

Desarrollo de un software para elevar la seguridad perimetral de un centro de datos basado en Raspberry Pi

Rússel Domínguez-Domínguez, Francisco J. Pérez-Porras, Luis Rodríguez-Martínez, Matthew S. Herrera-Silva, Michael E. Da Rocha Artur, Ignacio Cruz-Dominguez

Facultad de Ingeniería y Tecnología, Universidad de Montemorelos, Nuevo León, México.

rusbel@um.edu.mx, 1190201@alumno.um.edu.mx,
1190455@alumno.um.edu.mx, 1190518@alumno.um.edu.mx,
1140768@alumno.um.edu.mx,

ORCID: 0000-0002-4250-0826

1 Introducción

El monitoreo y control perimetral son componentes principales para garantizar la seguridad en los diversos sectores de la sociedad. La información es uno de los activos más importantes en las organizaciones. Es por eso que el desarrollo de un sistema que es capaz de alertar a los administradores y encargados de la seguridad, sobre los riesgos de las vulnerabilidades de un centro de datos es una tarea crucial que incrementa el nivel de seguridad en los activos de una institución [2], [7].

La necesidad de crear sistemas de seguridad personalizados es una practica cada vez más común, la cual requiere de características específicas del usuario y tiene una amplia posibilidad de desarrollo. El monitoreo y control de perimetral, específicamente en el aspecto relacionado con un centro de datos, es un espacio poco documentado al momento de redacción de este escrito. Actualmente, la seguridad perimetral en los centros de datos de la institución en la que se realiza esta investigación no se cuenta con la integración ni el desarrollo en la integración de tecnologías para el monitoreo, el control de acceso, y la seguridad perimetral.

Con la evolución de la ciencia y la tecnología, el monitoreo de las reacciones y actividades humanas se ha convertido en algo realmente sencillo. Las nuevas tecnologías basadas en el internet de las cosas tienen el potencial de revolucionar el área de la seguridad en diferentes ámbitos de la sociedad. Los varios dispositivos de monitoreo disponibles para la realización de sistemas de seguridad permiten el ahorro de recursos y mejoramiento de procesos [2], [3].

El proyecto por desarrollar tiene como objetivo incrementar el nivel de seguridad perimetral en los centros de datos de una institución educativa. Además,

como otro objetivo se tiene el incrementar el nivel de seguridad para la protección de datos. La implementación de sensores, actuadores y microcontroladores que permitan el desarrollo de una herramienta para la integración de tecnologías en el monitoreo, el control de acceso e incrementar el nivel de la seguridad perimetral de un centro de datos.

Definición del problema

En diversas instalaciones institucionales, se ha descubierto que ciertos procesos de seguridad rozan la obsolescencia, la vulnerabilidad o la falta de optimización, tales como la seguridad en las puertas al ser cerradas con llaves duplicables, el embotellamiento en casetas de entrada y salida de vehículos, y la detección de personas que entran y salen de instalaciones. Esto suele ser por razones económicas, falta de logística o negligencia humana. Económicamente hablando, el alto costo de los sistemas de seguridad perimetral aplaza la mejora de seguridad en pequeñas y medianas empresas, así como áreas institucionales o de desarrollo.

De igual forma, se ha encontrado que, en la institución educativa en estudio, no se cuenta con la implementación de medidas de seguridad perimetral adecuada para la protección y resguardo de un centro de datos, por tal motivo, investigadores de la misma institución en estudio desarrollaron un instrumento para medir el grado de un sistema de seguridad en la información basado en ISO/IEC 27001 en la cual analizaron los factores que inhiben la implementación de un sistema de gestión en seguridad de la información (SGSI), los cuales se dividen en cuatro factores: políticas y regulaciones, privacidad, integridad y autenticidad [4].

En este mismo sentido, en el año 2021 investigadores realizaron un estudio para evaluar el nivel de seguridad en la red inalámbrica de la institución en estudio, la cual permitía detectar fallas en la red exponiendo las vulnerabilidades. En esta investigación realizada, utilizaron una metodología basada en el hacking ético [5].

A finales del año 2021, en la institución en estudio se realizó una investigación en la alta gerencia, gerencia media, mandos medios y personal de TI. Este estudio consistía en conocer (a) el grado de efectividad de la administración para la seguridad de la información basada en la norma ISO/IEC 27001, (b) el grado de compromiso de los administradores, (c) el grado de capacitación del recurso humano y (d) el grado de conocimiento administrativo.

