

# **Taller de Proyecto II**

2020

## **Informe Final del Proyecto**

### **PS.4.1**

#### **Investigación sobre monitoreo en Raspberry Pi**

#### **Grupo de Desarrollo**

- Gonzalez Allende, Franco - 1681/9
- Verdolotti, Juan Cruz - 1634/0

## **1.- Proyecto**

El proyecto de investigación sobre monitoreo de una Raspberry Pi correspondiente a la materia Taller de Proyecto II, tiene la finalidad de desarrollar y dar a conocer las diversas herramientas de monitoreo y visualización para sistemas embebidos.

Inicialmente se propuso el monitoreo de Raspberry Pi con Prometheus y Grafana.

Al grupo se le propuso hacer una investigación profunda de las herramientas que se encuentran en el mercado, y hacer eventuales pruebas en Raspberry simuladas.

Se definió simular las más relevantes, por el motivo de los acotados plazos.

El principal objetivo es comparar cada una de estas, y definir las ventajas y desventajas que tienen, y cuál tiene mejor utilidad en sistemas embebidos.

## **2.- Materiales y Presupuesto**

Debido al contexto de aislamiento social, en principio se simulará en VirtualBox varias Raspberry Pi, para poder instalar y probar las herramientas.

Además se podría tener que utilizar una herramienta paga, igualmente se intentará utilizar las gratuitas, y las pagas nombrar y mostrar cómo trabaja a través de información de internet, para no hacer gastos extras.

Los principales elementos de hardware necesarios para el proyecto, son Raspberry Pi, las cuales según precios publicados en mercadolibre, una Raspberry Pi 3 model B+ vale \$6.840.

De todas maneras, es a modo informativo, ya que se acordó con la cátedra, que no será necesario la compra del utilitario porque se tiene disponibilidad de estas en el laboratorio, y comprar otra no tendría sentido.

### 3.- Descripción del Proyecto

El proyecto consta de dos etapas, que son las siguientes:

- 1er Etapa: Investigación de las diferentes herramientas de monitoreo y visualización de datos del mercado para una Raspberry Pi. En esta etapa solo se buscó información de herramientas que puedan utilizarse para monitorear una Raspberry Pi, cómo funcionan cada una de estas, y sus especificaciones.



- 2da Etapa: Instalación y prueba de algunas de las herramientas que se investigaron, para hacer comparativas y llegar a una conclusión. Para esta etapa, se simuló una Raspberry Pi en Virtualbox, para poder realizar las pruebas de las herramientas. Esto se llevó a cabo, instalando la imagen del sistema operativo de una Raspberry Pi (Raspbian) desde la página oficial de Raspberry, y montando la imagen en la máquina virtual.

La diferencia de la simulación con respecto al sistema físico es que a la hora de realizar los monitoreos con las herramientas, monitorea los datos de la PC en la que se está realizando el monitoreo. La simulación es una aproximación de cómo debería funcionar la herramienta más allá de no tener el hardware.



A continuación se muestra el enlace para acceder a la bitácora realizada, donde se muestra el avance cronológico del proyecto, con las páginas web utilizadas de guía.

<https://github.com/tpII/2020-ps-4.1-prometheus-grafana/wiki/Bitácora>

## 4- Guía de Instalación: Proyecto y Ambiente de Desarrollo

### 1er Etapa - Investigación

A continuación se muestra la información obtenida de las herramientas más relevantes presentes en el mercado, para el monitoreo de un sistema embebido:

#### **RPI MONITOR:**

Es un software que nos permite monitorizar la Raspberry Pi, presentándonos, en una sencilla pero eficaz página web, el estado de salud de la Raspberry Pi. Esto es particularmente interesante para los que, como en mi caso, no hemos instalado un entorno gráfico en la Raspberry Pi.

Por cuestiones de seguridad y rendimiento RPi Monitor separa la parte de obtener y almacenar los datos, de la parte de mostrar los datos. La parte de obtención de todos los datos es realizada por un proceso que funciona en modo demonio ejecutada con derechos de administrador. Todos los datos extraídos se guardan en una base de datos. Esto nos permite conocer el histórico sobre la salud de nuestra Raspberry Pi.

Para la presentación de la información se utiliza páginas HTML5, con bootstrap, jquery, y otra tecnología. Todo ello, tiene la ventaja de que descarga al servidor de trabajo, porque el proceso de renderizado es por cuenta del cliente. Menos trabajo para la Raspberry Pi.

Lo primero que nos encontramos con RPi Monitor, y me refiero a la página web que aparecerá al conectarnos a la Raspberry Pi, es un resumen del estado de salud.

En este resumen del estado de salud de nuestra Raspberry Pi, encontramos los siguientes parámetros:

- Versión: Nos da datos sobre el procesador, la distribución y versión instalada. Igualmente nos indica la versión del núcleo de Linux, y la del firmware. Por último, en este apartado, nos indica si hay paquetes por instalar.

- Uptime: En este apartado nos indica la hora del reloj de nuestra Raspberry Pi, y cuanto tiempo lleva en funcionamiento desde el último arranque

- CPU: Nos indica la carga de la CPU, y otros parámetros como frecuencia, tensión, etc.

- Temperature: Nos indica la temperatura medida en la CPU.

- Memory: Aquí podemos ver la cantidad de memoria en uso y disponible.

- Swap: En este punto encontraremos el uso y estado de la memoria de intercambio.

- SD Card: Lo mismo que en el apartado anterior pero para el caso de nuestra tarjeta SD.

-Network: Aquí verás los paquetes enviados y recibidos según tu conexión.

## **GRAPHITE:**

Es una herramienta gratuita de software de código abierto que monitorea y gráfica.

datos numéricos de series temporales, como el rendimiento de los sistemas informáticos. Graphite no recopila datos, hay un componente, un demonio Twisted llamado Carbon, que escucha pasivamente datos de series de tiempo. Los datos se almacenan en una biblioteca simple llamada Whisper . Finalmente, los gráficos se pueden renderizar bajo demanda a través de una aplicación web Django simple.

## **PROMETHEUS:**

Es una aplicación de software gratuita utilizada para la supervisión y alerta de

eventos. Registra métricas en tiempo real en una base de datos de series de tiempo construida utilizando un modelo de extracción HTTP, con consultas flexibles y alertas en tiempo real.

Desarrollado más recientemente, asume el desafío adicional de la escala y contiene numerosas características, incluido un lenguaje de consulta flexible (PromQL), una puerta de enlace push (para recopilar métricas de trabajos efímeros o por lotes), una variedad de exportadores y otras herramientas.

PromQL es el lenguaje de consulta que se utiliza para crear paneles y alertas.

Prometheus recopila datos de forma activa , los almacena y admite consultas, gráficos y alertas , además de proporcionar puntos finales a otros consumidores de API como Grafana o incluso Graphite.

Hace todo esto a través de los siguientes componentes:

-Bibliotecas cliente: instrumentación del código de la aplicación (para generar eventos);

-Servidor Prometheus: raspando y almacenando estos eventos, cuando se disparan, como datos de series de tiempo;

-Pushgateway: admite trabajos de importación de datos de corta duración;

-Exportadores de datos : exportan a servicios como HAProxy, StatsD, Graphite, etc .;

-Alertmanager: manejo de alertas. Prometheus especifica una condición que debe mantenerse durante un período específico para que se active una alerta. Cuando se activan las alertas, se envían al servicio Alertmanager, que puede incluir lógica para silenciar las alertas y también para reenviarlas al correo electrónico, Slack o servicios de notificación como PagerDuty.

## **GRAFANA:**

Es una aplicación web de visualización interactiva y analítica de código abierto multiplataforma . Proporciona cuadros, gráficos y alertas para la web cuando se conecta a fuentes de datos compatibles. Grafana tiene un backend muy ligero y admite más de 30 fuentes de datos comerciales y de código abierto con una infraestructura mínima.

Puede integrarse a la perfección con su flujo de trabajo para ayudar a explorar métricas, visualizar datos, explorar registros, evaluar puntos de datos y notificar problemas utilizando diferentes canales, y puede hacerlo todo en tiempo real.

Proporciona soporte integrado para importantes bases de datos de series temporales como Graphite, influxDB, Prometheus, Elasticsearch y muchas más.

Viene con una gran colección de complementos interactivos que van desde cuadros y gráficos simples hasta mapas de calor, geomapas, diagramas de flujo y muchos más.

Algunos paneles de Grafana le permiten definir reglas de alerta y evaluar métricas de forma continua contra umbrales predefinidos. Luego, puede enviar notificaciones a través de diferentes canales como correo electrónico, Slack, PagerDuty, VictorOps, etc.

Grafana le permite crear cuadros de mando dinámicos con la ayuda de variables de plantilla. Estos paneles muestran toneladas de datos utilizando la misma plantilla para aumentar la reutilización.

Grafana también permite consultar y mezclar diferentes fuentes de datos en el mismo gráfico, lo cual es una característica poco común. Además, otras características como el uso de filtros Ad-hoc y la adición de anotaciones lo convierten en un competidor duro en el mercado de herramientas analíticas y de visualización de datos.

