



Miguel Jaime Adrián David
Hernández Razo Juan Daniel

6CM3

Prof. Alemán Arce Miguel Ángel

05/01/2026

PROYECTO FINAL

***Sistema de monitoreo remoto de salud para
personas de la tercera edad***

Internet de las cosas / Embedded systems

Escuela Superior de Cómputo
Instituto Politécnico Nacional

CONTENIDO

1	Problema detectado.....	3
2	Planteamiento del problema.....	3
3	Objetivos	4
3.1	Objetivo general.....	4
3.2	Objetivos específicos	4
4	Justificación del proyecto	5
5	Marco teórico	5
5.1	Domótica aplicada a la salud.....	5
5.2	Internet de las Cosas (IoT)	6
5.3	Sensores biomédicos	6
5.4	Microcontrolador ESP32.....	6
5.5	Protocolos de comunicación IoT	6
5.6	Plataformas en la nube y mensajería instantánea.....	7
6	Plan de acción.....	7
	Fase 1. Análisis y Diseño	7
	Fase 2. Desarrollo de Prototipo de Hardware	7
	Fase 3. Plataforma en la Nube y Backend	8
	Fase 4. Desarrollo de Aplicación de Usuario.....	8
	Fase 5. Pruebas Piloto	9
	Fase 6. Optimización y Escalabilidad.....	9
7	Desarrollo y fases	10
8	Resultados y pruebas	16
9	Conclusiones	19
10	Anexos.....	20
10.1	Código usado en el ESP32	20
10.2	Script de Python.....	22
10.3	Configuración del Broker.....	25
10.4	Visualización del bot y alertas en Telegram.....	26
10.5	Documento de Excel generado	27

1 PROBLEMA DETECTADO

En los últimos años, el mundo ha enfrentado diversas emergencias sanitarias de gran impacto, entre las que destaca la pandemia de COVID-19, la cual evidenció importantes deficiencias en los sistemas de salud a nivel global. Durante este periodo, muchos servicios médicos se vieron saturados, lo que derivó en retrasos en la detección y atención de enfermedades, así como en una respuesta sanitaria ineficiente que contribuyó al aumento en la tasa de mortalidad. Esta situación puso de manifiesto la fragilidad de los modelos tradicionales de atención médica, especialmente en el primer nivel de atención.

La atención primaria de la salud representa el primer punto de contacto entre las personas y el sistema sanitario, y su función es brindar una atención integral, accesible y continua a lo largo de la vida. Un sistema de salud con una atención primaria sólida es fundamental para garantizar la cobertura sanitaria universal; sin embargo, en muchos países, particularmente aquellos de ingresos bajos y medios, la infraestructura, los recursos humanos y tecnológicos resultan insuficientes para satisfacer la demanda existente.

Ante este panorama, el uso de tecnologías emergentes ha cobrado relevancia como una alternativa viable para fortalecer los servicios de salud. En particular, los sistemas de monitoreo remoto de la salud han surgido como una solución tecnológica avanzada, capaz de recopilar datos biomédicos en tiempo real sobre los parámetros fisiológicos de una persona. Dichos sistemas permiten el seguimiento continuo de variables como la frecuencia cardíaca, la saturación de oxígeno en la sangre, la presión arterial y la temperatura corporal, facilitando la detección temprana de anomalías y la toma de decisiones oportunas por parte del personal médico.

En este contexto, el Internet de las Cosas (IoT) se ha convertido en un componente clave para el desarrollo de sistemas de monitoreo de salud, mediante la integración de sensores biomédicos, dispositivos inteligentes y plataformas en la nube. A través de esta tecnología, es posible adquirir, procesar y transmitir información médica de forma remota, permitiendo que los datos sean analizados por médicos o instituciones de salud, y compartidos con familiares o cuidadores.

El presente proyecto se enfoca en el desarrollo de un sistema domótico basado en IoT para el monitoreo remoto de signos vitales en adultos mayores, utilizando sensores biomédicos y tecnologías de comunicación inalámbrica. La solución propuesta busca contribuir a la mejora de la atención primaria de la salud, reducir la carga sobre los sistemas hospitalarios y ofrecer una herramienta accesible que permita el seguimiento continuo del estado de salud de personas en situación de vulnerabilidad.

2 PLANTEAMIENTO DEL PROBLEMA

El cuidado de la salud en adultos mayores representa uno de los principales retos para los sistemas sanitarios actuales, especialmente en contextos donde la atención primaria resulta insuficiente o limitada por la falta de recursos humanos, infraestructura médica y acceso oportuno a servicios especializados. Las personas de la tercera edad suelen presentar enfermedades crónicas o condiciones que requieren un monitoreo constante de signos vitales, tales como la frecuencia cardíaca y la saturación de oxígeno en la sangre, variables fundamentales para la detección temprana de complicaciones médicas.

En muchos casos, el seguimiento continuo de estos parámetros depende de visitas presenciales a centros de salud o del uso de equipos médicos especializados, los cuales pueden resultar costosos, poco accesibles o difíciles de operar para los propios pacientes y sus cuidadores. Esta situación se agrava cuando los adultos mayores viven solos o cuentan con una supervisión limitada, lo que incrementa el riesgo de que eventos críticos no sean detectados a tiempo.

Si bien existen soluciones tecnológicas avanzadas para el monitoreo de la salud, estas suelen estar orientadas a entornos hospitalarios o requieren infraestructuras complejas que no siempre son viables en el hogar. Por ello, se identifica la necesidad de desarrollar un sistema de monitoreo remoto que sea accesible, de bajo costo y capaz de operar de forma continua, permitiendo la adquisición, transmisión y análisis de datos biomédicos en tiempo real.

En este contexto, el uso de tecnologías de Internet de las Cosas (IoT) ofrece una alternativa eficiente para integrar sensores biomédicos, dispositivos inteligentes y plataformas en la nube, facilitando el monitoreo remoto de signos vitales y la notificación oportuna de eventos anómalos a cuidadores, familiares o personal médico. No obstante, aún existe una brecha en la implementación de sistemas IoT enfocados específicamente en el cuidado de adultos mayores, que combinen simplicidad, confiabilidad y capacidades de alerta en tiempo real.