Encontrando que el personal administrativo de menor rango tiene mayor problema para involucrarse en la toma de decisiones para la implementación de un SGSI. Así como los mandos medios no están plenamente capacitados en temas de seguridad de información. Con estas falencias en el personal de la alta gerencia se toman decisiones que pueden generar brechas de seguridad [6].

Actualmente, en instituciones educativas se busca incrementar el nivel de seguridad de manera física y lógica en centros de datos mediante los sistemas de gestión en seguridad de información que permitan una mejora continua, el uso e implementación de buenas prácticas que benefician en el cuidado y protección de

datos personales en posesión de particulares [8]. Durante esta investigación se plantea poner en marcha una serie de pruebas experimentales y desarrollo tecnológico para así lograr tener los mejores resultados económicos.

No obstante, después de un análisis a la institución en estudio, se encontró que no se cuenta con una metodología para el monitoreo y detección de amenazas en la seguridad perimetral de los centros de datos existentes. Es así que con el desarrollo de este proyecto se busca minimizar las vulnerabilidades en seguridad perimetral de una institución privada de educación superior en el noreste de México.

2 Metodología

Para el desarrollo y el diseño general de este proyecto se basó en una metodología sugerida por el Dr. N. Surantha y W. Wicaksono en su investigación [7]. Esta investigación consta de tres fases: diseño general del sistema, diseño de hardware y diseño de software.

Diseño general del sistema

Escenarios posibles en el sistema: En esta fase se plantearon todos los escenarios posibles de vulnerabilidad que sufren las diferentes áreas analizadas, con el fin de obtener resultados óptimos al momento de utilizar componentes electrónicos.

Arquitectura del sistema: Esta fase consiste en analizar cada una de las áreas a las cuales se piensa aplicar el proyecto. Estas áreas involucran centros de datos, áreas concurridas, y zonas delicadas.

Adaptación a las áreas: Esta fase consiste en buscar que elementos electrónicos y eléctricos pueden ser adaptados en cada una de las áreas orientadas a la seguridad según estas las requieran

Implementación tecnológica: Consiste en hacer uso de herramientas y aparatos tecnológicos para lograr una sinergia entre lo electrónico analógico y lo digital.

Diseño de Hardware

Selección de componentes electrónicos: En esta fase se busca encontrar elementos óptimos para la realización de la investigación presente.

Integración de los componentes: Consiste en emplear elementos electrónicos como sensores y actuadores de manera eficiente y adecuada de acuerdo con las necesidades de las áreas, así como la interconexión entre ellos por medio de microcontroladores y microprocesadores.

Detección temprana de fallas en el hardware: En esta fase se buscan cada uno de los errores o fallas que puedan dar los sensores y actuadores y se determina si funcionan de manera óptima

Diseño de Software

Creación de herramientas para la organización del software: En esta fase se pretende hacer un Diagrama de caso de uso, así como un diagrama de flujo general del sistema para mejorar la comprensión de lo que se realizará en cada etapa del sistema y cómo se relaciona con las herramientas tecnológicas con las que se cuenta.

Desarrollo del software: En esta parte de la metodología descansa la creación de los programas que se utilizarán. Se implementan diferentes algoritmos y técnicas de programación adecuados a la problemática.

Detección temprana de fallas en el software: La detección de errores de sintaxis, de ejecución y lógicos son estudiados en esta etapa.

Componentes del sistema

Como parte del desarrollo se dividió la estructura del proyecto en distintos módulos, los cuales tienen una tarea específica e individual que a la vez se entrelaza con otro módulo para completar el sistema. Los módulos son los siguientes.

Módulo de lectura de sensores

Este módulo se encarga de recibir los datos de los sensores de humo y gas, sensores de movimiento, sensor de líquido, sensor de polvo, y sensor de humedad y temperatura. Mediante librerías de uso libre disponibles para Python, los sensores se comunican con el sistema enviando la información recabada para su almacenamiento y una vez esta información ha sido recibida, el programa exporta esta información al siguiente módulo para el manejo de los datos.

Módulo de procesamiento de datos

Este módulo se encarga de procesar los datos recibidos de los sensores almacenados en el módulo anterior. Esta parte del sistema analiza y verifica los datos recibidos comparándolos con métricas establecidas en el sistema, identificando así anomalías en el patrón de los datos. Se utilizan técnicas de análisis de datos y aprendizaje automático para detectar dichas anomalías y generar alertas. Una vez procesados los datos, el programa exporta los resultados al siguiente módulo para la evaluación de estos, el control de los actuadores.