## **ZABBIX:**

Zabbix es una herramienta de software de monitoreo de código abierto para diversos componentes de TI, incluidas redes , servidores , máquinas virtuales (VM) y servicios en la nube. Zabbix proporciona métricas de monitoreo, entre otras, la utilización de la red, la carga de la CPU y el consumo de espacio en disco. La configuración de monitoreo de Zabbix se puede realizar utilizando plantillas basadas en XML que contienen elementos para monitorear. También ofrece varias opciones de monitoreo:

- Las comprobaciones simples pueden verificar la disponibilidad y la capacidad de respuesta de los servicios estándar como SMTP o HTTP sin instalar ningún software en el host monitoreado.

- También se puede instalar un agente Zabbix en hosts UNIX y Windows para monitorear estadísticas como la carga de la CPU, la utilización de la red, el espacio en disco, etc.

-Como alternativa a la instalación de un agente en los hosts, Zabbix incluye soporte para el monitoreo a través de comprobaciones SNMP , TCP e ICMP , así como a través de IPMI , JMX , SSH , Telnet y el uso de parámetros personalizados. Zabbix admite una variedad de mecanismos de notificación casi en tiempo real, incluido XMPP .

### **DATADOG:**

Datadog le permite ver las métricas de la infraestructura, los seguimientos y los registros, todo en el mismo panel. Datadog admite visualización, resolución de problemas, monitoreo y alertas. Datadog se integra con cientos de aplicaciones o servicios diferentes y puede comunicarse con cualquier entorno, como servidores, contenedores, dispositivos móviles, navegadores web y servicios en la nube. El objetivo principal de Datadog es ver dentro de cualquier pila y cualquier aplicación. Datadog también admite alertas, colaboración y le permite combinar datos de varias fuentes en una sola imagen.

### **ICINGA:**

Icinga es una aplicación para ordenador escrita en código abierto con el fin de monitorizar sistemas y monitorizar redes. Originalmente fue creada a partir una bifurcación del software Nagios en el año 2009.

Intenta superar las deficiencias percibidas en el proceso de desarrollo de Nagios,<sup>4</sup> así como la adición de nuevas características tales como una moderna interfaz web estilizada al usuario,<sup>56</sup> métodos de conexión a varias bases de datos (MySQL, Oracle y PostgreSQL),<sup>3</sup> y una REST API que permite a los administradores integrar numerosas extensiones sin complicadas modificaciones del núcleo de Icinga.

Icinga Core está escrito en lenguaje C y tiene una arquitectura modular con núcleo autónomo,<sup>18</sup> interfaz de usuario y base de datos en la que los usuarios pueden integrar diversos complementos.

### **SENSU:**

Sensu es una solución completa para el monitoreo y la observabilidad a escala. Sensu Go está diseñado para brindarle visibilidad de todo lo que le importa: armarios de servidores tradicionales, contenedores, aplicaciones, la nube y más.

Sensu es la solución preparada para el futuro para el monitoreo de múltiples nubes a escala. La canalización de eventos de monitoreo de Sensu permite a las empresas automatizar sus flujos de trabajo de monitoreo y obtener una visibilidad profunda de sus entornos de múltiples nubes. Sensu ofrece una solución de monitoreo integral para empresas, brindando visibilidad completa en todos los sistemas, todos los protocolos, en todo momento, desde Kubernetes hasta bare metal.



## **KIBANA:**

Kibana es una interfaz web escalable para la representación visual de datos. Junto con Elasticsearch y la herramienta de procesamiento de datos Logstash, forma la llamada “pila ELK” (también llamada Elastic Stack). Esta suite de código abierto permite a los usuarios recoger, organizar y preparar datos con fines analíticos desde diferentes servidores (y en cualquier formato). Además de la capacidad de visualizar los datos procesados por Logstash y Elasticsearch, Kibana también ofrece análisis automáticos en tiempo real, un algoritmo de búsqueda muy flexible y diferentes tipos de vistas (histogramas, gráficos, diagramas circulares, etc.) para los datos. En el panel de control (dashboard), las diversas visualizaciones interactivas pueden combinarse para formar una imagen general dinámica que permita su filtrado y examen.

Al ser una aplicación web escrita en JavaScript, Kibana puede utilizarse en todas las plataformas. Solo se incurrirá en gastos si se utiliza el servicio de hosting Elastic Cloud, ofrecido por el desarrollador. Este servicio de pago permite implementar y organizar un clúster seguro Kibana-Elasticsearch en Amazon o Google sin tener que proporcionar recursos propios.

Kibana envía datos de forma central con los clásicos: histogramas, grafos de líneas, gráficos circulares, proyecciones solares y más. Y, por supuesto, puedes buscar en todos tus documentos.

## **VENTAJAS:**

- Crea visualizaciones de forma sencilla e intuitiva
- Dashboards que impulsan el conocimiento
- Creación de alertas que desencadenan acciones personalizadas

## **NETDATA:**

Es una herramienta para visualizar y monitorear métricas en tiempo real, optimizada para acumular todo tipo de datos, como uso de CPU, actividad de disco, consultas SQL, visitas a un sitio web, etc.

Netdata lo ayuda a monitorear y solucionar todo tipo de dispositivos y las aplicaciones que ejecutan, incluidos los dispositivos de IoT.

Con Netdata instalado, también puede monitorear las métricas del sistema y cualquier otra aplicación que pueda estar ejecutando. De forma predeterminada, Netdata recopila métricas sobre el uso de CPU, E / S de disco, ancho de banda, uso de recursos por aplicación y mucho más.

Netdata recopila automáticamente alrededor de 1.500 métricas por segundo.

## **NAGIOS:**

Nagios es un sistema de monitorización de redes ampliamente utilizado, de código abierto, que vigila los equipos (hardware) y servicios (software) que se especifiquen, alertando cuando el comportamiento de los mismos no sea el deseado. Entre sus características principales figuran la monitorización de servicios de red (SMTP, POP3, HTTP, SNMP...), la monitorización de los recursos de sistemas hardware (carga del procesador, uso de los discos, memoria, estado de los puertos...), independencia de sistemas operativos, posibilidad de monitorización remota mediante túneles SSL cifrados o SSH, y la posibilidad de programar plugins específicos para nuevos sistemas.

Se trata de un software que proporciona una gran versatilidad para consultar prácticamente cualquier parámetro de interés de un sistema, y genera alertas, que pueden ser recibidas por los responsables correspondientes mediante (entre otros medios) correo electrónico y mensajes SMS, cuando estos parámetros exceden de los márgenes definidos por el administrador de red.

## **CACTI:**

Cacti es un programa para recoger información sobre el estado del ordenador y de la red y mostrarla gráficamente. En raspbian (al igual que en debian) la instalación por defecto de cacti se hace con el servidor web Apache. Como en mi caso he instalado lighttpd como servidor web es mejor hacer la instalación de cacti en varios pasos.

Es una completa solución para la generación de gráficos en red, diseñada para aprovechar el poder de almacenamiento y la funcionalidad para gráficas que poseen las aplicaciones RRDtool. Esta herramienta, desarrollada en PHP, provee un pooler ágil, plantillas de gráficos avanzadas, múltiples métodos para la recopilación de datos, y manejo de usuarios. Tiene una interfaz de usuario fácil de usar, que resulta conveniente para instalaciones del tamaño de una LAN, así como también para redes complejas con cientos de dispositivos.

Puedo, a través de Cacti , representar gráficamente los datos almacenados en la RRD: uso de conexión a internet, datos como temperatura, velocidad, voltaje, número de impresiones, etc. La RRD va a ser utilizada para almacenar y procesar datos recolectados vía SNMP.

## **PANDORA FMS:**

Pandora FMS es un software de código abierto que sirve para monitorear (monitorizar) y medir todo tipo de elementos. Monitoriza sistemas, aplicaciones o dispositivos de red. Permite conocer el estado de cada elemento de un sistema a lo largo del tiempo ya que dispone de histórico de datos y eventos. Pandora FMS está orientado a grandes entornos, y permite gestionar con y sin agentes, varios miles de sistemas, por lo que se puede emplear en grandes clusters, centros de datos y redes de todo tipo.

Pandora FMS está formado por tres componentes: servidor, consola y agente.

El servidor de Pandora FMS es quien procesa los datos recolectados de diferentes maneras; también son los que ejecutan alertas y guardan la información en la base de datos.

La consola es la interfaz web con la interfaz al usuario para administrar los servidores, catalogar la información, crear alertas, crear incidentes, cambiar contraseñas de acceso y en general permiten toda la configuración del sistema de manera horizontal. Aquí se realiza la conversión de lenguaje de bajo nivel al lenguaje de alto nivel.

Los agentes de Pandora FMS son entidades organizativas, generalmente un ordenador. Los agentes tienen la información, y pertenecen a un solo grupo.

## **2da Etapa - Instalación y prueba:**

Para realizar el proyecto, se necesito en primera instancia la instalación del Virtualbox para poder simular el sistema operativo que utiliza la Raspberry Pi.