3 OBJETIVOS

3.1 OBJETIVO GENERAL

Desarrollar un sistema de monitoreo remoto de signos vitales basado en tecnologías de Internet de las Cosas (IoT), orientado al cuidado de adultos mayores, que permita la adquisición, transmisión y registro en tiempo real de variables fisiológicas, con el fin de facilitar la detección temprana de anomalías y mejorar la calidad de vida de los usuarios.

3.2 OBJETIVOS ESPECÍFICOS

- Monitorear variables fisiológicas clave, específicamente la frecuencia cardíaca y la saturación de oxígeno en la sangre, mediante el uso de sensores biomédicos.
- Implementar una plataforma de procesamiento local utilizando un microcontrolador con conectividad inalámbrica para la recolección y envío de datos.
- Transmitir de manera inalámbrica la información obtenida hacia una plataforma en la nube utilizando protocolos de comunicación IoT.
- Almacenar los datos recolectados para su posterior análisis y generación de reportes históricos.
- Desarrollar un sistema de notificación automática que envíe alertas en tiempo real a través de aplicaciones de mensajería, como Telegram, ante la detección de valores fuera de rangos normales.
- Facilitar el acceso a la información de salud a cuidadores y familiares, permitiendo la consulta del estado de salud en tiempo real y el seguimiento histórico de los registros.
- Proporcionar una solución tecnológica de bajo costo, no invasiva y escalable, adecuada para entornos domésticos.

4 JUSTIFICACIÓN DEL PROYECTO

El incremento en la población adulta mayor y la alta prevalencia de enfermedades crónicas asociadas a la edad hacen indispensable el desarrollo de soluciones tecnológicas que permitan un monitoreo continuo y oportuno del estado de salud. La detección tardía de alteraciones en los signos vitales puede derivar en complicaciones graves que, en muchos casos, podrían prevenirse mediante un seguimiento constante y una atención temprana.

La atención médica tradicional presenta diversas limitaciones, como la dependencia de visitas presenciales, la saturación de los servicios de salud y la falta de recursos en determinados contextos socioeconómicos. Estas condiciones dificultan el monitoreo permanente de los pacientes y aumentan la carga tanto para los sistemas sanitarios como para los cuidadores y familiares.

En este sentido, el presente proyecto se justifica por la necesidad de contar con un sistema de monitoreo remoto de signos vitales que sea accesible, confiable y fácil de implementar en el hogar. La utilización de tecnologías de Internet de las Cosas (IoT) permite integrar sensores biomédicos de bajo costo con dispositivos de procesamiento y plataformas en la nube, posibilitando la adquisición y transmisión de información médica en tiempo real.

Asimismo, la integración de un sistema de alertas automáticas mediante aplicaciones de mensajería instantánea, como Telegram, ofrece un medio eficaz para notificar de manera inmediata a familiares o cuidadores ante la detección de valores anómalos, contribuyendo a una respuesta rápida ante posibles emergencias. De esta manera, el proyecto no solo busca mejorar la calidad de vida de los adultos mayores, sino también fortalecer el rol de los cuidadores y apoyar la toma de decisiones médicas basadas en datos.

Finalmente, este sistema sienta las bases para futuras mejoras, como la incorporación de análisis predictivo o algoritmos de aprendizaje automático, lo que incrementa su potencial de escalabilidad y su impacto en el ámbito de la salud digital.

5 MARCO TEÓRICO

5.1 DOMÓTICA APLICADA A LA SALUD

La domótica se define como el conjunto de tecnologías orientadas a la automatización y control inteligente de entornos habitacionales. En el ámbito de la salud, la domótica permite el desarrollo de sistemas de asistencia que facilitan la supervisión del estado físico de los usuarios, especialmente en adultos mayores o personas con movilidad reducida. Estas soluciones contribuyen a mejorar la seguridad, autonomía y calidad de vida mediante el uso de dispositivos inteligentes integrados en el hogar.

5.2 INTERNET DE LAS COSAS (IoT)

El Internet de las Cosas (IoT) es un paradigma tecnológico que permite la interconexión de objetos físicos capaces de recopilar, procesar y transmitir datos a través de redes de comunicación. En aplicaciones de salud, el IoT posibilita el monitoreo remoto de parámetros fisiológicos mediante sensores biomédicos, permitiendo el acceso a la información en tiempo real desde cualquier ubicación con conexión a Internet.

Los sistemas IoT se caracterizan por su escalabilidad, bajo consumo energético y capacidad de integración con plataformas en la nube, lo que los convierte en una solución adecuada para el monitoreo continuo de la salud.

5.3 SENSORES BIOMÉDICOS

Los sensores biomédicos son dispositivos diseñados para medir variables fisiológicas del cuerpo humano. En este proyecto se emplea un sensor óptico para la medición de la frecuencia cardíaca y la saturación de oxígeno en la sangre (SpO_2), variables fundamentales para evaluar el estado cardiovascular y respiratorio de una persona.

El sensor MAX30102 utiliza técnicas de fotopletismografía, basadas en la emisión y detección de luz infrarroja y roja, para estimar los niveles de oxigenación y el pulso cardíaco. Su bajo consumo de energía y tamaño compacto lo hacen adecuado para aplicaciones portátiles y sistemas IoT.

5.4 MICROCONTROLADOR ESP32

El ESP32 es un microcontrolador de bajo consumo que integra conectividad WiFi y Bluetooth, lo que lo convierte en una plataforma ideal para aplicaciones IoT. Su capacidad de procesamiento permite la adquisición de datos desde sensores externos, el preprocesamiento de la información y su posterior transmisión hacia servidores remotos.

Además, el ESP32 soporta múltiples protocolos de comunicación, lo que facilita su integración con plataformas en la nube y sistemas de mensajería.

5.5 PROTOCOLOS DE COMUNICACIÓN IoT

Los protocolos de comunicación IoT permiten el intercambio eficiente de información entre dispositivos y servidores. En este proyecto se utiliza el protocolo MQTT (Message Queueing Telemetry Transport), el cual se basa en un modelo publicador/suscriptor y está diseñado para aplicaciones que requieren bajo consumo de ancho de banda y alta confiabilidad.

MQTT resulta especialmente adecuado para el envío de datos biomédicos en tiempo real, ya que permite una comunicación eficiente incluso en redes con conectividad limitada.

5.6 PLATAFORMAS EN LA NUBE Y MENSAJERÍA INSTANTÁNEA

Las plataformas en la nube proporcionan servicios de almacenamiento y procesamiento de datos accesibles desde cualquier ubicación. En este proyecto, los datos enviados por el ESP32 son almacenados para su análisis y generación de reportes históricos.