Módulo de control de actuadores

Este módulo es el encargado de recibir la información procesada de los sensores y dependiendo de los resultados genera señales para los actuadores electrónicos como estrobos, sirenas, cerrojos magnéticos e iluminación. En este módulo se emplean bibliotecas para Python que interactúan con los actuadores para activar o desactivar los mismos.

Módulo de notificación

Este módulo es el encargado de generar y enviar notificaciones de alerta al administrador del sistema mediante una aplicación móvil. Para el funcionamiento de este módulo, el programa recibe, al igual que el módulo anterior, el resultado de los datos ya procesados y dependiendo de esto, generar las alertas. Estas alertas son entonces enviadas por el programa a una API web, la cual se comunica con la aplicación móvil para desplegar notificaciones de anomalías o información relevante para el administrador.

Herramientas

El lenguaje de programación seleccionado para el desarrollo es Python en su versión más reciente al momento de redacción de este documento (3.11.2). Python es un lenguaje de programación muy popular para desarrollar infraestructuras IoT por varias razones. Python es un lenguaje de alto nivel con una sintaxis sencilla y una amplia colección de bibliotecas y frameworks que facilitan el desarrollo de aplicaciones complejas. También es fácil de leer y escribir, lo que puede reducir el tiempo de desarrollo y mejorar la calidad del código. Debido al propósito del sistema y a la funcionalidad deseada, es necesario incluir herramientas de programación las cuales puedan aunarse a Python para añadir funcionalidades necesarias para el desarrollo [2].

Las siguientes bibliotecas y herramientas fueron utilizadas para el desarrollo del sistema:

RPi.GPIO

RPi.GPIO es una librería Python que permite a los desarrolladores acceder y controlar los pines GPIO (General Purpose Input/Output) de una Raspberry Pi. Los pines GPIO se utilizan para comunicarse con sensores externos, actuadores y otros dispositivos [10].

Adafruit.

La biblioteca DHT de Adafruit es una biblioteca Python que permite a los desarrolladores leer datos de la serie DHT de sensores de temperatura y humedad. Esta biblioteca se emplea para recopilar datos ambientales [11].

PySerial

PySerial es una librería Python que proporciona soporte para la comunicación serie entre un ordenador y dispositivos externos, incluyendo microcontroladores y sensores. Esta biblioteca puede utilizarse para comunicarse con dispositivos externos [12].

Flask

Flask es un framework web ligero para Python que puede utilizarse para crear aplicaciones IoT basadas en web. Flask ofrece a los desarrolladores herramientas para crear API RESTful, que pueden utilizarse para interactuar con dispositivos IoT y recopilar datos de sensores [13].

Mosquitto

Mosquitto es un broker de mensajería de código abierto que implementa el protocolo MQTT (MQ Telemetry Transport). MQTT es un protocolo de mensajería ligero diseñado para la transferencia eficaz de mensajes entre dispositivos con recursos informáticos y ancho de banda limitados [14].

En este proyecto, Mosquitto se utiliza para establecer una conexión entre el dispositivo Raspberry Pi y un broker local de Mosquitto

Firebase

Firebase Messaging Service es un servicio de mensajería basado en la nube proporcionado por Google como parte de su plataforma de desarrollo móvil Firebase. Permite a los desarrolladores enviar mensajes y notificaciones a sus usuarios a través de múltiples plataformas, incluyendo Android, iOS y web, a través de una API única. Firebase Messaging Service utiliza el protocolo Firebase Cloud Messaging (FCM), una solución de mensajería multiplataforma que permite la entrega fiable y eficiente de mensajes a las aplicaciones cliente [15].

One Signal

OneSignal es un servicio completo y potente de notificaciones push basado en la nube que proporciona a los desarrolladores un potente conjunto de herramientas para atraer a sus usuarios a través de múltiples plataformas, incluyendo web, móvil y escritorio [16].

Los materiales necesarios para el desarrollo del sistema fueron seleccionados teniendo como consideración el ahorro de recursos. Para el desarrollo del software en Python que se encarga de recibir y manejar la información dada por los sensores y actuadores, se utilizó un Raspberry Pi 4 modelo B el cual actúa como la unidad de procesamiento central del sistema. La Raspberry Pi es una opción popular para los sistemas IoT debido a su rentabilidad, bajo consumo de energía, pequeño tamaño, amplia gama de periféricos, compatibilidad con lenguajes de programación populares y fuerte apoyo de la comunidad [1]. Además de esto se utilizaron los siguientes sensores y actuadores:

- Sensores de humo y gas.
- Sensores de movimiento.

- Sensores de humedad y temperatura.

