Informaticae Abstracta. ISSN: 3061-8533
Año 2025, Vol. 3, No. 2, pp. 62-74
JUL | DIC 2025
https://doi.org/10.36677/ia.v3n2.27650

Arquitectura y Sincronización de un Sistema de Monitoreo Térmico con Notificaciones Push mediante Aplicaciones Web Progresivas (PWA) empleando una Raspberry Pi

Javier Salas-García; Otniel Portillo-Rodríguez; Adriana H. Vilchis-González

Información del Artículo

Recibido: 1 de octubre, 2025
Aceptado: 30 de noviembre, 2025

Palabras clave: Raspberry Pi, Temperatura, PWA, Notificaciones, IoT, Python, PHP, MariaDB

Resumen

Este trabajo explica el desarrollo de un sistema de monitoreo térmico, denominado «Monitor térmico», basado en una Raspberry Pi y una Aplicación Web Progresiva (PWA). El objetivo central es superar las restricciones de los sistemas operativos móviles modernos que limitan la ejecución de procesos en segundo plano para sitios web convencionales. A diferencia de las aplicaciones nativas distribuidas por tiendas oficiales como App Store o Play Store, esta solución permite al usuario recibir notificaciones push proactivas en su dispositivo móvil incluso cuando la página web del sistema no se encuentra abierta. Se detalla la integración del sensor MAX6675, el almacenamiento persistente en MariaDB y la implementación del protocolo Web Push mediante llaves criptográficas VAPID para asegurar la entrega de alertas de sobrecalentamiento. El artículo ofrece una metodología que escala desde implementaciones locales gratuitas, abordando el manejo de direcciones IP dinámicas, hasta el despliegue en entornos de hosting profesional, proporcionando una alternativa robusta y segura para el monitoreo proactivo sin las barreras de los ecosistemas cerrados.

1. Introducción

Construir sistemas para la supervisión remota de dispositivos electrónicos suele ser costoso debido a la dependencia de plataformas propietarias o aplicaciones móviles complejas de desarrollar. Sin embargo, el uso de Tecnologías Web Progresivas (PWA) permite crear sistemas de comunicación bidireccional donde no solo se monitorea información en tiempo real, sino que también se pueden ejecutar acciones sobre un hardware remoto directamente desde la pantalla de un celular. El presente artículo utiliza el desarrollo de un controlador de temperatura, denominado de ahora en adelante como el «Monitor térmico», como un pretexto práctico para aterrizar conceptos de ingeniería de software y electrónica. El objetivo es demostrar que, mediante una Raspberry Pi y estándares web abiertos, es posible implementar una solución reproducible que va más allá de un simple sensor: es una arquitectura completa para el envío de notificaciones y el control remoto de procesos, eliminando los costos de licencias y las barreras de las tiendas de aplicaciones tradicionales.

A pesar del creciente interés académico en las aplicaciones web, existe una brecha notable entre la teoría y la ejecución práctica en la literatura científica actual. Diversas investigaciones se centran en revisiones exhaustivas de la arquitectura PWA [1], o en comparaciones de su potencial frente a aplicaciones nativas [2]. No obstante, estos trabajos suelen limitarse a describir el «qué» —es decir, las ventajas y la estructura teórica— sin profundizar en el «cómo» implementar estas soluciones de manera granular. La falta de guías detalladas que aborden

desde la configuración de llaves criptográficas VAPID hasta la sincronización física de sensores con lógica de control persistente representa un obstáculo para la comunidad científica y técnica. Este artículo se distingue al proporcionar una metodología de implementación rigurosa y completamente reproducible, resolviendo los retos de configuración técnica que son omitidos en las fuentes académicas tradicionales.

Para que el usuario cuente con un sistema de alerta proactivo, se diseñó el «Monitor térmico» como una Aplicación Web Progresiva (PWA). Esta elección tecnológica es clave para resolver el principal problema de los dispositivos móviles modernos: las estrictas restricciones de ahorro de energía y gestión de memoria que suspenden la actividad de los navegadores cuando una pestaña no está visible. El objetivo fundamental del sistema es garantizar que el usuario reciba notificaciones push incluso cuando la página web del monitor no se encuentra abierta en el dispositivo, logrando un comportamiento idéntico al de una aplicación industrial nativa pero sin la necesidad de depender de los procesos de revisión y costos de las tiendas oficiales (App Store o Play Store). Para asegurar la viabilidad técnica en diversos contextos, el artículo detalla tres rutas de despliegue: el uso de herramientas gratuitas para una implementación local básica, la contratación de servicios de túnel para obtener direcciones permanentes y la migración hacia un servidor de hosting para una solución robusta.

Para reproducir este proyecto, se deben obtener los archivos de código desde el repositorio oficial. El código completo se encuentra disponible en GitHub mediante la dirección `https://github.com/JavierSalasGarcia/notificaciones_PWA_raspberry`, el cual contiene toda la estructura necesaria para el funcionamiento del servidor web, la lógica de control en Python y los esquemas de la base de datos. En la Sección 5 se detalla exhaustivamente el procedimiento técnico para la obtención de estos archivos y su correcta ubicación dentro del sistema de archivos de la Raspberry Pi.

