and query the databases from the ESP.

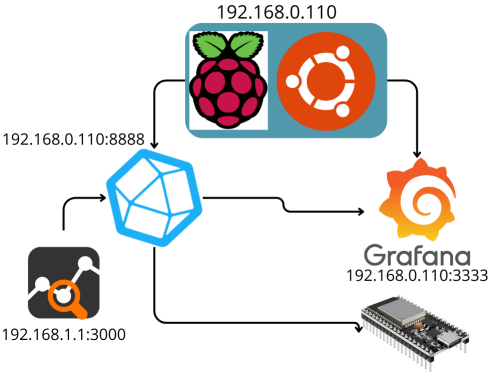


Figure: Configuration of the Raspberry Pi as the final system.

Implementation and Testing: Connect Influx with NTopng

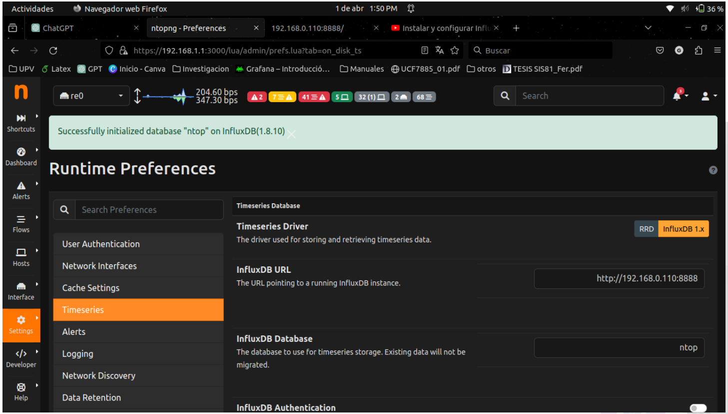


Figure: Ntopng: InfluxDB configuration inside Ntopng in the preferences section.

Implementation and Testing: Connect Influx with NTopng: Database

```
ubuntu@raspberrypi: ~
> SELECT * FROM "host:traffic" LIMIT 10
name: host:traffic
time                bytes_rcvd bytes_sent host          ifid
----                ---------- ---------- ----          ----
1712001061000000000 1573360    25446072   192.168.0.110 0
1712001061000000000 180938069  19947051   192.168.1.13  0
1712001061000000000 3698589    14724634   192.168.1.1   0
1712001121000000000 182091913  20082665   192.168.1.13  0
1712001121000000000 3721657    14748343   192.168.1.1   0
1712001121000000000 1650884    26534621   192.168.0.110 0
1712001181000000000 3748829    14774822   192.168.1.1   0
1712001181000000000 1716033    27105030   192.168.0.110 0
1712001181000000000 183404192  20282277   192.168.1.13  0
1712001241000000000 1729480    27226756   192.168.0.110 0
> SELECT * FROM "host:total_flows" LIMIT 10
name: host:total_flows
time                flows_as_client flows_as_server host          ifid
----                --------------- --------------- ----          ----
1712001061000000000 0               117             192.168.0.110 0
1712001061000000000 5468            37              192.168.1.13  0
1712001061000000000 28              3490            192.168.1.1   0
1712001121000000000 5523            37              192.168.1.13  0
1712001121000000000 28              3526            192.168.1.1   0
1712001121000000000 0               127             192.168.0.110 0
1712001181000000000 28              3564            192.168.1.1   0
1712001181000000000 0               132             192.168.0.110 0
1712001181000000000 5591            37              192.168.1.13  0
1712001241000000000 0               132             192.168.0.110 0
> SELECT * FROM "host:tcp_packets" LIMIT 10
```

Figure: InfluxDb: Query data stored within the ntop database linked to ntopng.

Alarm structure

The first performance tests were made with LED's, designating the blue LED as a stable flow within the network, when the red LED lights up it means that the host occupying more resources has exceeded the calculated average. In the case of the Buzzer, it will be activated when the host with more resources exceeds the average traffic.

