PONTIFICIA UNIVERSIDAD CÁTOLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

MAESTRÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

MENCIÓN REDES DE COMUNICACIONES

TRABAJO DE TITULACIÓN

Título De La Disertación

Diseño de un Sistema de Monitoreo de Voltaje y Parámetros Ambientales en Racks Exteriores de Telecomunicaciones con el Uso de Software Especializado y Arduino

Autor: Adrian Rubio Mena

Director: M. Sc. Juan Francisco Chafla A.

Quito, 2022

DEDICATORIA

Dedico el presente proyecto a mis padres que siempre han estado apoyándome en todas las etapas de mi vida. Por haberme forjado la persona que soy y seguirme motivando para superarme cada vez.

A mi hermano por ser mi compañero, mi mejor amigo y apoyarnos hombro con hombro en cada aspecto de la vida.

Cada logro y cada paso va dedicado a mi familia ya que sin su apoyo y motivación no sería la persona que soy ahora.

AGRADECIMIENTO

Agradezco a mi padre por todas las enseñanzas y consejos que me dio, por toda esa motivación para cada día ser mejor y nunca dejar de prepárame académicamente como personalmente. Estoy seguro de que estará orgulloso en este momento.

A mi madre por toda la paciencia y nobleza que ha tenido conmigo, por enseñarme los valores éticos y morales con los que me he formado, y ser mi apoyo tanto emocional como espiritual en cada paso de mi vida.

A mi novia Cristina por apoyarme incondicionalmente estos años, y día a día crecer juntos en todos los sentidos con muchas metas en común, como lo es seguirnos formando académicamente para poder contribuir positivamente a la sociedad.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Debido al crecimiento de las redes de telecomunicaciones en cuanto al ancho de banda, cobertura y servicios; en muchas ocasiones es necesario la instalación de racks externos, los mismos que se encuentran en la intemperie y su ubicación puede ser de difícil acceso, es por ello por lo que es de importancia mantener un correcto monitoreo de factores críticos para el funcionamiento de los equipos de telecomunicaciones.

En la actualidad es posible adquirir tarjetas de monitoreo para los parámetros mencionados, sin embargo, las soluciones actuales implican disponer de dispositivos compatibles o costos elevados para su implementación y en muchos casos los dispositivos solo toman medidas de un solo parámetro.

En el presente trabajo de titulación se diseñará una solución que permite el monitoreo proactivo de diversas variables dentro de los racks externos como son: voltaje, temperatura, humedad, acceso; a un costo reducido utilizando el protocolo SNMP (Simple Network Management Protocol) en Arduino. De tal manera que se pueda prever inconvenientes a futuro que puedan causar afectación de equipos y servicios, con la ayuda de un monitoreo centralizado.

ABSTRACT

Due to the increasing of telecommunications networks in terms of bandwidth, coverage, and services; in most cases it is necessary to install external racks. For that reason, is important to keep a correct monitoring of critical factors for the operation of the equipment. Nowadays it is possible to acquire monitoring cards, however, current solutions imply having compatible devices or high cost for their implementation and in many cases the devices only take measurements of a single parameter. In this document a solution was designed to allow the proactive monitoring of some variables in external racks such as: voltage, temperature, humidity and intrusion, with reduced cost using SNMP protocol and Arduino.

PROBLEMÁTICA

Muchas empresas de telecomunicaciones disponen de racks externos que son económicos y en muchos casos antiguos, los cuales no tienen embebidos sensores de monitoreo para diversos factores a los cuales están expuestos, como variaciones de voltaje no deseadas, incrementos de temperatura y/o humedad, acceso de personal no deseado a los racks. Esto ocasiona que no se pueda prever problemas que causen afectación a los equipos de telecomunicaciones y por consiguiente a los servicios de conectividad de los usuarios finales.

ANTECENDENTES

Actualmente existe gran variedad de dispositivos en el mercado cuyo enfoque es el monitoreo de los parámetros eléctricos y ambientales para racks y data centers en general. Se han tomado algunas marcas y modelos de sensores como referencia de costo a nivel comercial, como se puede observar en la siguiente tabla:

Tabla1. Cuadro comparativo de costo entre diferentes marcas y modelos

Marca	Modelo	Costo (USD)
HW Group	Poseidon 2 3266	450
APC	AP9520TH	254
Didactum	Monitoring System 100 IT Basic Protection	395
CDP	SNMP-MCY-EN	199
Emerson	Liebert IntelliSlot Unity	299

Fuente: (Group, 2020), (Dictum, 2020), (Industry Research, 2020), (Amazon, 2020)

En base a la información recopilada en la Tabla1, el presente Trabajo de Titulación se orientará a la elaboración de un dispositivo a un costo menor con los respectivos sistemas de gestión y monitoreo.

Existen varios trabajos de investigación relacionados con el monitoreo de parámetros externos y eléctricos, por lo que se mencionarán aquellos que tienen mayor afinidad al presente Trabajo de Titulación. (Veslateguí, 2019) orientó su trabajo al monitoreo de voltaje, corriente, temperatura en un Centro de Datos y el número de veces que se abre la puerta de un PDU (Unidades de Protocolo de Datos) con el uso de Arduino, utilizando conectividad IP. Siendo la mayor diferencia con el trabajo presente el uso de SNMP.

(Pastori & López, 2016) en su proyecto denominado Argos, realizaron un sistema de monitoreo para Centros de Datos utilizando Arduino con la ayuda de SNMP, cuyo principal objetivo es garantizar las condiciones adecuadas para el

funcionamiento de los equipos de red. La principal diferencia con el trabajo propuesto radica en el análisis de los valores límite tanto eléctricos como ambientales, de tal manera que se pueda predecir con suficiente antelación una falla que afecte a los equipos Mikrotik.

(Santana, 2017) ha realizado un proyecto de implementación en el cual monitorea los valores de temperatura y humedad en un Centro de Datos con la ayuda de Raspberry Pi y SNMP a un costo reducido. Una de las diferencias en este proyecto (a parte de la marca de hardware), es el monitoreo de voltaje y accesos dentro de un rack de telecomunicaciones.

Objetivos

Objetivo General

- Diseñar un sistema de monitoreo de voltaje y parámetros ambientales en racks exteriores de telecomunicaciones a bajo costo con el uso de software especializado y Arduino.

Objetivos Específicos

- Diseñar un dispositivo en Arduino para medir valores de voltaje, temperatura y humedad, así como la verificación del estado de la puerta de acceso.
- Configuración de la comunicación IP y SNMP en Arduino utilizando una conexión Ethernet.
- Configuración del software PRTG para interpretar la información que recibe del dispositivo Arduino.
- Análisis de las condiciones críticas en la cuales los equipos Mikrotik empiezan a tener problemas.
- Implementación en una empresa de telecomunicaciones local mediante la conexión física del dispositivo Arduino en un rack externo, así como en la red, y verificar el correcto funcionamiento de la solución implementada.

CAPÍTULO I

FUNDAMENTOS TEÓRICOS

1.1 Protocolo SNMP (Simple Network Management Protocol)

Es un protocolo de la capa de aplicación mismo que utiliza UDP (User Datagram Protocol) para el intercambio de información. Implícitamente en el modelo SNMP se conforma por: 1) Estaciones de administración, las cuales ejecutan aplicaciones donde se monitorea y controla los elementos de red. 2) Elementos de red, que son los dispositivos como router, firewalls, servidores, etc. (Case, Fedor, Schoffstall, & Davin, 1990).

La estructura de SNMP consta de los siguientes elementos:

a) Agente de Gestión. – Se encarga de la supervisión de los elementos de red, este se comunica con el gestor para infórmale sobre eventos en el objeto gestionado. Este agente es parte del dispositivo.

b) Gestor. – Es el software que reside en una estación de gestión, se comunica con los agentes y posee una interfaz para el usuario a través de la cual puede comunicarse con todos los elementos de gestión.

c) Objeto Gestionado. – Son los elementos físicos de la red que se gestionan, como router, firewall, servidores, etc. La base de datos de gestión (MIB) está formada por todos los objetos gestionados (Rossen & Popnikolov, 2005).

La comunicación cliente-servidor realiza de la siguiente manera: El software de administración envía (como cliente) un paquete UDP al servidor, llamado Agente SNMP, el cual normalmente es un software alojado en un dispositivo. Este responde un paquete UDP con una respuesta SNMP. Cada ciclo de pregunta/respuesta otorga al cliente una medida del dispositivo como por ejemplo tráfico, CPU, temperatura, etc. (Rupp & Zobel, 2017).

Los tipos de mensajes en SNMP son:

a) Get: Es enviado por la estación gestora para obtener las variables de la MIB.

b) Get-Next: Es enviado por la estación gestora para obtener las variables de la MIB siguiente a la gestionada.

c) Get-Response: es enviado por el nodo gestionado a la estación gestora como respuesta a get, get-next o set.

d) Set: Es enviado por la estación gestora para dar un valor determinado a una variable en la MIB de un nodo gestionado.

e) Trap: Son paquetes de información que son enviados por el nodo gestionado hacia la estación gestora tan pronto como algo específico ocurra en este, sin necesidad de haber existido una petición explícita. De tal manera que la estación gestora pueda reaccionar inmediatamente ante incidentes enviando mensajes de alerta (Rossen & Popnikolov, 2005).

Para la correcta comunicación entre estación gestora y nodo gestionado se debe configurar Comunidad en ambos, en SNMP esto funciona al igual que una contraseña, si la comunidad es igual en ambos dispositivos la comunicación puede establecerse (Rupp & Zobel, 2017).

Los objetos monitoreados se identifican con un OID (Object Idenfier), estos identificadores se agrupan en estructuras jerárquicas en los cuales los elementos se enumeran acorde a la siguiente figura:

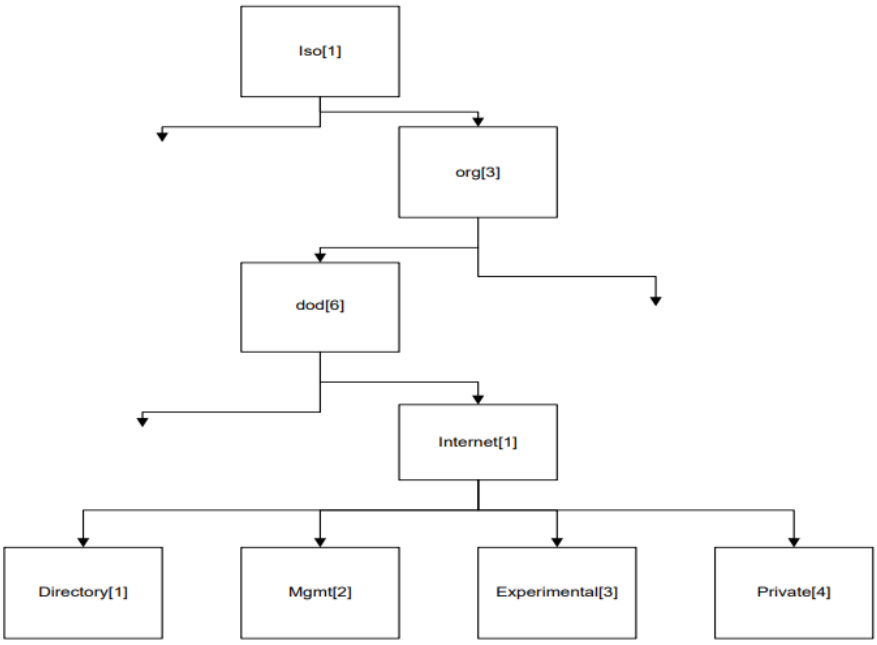

Figura 1. Arreglo OID en un esquema tipo árbol

Fuente: (Martínez y Medina, 2014)

Del esquema mostrado en la Figura 1, el objeto “Mgmt” se lo puede expresar como 1.3.6.1.2. Para poder monitorear los dispositivos, el administrador debe primero compilar los archivos de bases de datos (MIB) (Martínez, Medina, Guido, & Estaban, 2014).