Actuadores electrónicos (estrobos, sirenas, cerrojo magnético, iluminación).

El desarrollo del hardware fue completado utilizando los sensores anteriormente mencionados conectados directamente a la Raspberry Pi. Para hacer posible el uso de varios sensores conectados en la tarjeta madre es necesario considerar los pines de conexión disponibles en la Raspberry Pi. La salida de voltaje de los pines es de 3.3 v y 5 v, pines tierra y pines tipo GPIO los cuales funcionan como entradas y salidas digitales; algunos incluso pueden ser utilizados como pines de comunicación con diferentes protocolos como puede ser SPI o I2C.

Una vez definidas las características de la placa, se conectó cada uno de los sensores y se hicieron pruebas individuales de funcionamiento. El sensor de humedad y temperatura (DHT11) fue el primero en conectarse. La salida de este sensor es de tipo digital y puede ser conectado directamente a la placa. Para este sensor se utilizó el pin GPIO 4.

Seguido de esto se conectó el sensor infrarrojo de movimiento de tipo PIR HC-SR50, dicho sensor necesita una alimentación de 5V y entrega una salida digital de 3.3V. De la misma manera el sensor puede ser conectado directamente a la placa mediante el pin GPIO 23.

Por último, se conectó el sensor de humo y gas MQ2, este está incluido en un módulo que da salida analógica y digital, para evitar el uso de protocolos de comunicación y reducir la cantidad de pines utilizados en la placa. Para esto se utilizó un amplificador operacional que compara la salida digital con un voltaje de 300mV, si el voltaje sobrepasa los 300 mV, manda un voltaje de 5V a un Logic Level Converter que reduce el voltaje a 3.3V para su uso adecuado como entrada hacia el pin GPIO 18 en la placa.

Tabla 1. Tabla de componentes electrónicos y sus conexiones a los puertos de la Raspberry Pi.

No.	Nombre	Descripción	Puerto GPIO
1	Buzzer	Actuador	GPIO 24
2	LED	Actuador	GPIO 27
3	Servomotor	Actuador	GPIO 12
4	Sensor de movimiento PIR	Sensor de movimiento pasivo	GPIO 23
5	Sensor DHT11	Sensor de detección de temperatura y humedad	GPIO 4
6	Potenciómetro	Divisor de tensión que da una voltaje de 300mV para comparar con el sensor MQ2	No aplica
7	Sensor MQ2	Sensor de detección de humo y gas	No aplica
8	OPAMP	Comparador entre el sensor MQ2 y el potenciómetro	
9	Logic Level Converter	Disminuye el voltaje del OPAMP a 3.3V para ser recibido por la Raspberry Pi 3b+	GPIO 18

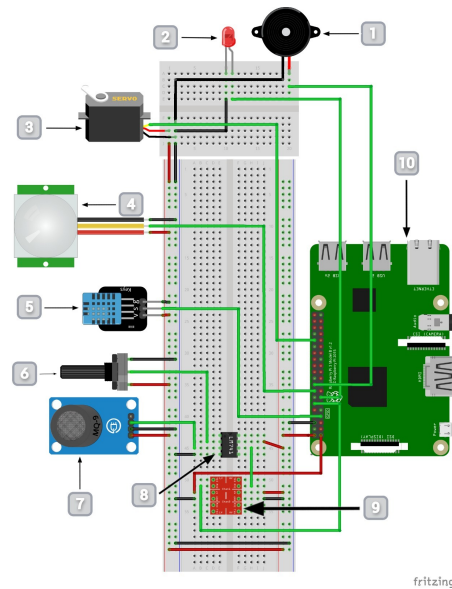


Figura 1. Diagrama de los componentes electrónicos y sus conexiones a los puertos de la Raspberry Pi.

3 Análisis de resultados

En esta sección, se analizará la implementación del software para el Incremento de Seguridad Física en un Centro de Datos Basado en Raspberry Pi y los resultados de su evaluación.

Pruebas de Sensores

Las pruebas de los sensores se realizaron con todos los sensores conectados en la configuración que se muestra en la Figura 1. Para cada uno se especificaron condiciones específicas para poder habilitar la repetibilidad de las pruebas.

Prueba de Sensor de Movimiento.

Las pruebas de detección de movimiento se realizaron dentro de un perímetro de 2 m x 3.5 m. El sensor de movimiento se eleva a una altura de 1.5 m y se coloca en una de las esquinas del perímetro. A continuación se muestran los resultados de las pruebas de detección de movimiento. Con los resultados de las siguientes pruebas, se puede asumir que si el sensor de movimiento se instala con una altura

de 1.5 m, entonces el sensor puede detectar con precisión hasta 3,9 metros por delante.