Por otro lado, las aplicaciones de mensajería instantánea, como Telegram, permiten la implementación de bots capaces de recibir información, generar reportes automáticos y enviar alertas en tiempo real. Esta integración mejora la comunicación entre el sistema y los usuarios finales, facilitando el seguimiento del estado de salud del adulto mayor.

6 PLAN DE ACCIÓN

FASE 1. ANÁLISIS Y DISEÑO

Objetivo: Definir requerimientos técnicos, variables a medir y la arquitectura IoT.

Esta fase constituye los cimientos del sistema; un error en la definición de requerimientos aquí puede llevar a fallos de seguridad o lecturas inexactas en etapas posteriores. Es importante establecer con precisión las variables fisiológicas (SpO_2 y BPM) y la arquitectura IoT, ya que de esto depende la selección de hardware capaz de procesar señales biomédicas en tiempo real. La elección del ESP32 sobre otras placas se justifica debido a su conectividad WiFi nativa y bajo consumo, ideal para dispositivos vestibles de monitoreo continuo.

Actividades:

- Levantamiento de requerimientos.
- Selección de variables fisiológicas y ambientales prioritarias (SpO_2 , BP,).
- Elección de sensores, microcontroladores y servicios en la nube adecuados.
- Diseño de la arquitectura IoT (diagrama de flujo: sensores → ESP32 → nube → app).
- Elaboración de prototipo de interfaz para cuidadores/familiares.

FASE 2. DESARROLLO DE PROTOTIPO DE HARDWARE

Objetivo: Construir el dispositivo de monitoreo inicial.

En esta fase transformara el diseño teórico en un prototipo físico funcional, donde la capturar de manera correcta la información que proporciona el sensor MAX30102 es clave para las siguientes fases. Esta fase también incluye la correcta implementación del protocolo I2C y la calibración del buffer de datos en el ESP32 son vitales para filtrar el ruido ambiental y los

movimientos del usuario, los cuales suelen causar falsos positivos en las lecturas de salud que hará el sensor MAX30102.

Asimismo, se empieza la búsqueda de la conexión con un bróker para implementar el protocolo MQTT para asegurar que los datos fluyan hacia la nube de manera ligera y constante.

Actividades:

- Adquisición de sensores (MAX30102).
- Integración con microcontrolador ESP32.
- Desarrollo de firmware básico para lectura de sensores.
- Envío de datos de prueba mediante MQTT/HTTP a la nube.

FASE 3. PLATAFORMA EN LA NUBE Y BACKEND

Objetivo: Configurar el sistema de almacenamiento y análisis de datos.

La configuración de un backend sólido es lo que permite que el proyecto funcione de forma autónoma, garantizando que la información del paciente sea procesada, analizada y almacenada sin intervención humana constante. Sin esta etapa, el proyecto sería un simple visualizador local.

El desarrollo de la base de datos histórica mediante archivos csv asegura que no se pierda ninguna métrica, permitiendo que el personal médico realice auditorías de salud retrospectivas. Por otro lado, la implementación de reglas de negocio para la detección de anomalías permite diferenciar un estado de reposo normal de una emergencia real (como taquicardia o hipoxia). Finalmente, la programación de las alertas garantiza que la comunicación con el usuario final sea oportuna y de fácil acceso a cuidadores y familiares

Actividades:

- Desarrollo de base de datos para almacenamiento histórico.
- Implementación de reglas de negocio: detección de valores fuera de rango.
- Programación de alertas automáticas (SMS/email/notificaciones push).

FASE 4. DESARROLLO DE APLICACIÓN DE USUARIO

Objetivo: Proveer acceso amigable a cuidadores y familiares.

Esta fase se prepara lo que será la entrega final del proyecto, donde el planteamiento que se realizaron en las fases anteriores se implementa y se corrigen para asegurar una experiencia de usuario simplificada y accionable. Teniendo en cuenta que a nivel salud el proyecto y sistema que se desarrolla no sea solo una herramienta de recolección de datos, sino un canal de comunicación efectivo entre el paciente y su entorno de apoyo.

La implementación de un bot de Telegram como interfaz principal permite aprovechar una plataforma que los usuarios ya conocen, eliminando la curva de aprendizaje y facilitando la recepción de reportes periódicos. Esta fase asegura que la solución técnica sea adoptada con éxito por las familias, cumpliendo con el impacto social de brindar tranquilidad y vigilancia constante.

Actividades:

- Implementación de bot para generar reportes semanales/mensuales.
- Pruebas de usabilidad con usuarios finales.
- Configuración de notificaciones específicas para diferenciar una lectura normal de una alerta crítica.

FASE 5. PRUEBAS PILOTO

Objetivo: Validar el sistema en un entorno real con usuarios.

Esta fase es determinante para garantizar el sistema en condiciones de uso cotidiano. Las pruebas piloto permiten identificar variables que no son evidentes en un entorno controlado, como la estabilidad de la conexión WiFi en diferentes áreas del hogar o la comodidad del sensor durante periodos prolongados de uso.

El monitoreo durante un periodo de 2 a 3 semanas es crucial para validar la confiabilidad del backend y el sistema de almacenamiento en Excel (CSV), asegurando que el registro de datos sea ininterrumpido y útil para una revisión médica posterior. Además, la evaluación de usabilidad permite confirmar si las alertas de Telegram realmente cumplen su propósito de avisar a los cuidadores de forma oportuna y comprensible ante una emergencia.

Actividades:

- Selección de grupo reducido de adultos mayores para pruebas.
- Monitoreo durante un periodo definido (2–3 semanas).
- Evaluación de usabilidad y confiabilidad del sistema.

FASE 6. OPTIMIZACIÓN Y ESCALABILIDAD

Objetivo: Ajustar el sistema para mayor robustez y crecimiento.

Esta fase final tiene como objetivo elevar los estándares de calidad del sistema, garantizando su sostenibilidad técnica y su capacidad de crecimiento. La optimización es crítica para dispositivos de salud vestibles, donde la eficiencia energética determina si el usuario puede

mantener el monitoreo durante todo el día o si el dispositivo se vuelve una carga por recargas constantes. Al mismo tiempo, preparar el sistema para la escalabilidad asegura que el backend en Python y el Broker MQTT puedan gestionar múltiples pacientes simultáneamente sin comprometer la velocidad de las alertas de emergencia.