1.2 Hardware libre Arduino

El Hardware libre está basado en los mismos principios en los cuales está basado el Software Libre, pero aplicando sus conceptos y bases a su campo. Para que un Hardware sea considerado Hardware Libre sus especificaciones y esquemas deben ser de acceso público, indistintamente del costo de estos. La placa Arduino es hardware libre ya sus ficheros esquemáticos se encuentran en la página web de Arduino, el mismo que permite la creación de diseños siempre y cuando sean publicados bajo la misma licencia (Martínez, Medina, Guido, & Estaban, 2014).

Arduino está construido con el microcontrolador ATmega328 cuya alimentación puede ser por medio de la interfaz USB o por fuente externa, el rango de voltaje es de 7 a 12V. Posee 14 pines de entradas y salidas de los cuales 6 tienen salida PWM (Pulse-Width Modulation), 6 pines de entradas analógicas, 32KB de memoria Flash, trabaja con una frecuencia de reloj de 16Mhz (Veslateguí, 2019).

Para interactuar con el medio es necesario conectar dispositivos llamados sensores a la placa de Arduino, los cuales son capaces de medir magnitudes físicas o químicas denominadas variables de instrumentación y transformarlas en magnitudes eléctricas, como por ejemplo voltaje, temperatura, presión, etc. Los sensores pueden ser analógicos o digitales, siendo los analógicos los más precisos ya que tienen que leerse/escribir un voltaje de 0 a 5 voltios que representan 10bits (lectura) o en 8 bits (escritura), es decir la lectura puede tener 1024 valores distintos mientras que la escritura puede tener 256 valores (Moreno & Córcoles, 2018).

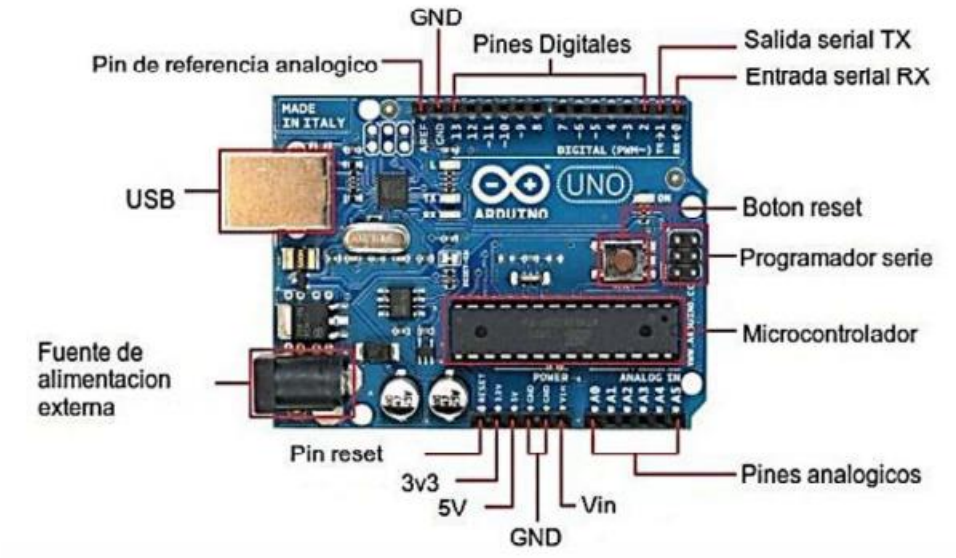

Figura 2. Placa electrónica Arduino

Fuente: (Veslateguí, 2019)

El Pin de voltaje de entrada VIN, es el voltaje de alimentación del Arduino. El PIN con etiqueta 5V tiene n voltaje regulado de 5V mismo que puede proceder de un USB, o una fuente externa.

El PIN con etiqueta 3.3V es una fuente que ha sido generada y regulada internamente, el consumo de corriente se encuentra alrededor de los 50mA.

El PIN GND es la puesta a tierra.

Arduino cuenta con 14 PIN digitales mismos que pueden ser entradas o salidas, en caso de implementarse como entradas soportan una corriente máxima de 40mA. Los PIN 3,5,6,9,10 y 11 son moduladores por ancho de pulso con una salida de 8bits. Los PIN 10,11,12,13 tienen funciones para realizar comunicación SPI.

Las entradas analógicas están etiquetadas desde A0 hasta A5 dando 6 entradas en total, con una resolución de 10 bits, estos PIN soportan un voltaje de entrada de 0V a 5V, por lo que es necesario acondicionar las señales analógicas para poder ser procesadas por Arduino.

1.3 Rack de Telecomunicaciones

El rack de telecomunicaciones también llamado armario, es un soporte metálico cuyo fin es alojar equipamiento electrónico, informático y de comunicaciones. La medida estándar es de 19 pulgadas de ancho para alojar equipos de cualquier fabricante, el alto se mide en unidades de rack siendo 1U= 1.7” = 44.5mm. Estos también contienen accesorios como ventiladores, regletas eléctricas, banco de baterías, ups, etc. (Nelson, 2014).

1.4 Conectividad IP

Con el fin de transmitir y recibir tramas SNMP en la red es necesaria la implementación de un protocolo de comunicación, para esto se utiliza el protocolo ethernet, utilizando la placa Ethernet Shield. Esta es una placa basado en ATmega328m la cual permite la comunicación IP sobre TCP y UDP soportando hasta 8 sockets de conexiones simultáneos. El estándar utilizado por EthertShield es RJ45. La forma de alimentación puede ser con una fuente externa o mediante PoE (Power over ethernet), siendo el voltaje admitido de 6V a 20V, con voltajes inferiores a la respuesta de la placa se puede volver inestable, mientras que con voltajes superiores la placa puede verse dañada.

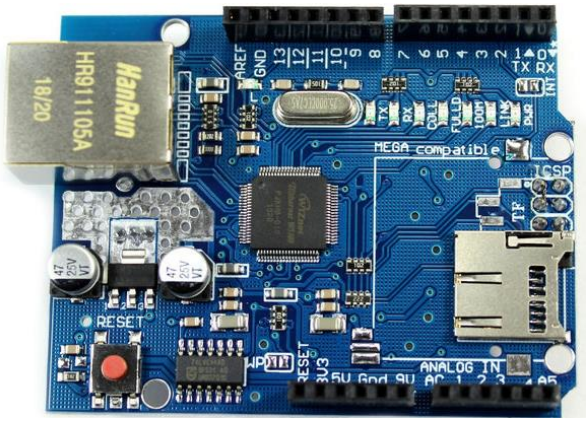

Figura 3. Módulo Ethernet

Fuente: (Electronics, 2022)

La forma de alimentación puede ser con una fuente externa o mediante PoE (Power over ethernet), siendo el voltaje admitido de 6V a 20V, con voltajes

inferiores a la respuesta de la placa se puede volver inestable, mientras que con voltajes superiores la placa puede verse dañada.

Para realizar la conectividad IP hacia la red, se utiliza el protocolo DCHP mismo que otorga una IP dentro del segmento 100.64.72.8/30, de esta manera se garantiza que adquiera una sola IP sin necesidad de configuración manual en la placa.

1.5 Rack de Telecomunicaciones exterior

El rack de telecomunicaciones, también llamado gabinete, en el cual se va a implementar el dispositivo del presente se muestra en el siguiente esquema:

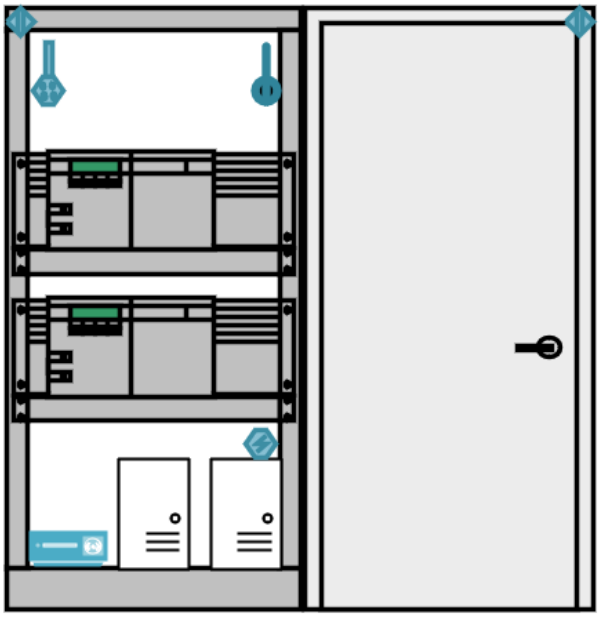

Figura 4. Esquema de Rack Exterior

Fuente: (Mena, 2022)

Este rack de telecomunicaciones posee las siguientes características:

- Dimensiones: 2 metros de ancho x 2 metros de alto x 2.5 metros de profundidad.
- Distribución de equipos: En la base del gabinete se encuentra el banco de baterías mismos que son conectados a un UPS, también se encuentra la acometida de la energía eléctrica comercial. En las bandejas de la mitad y

superior se encuentran localizados los equipos de red Mikrotik junto con las conexiones a los radios enlaces.

La ubicación de los sensores y el dispositivo a implementar se muestra en la siguiente figura 4 en color azul.

De esta manera el dispositivo se encuentra ubicado en la base del rack junto a las baterías, los sensores de humedad y temperatura se encuentran ubicados en la parte superior del rack. Los sensores de intrusión se encuentran ubicados en la puerta del rack.

Este dispositivo además de tomar las señales de cada uno de los sensores también está conectado hacia el ventilador, de tal manera que pueda controlarse de forma remota acorde a las condiciones de temperatura.