Este artículo explica cada detalle del proceso. Primero se describen las piezas que hacen posible que el navegador se comporte como una aplicación. Después se muestra cómo unir las piezas físicas con el código de manera segura, detallando la conexión del sensor y el manejo de los datos. También se explica cómo configurar el sistema para que los avisos lleguen correctamente a teléfonos iPhone y Android, y finalmente se detallan los pasos para que todo quede funcionando en internet de forma permanente.

2. La Web Moderna y las Aplicaciones Progresivas

En los últimos años, la forma en que se usa el internet ha cambiado mucho. Antes, los sitios web solo servían para leer información, y si se requería una aplicación que enviara avisos al celular, se tenía que descargar de una tienda oficial. Esto era un problema porque crear aplicaciones para diferentes tipos de teléfonos es tardado y difícil. Las Aplicaciones Web Progresivas (PWA) nacieron para solucionar esto. Una PWA es básicamente un sitio web que puede instalarse en el celular y funcionar casi igual que una aplicación normal [1].

Para que un sitio web se convierta en una PWA, se necesitan cuatro piezas principales que se encuentran en el código del «Monitor térmico». La primera pieza es el «Manifiesto», que es el archivo `Repositorio/public/manifest.json`. Este archivo es como una carta de presentación para el teléfono donde se escribe el nombre de la aplicación y se eligen los

colores y el icono que se verá en la pantalla de inicio. La segunda pieza es el «Service Worker», un programa en JavaScript ubicado en `Repositorio/public/sw.js` que corre en el fondo del teléfono para recibir alertas incluso si la página está cerrada [3]. La tercera pieza es la seguridad con HTTPS, que asegura que la información de temperatura viaje protegida y que el navegador confíe en los permisos otorgados.

La cuarta pieza fundamental, que no se encuentra directamente en el código descargado del repositorio, es la carpeta llamada «vendor». Esta carpeta es de gran importancia porque contiene todas las librerías externas que permiten que el sistema de notificaciones funcione. Se tiene que tener esta carpeta porque en ella están los programas que manejan el cifrado y la comunicación con los servidores de Google y Apple. Para crearla, se utiliza una herramienta llamada Composer que descarga automáticamente miles de archivos necesarios. No se incluye en el repositorio de GitHub debido a que la enorme cantidad de archivos que genera haría que la descarga fuera innecesariamente pesada y lenta. Sin la carpeta «vendor», un iPhone no tendría forma de detectar las funciones de envío de mensajes, por lo que su creación es un paso obligatorio para que el sistema sea profesional. Además, el «Monitor térmico» incluye una sección de ayuda en la página principal que guía al usuario con instrucciones claras para «instalar» la aplicación, diferenciando entre los pasos para sistemas Android y los requisitos específicos de los equipos de Apple.

3. Almacenamiento de Datos y Conexión de Hardware

Para que el sistema sea de utilidad, se deben conectar las piezas físicas con el código de computadora de manera segura. El cerebro del «Monitor térmico» es una Raspberry Pi 4 conectada a un sensor MAX6675 para medir la temperatura con gran precisión [4]. Para conectar estos dos componentes, se deben usar los pines de la Raspberry y el sensor mediante cables de puente, tal como se muestra en el diagrama de conexiones de la Figura 1. Es fundamental distinguir entre las dos formas de contar los pines en la Raspberry Pi. La primera es la numeración física o de placa (BOARD), que simplemente numera los pines del 1 al 40 según su posición física en dos columnas. La segunda es la numeración del procesador (BCM), la cual se basa en el nombre funcional de cada pin según el diseño interno del equipo. El «Monitor térmico» utiliza exclusivamente la numeración BCM porque es el estándar compatible con la mayoría de las librerías de programación profesionales. Para la conexión SPI del sensor MAX6675, se deben identificar los pines físicos: el Reloj (SCK) se conecta al pin BCM 11 (Pin físico 23), la salida de datos (SO) al pin BCM 9 (Pin físico 21) y la selección del chip (CS) al pin BCM 8 (Pin físico 24). La alimentación del sensor debe realizarse estrictamente con el pin de 3.3V (Pin físico 1) para no dañar los componentes, compartiendo la tierra común (GND) en el pin físico 9. Para identificar el pin número uno, se observa la esquina más alejada de los puertos USB, junto al pequeño foco rojo de encendido; a partir de ahí, los pines nones están en la fila interior y los pares en la fila exterior. La conexión física requiere precisión para evitar daños permanentes en la placa, por lo que se debe seguir un mapeo riguroso basado en la función de cada componente. En la Tabla 1 se presenta la relación exacta entre los pines del procesador y su ubicación física, facilitando el cableado del sensor y los actuadores.

Tabla 1: Mapeo de conexiones entre la Raspberry Pi y los componentes externos.

Dispositivo y Señal	Pin Físico (Raspberry Pi)	Pin BCM (Código)
Sensor MAX6675		
SCK (Reloj)	23	11
MISO (Datos)	21	9
CS (Selección)	24	8
VCC (3.3V)	1	-
GND (Tierra)	9	-
Actuadores		
LED de Control	12	18
Relevador (Calefactor)	16	23

Contar con este mapeo específico previene cortocircuitos eléctricos y asegura que el controlador de Python se comunique con la interfaz de hardware correcta desde el primer intento.