La implementación de algoritmos de análisis predictivo permite que el sistema no solo reaccione a un evento presente, sino que identifique patrones de riesgo antes de que se conviertan en una crisis, mejorando drásticamente la precisión del monitoreo.

Actividades:

- Optimización de consumo energético en el dispositivo (para uso continuo).
- Mejora en la precisión de detección de anomalías con algoritmos de análisis predictivo.
- Documentación técnica y manual de usuario.
- Preparación para escalamiento a mayor número de usuarios.

7 DESARROLLO Y FASES

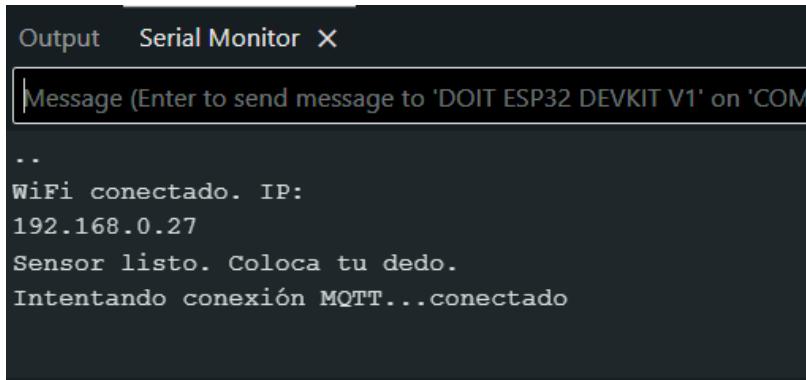
Fase 1: Captura de Datos y Procesamiento Local (Edge Computing)

El desarrollo del sistema comenzó con la implementación de la capa física y de procesamiento primario. En esta etapa, el sensor biomédico MAX30102 se configuró para interactuar con el microcontrolador ESP32 a través del protocolo de comunicación industrial I2C, lo que permitió una transferencia de datos síncrona y eficiente entre ambos dispositivos. La prioridad técnica en este punto fue garantizar que la señal capturada del dedo del paciente fuera lo más limpia posible antes de ser enviada a la red.

Para lograr esta limpieza de datos, se desarrolló e implementó un algoritmo de filtrado digital directamente en el firmware del ESP32. Utilizando un buffer circular (RATE_SIZE), el sistema realiza un promedio móvil de las lecturas de pulsaciones por minuto (BPM), descartando automáticamente valores atípicos o ruidosos causados por el movimiento (como lecturas inferiores a 50 o superiores a 120 BPM en reposo). Auxiliado por la IDE de Arduino fue lo que permitió la conexión con el ESP32, tanto para una configuración inicial y que se logrará una comunicación con la computadora y posteriormente se logrará recibir y visualizar los datos provenientes del sensor MAX30102. Este es un procesamiento local y es la primera fase de los sistemas IoT conocido como Edge Computing, fundamental ya que reduce la carga de trabajo en la nube y asegura que solo la información estable y verificada sea procesada para las alertas.

Finalmente, ya que se logró el Edge Computing y una vez procesada la información, el ESP32 se configuró como un cliente WiFi activo. Utilizando la librería PubSubClient, se programó el empaquetado de las variables de salud (BPM y SpO2) en una estructura de datos ligera tipo JSON. Esta elección de formato facilita la interoperabilidad del sistema, permitiendo que cualquier plataforma moderna pueda interpretar los signos vitales del usuario de manera

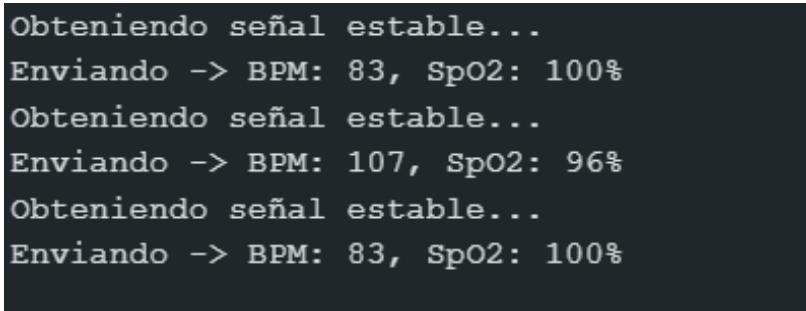
estandarizada, de este modo ya se prepara la información a la nube mediante protocolos como el MQTT que fue seleccionado en este proyecto y dar el salto a lo que vuelve el sistema parte del IoT.



The screenshot shows the Arduino Serial Monitor window. The title bar says "Output" and "Serial Monitor X". The main area displays the following text:

```
Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM
..
WiFi conectado. IP:
192.168.0.27
Sensor listo. Coloca tu dedo.
Intentando conexión MQTT...conectado
```

Ilustración 1- Serial Monitor de IDE Arduino mostrando conexión en ESP32



The screenshot shows the Arduino Serial Monitor window displaying transmitted data:

```
Obteniendo señal estable...
Enviando -> BPM: 83, SpO2: 100%
Obteniendo señal estable...
Enviando -> BPM: 107, SpO2: 96%
Obteniendo señal estable...
Enviando -> BPM: 83, SpO2: 100%
```

Ilustración 2- Obtención de valores cardiacos a través del ESP32

Fase 2: Protocolo de Comunicación y Broker en la Nube

Una vez que los datos fueron procesados y estructurados localmente, el siguiente paso fue establecer un canal de comunicación eficiente y confiable hacia la nube. Para este paso, se seleccionó el protocolo MQTT (Message Queuing Telemetry Transport). Esta elección fue la mejor elección con base a otros sistemas IoT, ya que MQTT es reconocido en el ámbito industrial del Internet de las Cosas por su arquitectura de "publicación/suscripción" y su extrema ligereza, lo que permite transmitir paquetes de datos con un consumo mínimo de ancho de banda y energía, factores vitales para un dispositivo de monitoreo continuo. Además de que ayuda aún más a la portabilidad del proyecto y limita costos al no contar con un DNS o una computadora que pudiera usarse como servidor local, como lo podría ser el uso de una raspberry, así se eliminan elementos conservando el uso solamente de la ESP32 y el sensor MAX30102.

Para la gestión de estos mensajes, se implementó el uso del broker público EMQX (broker.emqx.io), actuando como el nodo central que recibe la información del ESP32 y la distribuye a los sistemas autorizados. La conexión se configuró a través del puerto 1883, que es el estándar para tráfico MQTT sin cifrado, facilitando una integración rápida y de baja latencia con el hardware.