Tabla 2. Resultados de pruebas del sensor de movimiento.

Zona	A	B	C	D	E	F	G	H
Exitoso	5	5	5	5	5	5	5	5
Fallido	0	0	0	0	0	0	0	0

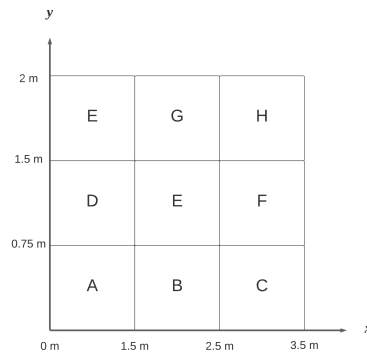


Figura 2. Zona de pruebas del sensor de movimiento.

Prueba de Sensor de Temperatura y Humedad

Para probar el sensor de temperatura y humedad se realizaron las pruebas en un ambiente controlado y al aire libre. En un cubículo de oficina, con un perímetro de 2 m x 3.5 m, se programó la temperatura para que el cubículo se mantuviera en un rango de 20 °C a 22 °C. Se ubicó el sensor en una de las esquinas del cubículo a una altura de 1.5 m y se registró la temperatura leída por el sensor. Una vez completada la lectura se retiraba el sensor de la habitación. Los resultados obtenidos de temperatura se muestran a continuación. Los resultados sugieren que el sensor de temperatura puede detectar cambios en la temperatura un rango de 3 °C.

Al mismo tiempo, se registró el porcentaje de humedad devuelto por el sensor en los distintos escenarios. Se compararon los resultados con los datos de humedad local brindados por The Weather Channel [9]. A continuación se muestran los valores devueltos por el sensor y el valor recolectado en la web.

Tabla 3. Resultados de pruebas de temperatura.

	Temperatura de Control	Lectura del Sensor
Resultado 1	22	23
Resultado 2	22	22
Resultado 3	22	20
Resultado 4	22	21

Resultado 5	22	20
-------------	----	----

Tabla 4. Resultados de pruebas de humedad.

	Humedad de Control	Ambiente controlado	Aire libre
Resultado 1	40 %	20%	38 %
Resultado 2	40 %	28%	38%
Resultado 3	40 %	28%	46%
Resultado 4	40 %	26%	46%
Resultado 5	40 %	26%	38%

Pruebas de Sensor de Humo y Gas.

El sensor de humo y gas se probó colocando el dispositivo en el centro de un cubículo de oficina de 2 m x 3.5 m, a una altura de 1.5 m sobre el suelo. La distancia máxima a la fuente en la que se detectó humo fue de 57 cm.

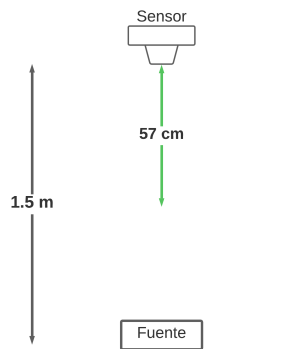


Figura 3. Prueba de sensor de humo y gas.

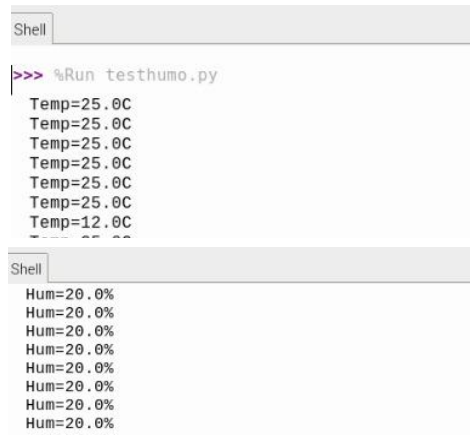
Pruebas de Software.

El correcto funcionamiento del software se dividió en dos fases, en la primera se probó la correcta comunicación entre los sensores y actuadores con el Raspberry Pi. En la segunda, se probó el envío de los datos a la API y el correcto procesamiento de los mismos para la comunicación con la aplicación Flutter.