CAPÍTULO II
DISEÑO DEL SISTEMA DE MONITOREO

2.1 Estructura física del dispositivo

La estructura física del dispositivo ha sido diseñada en el software de Solid Works, para posteriormente ser manufacturada en una impresora 3D. El dispositivo consta de las siguientes características:

a) Dimensiones: Las dimensiones son de 20cm de largo x 10 cm de ancho x 10 cm de alto.

b) Armazón: la estructura física donde están alojados todos los componentes electrónicos ha sido diseñado en una impresora 3D, por lo que este material es dieléctrico mismo que otorga mayor seguridad y aislamiento. Adicionalmente el equipo consta con puesta a tierra lo que da seguridad a la integridad de los componentes protegiéndolos de cualquier sobretensión. En la siguiente figura se muestra el case a utilizar:

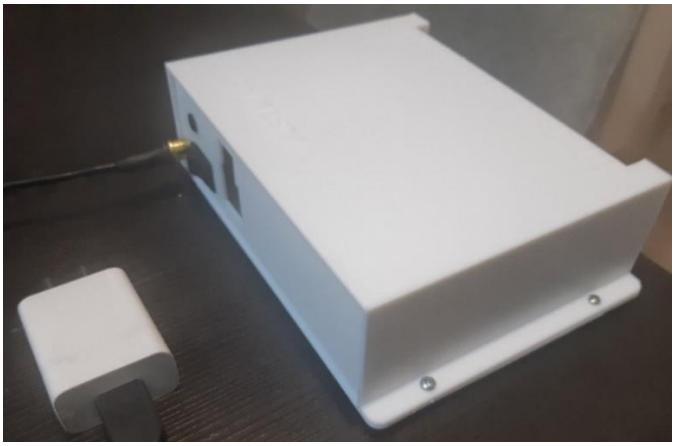

Figura 5. Chasis del dispositivo

Fuente: (Mena, 2022)

Como se puede observar se ha diseñado el case de tal manera que las borneras sean accesibles al usuario sin mayor esfuerzo de esta forma la inclusión de nuevos sensores o dispositivos actuadores se lo realice de una manera ágil y si intrusión a la placa. El case cuenta con etiquetado para cada bornera acorde a los sensores utilizados, esto ha sido implementado mediante la impresora 3D con

el objetivo de obtener un etiquetado duradero a fin de tener bien identificadas las conexiones de los sensores y actuadores respectivamente.

2.2 Desarrollo del diseño

En la siguiente imagen se muestra el diseño del dispositivo representado en diagrama de bloques del diseño a nivel eléctrico:

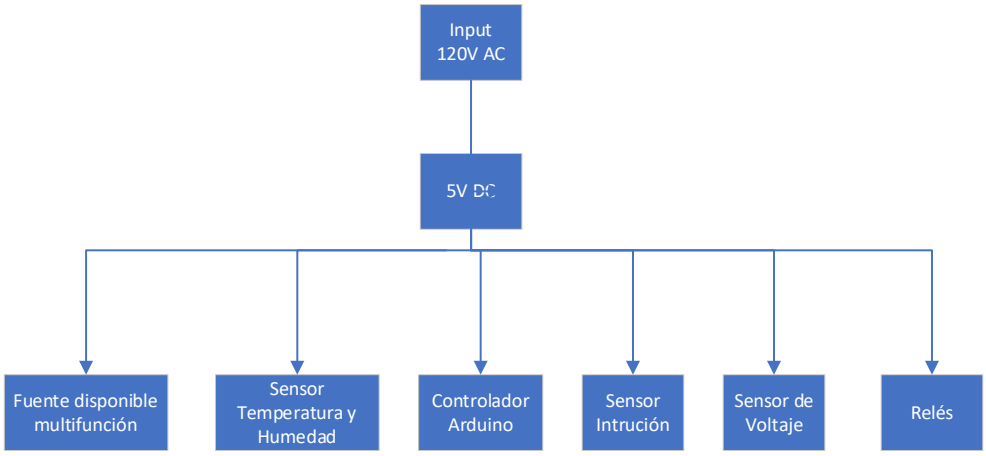

Figura 6. Diagrama de bloques

Fuente: (Mena, 2022)

Como se puede observar se tiene una alimentación de 120V alternos máximo, este voltaje es regulado a 5V DC mismos que alimentarán el controlador Arduino, los sensores de intrusión, voltaje, temperatura, humedad, así como los relés.

2.3 Diseño de adquisición de señales

Las señales que serán adquiridas son: voltaje, temperatura, humedad, intrusiones mismas que serán adquiridas de la siguiente manera:

- Voltaje. - Esta señal será obtenida mediante el sensor zmpt101b cuyo diagrama de funcionamiento está representado de la siguiente manera:

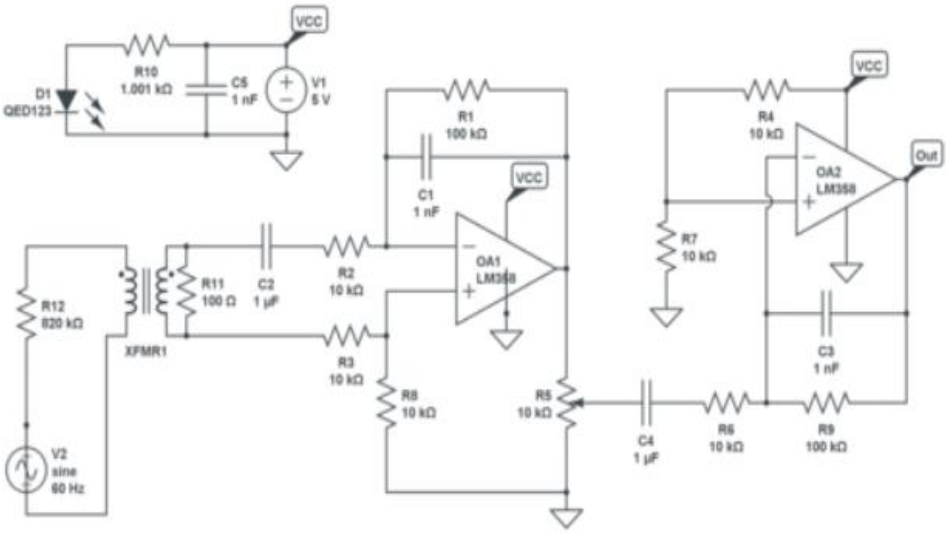

Figura 7. Diagrama del sensor zmpt101b

Fuente: (Lab, s.f.)

Este módulo permite medir el voltaje alterno reduciendo el voltaje de entrada (para este caso de alrededor 110V) a 5V que son los que puede medir el dispositivo Arduino. Este módulo está provisto de un aislamiento galvánico para una mayor seguridad lo cual es imperativo para la aplicación del presente proyecto. La señal de salida entregada por este dispositivo es del tipo sinusoidal desplazada positivamente para que no entregue valores negativos, este desplazamiento está definido por el voltaje de alimentación, siendo para nuestro caso de 5V, por lo que el desplazamiento de la señal será de 2.5V (definido por fabricante). (Electrónica, 2017)

Figura 8. Fotografía del módulo de medición de voltaje

Fuente: (Mechatronics, 2021)

La gráfica de voltaje que obtenida con este sensor es de la siguiente:

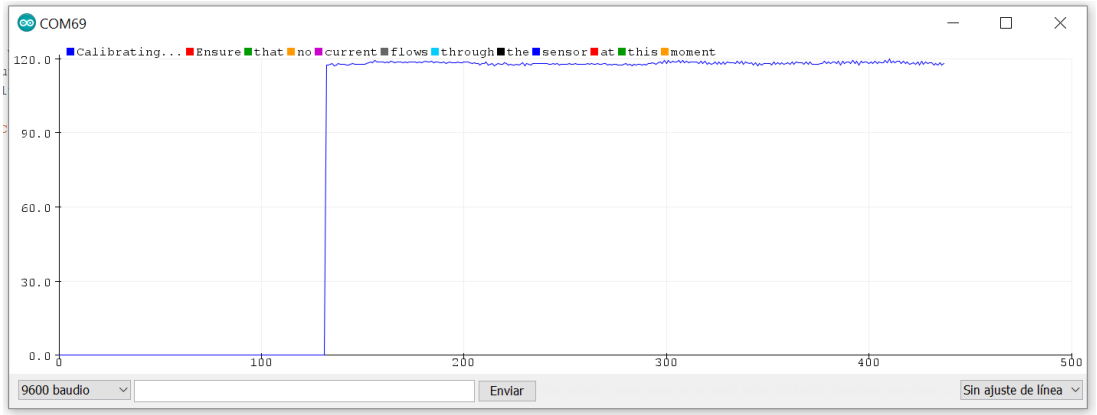

Figura 9. Medición obtenida con el sensor de voltaje

Fuente: (Mena, 2022)

- Temperatura y humedad: Estas señales serán adquiridas utilizando el sensor

DHT11. Este dispositivo utiliza un sensor digital capacitivo como componente de medición de temperatura y humedad. Está equipado con un circuito de procesamiento de señal estable y confiable para convertir los valores medidos en una señal estándar. El sensor de comunicación digital integrado garantiza la estabilidad a largo plazo del transmisor, un bajo retardo, amplio rango y precisión de medición, y una fuerte resistencia a la contaminación química. (Aosong, 2021)

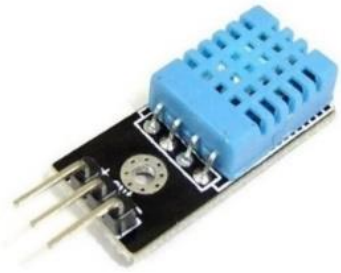

Figura 10. Fotografía del sensor DTH11

Fuente: (Libre, 2020; MGSystem, 2021)

- Intrusión: Los sensores utilizados son del tipo magnético, siendo los mismos ubicados en la puerta del rack, de tal manera que entregan un valor de 1 lógico al estar la puerta cerrado y 0 lógico al estar abierta la puerta.

Figura 11. Fotografía del sensor de intrusión

Fuente: (Marsagi, 2018)

2.4 Diseño de conexiones

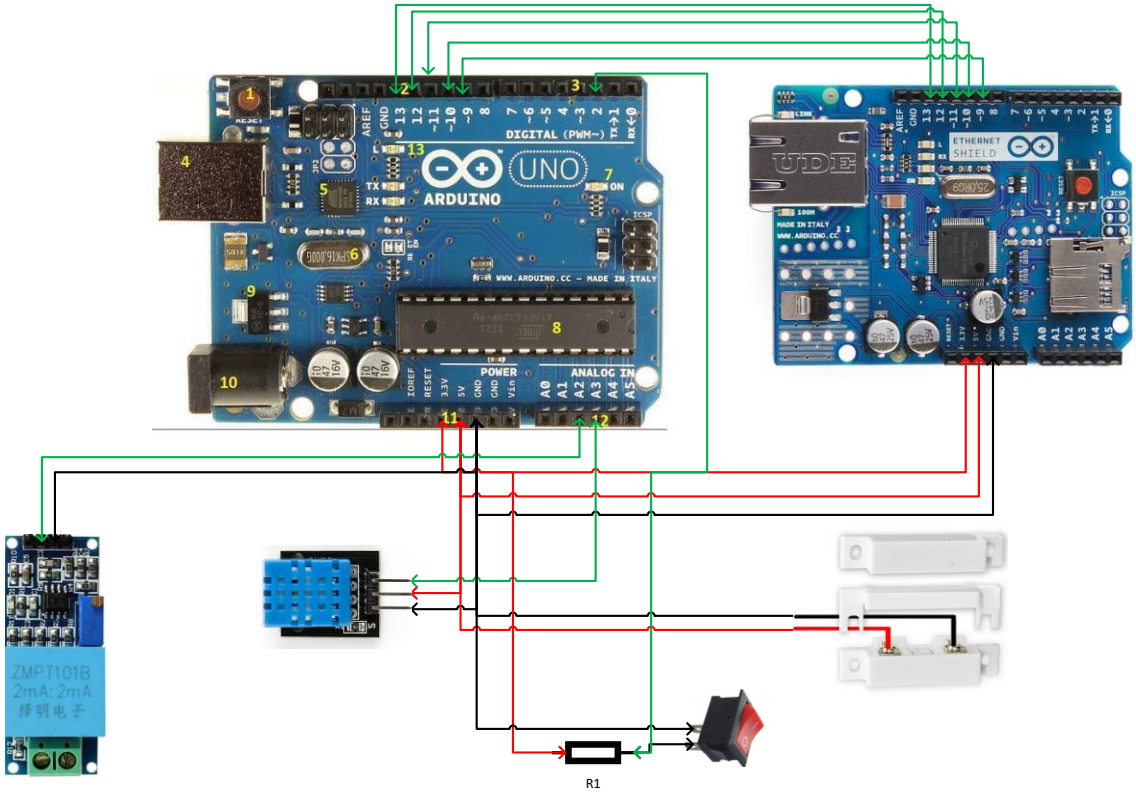

Figura 12. Diagrama de conexiones

Fuente: (Mena, 2022)

2.5 Conectividad IP

La conectividad entre la red y el dispositivo Arduino se ha establecido de la siguiente manera:

- Un servidor DCHP mismo que se encargará de otorgar el direccionamiento al dispositivo Arduino.
- La comunicación hacia el resto de la red se lo realizará mediante el anuncio del prefijo /30 vía BGP.