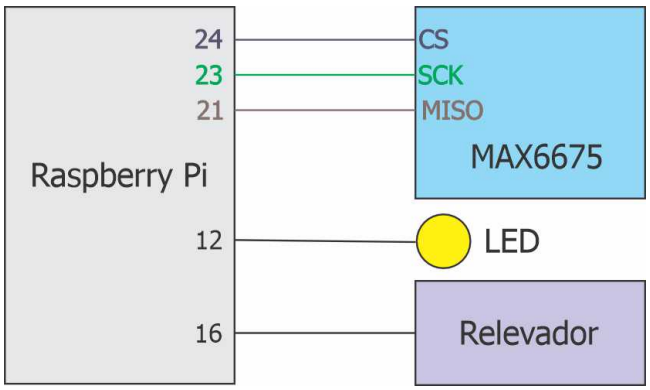

Figura 1: Diagrama de conexiones de pines entre el sensor, actuadores y Raspberry Pi (Elaboración propia).

Además del sensor, el sistema tiene dos salidas importantes operadas mediante pines GPIO. La primera va hacia un LED que sirve para probar que desde el celular se pueden enviar órdenes al sistema, verificando que la comunicación no es solo de lectura sino también de escritura de valores. La segunda salida va hacia un relevador que controla un elemento calefactor. En lugar del relevador, se puede usar otro LED para simular su funcionamiento durante las pruebas. El objetivo del «Monitor térmico» es controlar la temperatura de un recipiente que se debe mantener mucho más caliente que el ambiente. Por esta razón, cuando se alcanza la temperatura deseada, el sistema simplemente apaga el relevador y el recipiente se enfría solo por el contacto natural con el aire. Toda esta lógica requiere asegurar que los cables estén firmemente conectados, ya que un cable flojo podría enviar datos falsos que activarían el relevador de forma innecesaria o peligrosa.

La lógica de las notificaciones y el control de potencia es sumamente precisa para proteger el proceso sin molestar al usuario con avisos innecesarios. El sistema se basa en tres números configurables: la temperatura deseada, el umbral de operación y el offset de sobrecalentamiento. Por ejemplo, si se elige una temperatura deseada de 60 °C y un umbral de

2 °C, el sistema establece un rango aceptable de entre 58 °C y 62 °C. El «Monitor térmico» activará el relevador del calefactor en cuanto la temperatura baje de 58 °C y lo apagará únicamente cuando se alcancen los 62 °C, permitiendo que la temperatura baje de forma natural (ver Figura 2).

Para las alertas, se utiliza el tercer valor llamado offset. Si este fuera de 3 °C, el sistema sumará este valor al límite superior del rango aceptable. En el ejemplo anterior, el límite de peligro sería de 65 °C (60 + 2 + 3). Al llegar a esa temperatura crítica, se enviará una notificación de alerta al celular una sola vez para avisar sobre el sobrecalentamiento. El sistema mantendrá silencio hasta que la temperatura baje y vuelva a entrar en el rango de operación segura (58 °C a 62 °C), momento en el cual enviará un segundo aviso para informar que la situación se ha normalizado. Esta configuración de tres parámetros, detallada visualmente en la Figura 2, asegura un control industrial profesional y una comunicación eficiente con el usuario. Toda esta información se guarda en la base de datos MariaDB, que sirve como puente permanente entre el programa de Python (`main_controller.py`) y la página web en PHP (`dashboard.php`). Para que la comunicación sea exitosa, los archivos de Python `main_controller.py` y la librería `max6675.py` deben permanecer siempre en la misma carpeta, la cual se recomienda ubicar en la ruta personal del usuario para una ejecución sencilla.