Prueba de comunicación con sensores y actuadores

Se configuró el código de la aplicación para que registrará la lectura de los valores del sensor. En las siguientes figuras se muestran las impresiones a consola

con los valores de cada sensor. El sensor de temperatura y humedad registra los valores en grados centígrados de la temperatura y en porcentaje para la humedad. El sensor de movimiento registra en consola cada vez que detecta un valor positivo. Por último, el sensor de humo y gas registra a la consola cada vez que registra un valor positivo.



```
Shell
>>> %Run testhumo.py
Temp=25.0C
Temp=25.0C
Temp=25.0C
Temp=25.0C
Temp=25.0C
Temp=25.0C
Temp=25.0C
Temp=12.0C
...
Hum=20.0%
Hum=20.0%
Hum=20.0%
Hum=20.0%
Hum=20.0%
Hum=20.0%
Hum=20.0%
Hum=20.0%
```

Figura 3. Impresión a consola del sensor de temperatura y humedad.



```
Shell
>>> %Run testhumo.py
False
False
False
False
False
False
False
True
True
True
False
False
```

Figura 4. Impresión a consola del sensor de movimiento.



```
Shell
>>> %Run testhumo.py
Smoke or Gas detected
Smoke or Gas detected
```

Figura 5. Impresión a consola del sensor de humo y gas.

Prueba de comunicación con la API

La API es responsable de publicar y leer de varios temas de mensajes mqtt para recibir la información de los sensores y para comunicarse con los actuadores. Al mismo tiempo, se encarga de enviar notificaciones push mediante Firebase y One Signal a una aplicación Flutter. A continuación se presentan fragmentos del

código en la Raspberry Pi encargado de agregar mensajes a un tema mqtt y de la API en Fask encargada de procesar los datos.

```
def on_connectT(client, userdata, msg, rc):  
  
    print("connected with result code " + str(rc))  
    client.subscribe("topic/temp-humid")  
  
def on_connectH(client, userdata, msg, rc):  
  
    print("connected with result code " + str(rc))  
    client.subscribe("topic/humo")  
  
def on_connectM(client, userdata, msg, rc):  
  
    print("connected with result code " + str(rc))  
    client.subscribe("topic/mov")
```

Figura 6. Fragmento de código encargado de comunicar la API con el tema de MQTT.

```

@app.route('/prenderLED', methods = ['GET'])
def LED():
    client = mqtt.Client()
    client.connect('localhost', 1883, 60)
    json_body = {"id": 1}
    res = json.dumps(json_body)

    client.publish("topic/actuador", res)
    client.disconnect()

    return json_body

@app.route('/prenderBuzzer', methods = ['GET'])
def Buzzer():
    client = mqtt.Client()
    client.connect('localhost', 1883, 60)
    json_body = {"id": 0}
    res = json.dumps(json_body)

    client.publish("topic/actuador", res)
    client.disconnect()

    return json_body

@app.route('/prenderServo', methods = ['GET'])
def Servo():
    client = mqtt.Client()
    client.connect('localhost', 1883, 60)
    json_body = {"id": 2}
    res = json.dumps(json_body)

    client.publish("topic/actuador", res)
    client.disconnect()

    return json_body

```

Figura 7. Fragmento de código encargado de comunicar la API con los actuadores.

Prueba de Comunicación con la Aplicación Flutter

Para probar la funcionalidad de las notificaciones en el sistema, se utilizó una aplicación Flutter de demostración como plataforma receptora de notificaciones. En concreto, el sistema se configuró para enviar notificaciones a la aplicación cuando la API recibía un valor fuera de los parámetros establecidos de temperatura, humedad, movimiento, fuego o humo. A continuación, se muestra un ejemplo de la aplicación de Flutter y un fragmento del código que recibe las notificaciones.

```

5
6 void main() {
7   runApp(MyApp());
8
9   OneSignal.shared.setLogLevel(OSLogLevel.verbose, OSLogLevel.none);
10  OneSignal.shared.setAppId('ccc3e0ea-e6e8-44d1-93db-4287eb8507a8');
11  OneSignal.shared
12    .promptUserForPushNotificationPermission()
13    .then((accepted) => print('Accepted Permission: $accepted'));
14 }
15

```

Figura 8. Fragmento de código encargado de recibir notificaciones en la aplicación Flutter.

4 Discusión

La atención de este artículo se centra en la creación de un sistema de seguridad que alerte a los administradores y responsables de seguridad sobre los riesgos de las vulnerabilidades en áreas específicas como los centros de datos. Se hace hincapié en la necesidad de contar con sistemas de seguridad personalizables que puedan adaptarse a los requisitos específicos de los usuarios, y se analiza el potencial de las nuevas tecnologías basadas en el Internet de las Cosas (IoT).

El documento aborda el problema de la obsolescencia, vulnerabilidad o falta de optimización en determinados procesos de seguridad debido a razones económicas, falta de logística o negligencia humana. En concreto, se destaca la falta de medidas de seguridad perimetral adecuadas para proteger los centros de datos de la institución educativa objeto de estudio.