El esquema de conectividad es el siguiente:

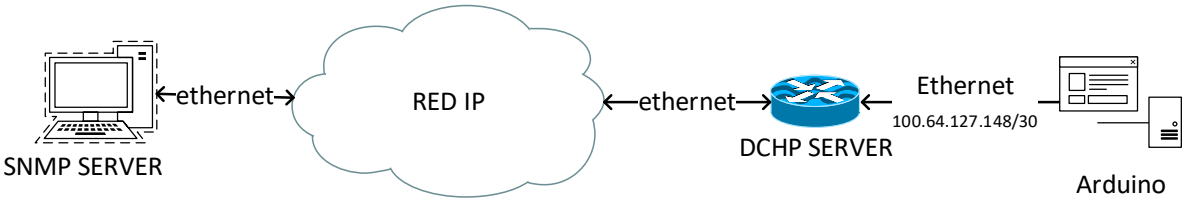

Figura 13. Esquema de conectividad IP

Fuente: (Mena, 2022)

Para la asignación de IP mediante DHCP es necesario incluir las librerías: ethernet.h y SPI.h, la programación de Arduino se muestra en el Anexo1.

CAPÍTULO III

IMPLEMENTACIÓN DEL SISTEMA DE MONITOREO

3.1 Obtención de valores críticos para monitoreo

Para la obtención de los valores críticos se han tomado en consideración varios escenarios y equipos a fin de obtener las curvas características del comportamiento de la respuesta de los equipos, esta información ha sido recopilada mediante sistemas de monitoreo implementados en data centers y en un laboratorio de pruebas siendo los resultados mostrados a continuación:

3.2 Pruebas en laboratorio

Para llevar a cabo estas pruebas se ha equipado un laboratorio con un router CCR1009-7G-1C-1S+, un generador de señales eléctricas, un calefactor y el dispositivo de monitoreo Arduino como se observa en la siguiente imagen:

Figura 14. Laboratorio implementado para pruebas

Fuente: (Mena, 2022)

Para el funcionamiento de este laboratorio se ha conectado el dispositivo de monitoreo Arduino en el equipo Mikrotik simulando la conexión en un entorno real.

3.3 Pruebas de voltaje

Para realizar estas pruebas se ha variado el voltaje con una curva que compete a la siguiente función:

$$V(t) = -4t + 120$$

Ec.1 Curva de variación de voltaje

Siendo:

t= tiempo

V= voltaje

Con esta señal de alimentación el dispositivo de monitoreo obtuvo el siguiente resultado:

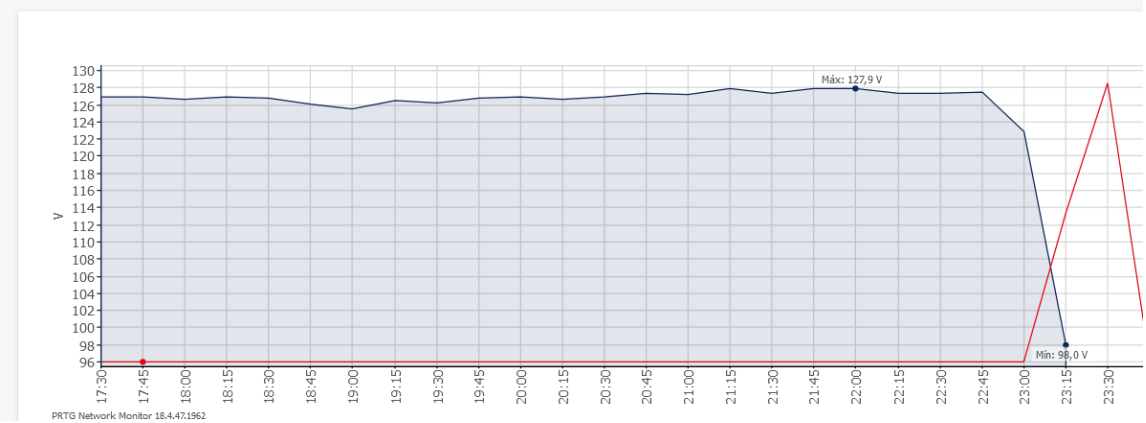

Figura 15. Resultado de pruebas de voltaje

Fuente: (Mena, 2022)

Como se puede observar en la Fig. 15 al llegar el voltaje a valores inferiores a 100V el equipo dejó de responder. Por tanto, el sistema de monitoreo debe enviar alertas de voltaje al llegar a un voltaje de 110 Voltios con este valor encontrado se logra lo siguiente:

- Se evita el envío de falsas alertas de problemas de energía por falla eléctrica comercial
- Con estos resultados al desencadenarse la alerta se obtiene tiempo suficiente para tomar acciones correctivas, dado que el presente proyecto está enfocado en nodos remotos este tiempo es muy importante ya que la movilización de personal técnico no es inmediata

3.4 Pruebas temperatura:

Para realizar pruebas de temperatura se optó por incrementar la temperatura de forma artificial mediante calefactores en un entorno cerrado donde se obtuvo el siguiente resultado:

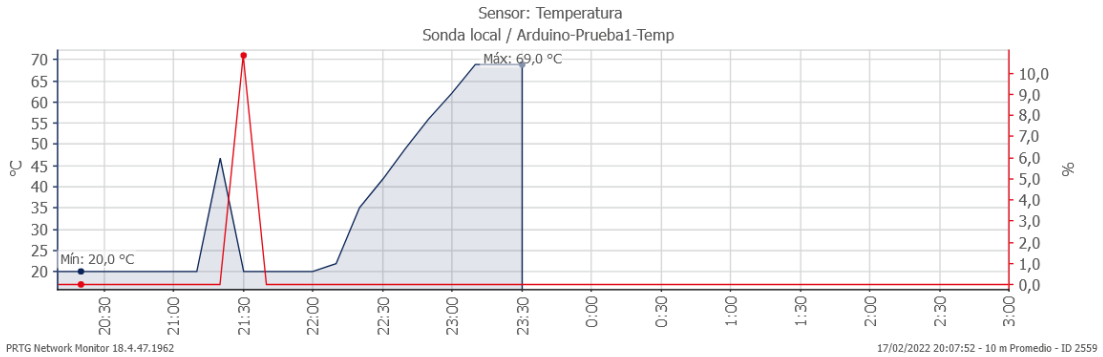

Figura 16. Resultado de pruebas de temperatura

Fuente: (Mena, 2022)

Como se puede observar en la Fig.16 inicialmente se realizaron pruebas hasta llegar a un valor de 45 grados Celsius sin presentar problemas. Sin embargo, al llegar a los 69 grados Celsius medidos, el equipo Mikrotik dejó de responder. En ese sentido este es el valor crítico al cual el equipo deja de funcionar y se apaga en su totalidad, por motivos de seguridad el equipo no se lo volvió a prender hasta el día siguiente.

3.5 Pruebas humedad:

Las pruebas de humedad han sido realizadas con equipamiento ubicado en un rack de telecomunicaciones exterior durante una semana, donde se obtuvo el siguiente resultado:

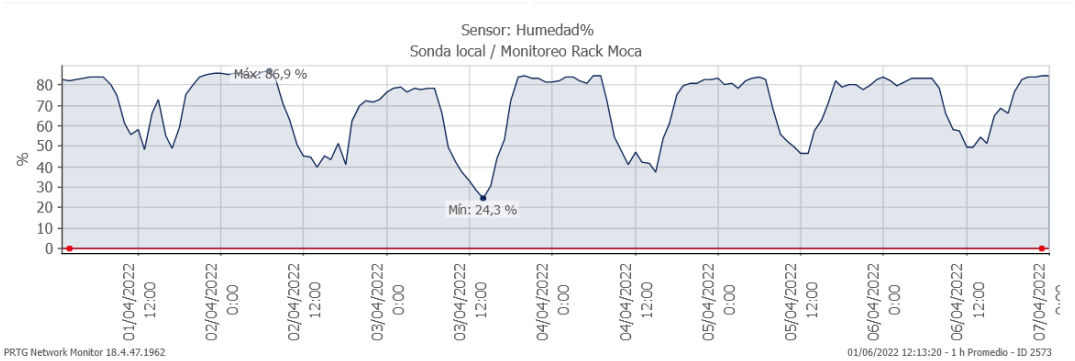

Figura 17. Resultado de pruebas de temperatura

Fuente: (Mena, 2022)

Como se puede observar en la Fig. 17 los valores de humedad oscilan entre el 24% y el 86% donde la operación del equipamiento ha sido normal. Por lo que los valores críticos serán configurados para valores superiores al 90% donde la humedad es muy alta y puede indicar la presencia de líquido.

3.6 Datos obtenidos en Centros de Datos equipados con sistemas de monitoreo de temperatura

Los siguientes resultados de temperatura han sido tomados mediante sensores alojados en algunos equipos.