Luego de este paso, se descargó la imagen del sistema operativo de la Raspberry Pi (Raspbian) de la página oficial. Esta misma fue instalada en la máquina virtual, repitiendo los pasos de la siguiente guía encontrada en internet para configurar todo de la mejor forma. (<https://roboticsbackend.com/install-raspbian-desktop-on-a-virtual-machine-virtualbox/>)

Al finalizar esta instalación, ya estaba funcionando la simulación de la Raspberry Pi correctamente. Ahora el siguiente paso, por las dificultades para realizar el proyecto a distancia, decidimos utilizar el protocolo SSH para poder acceder remotamente a la misma Raspberry Pi. SSH sigue un modelo cliente-servidor. El cliente inicia una petición al servidor, que autentifica la comunicación e inicia el entorno Shell. Múltiples clientes pueden conectarse a un mismo servidor. Para realizar la configuración SSH en nuestra Raspberry Pi simulada, seguimos la siguiente guía encontrada en internet: <https://www.luisllamas.es/raspberry-pi-ssh/>.

Ya teniendo la simulación funcionando correctamente, empezamos con la instalación y la prueba de las herramientas que según la investigación realizada son las más adecuadas y conocidas para sistemas embebidos.

## **PROMETHEUS Y GRAFANA**

Primero decidimos instalar Grafana y Prometheus, para hacer la prueba del proyecto original. Seguimos la siguiente guía para realizar la instalación de ambos: <https://leanpub.com/rpcmonitor/read#leanpub-auto-about-prometheus>. Esta instalación se pudo realizar sin ningún inconveniente siguiendo los pasos, y funcionó todo perfecto desde el comienzo. Primero procedimos con la instalación de Prometheus para la arquitectura Armv7, que es la que usa la CPU de una Raspberry Pi 4.

```
wget https://github.com/prometheus/prometheus/releases/download/v2.17.1/prometheus-2.17.1.linux-armv7.tar.gz
```

Luego creamos el archivo de servicio de prometheus, llamado prometheus.service con el próximo comando y copiamos el siguiente texto en el archivo:

```
sudo nano /etc/systemd/system/prometheus.service
```

```
[Unit]
Description=Prometheus Server
Documentation=https://prometheus.io/docs/introduction/overview/
After=network-online.target

[Service]
User=pi
Restart=on-failure

#Change this line if Prometheus is somewhere different
ExecStart=/home/pi/prometheus/prometheus \
  --config.file=/home/pi/prometheus/prometheus.yml \
  --storage.tsdb.path=/home/pi/prometheus/data

[Install]
WantedBy=multi-user.target
```

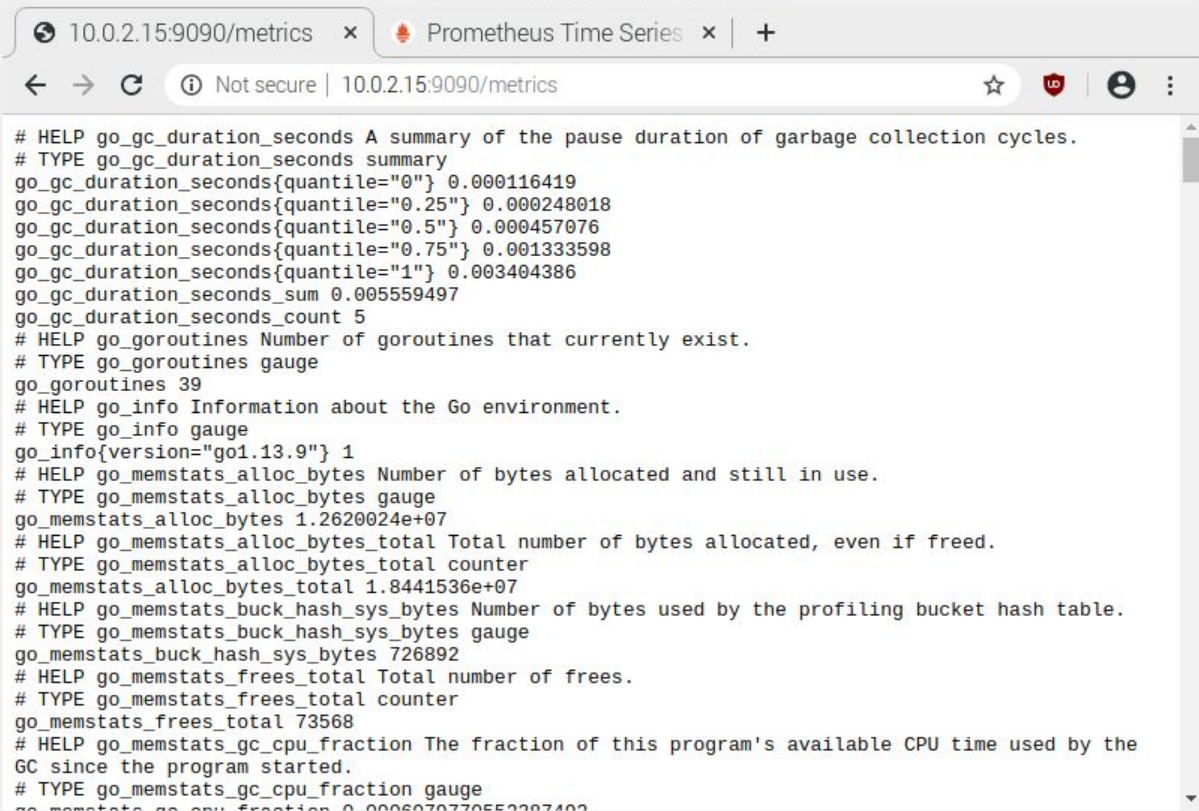
Y después ejecutamos el siguiente comando para que systemctl conozca el nuevo servicio:

```
sudo systemctl daemon-reload
```

Y para finalizar, iniciamos el servicio de prometheus con el siguiente comando:

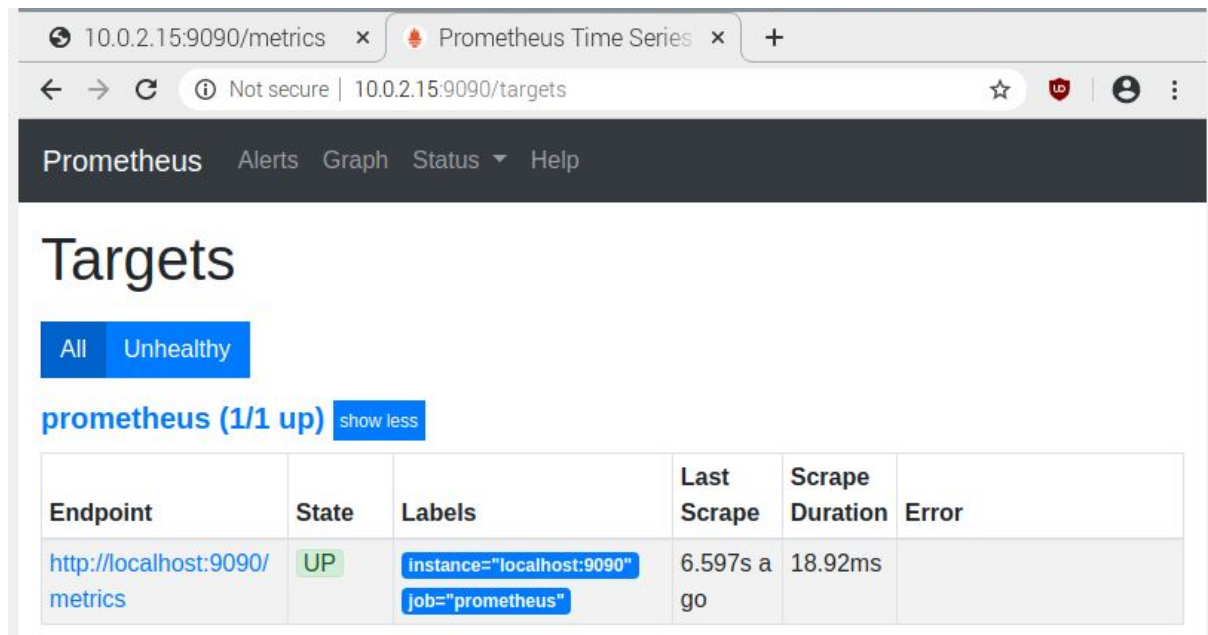
```
sudo systemctl start prometheus
```

A partir de ahí, para comprobar que todo funciona de manera correcta, entramos a la url: 10.0.2.15:9090/metrics, y se puede ver que Prometheus estaba proporcionando métricas, como se muestra en la siguiente captura:



```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0.000116419
go_gc_duration_seconds{quantile="0.25"} 0.000248018
go_gc_duration_seconds{quantile="0.5"} 0.000457076
go_gc_duration_seconds{quantile="0.75"} 0.001333598
go_gc_duration_seconds{quantile="1"} 0.003404386
go_gc_duration_seconds_sum 0.005559497
go_gc_duration_seconds_count 5
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 39
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.13.9"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.2620024e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.8441536e+07
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 726892
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 73568
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the program started.
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 0.0006070770552227402
```

Ahora podemos ingresar a la dirección 10.0.15.2:9090 para confirmar que Prometheus estaba funcionando correctamente, en el menú desplegable ‘Status’ y luego a ‘Targets’ podemos ver la lista de targets de los que Prometheus está extrayendo métricas, como se ve en la siguiente captura:



Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
<a href="http://localhost:9090/metrics">http://localhost:9090/metrics</a>	UP	<code>instance="localhost:9090"</code> <code>job="prometheus"</code>	6.597s ago	18.92ms	

Ya finalizada la instalación de Prometheus, el siguiente paso es la instalación de Grafana, que al igual que Prometheus, descargamos la versión para la arquitectura Armv7 de la página oficial.

```
wget https://dl.grafana.com/oss/release/grafana-6.5.3.linux-armv7.tar.gz
```

También al igual que con Prometheus, creamos el archivo de servicio y lo iniciamos:

```
sudo nano /etc/systemd/system/grafana.service
```

```
[Unit]
Description=Grafana Server
After=network.target

[Service]
Type=simple
User=pi
ExecStart=/home/pi/grafana/bin/grafana-server
WorkingDirectory=/home/pi/grafana/
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl daemon-reload
```

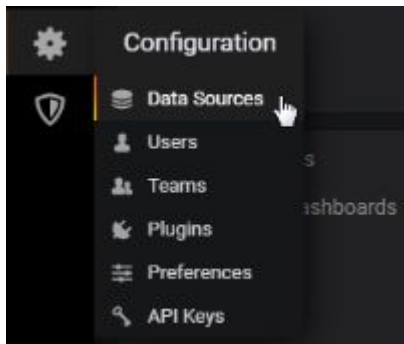
```
sudo systemctl start grafana
```

En este punto ya era posible conectarse a través del navegador al servidor Grafana por la dirección 10.0.15.2:3000.

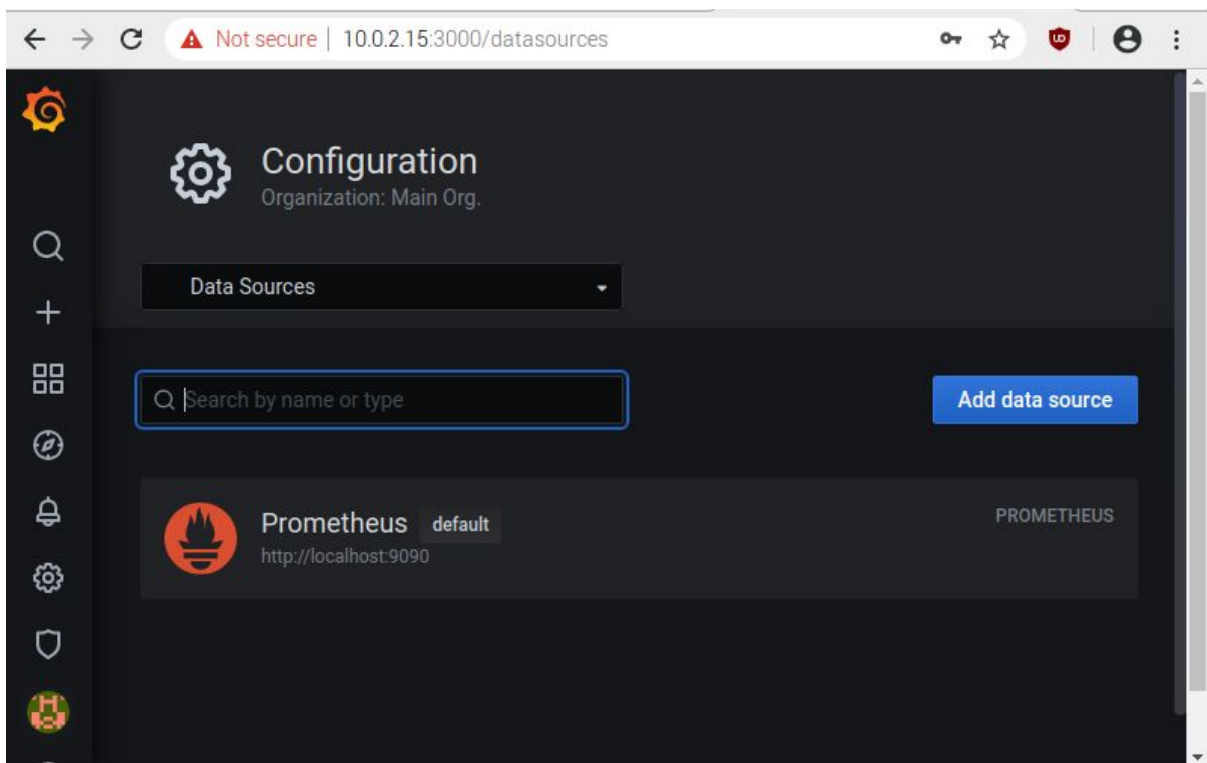


La cuenta es admin, y la contraseña es admin. Luego, Grafana pedirá que definamos una nueva contraseña.

El último paso fue la configuración para crear una fuente de datos que Grafana utiliza para recopilar métricas a partir de Prometheus. Acá ingresamos en 'Configuration' y luego en 'Data Sources':

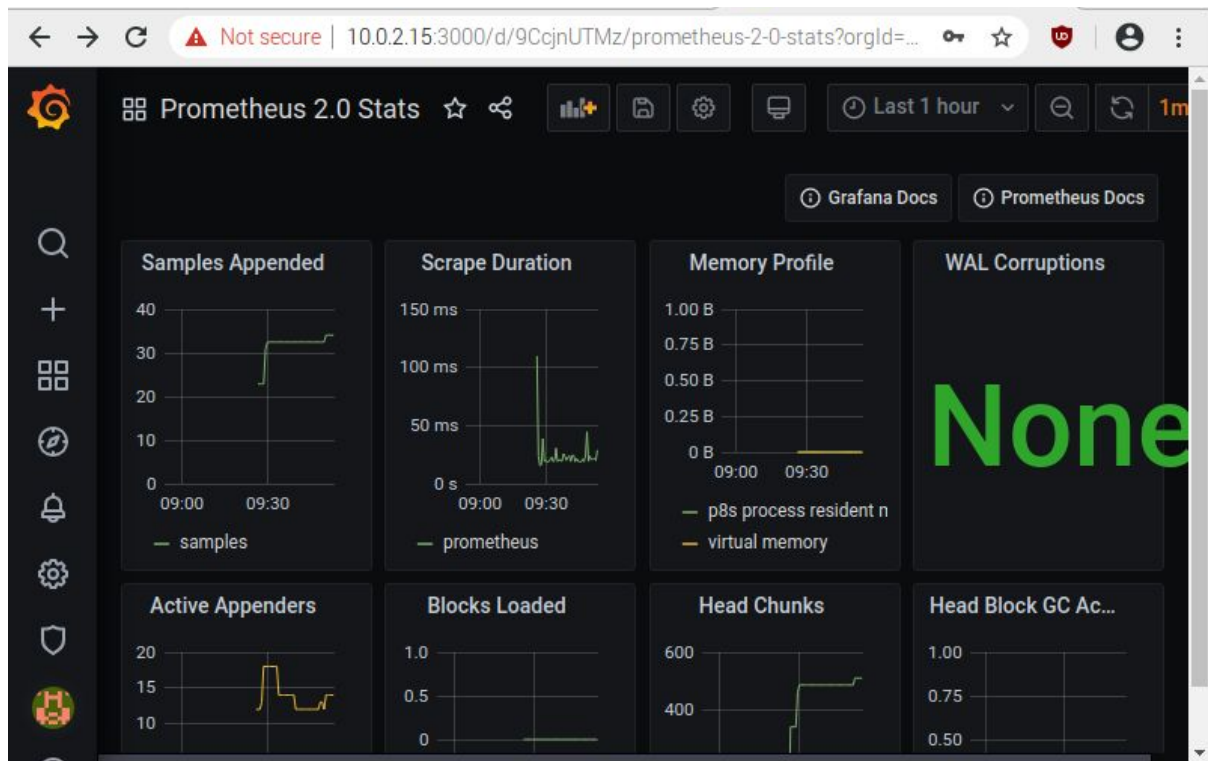


Después agregamos en Data Sources a Prometheus y configuramos unos ajustes para la conexión con el servicio de Prometheus:





Y luego de esto importamos los dashboards del panel de estadísticas de Prometheus, para poder visualizar las métricas recolectadas. Elegimos Prometheus 2.0 Stats que tira unas estadísticas por defecto, pero se puede agregar cualquier tipo de dashboard y modificar a gusto. En la siguiente captura se puede observar las métricas mostradas:



## GRAPHITE

Para la instalación de la herramienta se siguió las instrucciones presentes en esta página <https://arpitbhayani.me/blogs/setting-up-graphite-using-nginx-on-ubuntu> .

Se proponen los siguientes pasos:

En esta primera instancia se propone instalar NGINX que es un servidor web muy rápido, más rápido que la mayoría de los servidores web disponibles en el mercado.

```
sudo apt-get install nginx nginx-extras
```

Se instala el paquete Graphite Ubuntu.

```
sudo apt-get update  
sudo apt-get install graphite-web graphite-carbon
```

Se instala y configura la base de datos de PostgreSQL.

Script para instalar la base de datos y las bibliotecas utilizadas por Graphite para comunicarse con PostgreSQL.

```
sudo apt-get install postgresql libpq-dev python-psycopg2
```

Una vez instalado nuestro PostgreSQL se crea un usuario y una base de datos.