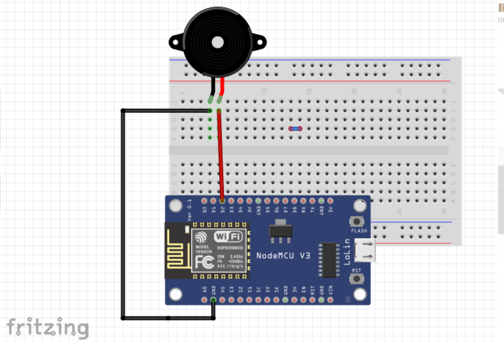


Figure: Configuration of the Raspberry Pi as the final system.

Implementation and Testing: InfluxDB Queries host:traffic

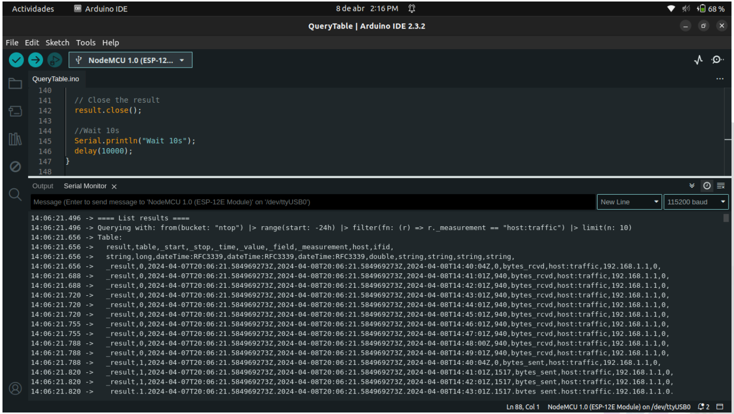


Figure: Arduino IDE: Querying the host:traffic table.

Implementation and Testing: InfluxDB Queries Host query with most bytes used.

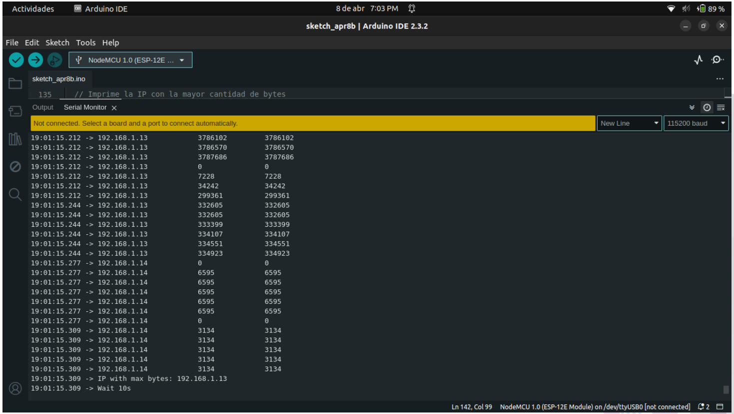


Figure: Arduino IDE: Host query with most bytes used.

Implementation and Testing: Grafana Dashboards

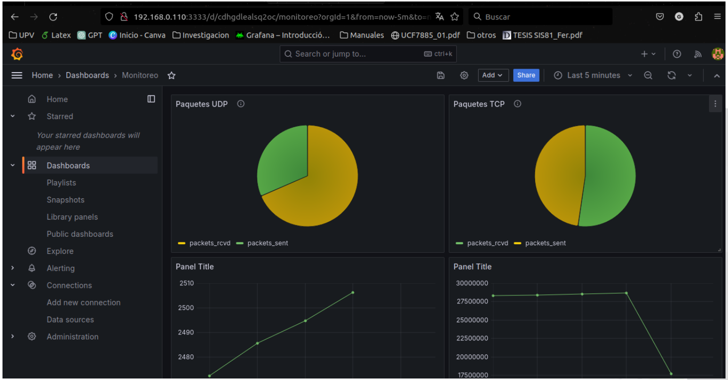


Figure: Grafana: Dashboard of Influx connections.