Un aspecto fundamental de esta fase fue la definición de un Tópico de Publicación específico y personalizado: proyecto/salud/equipo_adri/datos/oxigeno. La importancia de esta jerarquía radica en la organización y escalabilidad del sistema; el uso de tópicos bien definidos permite que el flujo de información sea ordenado y que cualquier suscriptor con las credenciales y el tópico correcto pueda recibir las constantes vitales del paciente en tiempo real desde cualquier ubicación geográfica. Esto elimina las barreras de distancia, cumpliendo con el objetivo de proporcionar un monitoreo remoto efectivo para los cuidadores.

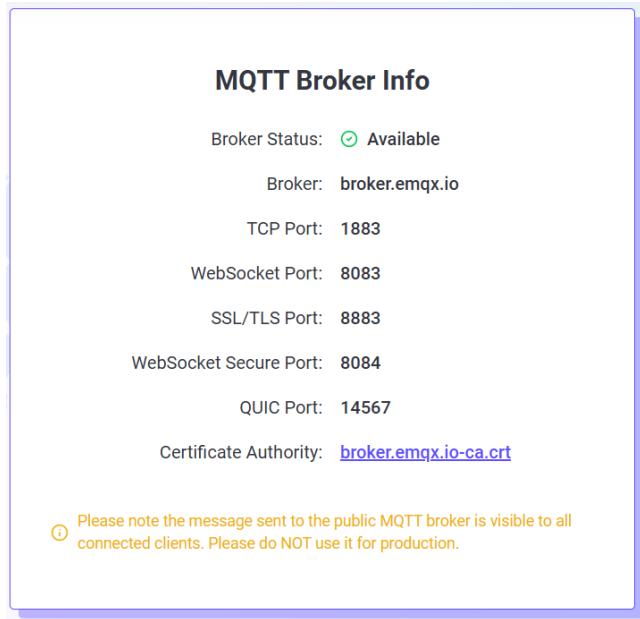


Ilustración 3- Datos del broker EMQX

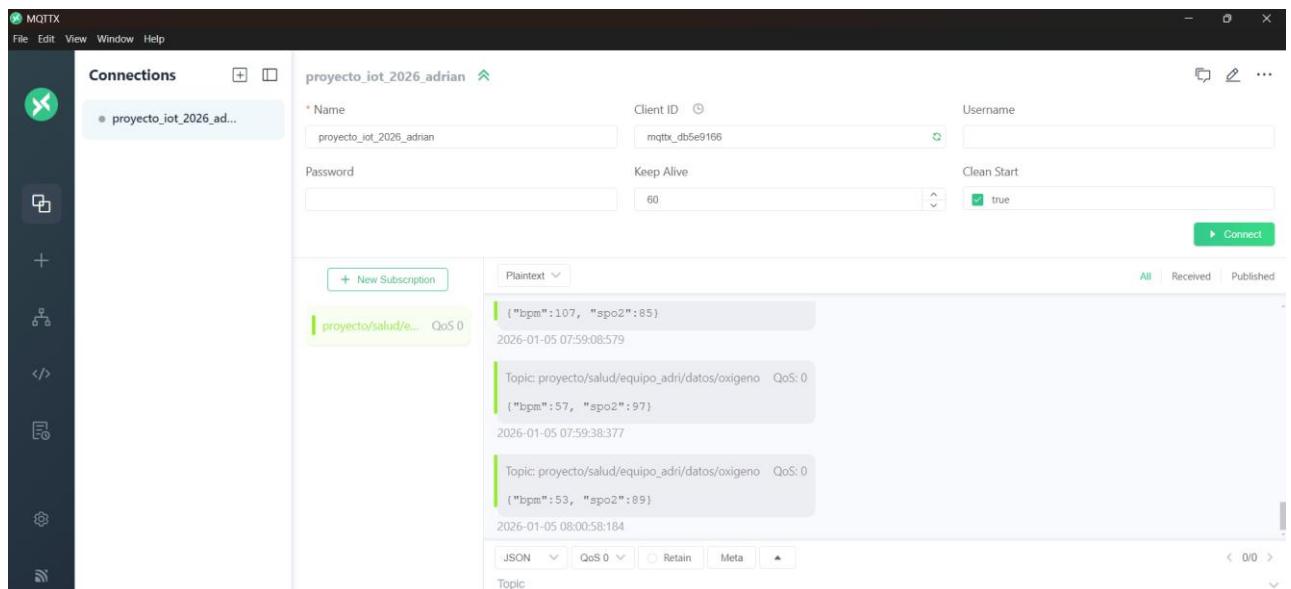


Ilustración 4- Configuración del broker en MQTTX

Fase 3: Puente de Inteligencia y Gestión de Alertas (Python Bridge)

En esta etapa, se centró en la creación de un script de Python, el cual actúa como el "cerebro" central del sistema en el lado del receptor. Mientras que el ESP32 se encarga de la captura, este puente de software asume la responsabilidad de procesar, discriminar y almacenar la información de manera inteligente. Permitiendo la comunicación entre lo que se procesa localmente (Edge computing) y el broker que se configuró previamente. Este script se ejecuta de manera local en la terminal del sistema operativo, en este caso Windows e informa el estatus de la conexión en todo momento, esta función no cobra relevancia para el usuario final, sin embargo, si ocurre alguna falla es de ayuda para que el desarrollador pueda identificar el problema oportunamente. Lo que sí cobra relevancia para el usuario final es la integración con un bot de Telegram que envía la información recopilada a uno o varios usuarios seleccionados. El script se diseñó para operar de forma ininterrumpida, escuchando los mensajes que llegan al broker y aplicando tres funciones críticas que garantizan la utilidad del sistema:

En primer lugar, se implementó la persistencia de datos. Cada lectura recibida se registra automáticamente en un archivo local con formato CSV (historial_salud.csv), vinculando cada medición con su respectiva fecha y hora. Esta base de datos histórica es fundamental, ya que permite la generación de gráficos de tendencia y reportes médicos que facilitan el análisis retrospectivo del estado de salud del paciente.

En segundo lugar, el script ejecuta las reglas de negocio que analizan las pulsaciones por minuto (BPM) en tiempo real. Se establecieron umbrales clínicos para detectar anomalías: si el pulso desciende de 50 BPM (indicativo de bradicardia) o supera los 120 BPM (indicativo de taquicardia), el sistema clasifica el dato como una emergencia potencial.