Se inicia sesión en la consola de PostgreSQL.

```
sudo -u postgres psql
```

Y se procede a seguir los pasos de la página, previamente mencionados.

- Se configura la aplicación Web Graphite, también siguiendo los pasos mencionados en la página.
- Se sincroniza la base de datos

Aquí la página sugiere utilizar el comando:

```
sudo graphite-manage syncdb
```

En nuestra simulación de Raspberry Pi, este comando “syncdb” no funciona. Como no hicimos una migración anteriormente, primero se debe correr el comando **sudo graphite-manage makemigrations** y luego **sudo graphite-manage migrate**.

- Se configura Carbon, que es el backend de almacenamiento de Graphite.
- Se configura esquemas de almacenamiento.
- Se configura métodos de agregación de almacenamiento.
- Se configura uwsgi y script de inicio.

Aquí los pasos sugeridos en la página siguen, pero se presentaron inconvenientes que no fueron posibles resolver.

Los mismos se van a encontrar en el próximo apartado “Problemas y soluciones”.

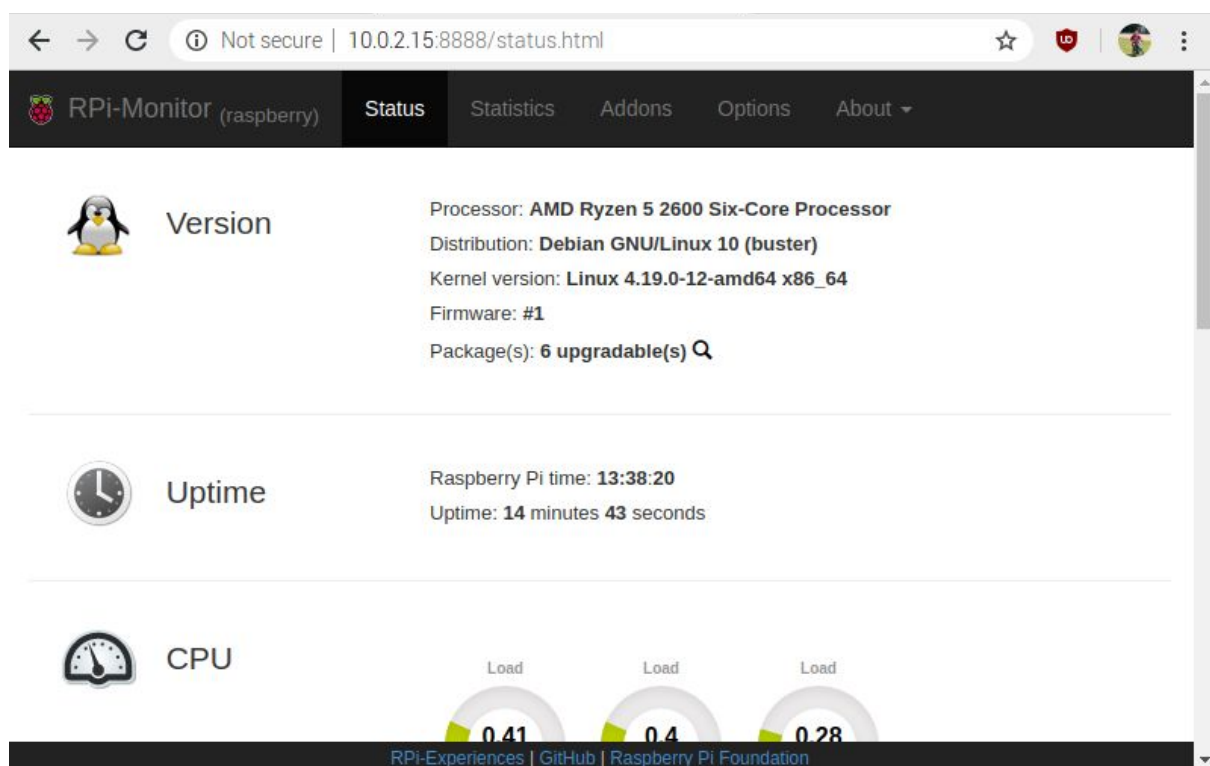
## RPI MONITOR

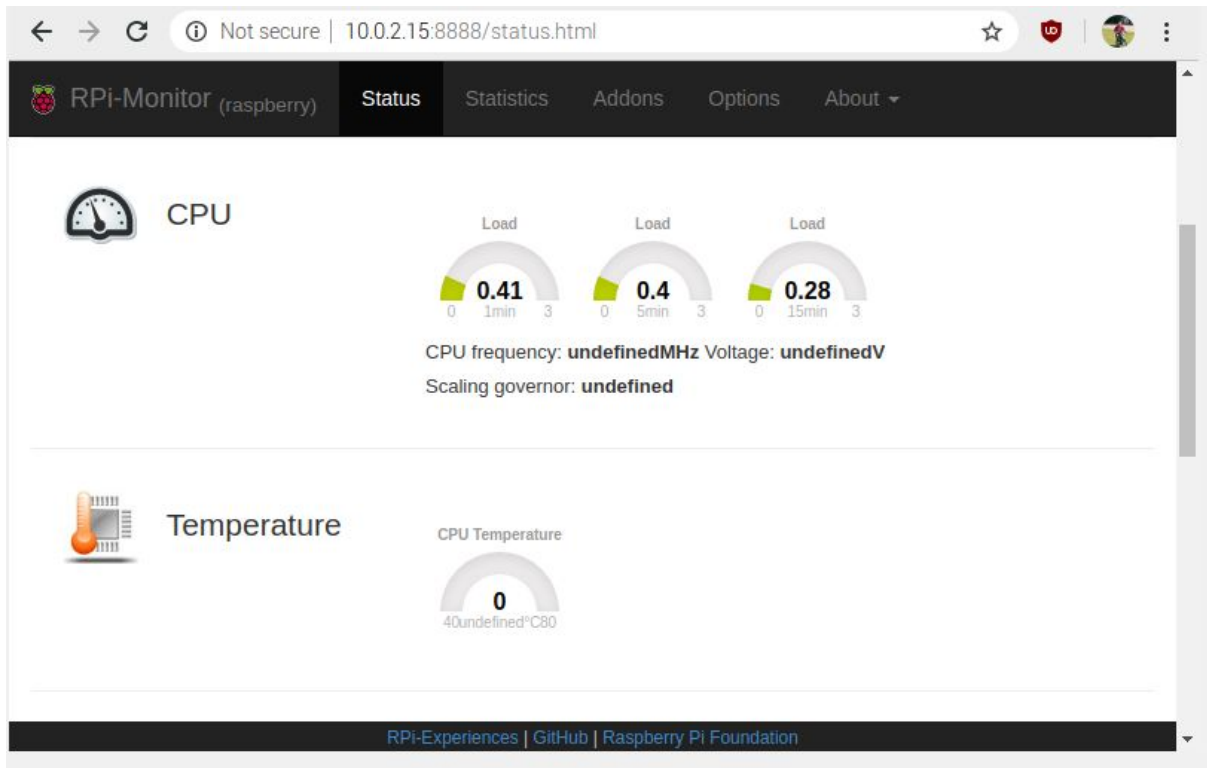
Decidimos realizar la prueba de esta herramienta, ya que es una herramienta hecha específicamente para Raspberry Pi. Para la instalación seguimos los pasos de la siguiente guía <https://alexpro.sytes.net/como-instalar-rpi-monitor/>.

Instalamos varias dependencias y la herramienta con los siguientes comandos:

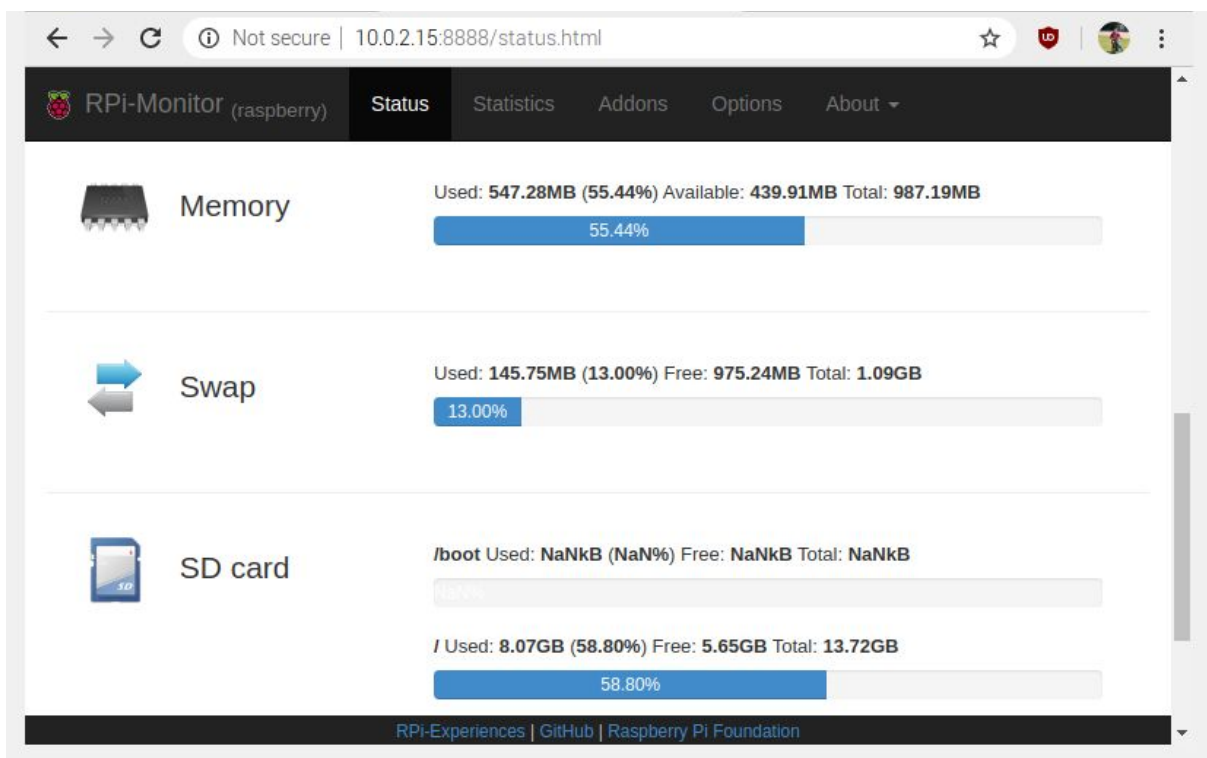
```
sudo apt-get install apt-transport-https ca-certificates
sudo wget https://goo.gl/vewCLL -O /etc/apt/sources.list.d/rpimonitor.list
sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 2C0D3C0F
sudo apt update
sudo apt install rpimonitor
```