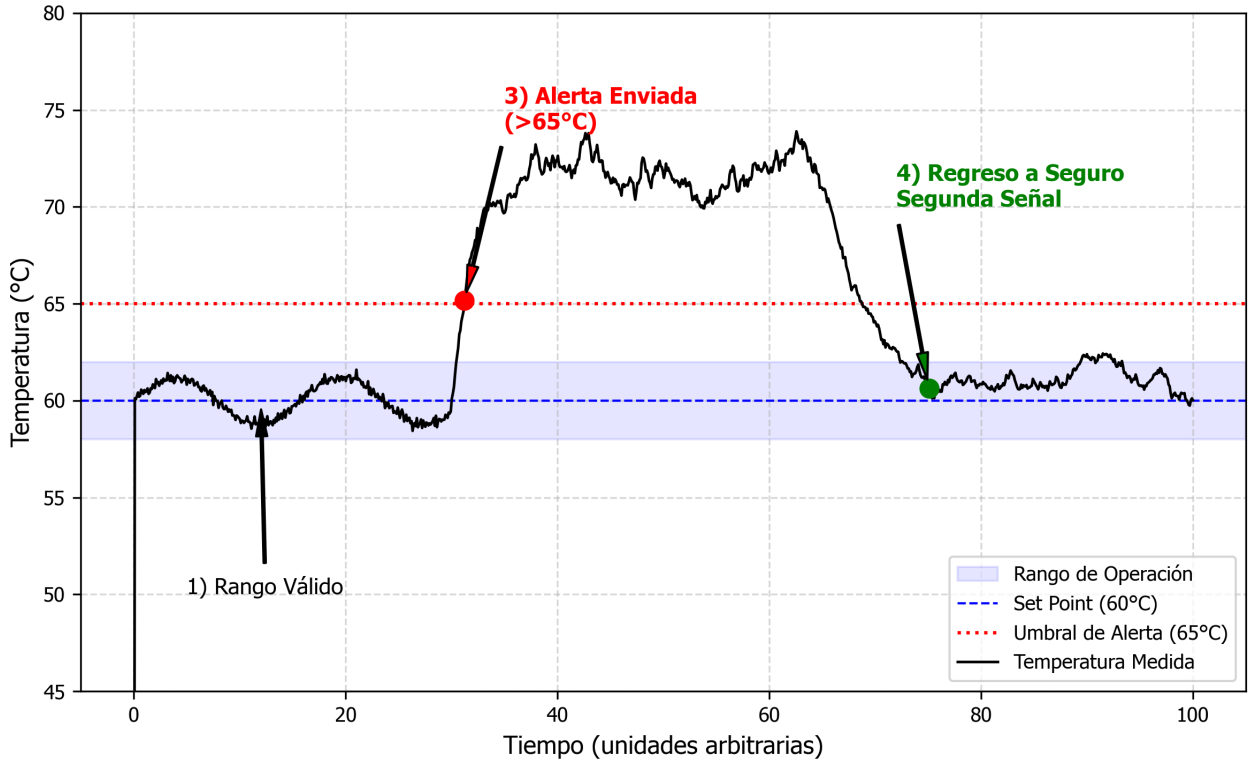

Figura 2: Dinámica de la lógica de control térmico e histéresis del sistema. Se ilustran los cuatro casos críticos: 1) operación en rango válido, 2) salida de umbral operativa, 3) cruce del umbral de alerta ($> 65°C$) con envío de notificación y 4) recuperación del sistema con envío de señal de normalización (Elaboración propia).

4. Seguridad y el Reto de los Teléfonos iPhone

Para que el «Monitor térmico» sea confiable, se utilizan las denominadas «llaves VAPID» [5]. Estas llaves son códigos secretos guardados en el archivo `config.php` (generado previamente a partir de su respectiva plantilla `.sample`) que permiten que el teléfono verifique que los avisos vienen del equipo autorizado. Es posible generar llaves propias usando herramientas en línea o comandos de terminal para asegurar que el sistema sea único y privado. Un requerimiento técnico fundamental para la seguridad es la generación de llaves VAPID únicas, las cuales actúan como la firma criptográfica exclusiva de las notificaciones. Para realizar este proceso paso a paso, se debe ejecutar en la terminal la orden `npx web-push generate-vapid-keys`. Una vez procesado, el sistema devolverá en pantalla dos bloques de texto: la «Public Key» y la «Private Key». La secuencia de configuración consiste en copiar el valor de la Public Key y pegarlo cuidadosamente dentro del archivo `sw.js` en la raíz de la web, así como en la variable correspondiente dentro de `dashboard.php`. Posteriormente, se debe tomar la Private Key y guardarla en el archivo de configuración `config.php`. Este procedimiento secuencial garantiza que solo el servidor autorizado pueda enviar mensajes a los teléfonos suscritos, impidiendo que terceros puedan suplantar la identidad del sistema. Además, los datos se protegen con cifrado de extremo a extremo, lo que significa que nadie puede leer la temperatura mientras viaja por internet [6]. Este nivel de seguridad es posible gracias a las librerías guardadas en la carpeta «vendor», las cuales permiten que el sistema cumpla con las reglas estrictas de los teléfonos modernos.

En el caso de los teléfonos iPhone, existe un reto especial debido a que Apple bloquea las notificaciones de sitios web automáticos para ahorrar batería [7]. Para que un iPhone permita recibir avisos del «Monitor térmico», el usuario debe realizar un proceso manual sencillo que se explica detalladamente en la opción de ayuda de la aplicación. Es fundamental que el usuario abra la dirección del sistema utilizando únicamente el navegador Safari, ya que otros exploradores como Chrome o Firefox no tienen permitido activar estas funciones en iOS. El proceso requiere tocar el botón de compartir (el icono del cuadrado con la flecha) y seleccionar la opción de agregar a la pantalla de inicio. Una vez que se abre la aplicación desde el nuevo icono creado, el usuario debe pulsar explícitamente una opción claramente identificada como «Activar Alertas» dentro de la configuración de la app para otorgar los permisos finales. Durante este proceso, el sistema detecta automáticamente si se trata de un equipo Android o iOS y guarda esta información en la tabla de suscripciones de MariaDB, lo que permite gestionar el envío de mensajes de forma más eficiente y personalizada.

5. Instalación del Sistema y Conexión a Internet

Para instalar el «Monitor térmico», primero se debe abrir la terminal de comandos en la Raspberry Pi. La secuencia de comandos comienza con la actualización del sistema mediante `sudo apt update`. Luego, se instalan los programas del servidor con el comando `sudo apt install apache2 mariadb-server php libapache2-mod-php php-mysql composer`. Para el control del hardware se requiere instalar las herramientas de Python con `sudo apt install python3-pip` y posteriormente asegurar la presencia de las li-

brerías de conexión mediante `pip install mysql-connector-python requests`. Finalmente, es obligatorio escribir `sudo raspi-config`, entrar a «Interfacing Options», seleccionar «SPI» y marcar «Yes» para activar la comunicación con el sensor. Antes de iniciar las pruebas del software, se recomienda confirmar que la comunicación física entre el procesador y el sensor está habilitada correctamente. Una vez realizada la activación en el menú de configuración del sistema y tras haber reiniciado el equipo, se debe ejecutar el comando `ls /dev/spi*` en la terminal. Si la activación fue exitosa, el sistema responderá mostrando nombres como `spidev0.0` o `spidev0.1`. En caso de no obtener respuesta, significa que el bus de datos no es visible para el sistema operativo y cualquier intento de correr el código de Python resultará en un error de hardware. Esta verificación sencilla permite descartar fallos de configuración básica antes de avanzar a etapas más complejas de programación.