Finalmente, para asegurar la estabilidad de la comunicación, se integró un Control de Saturación. Mediante un filtro de tiempo de 20 segundos, el script regula el envío de notificaciones hacia la interfaz de usuario. Esta medida técnica es vital para evitar el bloqueo del bot por parte de los servidores de Telegram debido al exceso de mensajes (anti-spam), garantizando que las alertas lleguen de manera fluida y oportuna sin comprometer la integridad del canal de comunicación.

```
PS C:\Users\adri_\Desktop\ESCOM\8VO SEMESTRE\IoT> python puente_telegram.py
🔗 Puente activado. Iniciando monitoreo...
✅ Conectado y listo para monitorear
🚀 [TELEGRAM] Alerta enviada con 107 BPM.
🚀 [TELEGRAM] Alerta enviada con 93 BPM.
🕒 Dato (83 BPM) guardado en Excel. Esperando para Telegram...
🚀 [TELEGRAM] Alerta enviada con 53 BPM.
```

Ilustración 5-Ejecución en terminal del script de Python

Fase 4: Interfaz de Usuario y Notificaciones de Emergencia

La etapa final del desarrollo consistió entre la interacción humana y el sistema final, para lo cual se integró la API de Telegram a través del script de Python para conectar un bot de Telegram que enviará los datos que ya se procesaron y filtraron. La elección de Telegram fue por su alta confiabilidad, su robusto cifrado de extremo a extremo y su capacidad para gestionar "bots" de manera eficiente sin costo operativo. Al ser una aplicación multiplataforma presente en la mayoría de los dispositivos móviles, garantiza que el cuidador reciba las notificaciones de manera instantánea, sin importar su ubicación geográfica, aprovechando una infraestructura de mensajería ya establecida y estable.

Dentro de esta integración, se desarrolló un Bot de Monitoreo personalizado que actúa como el receptor final de la lógica procesada en Python. El script de Python traduce los datos técnicos en reportes visualmente intuitivos utilizando el formato Markdown, lo que permite resaltar información crítica en negritas y organizar los datos de manera clara. Además, se implementó un sistema de semáforo visual mediante emojis (🟢 para niveles normales, 🟥 para precaución y 🚨 para emergencias), permitiendo que el familiar identifique el estado de salud del paciente con un solo vistazo, incluso antes de leer las cifras exactas.

Un componente importante en el diseño de esta interfaz es el acceso directo a emergencias. En el momento en que el script de Python detecta que los niveles de BPM o SpO2 están en un rango peligroso, el mensaje enviado al bot incluye automáticamente un enlace directo de llamada al 911. Esta funcionalidad reduce drásticamente el tiempo de reacción ante una crisis, ya que el cuidador no necesita salir de la aplicación ni marcar manualmente el número; la tecnología actúa como un facilitador que agiliza la asistencia médica en los segundos más críticos de una emergencia. A su vez se puede implementar el contactar de manera automática un contacto de emergencia según el paciente o usuarios elijan.

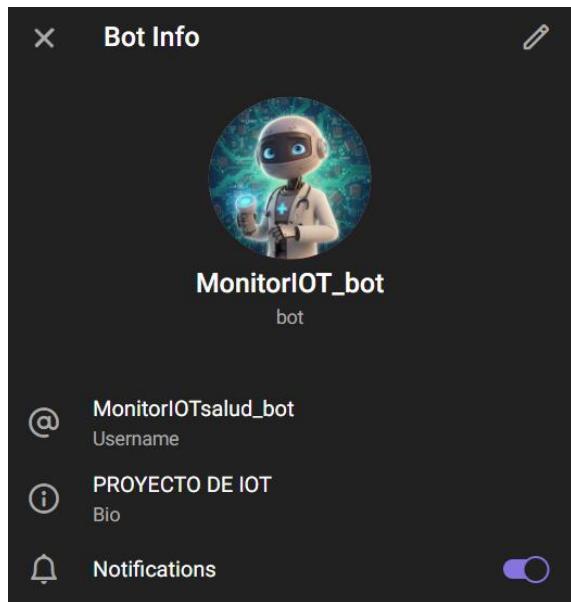


Ilustración 6- Información del Bot de Telegram creado

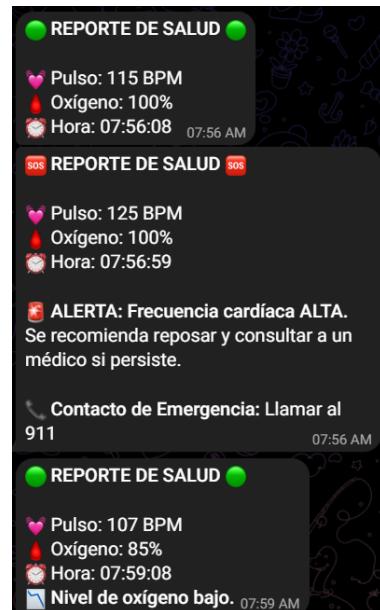


Ilustración 7- Mensajes enviados por el bot de Telegram

Fase 5: Almacenamiento y Gestión de Base de Datos (Data Logging)

La última etapa del desarrollo se centró en garantizar la integridad y utilidad de la información a largo plazo mediante una capa de persistencia de datos. Mientras que la interfaz de Telegram prioriza la inmediatez y la gestión de crisis, esta fase se enfoca en el registro continuo y detallado de cada métrica capturada por el hardware. El script de Python fue programado para realizar un "data logging" exhaustivo: a diferencia de las notificaciones móviles que se filtran cada 20 segundos, el archivo local registra el 100% de las lecturas enviadas por el ESP32, asegurando que no se pierda ningún evento fisiológico por breve que sea.

Para facilitar la auditoría de estos datos, se diseñó una estructura de información estandarizada dentro de un archivo CSV (historial_salud.csv). El archivo organiza las mediciones en cuatro ejes fundamentales: Fecha, Hora, BPM y SpO2, permitiendo una trazabilidad exacta del comportamiento del organismo en momentos específicos del día o la noche. Esta organización permite que cualquier software de hoja de cálculo pueda procesar la información de inmediato, facilitando la creación de gráficos comparativos que visualizan la evolución de la salud del paciente.