Luego de realizar estos pasos, y ya teniendo instalada la herramienta, podemos acceder a la página web, el servicio se ofrece en el puerto 8888, entonces la dirección es <http://localhost:8888>. En las siguientes capturas se puede ver como es la interfaz web de la herramienta, y las métricas que recolecta:





Como se ve en la imagen mostrada anteriormente, la temperatura no se mostraba de manera correcta, intentamos varias cosas para solucionarlo. Es un problema de un archivo Python que recolecta las estadísticas, pero no hay mucha información en internet para arreglarlo, y al no tener conocimientos de Python, no pudimos llegar a solucionar el inconveniente.



También se puede ver en la parte superior de la interfaz web el apartado 'Statics', en donde se muestra los datos recolectados en cualquier momento de forma de gráfico. Y se ve de la siguiente manera:



Como se ve en la imagen, desde el menú desplegable de la izquierda, se puede seleccionar las diferentes opciones de métricas para mostrar. Y en el otro desplegable se puede elegir el periodo que se desea mostrar.

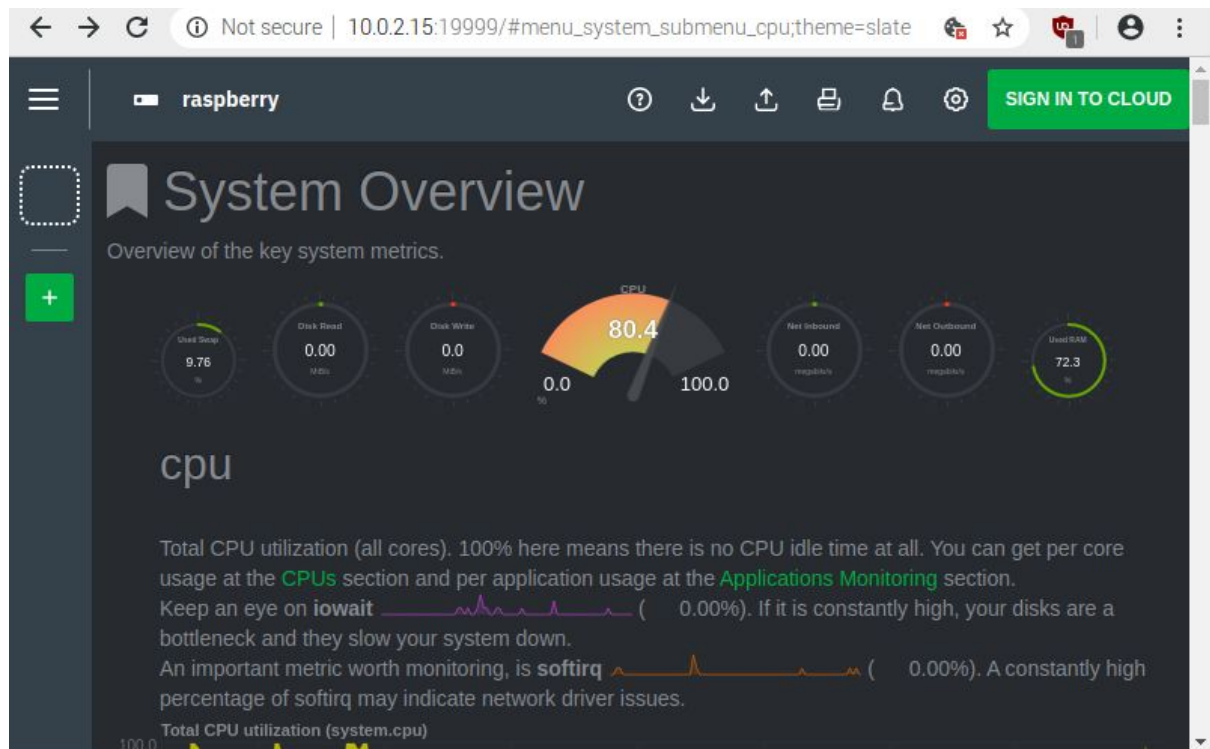
## NETDATA

Para la instalación de Netdata, nos guiamos con los pasos de la siguiente página web: <https://samourai47.com/?p=192>.

Al igual que con Rpi Monitor, es una instalación bastante sencilla, ejecutando una sola línea de código se puede instalar:

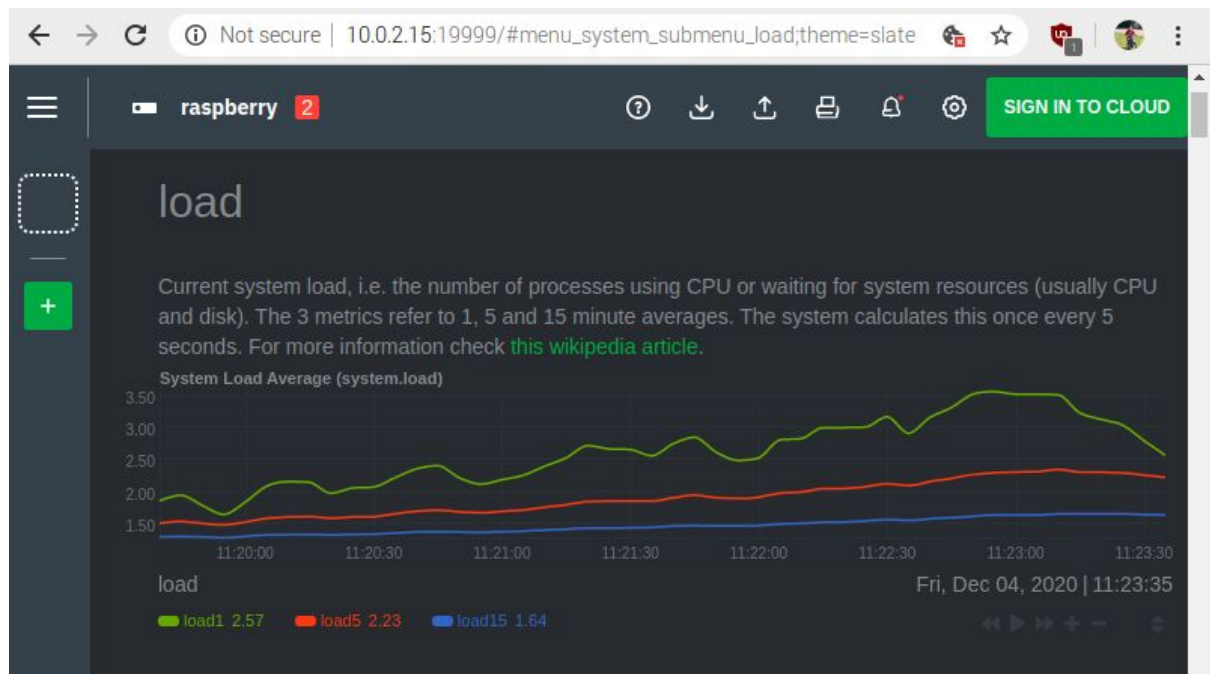
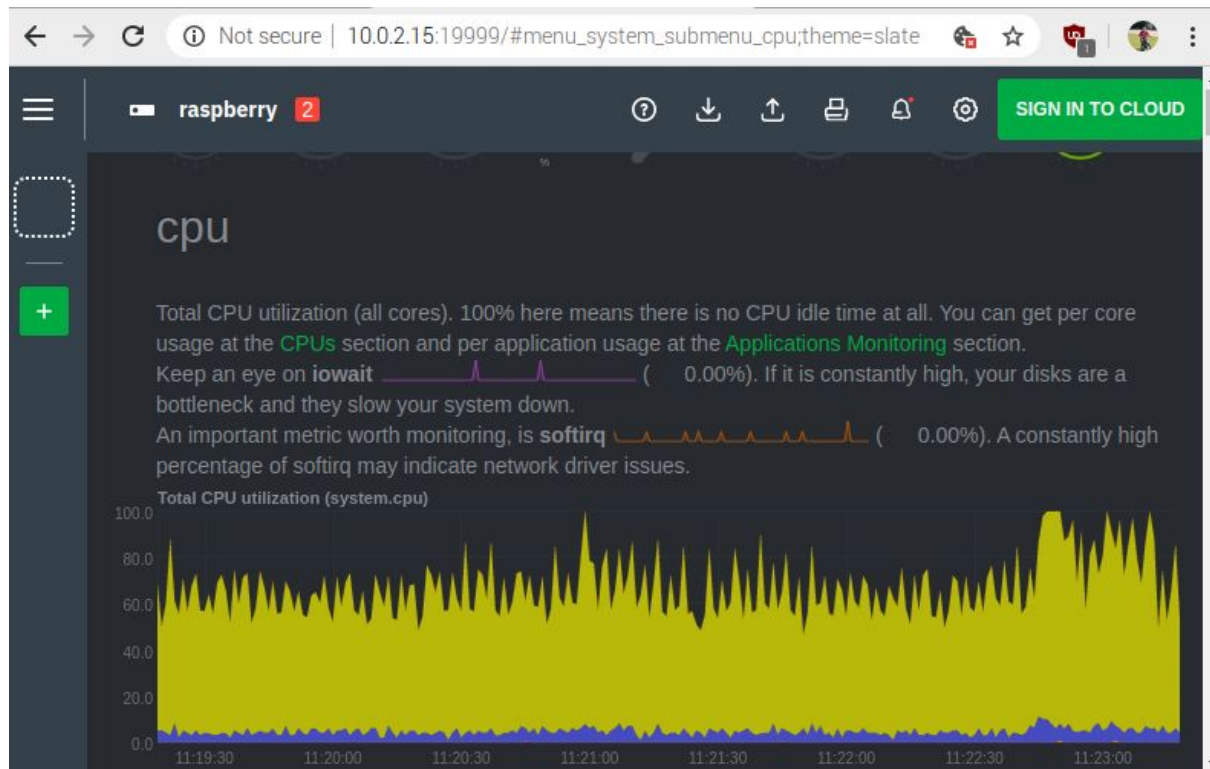
```
1. bash <( curl -Ss https : //my-netdata.io/kickstart.sh)
```