En la siguiente gráfica se observa un pico de temperatura superior a los 40 grados celcius (color azul) y superior a los 50 grados celcius en las interfaces ópticas el día jueves 07 de enero de 2021, posteriormente el día viernes 8 se corrigió el problema de temperatura mismo que fue localizado en el sistema de aire acondicionado :

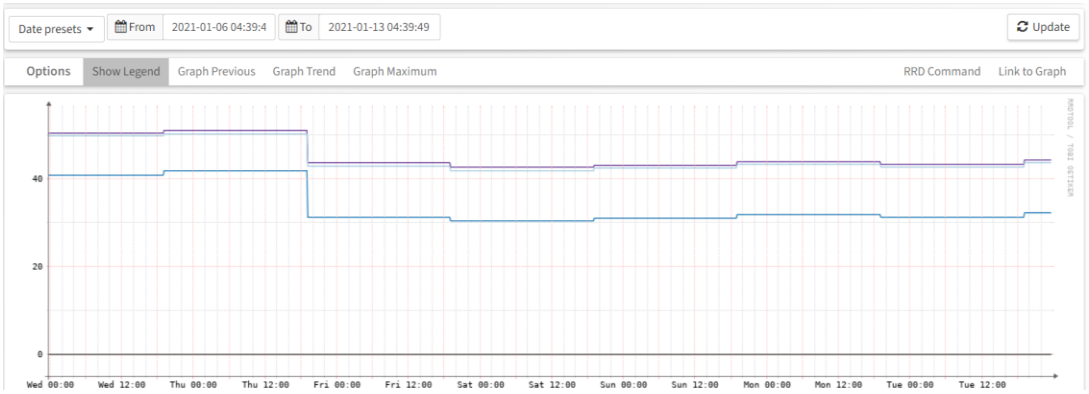

Figura 18. Monitoreo de Temperatura en Data Center

Fuente: (Mena, 2022)

En estas condiciones se observaron problemas de desconexión del equipo, en la siguiente gráfica se observa que el equipo presentó 3 desconexiones de red (líneas color rojo), posterior a los cambios realizados el comportamiento del equipo se normalizó:

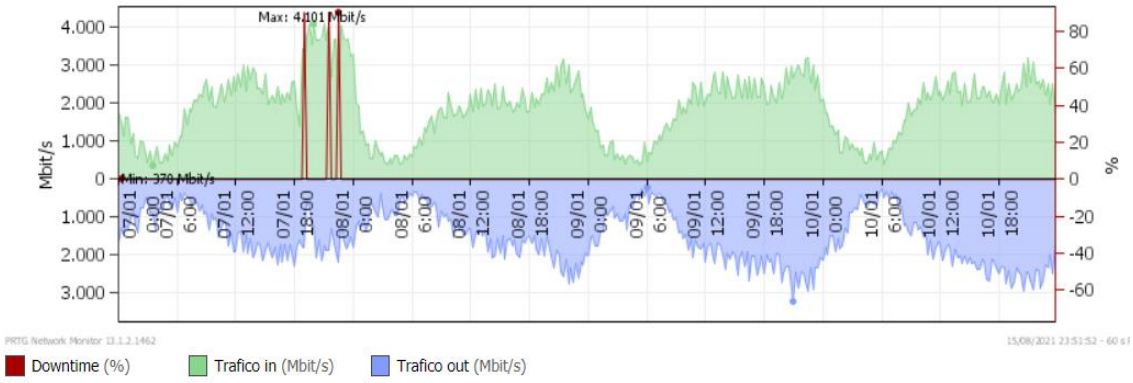

Figura 19. Comportamiento de equipamiento Mikrotik a altas temperaturas

Fuente: (Mena, 2022)

En el siguiente caso el equipo tuvo un incremento de temperatura llegando a 50 grados Celsius aproximadamente como se puede observar en la siguiente figura:

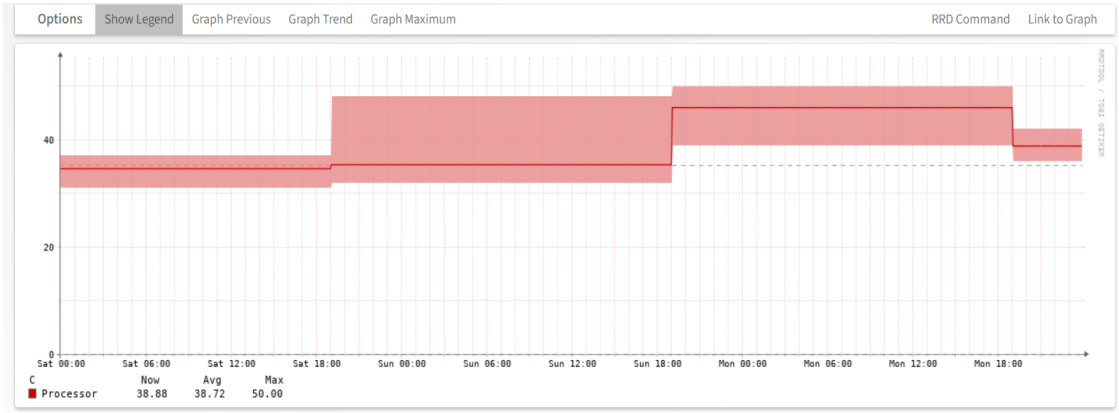

Figura 20. Caso 2 de monitoreo de Temperatura

Fuente: (Mena, 2022)

Durante este incremento de temperatura el equipo tuvo una desconexión de red como se puede observar en la siguiente gráfica:

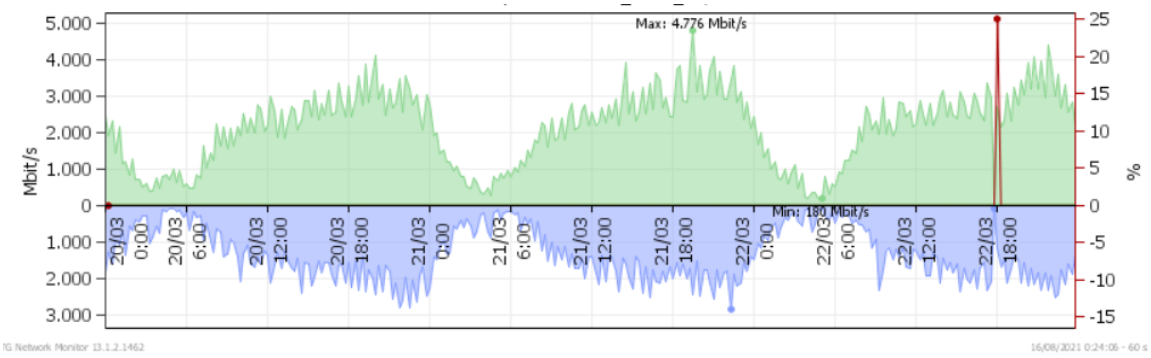

Figura 21. Caso 2 Comportamiento de equipamiento Mikrotik a altas temperaturas

Fuente: (Mena, 2022)

Por lo que el valor límite a ser configurado en el monitoreo será de 30 grados Celsius de esta manera se tiene holgura antes de presentar inconvenientes en los equipos.

3.7 Configuración de desencadenadores de notificaciones en PRTG

Con la información recopilada en las pruebas realizadas se configuran los desencadenadores de notificaciones en el PRTG de tal manera que envíen mensajes de alerta a Telegram cuando se alcanza un valor crítico de cualquier parámetro monitoreado.

Esta configuración se la realiza de la siguiente manera:

a) Se selecciona el sensor deseado donde se crea un desencadenador de notificaciones.
b) Se configuran los valores límites acorde a los valores obtenidos en el punto 3.4.2, de esta manera las notificaciones quedan de la siguiente manera:

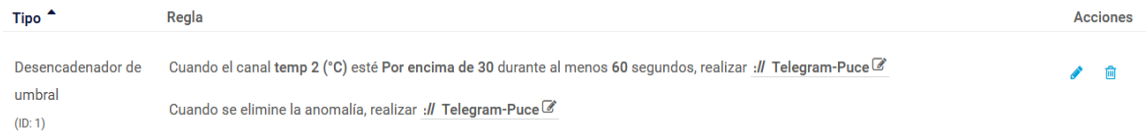

Figura 22. Ejemplo de configuración de alarmas
Fuente: (Mena, 2022)

Este procedimiento se realiza en todos los sensores.

3.8 Conexión del dispositivo Arduino en rack de Telecomunicaciones

La locación elegida para esta implementación es un nodo antiguo ubicado en el pueblo de Moca, se ha escogido este lugar debido al difícil acceso y a la distancia de este por lo que el monitoreo proactivo se vuelve indispensable ya que el tiempo de movilización del personal conllevar un tiempo considerable.

Figura 23. Implementación del dispositivo de monitoreo
Fuente: (Mena, 2022)

Como se puede observar en al Fig.22 el equipo se ha ubicado en la segunda bandeja del rack, el cableado de los sensores se sujetó con amarras para organizarlos y la alimentación se lo realiza directamente desde el UPS.
Una vez conectado el cable de red en el puerto designado del equipamiento en sitio la conectividad entre la estación gestora y el dispositivo de monitoreo se estableció inmediatamente.

3.9 Configuraciones de seguridad

Con el fin de evitar la conectividad vía SNMP al equipo de monitoreo desde dispositivos no permitidos se tomaron las siguientes medidas de seguridad:

- Cambio de comunidad. - este cambio se lo realizó en la librería agentuino.cpp en las siguientes líneas de código:

_getCommName = "public";
_setCommName = "private";

En donde las comnidades “public” y “private” son reemplazadas por una cadena de caracteres diferente.

3.10 Análisis de costos empleados en el dispositivo de monitoreo:

Para este análisis se han tomado en cuenta el costo de todos los materiales empleados en la fabricación del dispositivo de monitoreo:

Tabla 2. Tabla de Costos empleados

Item	Costo
Placa Arduino	9.5 USD
Sensor de Temperatura/Humedad	2.5 USD
Sensor de Voltaje alterno	12.0 USD
Placa ethernet	11.0 USD
Sensor magnético	2.0 USD
Chasis	17.0 USD

Fuente de alimentación	3.0 USD
Cable con tomacorriente	2.0 USD
Interruptor	0.1 USD
Bornes	4.0 USD
Total	63.1

Fuente: (Mena, 2022)

Como se puede observar en la Tabla 2, el costo total fue de 63.1USD siendo este un valor menor a los que se puede observar en la Tabla 1, de esta manera se alcanza el objetivo de la construcción del sistema de monitoreo a bajo costo.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- En las pruebas realizadas se encontraron valores de voltaje y temperatura en los cuales los equipos Mikrotik dejan de operar, con estos valores se configuraron alertas con las cuales se tiene un margen de tiempo de acción para tomar las medidas correctivas sin mayor afectación de servicios.
- En las pruebas de voltaje se encontró que el equipamiento Mikrotik opera sin problemas con valores de hasta 100 voltios, con valores inferiores el equipo empieza a tener problemas dando como resultado el apagado del mismo.
- En las pruebas de Temperatura se determinó que la temperatura máxima a la cual el equipo Mikrotik puede operar es de 60 grados Celsius, con valores superiores el equipo empieza a tener problemas de rendimiento siendo la temperatura crítica de 69 grados Celsius donde el equipo deja de responder.
- El costo total del sistema de monitoreo es menor al que se encuentra comúnmente en el mercado, siendo la construcción y diseño específicos para la implementación del presente proyecto.
- El monitoreo adecuado de las condiciones ambientales dentro del rack de comunicaciones externos ayuda a prevenir afectaciones de servicios.
- La integración del sistema de monitoreo en Telegram permite que las alertas sean enviadas a todas las personas participantes del grupo, de esta manera se logra una mayor difusión y se elimina la dependencia de una persona encargada de monitorear las alarmas en el software PRTG constantemente.
- El dispositivo de monitoreo ha sido diseñado de tal manera que no requiere configuraciones especiales para su funcionamiento, siendo así que el único requerimiento es la conexión a un puerto ethernet y a la fuente de alimentación.
- Con implementación del dispositivo en el rack de telecomunicaciones se ha logrado monitorear adecuadamente las condiciones ambientales a las cuales están operando los equipos de red, logrando prevenir afectaciones de servicios.

4.2 Recomendaciones

- Este tipo de monitoreos debe ser empleado en todos los racks y Centros de Datos, de tal manera que se pueda prevenir afectaciones de servicio y pérdida de equipos.
- El presente trabajo ha sido orientado para el gestor de monitoreo SNMP PRTG, sin embargo, puede utilizarse cualquier gestor SNMP que existe en el mercado, sea pagado o gratuito.
- Las condiciones ambientales en los racks de comunicaciones no deben superar los valores críticos a fin de evitar problemas con el equipamiento, así como con los servicios.