Finalmente, el verdadero impacto de esta fase radica en su valor clínico. Al contar con un historial ininterrumpido, el personal médico puede observar tendencias y patrones, como; taquicardias recurrentes en horarios específicos o caídas leves de oxígeno durante el sueño, que serían imposibles de detectar en una consulta presencial aislada. De este modo, el sistema deja de ser un monitor de emergencia para convertirse en una herramienta de atención primaria eficiente, proporcionando al médico datos reales y objetivos que fundamentan un diagnóstico preventivo y una atención personalizada para el adulto mayor.

Fecha	Hora	BPM	SpO2
05/01/2026	00:14:43	51	99
05/01/2026	00:15:23	115	96
05/01/2026	00:16:03	93	100
05/01/2026	00:24:12	125	99
05/01/2026	00:24:22	125	99
05/01/2026	00:24:32	68	98
05/01/2026	00:24:42	115	98
05/01/2026	00:24:52	100	100
05/01/2026	00:41:40	115	99
05/01/2026	00:42:20	107	99
05/01/2026	00:44:20	60	47
05/01/2026	00:46:00	125	93
05/01/2026	00:46:40	48	92
05/01/2026	00:47:50	65	93

Ilustración 8-Primeros registros en el documento de Excel

8 RESULTADOS Y PRUEBAS

Con el objetivo de validar el correcto funcionamiento del sistema propuesto, se realizaron diversas pruebas experimentales que permitieron comprobar la adquisición, transmisión y visualización de los datos biomédicos obtenidos. A continuación, se presentan los resultados más representativos del proyecto.

La Ilustración 9 muestra el proceso de medición de la frecuencia cardíaca y la saturación de oxígeno en la sangre (SpO_2) utilizando el sensor biomédico MAX30102, integrado al microcontrolador ESP32. Durante la prueba, el sensor fue colocado correctamente debajo del dedo del usuario, permitiendo la adquisición de las señales ópticas necesarias para el cálculo de los signos vitales.

Los valores obtenidos se mantuvieron dentro de rangos fisiológicos normales, lo que confirma la correcta calibración del sensor y su adecuada integración con el sistema de procesamiento. Esta prueba valida la capacidad del sistema para realizar mediciones no invasivas y en tiempo real, condición indispensable para aplicaciones de monitoreo continuo en adultos mayores.

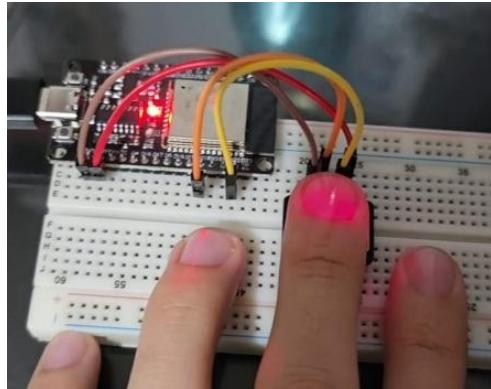


Ilustración 9- Medición de pulso cardiaco con sensor y ESP32

En la Ilustración 10 se presenta la obtención de los datos de frecuencia cardíaca y saturación de oxígeno visualizados a través del monitor serial del entorno de desarrollo Arduino IDE. Los datos capturados por el ESP32 fueron enviados de manera inalámbrica hacia la plataforma en la nube mediante el protocolo MQTT al bróker configurado en MQTXX.

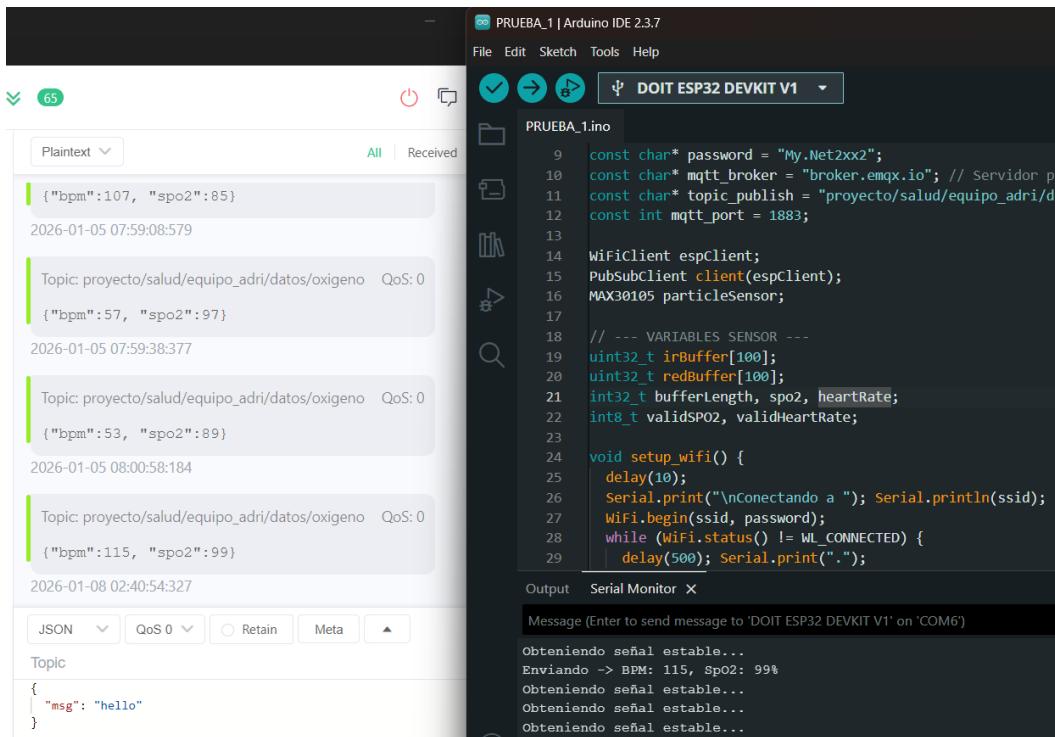


Ilustración 10-Visualización de envío de datos al broker MQTTX

Posteriormente, como se puede observar en la ilustración 11 el script desarrollado en Python se encargó de recibir y procesar la información transmitida al bróker MQTTX confirmando la correcta comunicación entre el dispositivo IoT y el servidor. Para que después filtre las mediciones y mandar los mensajes y/o alertas correspondientes al bot de Telegram y que este a su vez envíe el mensaje al usuario final. Esta prueba demuestra la confiabilidad del sistema de transmisión de datos y la correcta interoperabilidad entre el hardware embebido y el software de backend.

```

PS C:\Users\adri_> cd C:\Users\adri_\Desktop\ESCOM\"8VO SEMESTRE"\IoT
PS C:\Users\adri_\Desktop\ESCOM\8VO SEMESTRE\IoT> python puente_telegram.py
🔗 Puente activado. Iniciando monitoreo...
✅ Conectado y listo para monitorear
🚀 [TELEGRAM] Alerta enviada con 71 BPM.
🚀 [TELEGRAM] Alerta enviada con 88 BPM.

Message (Enter to send message to 'DOIT ESP32 DEVKIT V1' on 'COM6')

Obteniendo señal estable...
Obteniendo señal estable...
Enviando -> BPM: 71, SpO2: 98%
Obteniendo señal estable...
Obteniendo señal estable...
Enviando -> BPM: 88, SpO2: 99%