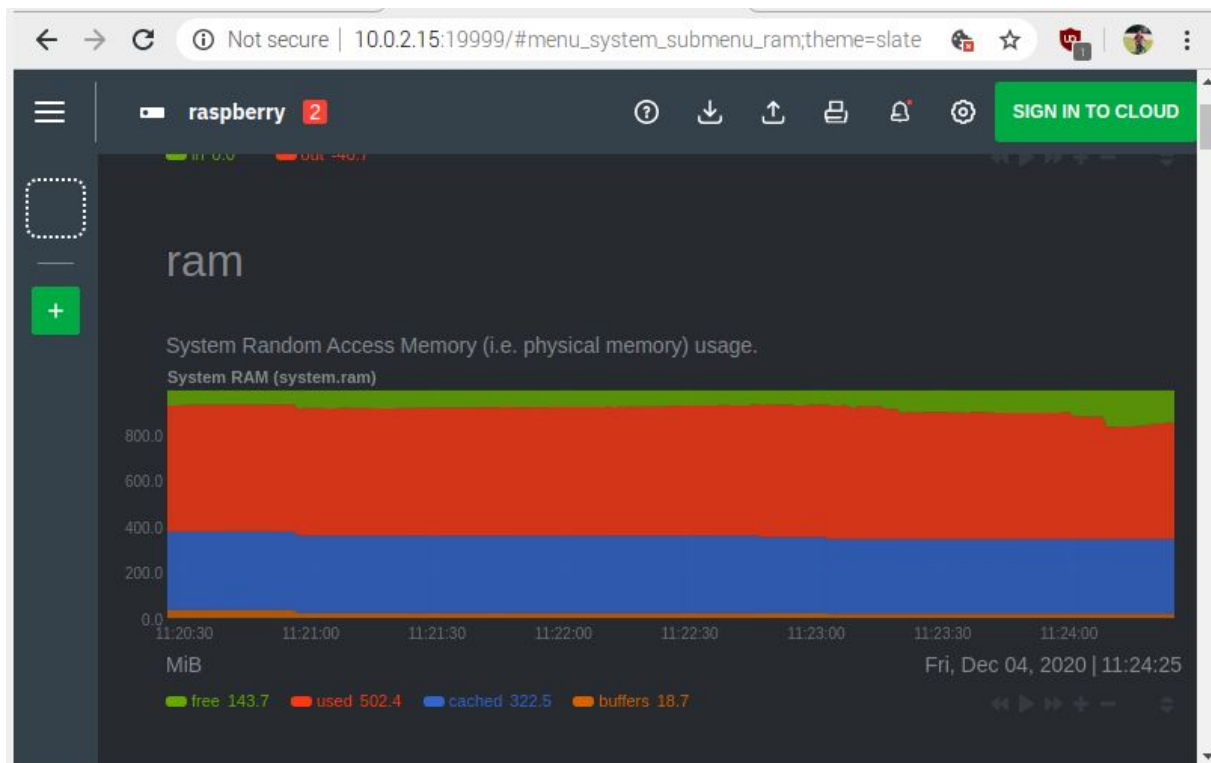
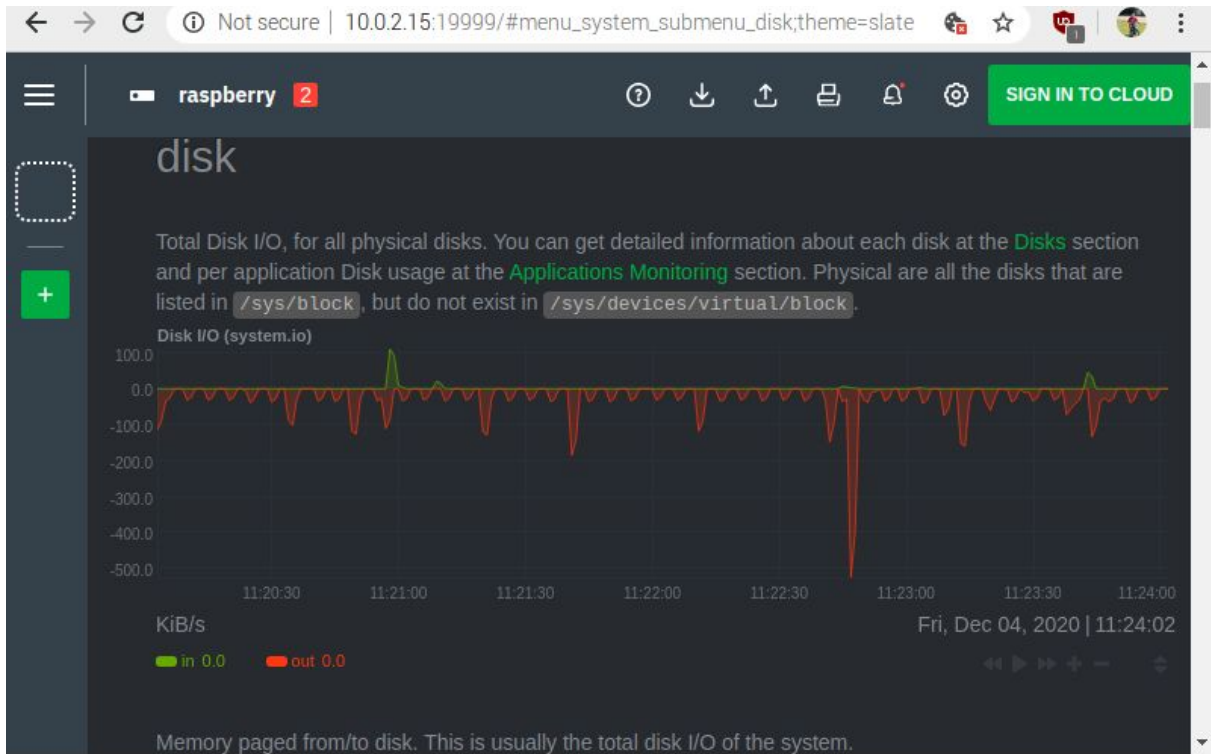
Este comando instala la herramienta y todas sus dependencias. Una vez completada la instalación solo queda abrir en el navegador la página web con el siguiente enlace <http://localhost:19999>, y nos aparece la interfaz web de la herramienta. A continuación se muestra la captura de como se ve la interfaz web:





Esta herramienta es muy completa, ya que muestra una gran cantidad de datos, a continuación se muestra las capturas de algunas de las métricas que recolecta:







Además esta herramienta muestra alertas, por ejemplo en la siguiente captura salta una alarma en la que nos informa que no hay más espacio disponible en el disco.

The screenshot shows a web browser window with the address bar displaying "10.0.2.15:19999/#menu\_system\_submenu\_cpu;theme=slate". The page has a dark theme and a top navigation bar with "Active", "All", and "Log" tabs. The main heading is "Raised Alarms". Below this, the alarm is identified as "disk - /".