El objetivo de este estudio era evaluar el rendimiento de un sistema de seguridad para centros de datos basado en IoT y Raspberry Pi, centrándose específicamente en las pruebas de sensores y las pruebas de software.

Los resultados de las pruebas de los sensores indican que el sensor de movimiento puede detectar con precisión movimientos hasta 3,9 metros por delante, lo que sugiere que puede utilizarse eficazmente para detectar movimientos no autorizados en un centro de datos. El sensor de temperatura detectó cambios de temperatura en un intervalo de 3 °C, lo que resulta útil para controlar la temperatura en un centro de datos. Además, los resultados del sensor de humedad se compararon con los datos locales de humedad, lo que indica que el sensor puede medir con precisión los niveles de humedad.

Los resultados de este artículo sugieren que el sistema de seguridad de centros de datos basado en IoT y Raspberry Pi puede monitorear eficazmente el entorno físico de un centro de datos y prevenir posibles amenazas a la seguridad. Sin embargo, se necesita más investigación para evaluar el rendimiento del sistema en condiciones más complejas y realistas. El trabajo futuro podría centrarse en probar el sistema en un centro de datos más grande, con un conjunto más diverso de amenazas a la seguridad para evaluar su eficacia en un escenario del mundo real.

El sistema obtenido como resultado del proyecto añade, a trabajos realizados previamente por otros investigadores, una muestra exitosa del empleo de metodologías y tecnologías para la construcción de un sistema de seguridad perimetral. El uso de metodologías ágiles proporcionan herramientas útiles para el desarrollo de proyectos basados en IoT [17] [21].

El proyecto actual tiene como objetivo implementar un sistema de seguridad basado en IoT y Raspberry Pi. Para evaluar la eficacia de este sistema de seguridad, el proyecto se basa en estudios de investigación existentes como [18], [19] y [20], que han explorado la implementación de IoT y Raspberry Pi en diversos sistemas de seguridad. En particular, [18] se centró en la implementación de la biometría, [19] estudió el uso de la detección de intrusiones en IoT, y [20] propuso un sistema de seguridad doméstica contra la intrusión humana utilizando Raspberry Pi.

Basándose en los resultados presentados en estos estudios y en el proyecto actual, se puede concluir que los sistemas de seguridad basados en IoT y Raspberry Pi son una alternativa prometedora a las soluciones comerciales. La implementación de IoT y Raspberry Pi en sistemas de seguridad ha demostrado proporcionar soluciones fiables y rentables para diferentes tareas relacionadas con la seguridad. Por lo tanto, el presente proyecto se suma al conjunto de proyectos de sistemas de seguridad basados en IoT y Raspberry Pi y proporciona resultados que podrían sugerir la utilidad de los sistemas de seguridad basados en IoT y Raspberry Pi en aplicaciones prácticas.

5 Conclusión

Este artículo centra en el desarrollo de un sistema de seguridad personalizable para centros de datos, con especial énfasis en el uso de las tecnologías IoT y Raspberry Pi. El estudio pretende evaluar el rendimiento del sistema desarrollado probando los sensores y el software. Los resultados muestran que el sistema puede supervisar eficazmente el entorno físico de un centro de datos y prevenir posibles amenazas a la seguridad. Sin embargo, es necesario continuar investigando para evaluar el rendimiento del sistema en condiciones más complejas y realistas.

Los resultados del proyecto contribuyen al corpus de investigación sobre el desarrollo de sistemas de seguridad perimetral, con un ejemplo de éxito en el uso de metodologías ágiles y tecnologías IoT.

En general, se aportan pruebas de la utilidad potencial de los sistemas de seguridad basados en IoT y Raspberry Pi en aplicaciones prácticas y se sugiere que pueden ser una alternativa prometedora a las soluciones comerciales. El trabajo futuro podría centrarse en probar el sistema en entornos de centros de datos más amplios y diversos para seguir evaluando su eficacia y fiabilidad.