Una vez que el servidor web ha sido instalado y la ruta base `/var/www/html` ha sido creada por el sistema, se debe proceder a la obtención y organización minuciosa de los archivos de código. Para asegurar que la estructura del sistema coincida exactamente con lo direccionado en el repositorio oficial, se debe abrir una terminal y ejecutar la descarga mediante la orden `git clone https://github.com/JavierSalasGarcia/notificaciones_-PWA_raspberry Repositorio`. En caso de optar por una descarga manual en formato ZIP, se debe utilizar el comando `unzip` para extraer el contenido. Una vez obtenida la carpeta `Repositorio`, es indispensable crear el directorio de destino para la interfaz web con el comando `sudo mkdir -p /var/www/html/monitor`. La correcta transferencia de los archivos se realiza moviendo únicamente el contenido de la subcarpeta `public` del repositorio hacia la nueva ruta del servidor mediante el comando `sudo cp -r Repositorio/public/* /var/www/html/monitor/`. De forma paralela, para que la lógica de control funcione sin errores de direccionamiento, la carpeta de Python debe ubicarse en la raíz del usuario mediante la orden `mv Repositorio/python /home/pi/`. Este procedimiento garantiza que tanto el servidor web como el controlador de hardware localicen sus archivos de configuración y datos en las rutas previstas, asegurando la integridad del sistema.

Para que la lógica de control funcione sin errores, se debe seguir la siguiente organización jerárquica de archivos:

```
/var/www/html/monitor/ (Interfaz Web)
  |-- config.php.sample (Plantilla de ajustes)
  |-- config.php (Archivo configurado)
  |-- dashboard.php (Panel de control)
  |-- vendor/ (Librerias externas)
  |-- sw.js (Service Worker)

/home/pi/python/ (Logica de Control)
  |-- main_controller.py.sample (Plantilla de logica)
  |-- main_controller.py (Cerebro configurado)
  |-- max6675.py (Libreria del sensor)
```

Esta separación es intencional para mantener el código de control fuera del alcance del servidor web público. Para evitar errores y facilitar la personalización, el repositorio no incluye archivos de configuración definitivos, sino plantillas denominadas «archivos sample». El primer paso para configurar la web es entrar a la carpeta del servidor y crear una copia

de la plantilla mediante el comando `cp config.php.sample config.php`. Una vez creada la copia, se debe abrir mediante el terminal utilizando el editor `nano`. Al ejecutar `sudo nano config.php`, se abrirá una interfaz con comentarios que guían al usuario sobre dónde capturar los datos de la base de datos y las llaves VAPID. Una vez realizados los cambios, la información se almacena presionando la combinación de teclas `Ctrl + O`, confirmando con la tecla `Entrar`, y se finaliza el proceso pulsando la combinación `Ctrl + X`.

Un paso obligatorio para habilitar las notificaciones es la creación de la carpeta «vendor» mediante el gestor de dependencias Composer. Debido a que el repositorio no incluye estas librerías por su gran volumen, el sistema web no funcionará hasta que se descarguen manualmente. Para realizar esta tarea, se debe utilizar la terminal para entrar al directorio del servidor escribiendo la orden `cd /var/www/html/monitor/`. Una vez dentro de la ruta correcta, se debe ejecutar la instrucción `composer install`. En caso de que el sistema devuelva un error de permisos, se debe anteponer la palabra `sudo` al comando anterior. Durante este proceso, se observará en la pantalla una lista de descargas; el programa lee automáticamente el archivo `composer.json` y obtiene las versiones exactas de las librerías de cifrado y protocolos push necesarias para comunicarse con los servidores de Google y Apple. Al finalizar, la carpeta «vendor» aparecerá en el directorio, permitiendo que las funciones de seguridad de la PWA se activen de inmediato.

El acceso continuo al «Monitor térmico», aunque cambie la dirección de internet de la casa, se logra mediante el DNS Dinámico. Normalmente, los proveedores de internet cambian el número de identificación (dirección IP) del módem cada pocos días. Si esto pasa, el usuario perdería la conexión con su Raspberry Pi desde fuera de casa. Para evitarlo, se utiliza un servicio que permite crear un nombre fijo como «mi-monitor.ddns.net». Se debe instalar el programa `ddclient` en la Raspberry con el comando `sudo apt install ddclient`. Este programa vigila constantemente si el número de IP cambió y, si lo hace, le avisa de inmediato al servidor de nombres para que el nombre fijo siempre dirija al usuario a su equipo de forma automática y transparente.

Para que el sistema sea visible en el navegador de cualquier dispositivo, se debe asegurar el correcto despliegue en el servidor web Apache. Tal como se detalló en el proceso de obtención de archivos anterior, la ubicación de los documentos en la ruta `/var/www/html/monitor/` es crítica para que Apache pueda servirlos correctamente. Esta organización permite que al escribir la dirección de la Raspberry seguida de la palabra «monitor», el navegador encuentre de inmediato el panel de control del sistema.