On the left side, there is a summary section for "disk\_space.\_" showing "disk space usage 100%". Below this, it says "current disk space usage" and "role: sysadmin". There are also icons for a location pin, a document, and a folder.

The main content area displays the alarm configuration details:

- warning when `$this > (($status >= SWARNING) ? (80) : (90))`
- critical when `$this > (($status == SCRITICAL) ? (90) : (98))`
- calculation `$used * 100 / ($avail + $used)`
- check every 1 min
- execute `/usr/libexec/netdata/plugins.d/alarm-notify.sh`
- hysteresis on escalation 1 min, on recovery 15 mins
- multiplied by 1.5, up to 1 hour
- source 12@/usr/lib/netdata/conf.d/health.d/disks.conf

On the right side of the interface, there is a vertical bar with a green "LOUD" indicator and a yellow-green waveform graph.

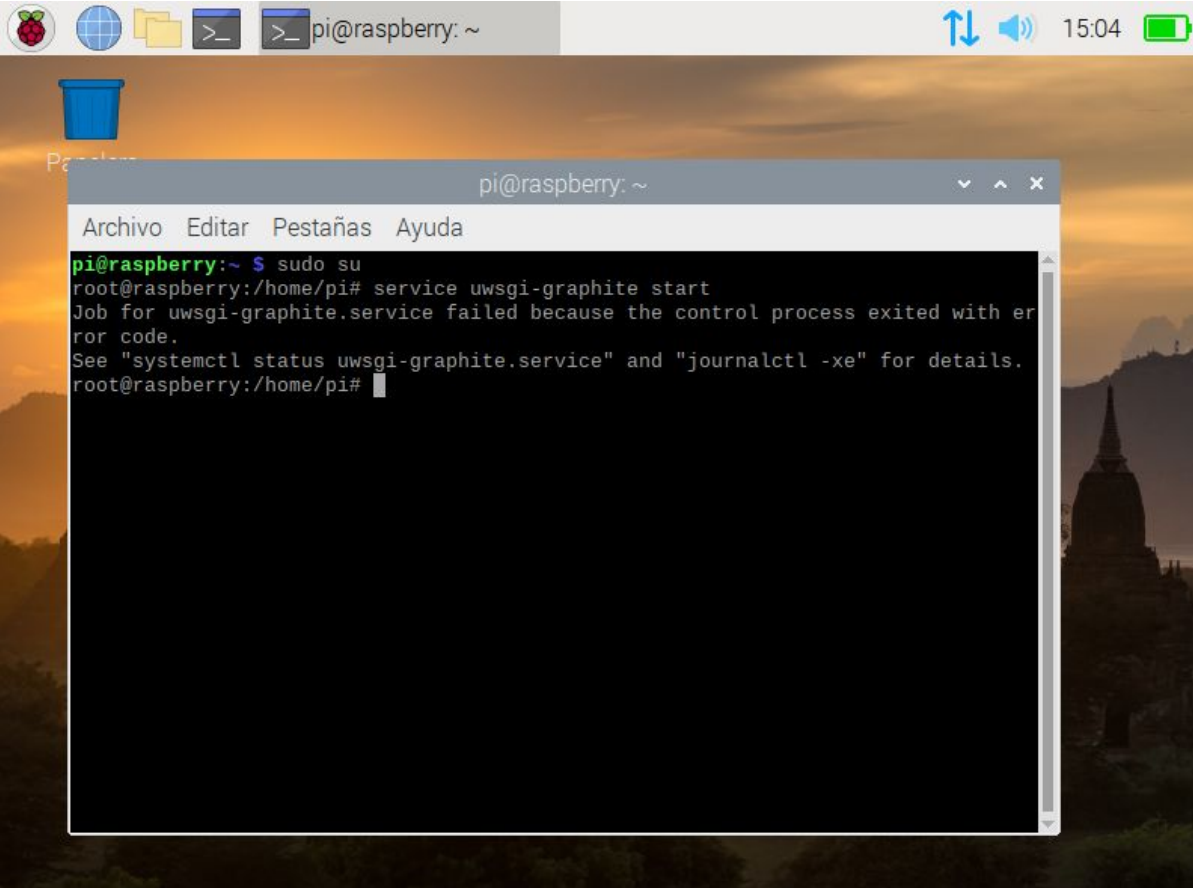
## 5.- Problemas y Soluciones

En primer lugar, debido al contexto de aislamiento social, el proyecto se encuentra limitado debido a que no se puede contar con los materiales “en mano” y se deben simular las Raspberry Pi a través de VirtualBox.

En cuanto a las herramientas utilizadas, se hizo dificultosa la configuración de **Graphite** para obtener métricas, almacenarlas y luego poder visualizarlas.

A la hora de correr el comando **sudo service uwsgi-graphite start**. No se puede iniciar el servicio ya que Raspberry Pi no lo reconoce. Se propuso hacer una investigación acerca del problema, en donde se pudo mostrar que los servicios se encuentran en la carpeta /etc/init.d en donde se lo debe configurar acá mismo para poder ser reconocido. No olvidar que antes de cada cambio realizado, se debe correr el comando (**sudo systemctl daemon-reload**) y así poder reflejar los cambios efectuados.

En la figura se puede observar el mensaje de error que arroja la consola.



The image shows a terminal window on a Raspberry Pi desktop. The terminal output is as follows:

```
pi@raspberrypi: ~  
$ sudo su  
root@raspberrypi:/home/pi# service uwsgi-graphite start  
Job for uwsgi-graphite.service failed because the control process exited with error code.  
See "systemctl status uwsgi-graphite.service" and "journalctl -xe" for details.  
root@raspberrypi:/home/pi#
```

El mismo NO se pudo solucionar y queda a disposición de un futuro proyecto que lo corrija, el grupo queda a disposición de consultas del próximo grupo que desee continuarlo.

A modo de “trabajo futuro” se propone culminar con la simulación de las herramientas que fueron investigadas y poder sacar una nueva conclusión de cuál sería la mejor herramienta para un sistema embebido.

Una vez culminada la simulación, otro de los objetivos que se plantea es poder monitorear un sistema de tiempo real que esté en funcionamiento, como podría ser la domótica de una casa. En donde se encuentran presentes sensores, dispositivos, motores, etc. Y es de gran importancia tener un control sobre ellos, favorecidos también por varias herramientas de monitoreo que ofrecen alertas configurables por el usuario.

Para esto necesitaremos además de lo mencionado en la domótica, una conexión activa de internet y una o más Raspberry PI.

## 6.- Documentación en Formato Gráfico y Video

En el siguiente enlace, se puede visualizar y descargar un video en el cual se muestra el funcionamiento de las herramientas de monitoreo Prometheus y Grafana.

[https://drive.google.com/file/d/1auGFkcf-jQAjng\\_u8LPfqpJoHWwi8LWo/view?usp=sharing](https://drive.google.com/file/d/1auGFkcf-jQAjng_u8LPfqpJoHWwi8LWo/view?usp=sharing)

## 7.- Conclusiones

Hay varias herramientas de monitoreo para utilizar en sistemas embebidos, no todas son específicas para estos sistemas, pero funcionan igualmente. Con el tiempo que tuvimos no llegamos a probar todas, pero entre las que probamos podemos decir que, Grafana y Prometheus son muy completas, tienen una instalación bastante sencilla, y se puede encontrar mucho material en internet sobre estas herramientas. Además tiene la posibilidad de agregar paneles con las métricas que se requieran. En este caso se utiliza Prometheus para recolectar los datos, y Grafana para la muestra de gráficos, ambas se complementan muy bien.

También intentamos utilizar Graphite con Grafana, a diferencia de Prometheus, este no puede configurar alertas. Para realizar esto se tiene que instalar solo la parte de recolección de datos de Graphite, que no es muy fácil y no se encuentra mucha información en internet. Con esta herramienta no logramos hacer que recopile los datos correctamente. Aunque Graphite tiene su propia interfaz web para mostrar gráficos además de hacer la recolección de métricas. Al instalar Graphite tuvimos varios inconvenientes, como se mencionó anteriormente. La conclusión que sacamos, es que la instalación de Graphite es ineficiente, en donde es bastante complejo de instalar, especialmente si se tiene un conocimiento mínimo de Python, con una serie de dependencias como Django y MySQL, y toda su configuración asociada. Además su interfaz web no tiene un diseño muy elegante.

Luego, con respecto a la herramienta RPI Monitor, es una herramienta especialmente hecha para monitorizar una Raspberry, su instalación es muy simple, solo con unas líneas de código ya se podía ver la interfaz web. Recolecta los datos principales de la Raspberry, y no muestra mucha cantidad de información.

Y la última herramienta que instalamos fue Netdata, su instalación al igual que con RPI Monitor fue muy sencilla, con tan solo 1 línea de código ya estaba funcionando. Esta herramienta muestra una gran cantidad de métricas, y además tiene la posibilidad de configurar alarmas para ver el estado de nuestra Raspberry Pi. También tiene la función de repositorios en la nube, que solo basta tener un correo electrónico, en donde llegará un correo con un link de enlace y se agrega un nodo, para poder monitorizar múltiples sistemas.