REFERENCIAS

Amazon. (11 de Noviembre de 2020). *Amazon*. Obtenido de Amazon: https://www.amazon.com/-/es/AP9520TH-unidad-temperatura-humedad-visualizaci%C3%B3n/dp/B0002JC2QY

Aosong. (2021). *Aosong Electronics*. Obtenido de www.aosong.com

Case, J., Fedor, M., Schoffstall, M., & Davin, J. (1990). RFC 1098. *A Simple Network Management Protocol (SNMP)*. IETF.

Dictum. (11 de Noviembre de 2020). *Dictum Ltd.* Obtenido de Dictum Ltd.: https://www.didactum-security.com/en/bundle/monitoring-system-100-it-basic-protection.html

Electrónica, M. (Febrero de 2017). *maxelectronica.com*. Obtenido de https://maxelectronica.cl/sensores/690-sensor-de-tension-ac-transformador-monofasico-zmpt101b.html

Electronics, T. (Febrero de 2022). *Talos Electronics*. Obtenido de www.taloselectronics.com

Group, H. (11 de Noviembre de 2020). *HW-Group*. Obtenido de HW-Group: https://www.hw-group.com/device/poseidon2-3266

Industry Research. (2020). *Global Communication Router Industry Market Research Report*. Obtenido de https://www.industryresearch.co/global-communication-router-industry-market-13935452

Lab, C. (s.f.). *circuitlab.com*. Obtenido de https://drive.google.com/file/d/1M4Q-TLvbLLLNFUuzeOaPjf8OgxQowewL/view

Libre, M. (11 de Noviembre de 2020). *Mercado Libre*. Obtenido de Mercado Libre: https://articulo.mercadolibre.com.ec/MEC-427953197-tarjeta-snmp-rj45-cdp-interna-monitoreo-para-ups-online-_JM?quantity=1#position=10&type=item&tracking_id=5df9ce15-12ac-4043-88d5-5c8cf11181b0

Marsagi. (Octubre de 2018). *forum.arduino.cc*. Obtenido de https://forum.arduino.cc/t/conexion-de-sensor-magnetico-de-puerta/550407

Martínez, A., Medina, V., Guido, F., & Estaban, A. (2014). Diseño e Implementación de un Módulo SNMP para el Control y Monitoreo de Nodos de la Empressa Etapa EP. Cuenca, Azuay, Ecuador: Universidad de Cuenca.

Mechatronics, N. (2021). *Naylamp Mechatronics SAC*. Obtenido de https://naylampmechatronics.com/sensores-corriente-voltaje/393-transformador-de-voltaje-ac-zmpt101b.html

Mena, A. R. (Marzo de 2022). Diseño de un Sistema de Monitoreo de Voltaje y Parámetros Ambientales. Quito, Pichincha, Ecuador: Pontificia Universidad Católica del Ecuador.

MGSystem. (2021). *Mercado Libre*. Obtenido de https://articulo.mercadolibre.com.ec/MEC-506357661-mgsystem-modulo-sensor-de-tempertura-y-humedad-dht11-arduino-_JM

Moreno, A., & Córcoles, S. (2018). *Arduino. Curso Práctico.* Madrid: RA-MA.

Nelson, R. (2014).). Diseño de un Sistema de Cableado Estructurado, con cable Tipo FTP CAT_6 (Blindado) 4 Pares para el nuevo Hospital Docente Calderón de la Ciudad de Quito del Ministerio de Salud Pública (MSP). Quito, Ecuador: Universidad de las Américas.

Pastori, D., & López, P. (2016). *ARGOS, Sistema de Control para CPD.* Santa Fe: Universidad Abierta Interamericana.

Rossen, P., & Popnikolov, A. (2005). Una Herramienta de Gestión de Redes Virtuales. Azcapotzalco, México: Universidad Autónoma Metropolitana.

Rupp, J., & Zobel, D. (2017). Quo Vadis SNMP? Nuremberg, Alemania: PAESSLER A.G.

Santana, M. (2017). Implementación de un Sistema de Sensores, Monitoreo y Alertas de la Temperatura y Humedad de un Centro de Datos. Guayaquil, Ecuador: Universidad de Guayaquil.

Siggins, M. (2009). *Top 5 causes of network failures.* Obtenido de dpstele: https://www.dpstele.com/blog/top-five-causes-of-network-failure.php

Veslateguí, V. (2019). Desarrollo de un Prototipo para la Monitorización Inalámbrica de Potencia de un Data Center Universitario. Quitio, Ecuador: Escuela Politécnica Nacional.

ANEXOS

Anexo 1: Programación en Arduino

```
#include <Streaming.h>
#include <Ethernet.h>
#include <SPI.h>
#include <MemoryFree.h>
#include <Agentuino.h>
#include <DHT.h>
#include <Filters.h>
#include <Wire.h>
//////////////Sensor de temperatura/humedad
// Se define el pin digital en el cual se conectará el sensor
#define DHTPIN A4
// Se define el tipo de sensor
#define DHTTYPE DHT11
// Inicializamos el sensor DHT11
DHT dht(DHTPIN, DHTTYPE);
//////////////entradas
int valInput1;
int valInput2;
int valInput3;
int valInput4;
//////////Sensor de Voltaje Alterno

float Frecuencia = 60;                          // Frecuencia (Hz)
float promsenal = 40.0/Frecuencia;          // promedio de la señal
int SensorA5 = 5;                                //A5
float calibraciónV = -0.04;                    // ajuste de calibración
float CPendiente = 0.0405;                          // ajuste de calibración
float voltios;                                   // Voltage
```

```
int voltsR;                                          // Voltage
unsigned long periodo = 1000;
unsigned long tiempoAnterior = 0;
////////////// definición de variables
int raw2 = 0;
 int raw3 = 0;
 int raw4 = 0;
 int raw5 = 0;
 float temp2 = 0;
 float temp3 = 0;
 float temp4 = 0;
 float temp5 = 0;
 float h;
 float t;
 int temp = 0;
 int hum = 0;
 float temperature = 0;
 float humidity = 0;
const char  reg= "C";                                // C=Temperatura en grados Celsius
static byte mac[] = { 0xEE, 0xAA, 0xBB, 0xEE, 0xFA, 0xED };      //Dirección MAC
Ethernet Shield                                                  //Comunicación Ethernet
EthernetClient client;
const int a2 = A2;
const int a3 = A3;
const int a4 = A4;
const int a5 = A5;
int term2=0;
int term3=0;
int term4=0;
int term5=0;
///////////////// Configuración OID
```

```
const char sysDescr[] PROGMEM = "1.3.6.1.2.1.1.1.0"; // Descripción del sistema
const char sysContact[] PROGMEM = "1.3.6.1.2.1.1.4.0";      // Contacto
const char sysName[] PROGMEM = "1.3.6.1.2.1.1.5.0";      // Nombre del sistema
const char sysLocation[] PROGMEM = "1.3.6.1.2.1.1.6.0";// Ubicación del sistema
const char sysServices[] PROGMEM = "1.3.6.1.2.1.1.7.0";  // System Services
const char temperature0[] PROGMEM = "1.3.6.1.3.2016.5.1.0";  //Temperatura
const char humidity1[] PROGMEM = "1.3.6.1.3.2016.5.1.1";    //Humedad en %
const char voltage2[] PROGMEM = "1.3.6.1.3.2016.5.1.2";     //Voltaje in Voltios
const char disInput1[] PROGMEM = "1.3.6.1.3.2016.5.1.3";   //Sensor Magenético
//OID para Relé y Control
static int32_t relayState;
//byte estático del relé;
int relay=6;
static int32_t relayCommand;
int32_t *relayPointer;
static char relay1OID [] = "1.3.6.1.3.2016.5.1.7";                          // Relé
// RFC1213 SNMP
static char locDescr[]= "SNMP monitoreo de temperatura";             // solo lectura
static char locContact[50]         = "PUCE";
static char locName[20]            = "Quito";
static char locLocation[20]        = "Ecuador";
static int32_t locServices         = 2;
uint32_t prevMillis = millis();
char oid[SNMP_MAX_OID_LEN];
SNMP_API_STAT_CODES api_status;
SNMP_ERR_CODES status;
void pduReceived()
{
///// Procesamiento de PDU
  SNMP_PDU pdu;
  api_status = Agentuino.requestPdu(&pdu);
```

```
  if ((pdu.type == SNMP_PDU_GET || pdu.type == SNMP_PDU_GET_NEXT ||
pdu.type == SNMP_PDU_SET)
    && pdu.error == SNMP_ERR_NO_ERROR && api_status ==
SNMP_API_STAT_SUCCESS ) {
////////////////// PDU de información de dispositivo
    pdu.OID.toString(oid);
      if ( strcmp_P(oid, sysDescr ) == 0 ) {
      if ( pdu.type == SNMP_PDU_SET ) {
        pdu.type = SNMP_PDU_RESPONSE;
        pdu.error = SNMP_ERR_READ_ONLY;
      } else {
        status = pdu.VALUE.encode(SNMP_SYNTAX_OCTETS, locDescr);
        pdu.type = SNMP_PDU_RESPONSE;
        pdu.error = status;
      }
    } else if ( strcmp_P(oid, sysName ) == 0 ) {
      if ( pdu.type == SNMP_PDU_SET ) {
        status = pdu.VALUE.decode(locName, strlen(locName));
        pdu.type = SNMP_PDU_RESPONSE;
        pdu.error = status;
      } else {
        status = pdu.VALUE.encode(SNMP_SYNTAX_OCTETS, locName);
        pdu.type = SNMP_PDU_RESPONSE;
        pdu.error = status;
      }
    } else if ( strcmp_P(oid, sysContact ) == 0 ) {
      if ( pdu.type == SNMP_PDU_SET ) {
        status = pdu.VALUE.decode(locContact, strlen(locContact));
        pdu.type = SNMP_PDU_RESPONSE;
        pdu.error = status;
```

```
    } else {
      status = pdu.VALUE.encode(SNMP_SYNTAX_OCTETS, locContact);
      pdu.type = SNMP_PDU_RESPONSE;
      pdu.error = status;
    }
  } else if ( strcmp_P(oid, sysLocation ) == 0 ) {
    if ( pdu.type == SNMP_PDU_SET ) {
      status = pdu.VALUE.decode(locLocation, strlen(locLocation));
      pdu.type = SNMP_PDU_RESPONSE;
      pdu.error = status;
    } else {
      status = pdu.VALUE.encode(SNMP_SYNTAX_OCTETS, locLocation);
      pdu.type = SNMP_PDU_RESPONSE;
      pdu.error = status;
    }
  } else if ( strcmp_P(oid, sysServices) == 0 ) {
    if ( pdu.type == SNMP_PDU_SET ) {
      pdu.type = SNMP_PDU_RESPONSE;
      pdu.error = SNMP_ERR_READ_ONLY;
    } else {
      status = pdu.VALUE.encode(SNMP_SYNTAX_INT, locServices);
      pdu.type = SNMP_PDU_RESPONSE;
      pdu.error = status;
    }

  }
////////////////////PDU para información de Sensores
  else if ( strcmp_P(oid, temperature0 ) == 0 )
  {
    if ( pdu.type == SNMP_PDU_SET )
    {
```

```
    pdu.type = SNMP_PDU_RESPONSE;
    pdu.error = SNMP_ERR_READ_ONLY;
  }
  else
  {
    status = pdu.VALUE.encode(SNMP_SYNTAX_INT, temp);
    pdu.type = SNMP_PDU_RESPONSE;
    pdu.error = status;
  }
}
     else if ( strcmp_P(oid, humidity1 ) == 0 )
{
  if ( pdu.type == SNMP_PDU_SET )
  {
    pdu.type = SNMP_PDU_RESPONSE;
    pdu.error = SNMP_ERR_READ_ONLY;
  }
  else
  {
    status = pdu.VALUE.encode(SNMP_SYNTAX_INT, hum);
    pdu.type = SNMP_PDU_RESPONSE;
    pdu.error = status;
  }
}
 else if ( strcmp_P(oid, voltage2 ) == 0 )
{
  if ( pdu.type == SNMP_PDU_SET )
  {
    pdu.type = SNMP_PDU_RESPONSE;
    pdu.error = SNMP_ERR_READ_ONLY;
  }
```

```
      else
      {
////////////// obtención de voltaje
         float voltios;
         RunningStatistics inputStats;
         inputStats.setWindowSecs(promsenal);
         int i=0;
         while(i<120) {
           SensorA5 = analogRead(A5);                    //Leer pin Analógico
           inputStats.input(SensorA5);
           if((unsigned long)(millis() - tiempoAnterior) >= periodo) {
             voltios = calibraciónV + CPendiente * inputStats.sigma();
          //offset y amplitud
             voltios = voltios*(40.3231);                          //calibración
             voltios=voltios-11;
             Serial.print("\tVoltage: ");
             Serial.println(voltios);
             tiempoAnterior = millis();
             tiempoAnterior = 0;
           }
         i=i+1;
         }
       voltsR = (int)voltios;
       status = pdu.VALUE.encode(SNMP_SYNTAX_INT, voltsR);
       pdu.type = SNMP_PDU_RESPONSE;
       pdu.error = status;
     }
   }
      else if ( strcmp_P(oid, disInput1 ) == 0 )
   {
     if ( pdu.type == SNMP_PDU_SET )
```

```
  {
    pdu.type = SNMP_PDU_RESPONSE;
    pdu.error = SNMP_ERR_READ_ONLY;
  }
  else
  {
    status = pdu.VALUE.encode(SNMP_SYNTAX_INT, valInput1);
    pdu.type = SNMP_PDU_RESPONSE;
    pdu.error = status;
  }
}
else if ( strcmp_P(oid, disInput2 ) == 0 )
{
  if ( pdu.type == SNMP_PDU_SET )
  {
    pdu.type = SNMP_PDU_RESPONSE;
    pdu.error = SNMP_ERR_READ_ONLY;
  }
  else
  {
    status = pdu.VALUE.encode(SNMP_SYNTAX_INT, valInput2);
    pdu.type = SNMP_PDU_RESPONSE;
    pdu.error = status;
  }
}

else if ( strcmp_P(oid, disInput3 ) == 0 )
{
  if ( pdu.type == SNMP_PDU_SET )
  {
    pdu.type = SNMP_PDU_RESPONSE;
```

```
    pdu.error = SNMP_ERR_READ_ONLY;
  }
  else
  {
    status = pdu.VALUE.encode(SNMP_SYNTAX_INT, valInput3);
    pdu.type = SNMP_PDU_RESPONSE;
    pdu.error = status;
  }
}
else if ( strcmp_P(oid, disInput4 ) == 0 )
{
  if ( pdu.type == SNMP_PDU_SET )
  {
    pdu.type = SNMP_PDU_RESPONSE;
    pdu.error = SNMP_ERR_READ_ONLY;
  }
  else
  {
    status = pdu.VALUE.encode(SNMP_SYNTAX_INT, valInput4);
    pdu.type = SNMP_PDU_RESPONSE;
    pdu.error = status;
  }
}
//////////////Relé
else if ( strcmp(oid, relay1OID ) == 0 ) {
  if ( pdu.type == SNMP_PDU_SET ) {
    status = pdu.VALUE.decode(relayPointer);
    if (status == 0) {
      relayCommand = *pdu.VALUE.data;
      if (relayCommand == 1) {
        digitalWrite(relay, HIGH);
```

```
          relayState = 1;
        }
        else {

          digitalWrite(relay, LOW);
          relayState = 0;
        }
      }
      pdu.type = SNMP_PDU_RESPONSE;
      pdu.error = status;
    } else {
      ///// Respuesta para una petición GET
      status = pdu.VALUE.encode(SNMP_SYNTAX_INT32, relayState);
      pdu.type = SNMP_PDU_RESPONSE;
      pdu.error = status;
    }
  }
    else {
    pdu.type = SNMP_PDU_RESPONSE;
    pdu.error = SNMP_ERR_NO_SUCH_NAME;
  }
  Agentuino.responsePdu(&pdu);
 }
  Agentuino.freePdu(&pdu);
}

////// Inicialización ethernet
 void setup()
{
  Serial.begin(9600);
  Ethernet.begin(mac);                                    // MAC Estática
```

```
  api_status = Agentuino.begin();              //Inicio de SNMP en Ethernet shield
  pinMode( A2, INPUT );
  pinMode( A3, INPUT );
  pinMode( A4, INPUT );
  pinMode( A5, INPUT );
  // Inicializador del sensor DHT
  dht.begin();
  pinMode(relay,OUTPUT);
  digitalWrite(relay, HIGH);
  relayState= 0;
  float aInput= 0;
  int dInput= 0;
  if ( api_status == SNMP_API_STAT_SUCCESS ) {
    Agentuino.onPduReceive(pduReceived);
    delay(10);
    return;
  }
  delay(10);
}
void loop()
{
  Agentuino.listen();
  if(millis()-prevMillis>2000){
    h = dht.readHumidity();                          //// Lectura de humedad relativa
    hum=(int)h;
    t = dht.readTemperature();                       //// Lectura de Temperatura
    temp=(int)t;
    prevMillis = millis();
    Serial.print("\tTemp: ");
    Serial.println(t);
    Serial.print("\tHum: ");
```

```
    Serial.println(h);
  }
  //////////Lectura de entradas
  valInput1 = digitalRead(5);
  valInput2 = digitalRead(3);
  valInput3 = digitalRead(2);
  valInput4 = digitalRead(4);
  Serial.print("\tInp: ");
  Serial.println(valInput1);
}
```