Un desafío técnico importante es que las notificaciones PWA exigen estrictamente el uso de conexiones seguras HTTPS, lo cual es difícil de configurar en la red de una casa. La diferencia fundamental reside en que HTTP (HyperText Transfer Protocol) envía la información en texto plano, lo que permite que sea interceptada, mientras que HTTPS (Protocolo Seguro de Transferencia de Hipertexto) utiliza certificados SSL/TLS para cifrar los datos del sensor y proteger la privacidad del usuario. Los navegadores modernos prohíben el uso de cámaras, ubicación y notificaciones push si no se detecta esta seguridad, a lo cual llaman «Contexto Seguro». Para solucionar este problema de forma profesional y gratuita, el «Monitor térmico» utiliza una herramienta llamada ngrok.

Esta utilidad fue seleccionada porque permite crear un túnel seguro desde internet hacia la Raspberry Pi sin necesidad de abrir puertos en el módem. Al activar la cuenta con el

comando `ngrok config add-authtoken` y ejecutar `ngrok http 80`, el sistema generará una dirección web especial que comienza con «https://». Es en este momento cuando el usuario podrá ver el candado verde en su navegador móvil; esto sucede porque ngrok actúa como un intermediario que ya posee los certificados de seguridad válidos que exigen los teléfonos Android e iPhone. Aunque existen otras herramientas como LocalTunnel o Cloudflare, se eligió ngrok por su gran estabilidad y facilidad de configuración para proyectos de desarrollo técnico.

El uso de ngrok en su modalidad gratuita conlleva una característica importante que debe comprenderse para mantener el acceso a largo plazo. Cada vez que el programa se detiene o se pierde la conexión a internet, la plataforma genera una dirección web completamente nueva y aleatoria. Esto implica que el enlace que se haya guardado o instalado en el teléfono dejará de funcionar de inmediato, mostrando un error de página no encontrada. Para recuperar el control, se debe observar la nueva dirección generada en la terminal de la Raspberry Pi y actualizarla manualmente tanto en los archivos de configuración del sistema como en el navegador del dispositivo móvil para volver a instalar el acceso directo.

Para escenarios donde se requiere disponibilidad permanente y profesional, existen alternativas que evitan el cambio constante de la dirección web. La opción más directa es utilizar una cuenta con nombre de dominio fijo, lo cual permite reservar una dirección que nunca cambiará a pesar de los reinicios del equipo. De igual manera, se pueden implementar soluciones como los túneles de Cloudflare, que operan de forma similar pero sobre un dominio propio registrado. Estas opciones transforman el proyecto de un prototipo local a una aplicación real capaz de operar durante meses sin intervención manual.

Para que la temperatura se mida de forma constante, se debe activar el programa cerebro. Siguiendo la misma lógica de orden, se debe entrar a la carpeta de Python y copiar la plantilla del controlador mediante `cp main_controller.py.sample main_controller.py`. Posteriormente, se debe editar mediante el comando `sudo nano main_controller.py`, asegurando que el nombre del servidor, usuario y clave coincidan plenamente con los de la interfaz web. Si esta información no es idéntica en ambos archivos, el sistema no podrá comunicar el estado real del sensor con el celular del usuario. Para asegurar que este control no se detenga por reinicios del equipo, se recomienda utilizar el programa de tareas automáticas crontab. Se debe abrir la configuración con `crontab -e` y añadir al final la línea `@reboot python3 /home/pi/python/main_controller.py`. Esta alternativa garantiza que el «Monitor térmico» inicie su vigilancia de forma autónoma.

6. Migración a Hosting Web y Escalabilidad

Para transformar el prototipo en una solución de nivel profesional, resulta conveniente migrar la arquitectura web hacia un servicio de hosting comercial. Esta transición ofrece ventajas significativas, como la obtención de una dirección IP fija y un nombre de dominio propio (por ejemplo, www.monitor-termico.com), eliminando la dependencia de herramientas de túnel temporal como ngrok y asegurando que las funciones de Aplicación Web Progresiva operen bajo certificados SSL/TLS nativos. El proceso de migración comienza con el respaldo de la información local. En la terminal de la Raspberry Pi, se debe ejecutar el comando de ex-

portación `mysqldump -u root -p pwa_monitor > respaldo.sql`; tras escribir la contraseña, se generará un archivo que contiene toda la estructura de tablas y los registros históricos. Este archivo debe descargarse a una computadora personal para su posterior carga en el servidor remoto.

Una vez obtenido el respaldo, se debe acceder al panel de control del hosting (comúnmente cPanel o Plesk) y buscar la sección dedicada a bases de datos MySQL. En esta interfaz, se tiene que crear una nueva base de datos y un usuario con una contraseña segura, asegurándose de asignar todos los privilegios al usuario sobre la base de datos recién creada. Posteriormente, se abre la herramienta phpMyAdmin del hosting, se selecciona la base de datos vacía en la barra lateral y se pulsa la pestaña superior de «Importar». Se debe elegir el archivo `respaldo.sql` y presionar el botón de ejecutar al final de la página. Si el proceso es correcto, el sistema mostrará un mensaje indicando que todas las tablas fueron creadas con éxito en la nube.