Implementation and Testing: Flowchart

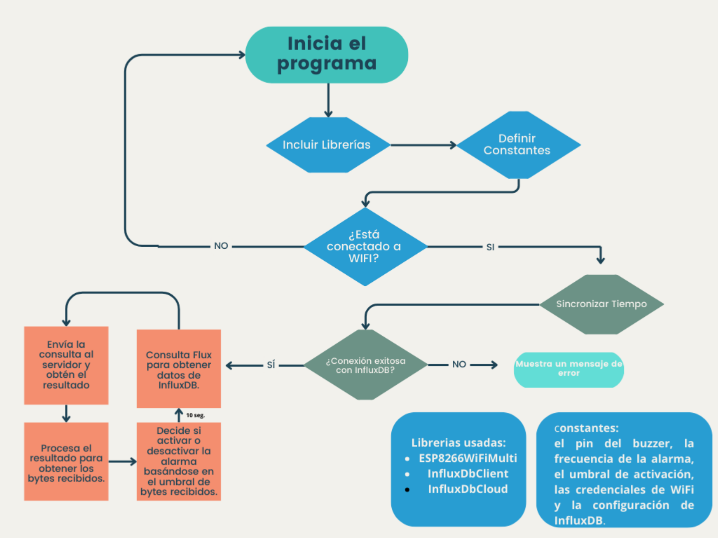


Figure: Flowchart of influxDB query code operation

Implementation and Testing: Test results with led's

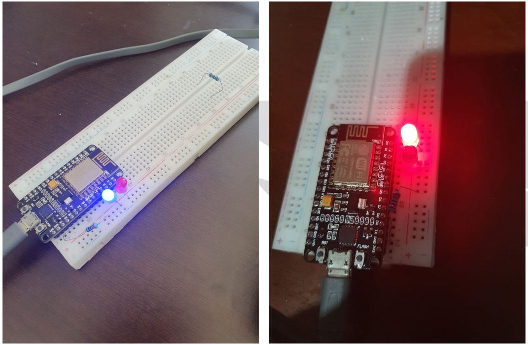


Figure: Alarm test with red and blue LED's to determine when a host is using more resources or if the host is following the designated parameters.

Conclusions

During the time of the project, the aspect of validating that the software selected for its development is equally efficient for the final objective that was developed was concluded. Ntopng was installed within pfSense to monitor the flow that pfSense filters according to the rules designated within it. Ntopng was connected to InfluxDb after verifying that its reports could not be viewed from an ESP and would have to be queried by other methods (with a Raspberry), as well as connecting Influx to Grafana for visual reference of the data obtained. An alarm was implemented using the ESP that functions as an alarm to notify anomalies based on average usage.

Future work

- Conducting tests on larger networks to designate the averages used by sectors, such as the institution's buildings, in order to monitor a larger number of users and determine those that could be generating network congestion.
- Identify usage patterns and possible bottlenecks in network traffic, which in turn could contribute to more efficient management and better optimization of available bandwidth.
- Another area of importance for the future development of the system is the implementation of specialized tools for detecting anomalous behavior in data consumption.

References I

[1] Damián Victorio González and Cruz Quezada Carrasco. "Sistema de monitoreo y análisis de tráfico en la red". In: *Licenciatura en Ingeniería en Sistemas Computacionales* (2015).

[2] *ntopng Documentation — ntopng 6.1 documentation*. https://www.ntop.org/guides/ntopng/. (Accessed on 03/26/2024).

[3] *pfSense Documentation — pfSense Documentation*. https://docs.netgate.com/pfsense/en/latest/. (Accessed on 03/26/2024).

[4] InfluxDB. *InfluxData Documentation*. https://docs.influxdata.com/. (Accessed on 03/26/2024).

[5] Mainak Chakraborty and Ajit Pratap Kundan. "Grafana". In: *Monitoring cloud-native applications: Lead agile operations confidently using open source software*. Springer, 2021, pp. 187–240.

Thank you for your time!