Anexo 2: Configuración de Software PRTG

Creación de sensores

Para crear los sensores en PRTG es necesario añadir la IP del dispositivo, para este caso la IP es 100.64.72.10 (obtenida vía DHCP), por la librería utilizada la versión del protocolo SNMP es la V1 y la cadena de comunidad es “public”, por lo que el sensor creado será de la siguiente manera:

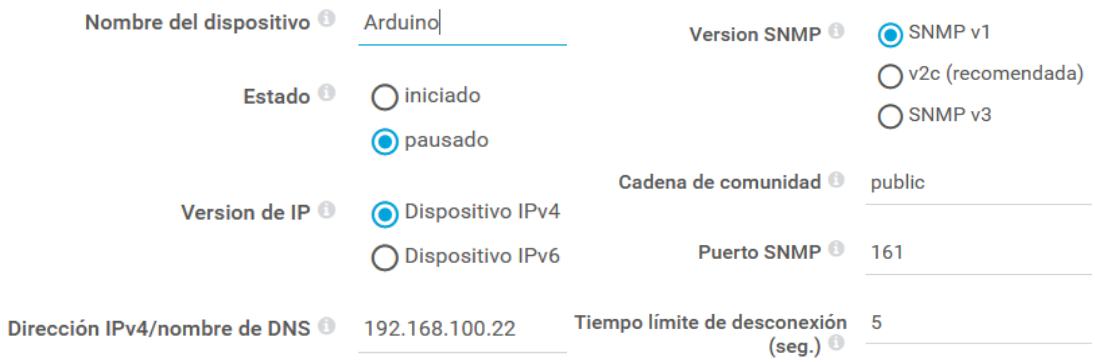

Figura 24. Configuración de Dispositivo PRTG

Fuente: (Mena, 2022)

Una vez creado el dispositivo en PRTG se crean los sensores del tipo SNMP avanzado, para esto se especifica el OID de cada sensor:

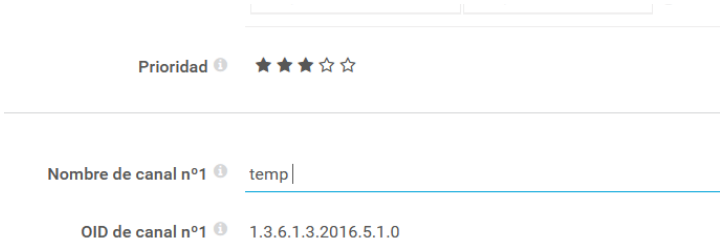

Figura 25. Configuración de sensores en PRTG

Fuente: (Mena, 2022)

De esta manera el sensor Temperatura se encuentra creado y se empieza a recibir información del sensor:

a) Humedad (para obtener variaciones se ha sumergido en agua):

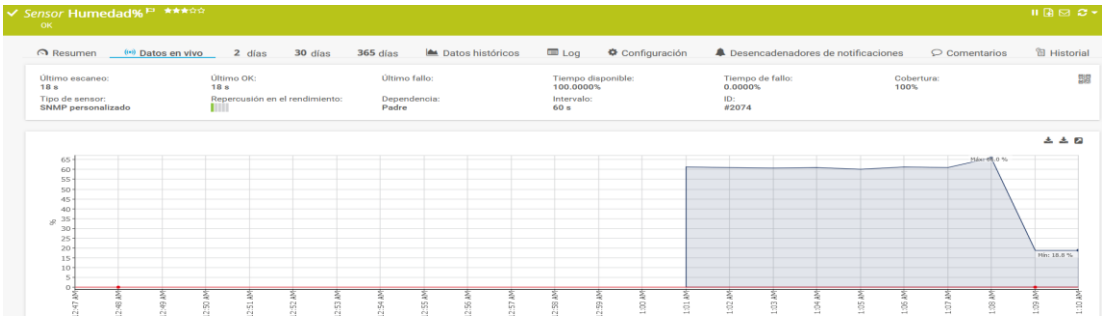

Figura 26. Respuesta del sensor de Humedad en PRTG

Fuente: (Mena, 2022)

b) Temperatura:

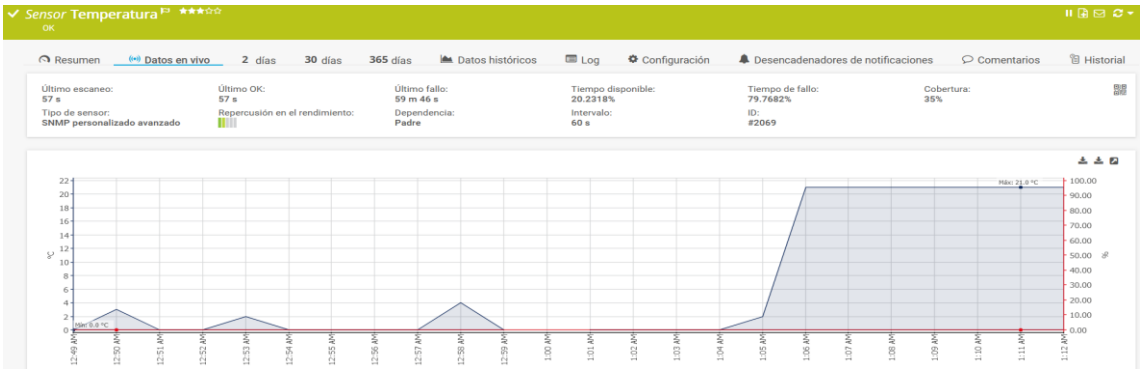

Figura 27. Respuesta del sensor de Temperatura en PRTG

Fuente: (Mena, 2022)

c) Intrusión, se ha variado el estado de la puerta (valor de 1 para cerrado y 0 para abierto)