La transferencia del código se realiza mediante el administrador de archivos del hosting o un cliente SFTP como FileZilla. Una ventaja clave de esta arquitectura es su capacidad de escalabilidad; el sistema no requiere necesariamente ser instalado en la raíz del dominio principal. Es posible organizar múltiples unidades de monitoreo bajo un mismo dominio utilizando subdirectorios (por ejemplo, `dominio.com/sensor1/`) o subdominios específicos. Esto es posible gracias a que el alcance o «scope» del Service Worker se limita automáticamente a la carpeta donde se encuentra alojado, permitiendo que cada instancia opere de forma independiente sin interferir con otras instalaciones en el mismo servidor. Por lo tanto, una empresa puede gestionar una flota de dispositivos bajo una sola infraestructura web centralizada.

En este punto, es fundamental comprender que la mayoría de los servicios de hosting compartido no permiten ejecutar el comando Composer directamente en la nube. Por esta razón, se debe subir la carpeta «vendor» completa que se generó previamente en la Raspberry Pi hacia el directorio elegido en el servidor remoto, junto con el resto de los archivos y la carpeta `api`. Se debe verificar que todos los recursos de JavaScript y el archivo del Service Worker (`sw.js`) se ubiquen en la misma carpeta del proyecto para que el navegador pueda registrarlos con el alcance correcto. Inmediatamente después de la subida, es obligatorio editar el archivo `config.php` en el hosting para actualizar las credenciales de acceso a la base de datos, asegurando que la conexión apunte al nombre de servidor y usuario asignados por el proveedor.

Finalmente, la lógica de control en la Raspberry Pi debe reprogramarse para que el sistema hardware envíe la temperatura al nuevo servidor. Se debe editar el archivo `main_controller.py` (creado previamente a partir de la plantilla `.sample`) en la Raspberry mediante el comando `sudo nano /home/pi/python/main_controller.py` y localizar la línea que define la dirección del servidor (usualmente escrita como una variable URL). Se debe cambiar `http://localhost/monitor` por la dirección completa del dominio contratado, por ejemplo, `https://www.monitor-termico.com/monitor`, asegurando siempre el uso del protocolo seguro `https`. Una característica fundamental de este cambio es que ahora la Raspberry Pi actúa como un cliente que «empuja» los datos hacia internet, por lo que ya no importa si su dirección IP doméstica cambia constantemente. Para que el control de encendido y apagado siga funcionando, el script de Python realiza una consulta cada segundo a la base de datos del

hosting para verificar si el usuario ha presionado algún botón en la web, garantizando una sincronización perfecta a larga distancia sin configuraciones de red complejas en el módem.

7. Pruebas y Puesta en Marcha del Sistema

Para verificar que el «Monitor térmico» funciona correctamente, se debe realizar una secuencia de pruebas que comienza con la base de datos. Si se utiliza una herramienta visual como phpMyAdmin, el primer paso es entrar a la dirección local del servidor y crear una nueva base de datos llamada «pwa_monitor». Una vez creada, se debe usar la opción de importar para cargar el archivo `setup.sql` (ubicado originalmente en la carpeta `database` del repositorio descargado), el cual creará automáticamente todas las tablas necesarias. Es vital asegurarse de que los datos de acceso en el archivo `config.php` (validado tras su creación desde el archivo `.sample`) coincidan con el usuario y la contraseña configurados en el servidor MariaDB. Si se prefiere usar la línea de comandos para mayor seguridad, se puede crear el usuario y darle permisos escribiendo los comandos SQL `CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'contraseña';` y después `GRANT ALL PRIVILEGES ON pwa_monitor.* TO 'usuario'@'localhost';`. Por defecto, el sistema incluye un usuario administrador con el nombre «admin» y la contraseña «admin123». Por seguridad, se recomienda entrar a la nueva sección de gestión de usuarios ubicada en `users.php` para dar de alta a nuevos operadores y eliminar o cambiar la clave del administrador original. Desde esta interfaz, se pueden registrar múltiples cuentas, lo que permite que diferentes personas supervisen el sistema con sus propios accesos.

Una vez configurada la base de datos, se debe probar la conexión desde los dispositivos móviles (usando Safari en iOS). Al entrar por primera vez, el sistema detecta el dispositivo y permite la instalación en la pantalla de inicio, mostrándose una interfaz gráfica que irá guiando al usuario. Tras abrir la aplicación instalada, se debe pulsar el botón «Activar Alertas».

Durante este paso, es probable que el navegador pregunte si se desea permitir el envío de mensajes. Si por error se presiona el botón de bloquear, la aplicación dejará de recibir avisos. Para solucionar esto, se debe tocar el pequeño candado que aparece junto a la dirección web en el navegador móvil y seleccionar la opción de restablecer los permisos del sitio. Para verificar que el registro fue exitoso, se puede consultar la tabla «suscripciones» en phpMyAdmin; allí se deberá observar una nueva fila con los códigos de seguridad VAPID y el nombre del dispositivo detectado (Android o iOS). El sistema está diseñado para enviar una notificación a todos los dispositivos registrados al mismo tiempo, lo que permite que un mismo usuario reciba alertas en varios celulares o que un equipo de trabajo esté informado simultáneamente sobre cualquier sobrecalentamiento.