Referencias

- [1] R. Singh and S. Singh, "Smart border surveillance system using wireless sensor networks," *International Journal of System Assurance Engineering and Management*, vol. 13, no. S2, pp. 880–894, Jun. 2022, doi: 10.1007/s13198-021-01208-6.
- [2] Andreas, C. R. Aldawira, H. W. Putra, N. Hanafiah, S. Surjarwo, and A. Wibisurya, "Door Security System for Home Monitoring Based on ESP32,"

- Procedia Comput Sci, vol. 157, pp. 673–682, 2019, doi: 10.1016/j.procs.2019.08.218.
- [3] B. Lajevardi, K. R. Haapala, and J. F. Junker, “Real-time monitoring and evaluation of energy efficiency and thermal management of data centers,” *J Manuf Syst*, vol. 37, pp. 511–516, Oct. 2015, doi: 10.1016/j.jmsy.2014.06.008.
 - [4] R. Domínguez Domínguez, O. A. Flores Laguna, and J. A. Sánchez Valdez, “Exploratory analysis of a measurement scale of an information security management system,” in *2021 International Conference on Computer Science, Computer Engineering, E Applied Computing (CSCE)*, Jul. 2021.
 - [5] D. L. C. de La Cruz Paola and D. D. Rusbel, “Evaluation of the security level of Aruba wireless networks in an educational center,” in *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec. 2021, pp. 1370–1376. doi: 10.1109/CSCI54926.2021.00275.
 - [6] R. Domínguez Domínguez, O. A. Flores Laguna, and J. A. del Valle López, “Evaluation of an information security management system at a Mexican higher education institution,” in *2022 International Conference on Computational Science and Computational Intelligence (CSCI)*, Jul. 2022.
 - [7] N. Surantha and W. R. Wicaksono, “Design of Smart Home Security System using Object Recognition and PIR Sensor,” *Procedia Comput Sci*, vol. 135, pp. 465–472, 2018, doi: 10.1016/j.procs.2018.08.198.
 - [8] D. K. Sharma, K. D. Gupta, and R. Dwivedi, “Data Center Security,” in *Green Computing in Network Security*, New York: CRC Press, 2021, pp. 25–25. doi: 10.1201/9781003097198.
 - [9] The Weather Channel. [Online]. Available: <https://weather.com/>. [Accessed: May 7, 2023].
 - [10] M. Croston, “RPi.GPIO: A Python module to control the GPIO on a Raspberry Pi”, 2012. [Online]. Available: <https://pypi.org/project/RPi.GPIO/>. [Accessed: May 8, 2023].
 - [11] Adafruit, “Adafruit Python GPIO Library”, 2021. [Online]. Available: https://github.com/adafruit/Adafruit_Python_GPIO. [Accessed: May 8, 2023].
 - [12] M. Zeller, “PySerial: Python Serial Port Extension”, 2011. [Online]. Available: <https://pypi.org/project/pyserial/>. [Accessed: May 8, 2023].
 - [13] A. Grinberg, “Flask: A Python microframework for building web applications”, 2010. [Online]. Available: <https://flask.palletsprojects.com/en/2.1.x/>. [Accessed: May 8, 2023].
 - [14] A. Banks, “Mosquitto: An Open Source MQTT Broker”, 2010. [Online]. Available: <https://mosquitto.org/>. [Accessed: May 8, 2023].
 - [15] Google LLC, “Firebase - App success made simple”, 2021. [Online]. Available: <https://firebase.google.com/>. [Accessed: May 8, 2023].
 - [16] OneSignal Inc., “OneSignal - Unified Push Notifications, Email, SMS, Web Push, In-App Messaging”, 2021. [Online]. Available: <https://onesignal.com/>. [Accessed: May 8, 2023].

- [17] G. Islam and T. Storer, "A case study of agile software development for safety-critical systems projects," *Reliab Eng Syst Saf*, vol. 200, p. 106954, Aug. 2020, doi: 10.1016/j.ress.2020.106954.
- [18] D. Shah and V. haradi, "IoT Based Biometrics Implementation on Raspberry Pi," *Procedia Comput Sci*, vol. 79, pp. 328–336, 2016, doi: 10.1016/j.procs.2016.03.043.
- [19] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *Journal of Network and Computer Applications*, vol. 84. Academic Press, pp. 25–37, Apr. 15, 2017. doi: 10.1016/j.jnca.2017.02.009.
- [20] R. A. Nadafa, S. M. Hatturea, V. M. Bonala, and S. P. Naikb, "Home Security against Human Intrusion using Raspberry Pi," *Procedia Comput Sci*, vol. 167, pp. 1811–1820, 2020, doi: 10.1016/j.procs.2020.03.200.
- [21] A. Srivastava, D. Mehrotra, P. K. Kapur, and A. G. Aggarwal, "Analytical evaluation of agile success factors influencing quality in software industry," *International Journal of System Assurance Engineering and Management*, vol. 11, pp. 247–257, Jul. 2020, doi: 10.1007/s13198-020-00966-z.