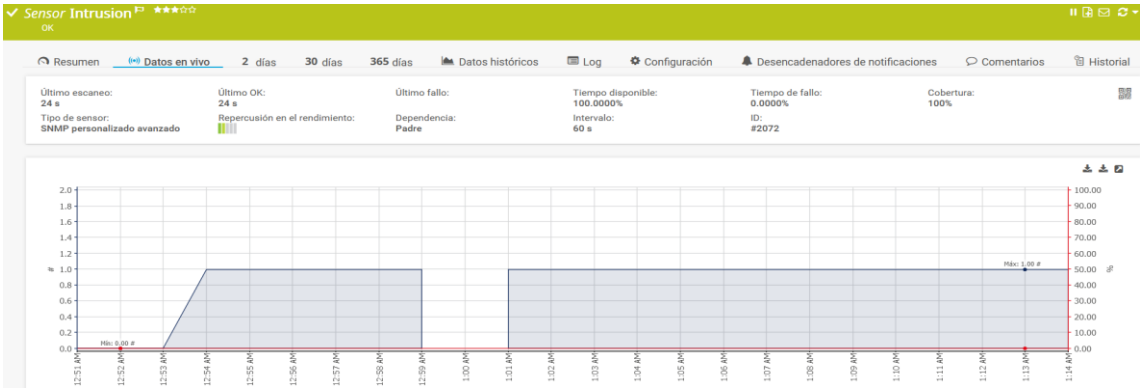

Figura 28. Respuesta del sensor de Intrusión en PRTG

Fuente: (Mena, 2022)

d) Voltaje:

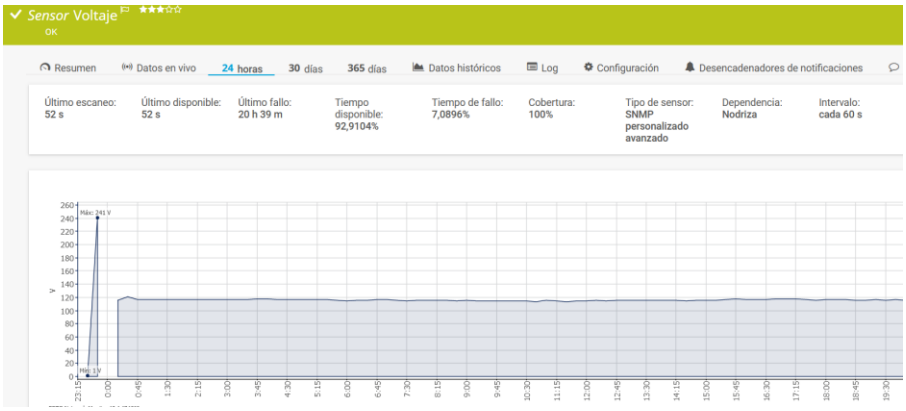

Figura 29. Respuesta del sensor de Voltaje en PRTG

Fuente: (Mena, 2022)

Integración de alertas con Telegram

Gracias a la flexibilidad de Telegram es posible integrar las notificaciones de alertas del PRTG con Telegram de la siguiente manera:

a) En primera instancia se debe crear una cuenta en Telegram e instalar la aplicación en una computadora.

b) Posteriormente se crea un “Bot”, mismo que es un software integrado en Telegram cuya función es el envío automatizado de mensajes en un chat, los cual es muy ventajoso ya que no depende de un ser humano para el envío de mensajes. Para añadirlo se debe abrir un chat con BotFather (el cual es un gestor de Bots) en donde se ingresará el comando /newbot.

c) A continuación, se siguen las instrucciones que se presentan, tales como nombre del Bot, usuario del Bot. De esta manera el Bot estará creado, es necesario prestar especial atención en el token que nos entrega el Bot ya que se utilizará posteriormente.

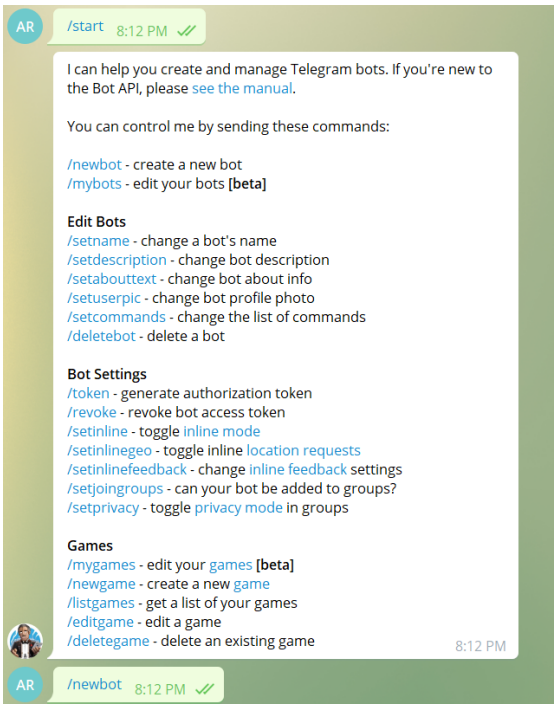

Figura 30. Creación de Bot en Telegram

Fuente: (Mena, 2022)

d) A fin de evitar cualquier inconveniente con el envío de mensajes desde el PRTG es necesario deshabilitar la privacidad del Bot, para esto se ejecutan los siguientes comandos (para la presente implementación se ha elegido el nombre de PUCE_PRTG_bot):

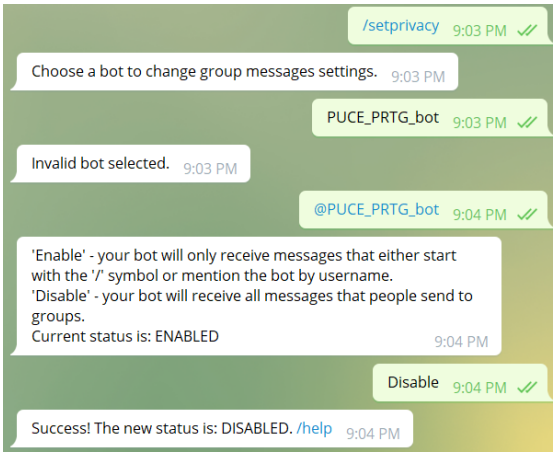

Figura 31. Des habilitación de privacidad del Bot en Telegram

Fuente: (Mena, 2022)

e) Opcionalmente se puede crear un chat de grupo en el cual se debe integrar el Bot creado a fin de que las alertas lleguen a todas las personas que se hayan agregado en el grupo, para la presente implementación se ha creado un grupo llamado PUCE-PRTG.

f) A continuación, es necesario identificar el ID del chat para esto es necesario ingresar a la URL, https://api.telegram.org/bot<token>/getMe, después se debe ingresar a: https://api.telegram.org/bot<token>/getUpdates, para el presente proyecto se obtuvo el siguiente Chat ID:

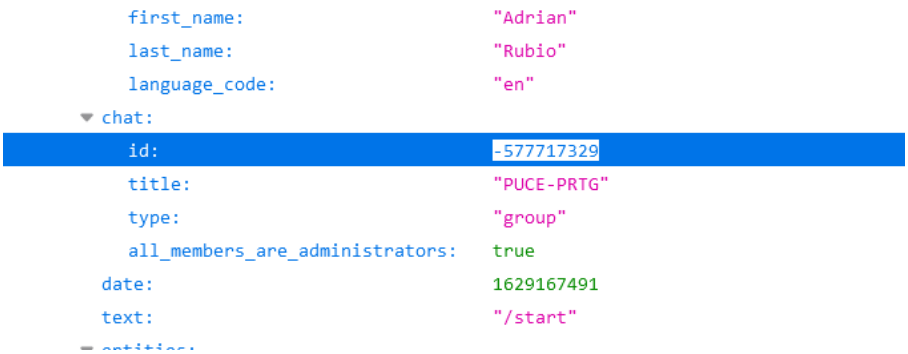

Figura 32. Chat ID de Telegram

Fuente: (Mena, 2022)

g) Con el ID obtenido en el PRTG se crea una plantilla de notificaciones, donde se asignará una acción del tipo HTTP utilizando la información del CHAT ID y el Token del Bot:

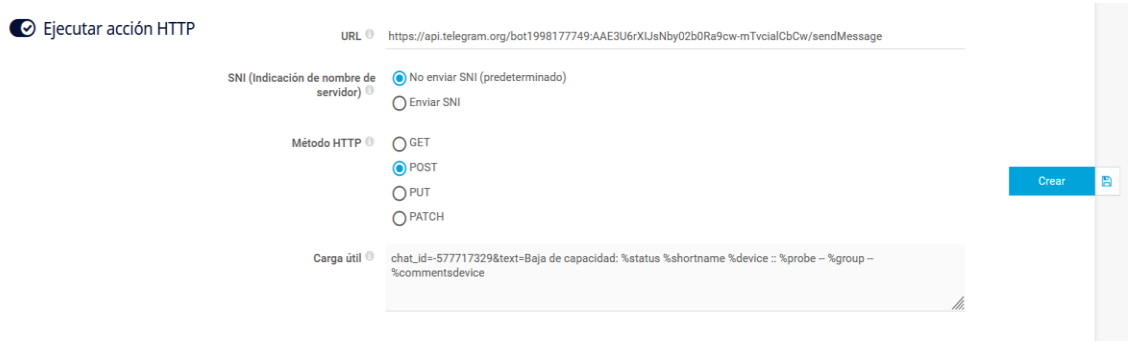

Figura 33. Integración de Telegram con PRTG

Fuente: (Mena, 2022)

h) Para validar que la comunicación entre el PRTG y Telegram es correcta se envía un mensaje de prueba obteniendo el siguiente resultado en el chat de Telegram:

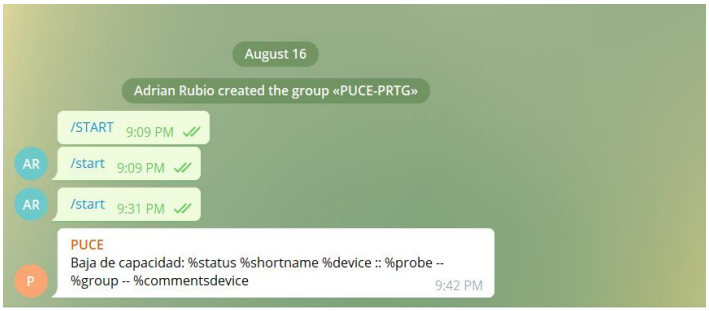

Figura 34. Prueba de conectividad entre Telegram y PRTG

Fuente: (Mena, 2022)