Es fundamental entender los riesgos de no gestionar correctamente las librerías con Composer. Si se intenta añadir una nueva función pesada y se olvida ejecutar el comando de instalación, el sistema experimentará fallos diversos. En un explorador web, esto se vería como una pantalla completamente blanca. Para diagnosticar este problema, se debe revisar el archivo de registro de errores en la ruta `/var/log/apache2/error.log`, donde se encontrará una descripción técnica del fallo, como la ausencia del archivo `autoload.php`. Para diagnosticar problemas visuales profundos, se puede presionar la tecla F12 en una computadora para

abrir las herramientas de desarrollo, donde en la pestaña llamada «Application» se podrá verificar si el Manifiesto y el Service Worker se cargaron sin errores. El «Monitor térmico» ha sido diseñado para funcionar de forma óptima con versiones de PHP 7.4 o superiores y con Python 3.x, por lo que es importante confirmar que se tienen instaladas estas ediciones para evitar fallos de compatibilidad en las funciones de cifrado. Cuando el proceso de instalación o la llegada de alertas no se comporta como se espera, se debe abrir el navegador en una computadora y pulsar la tecla F12. En la ventana que aparece, se debe acceder a la pestaña denominada «Application» y buscar en la barra lateral el menú de «Service Workers». Ahí se podrá confirmar visualmente si el archivo está en estado «Running» (en color verde). Si aparece cualquier advertencia en color rojo, indica un fallo en la estructura del código o en el certificado de seguridad que debe corregirse para que las funciones se activen.

8. Conclusiones

El desarrollo del «Monitor térmico» demuestra que es posible crear herramientas de supervisión industrial de alta calidad utilizando herramientas de código abierto y hardware accesible como la Raspberry Pi. El hallazgo principal de este trabajo radica en que la tecnología PWA ofrece una solución real para el envío de notificaciones proactivas en segundo plano, superando los bloqueos técnicos de los sistemas operativos móviles actuales que impiden que una web estándar entregue alertas si el usuario no mantiene la página abierta. Al integrar un Service Worker y el protocolo Web Push, se logra un sistema de vigilancia que informa al instante sobre eventos críticos sin obligar al desarrollador a someterse a las restricciones de las tiendas tradicionales de aplicaciones.

Uno de los hallazgos más importantes es que la seguridad no tiene por qué ser un obstáculo para la sencillez. Mediante el uso de llaves VAPID y el cifrado de datos, el sistema asegura que la información privada esté siempre protegida, cumpliendo con los estándares más exigentes de la web actual. Asimismo, el uso de herramientas de gestión como Composer y Pip facilita enormemente la instalación y el mantenimiento del software, permitiendo que el sistema sea fácil de replicar y actualizar. La solución propuesta para el problema de las direcciones de internet cambiantes asegura que el monitor esté siempre disponible, convirtiéndolo en una herramienta útil para el monitoreo de procesos críticos donde el sobrecalentamiento puede representar un riesgo serio.

Finalmente, se concluye que este enfoque de ingeniería no solo ahorra costos significativos, sino que también otorga una libertad total al desarrollador para adaptar el sistema a sus necesidades específicas. La metodología presentada permite escalar el proyecto desde un prototipo funcional basado en herramientas gratuitas, hasta un sistema de monitoreo de grado profesional alojado en la nube. La capacidad de controlar hardware desde un celular, medir variables en tiempo real y recibir alertas automáticas en cualquier parte del mundo abre un abanico de posibilidades para la automatización en talleres, laboratorios y hogares. El «Monitor térmico» se presenta como un ejemplo claro de cómo la unión de la electrónica moderna con los estándares web abiertos puede generar soluciones potentes, seguras y al alcance de todos para asegurar que los procesos importantes se mantengan siempre bajo control.

Direcciones de correo de los autores

Javier Salas-García: Universidad Autónoma del Estado de México jsalasg@uaemex.mx

Otniel Portillo-Rodríguez: Universidad Autónoma del Estado de México oportillo@uaemex.mx

Adriana H. Vilchis-González: Universidad Autónoma del Estado de México avilchisg@uaemex.mx

Referencias

[1] S. M. Gaikwad, V. Sharma, R. Verma y P. Jain, «Progressive Web App (PWA) - One Stop Solution for All Application Development Across All Platforms,» *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 5, n.º 2, 2019. DOI: 10.32628/cseit1952290

[2] A. Biørn-Hansen, T. A. Majchrzak y T. M. Grønli, «Progressive Web Apps: The Possible Web-native Unifier for Mobile Development,» en *Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST)*, 2017, págs. 344-351. DOI: 10.5220/0006353703440351

[3] W3C. «Push API,» visitado 31 de dic. de 2025. dirección: https://www.w3.org/TR/push-api/

[4] S. Sudasna y S. Inthachot, «The IoT Based Temperature Monitoring and Air Inlet Optimization Controlling for Gasification Stove,» en *2019 IEEE International Conference on Electrical, Computing and Communication Technologies (ICECCT)*, 2019, págs. 1-5. DOI: 10.1109/ECTI-NCON.2019.8692283

[5] M. Thomson y P. Beverloo. «Voluntary Application Server Identification (VAPID) for Web Push,» visitado 31 de dic. de 2025. dirección: https://datatracker.ietf.org/doc/html/rfc8292

[6] M. Thomson. «Message Encryption for Web Push,» visitado 31 de dic. de 2025. dirección: https://datatracker.ietf.org/doc/html/rfc8291

[7] WebKit Team. «Web Push for Web Apps on iOS and iPadOS,» visitado 31 de dic. de 2025. dirección: https://webkit.org/blog/13878/web-push-for-web-apps-on-ios-and-ipados/