```

Ilustración 10-Obtención de datos a través de IDE Arduino y obtención de datos con script de Python

La Ilustración 11 muestra finalmente la interacción entre las tres partes claves del proyecto, obtención y envío de datos desde la IDE de Arduino, recepción de datos en el broker de MQTDX y puente entre el broker y telegram con el script de Python. El bot recibe la información enviada desde el servidor, registra los valores medidos y envía mensajes automáticos al usuario autorizado, permitiendo la consulta del estado de salud en tiempo real o generación de alertas según sea el caso.

Esta funcionalidad valida el correcto funcionamiento del sistema de notificaciones, así como la accesibilidad de la información para familiares o cuidadores. La integración del bot de Telegram proporciona una interfaz sencilla e intuitiva, eliminando la necesidad de aplicaciones especializadas y facilitando la supervisión remota del adulto mayor.

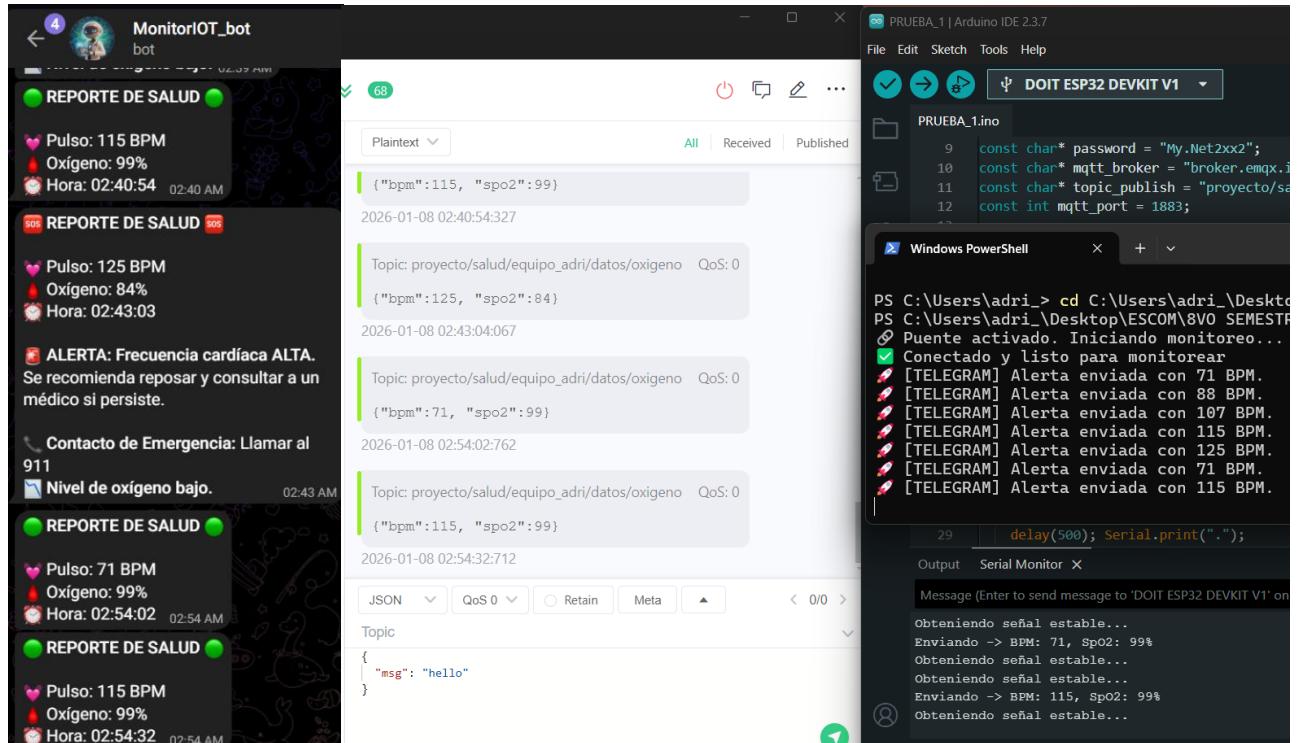


Ilustración 11-Interacción entre ESP32, broker, Python y Telegram

Los resultados obtenidos durante las pruebas demuestran que el sistema desarrollado cumple con los objetivos planteados, logrando la adquisición precisa de signos vitales, la transmisión confiable de los datos y su visualización en tiempo real mediante una plataforma de mensajería. El correcto funcionamiento de cada uno de los componentes valida la viabilidad del sistema como una solución de monitoreo remoto de bajo costo y fácil implementación.

9 CONCLUSIONES

La culminación de este proyecto permite concluir que es posible desarrollar un sistema de monitoreo de salud robusto, asequible y funcional integrando tecnologías de hardware libre y protocolos de comunicación industriales. Los resultados obtenidos demostraron que la combinación del sensor **MAX30102** con el microcontrolador **ESP32** proporciona lecturas de frecuencia cardíaca y saturación de oxígeno con un nivel de precisión suficiente para el monitoreo preventivo.

El flujo de datos implementado —desde la captura física, la transmisión vía MQTT a través de un broker en la nube, hasta el procesamiento en Python— probó ser una arquitectura eficiente y de baja latencia. El sistema no solo logró notificar alertas críticas en tiempo real a través de un Bot de Telegram, sino que garantizó la integridad de la información mediante la creación de un historial automatizado en formato CSV. Esta capacidad de almacenamiento transforma el dispositivo de un simple sensor a una herramienta de análisis clínico longitudinal.

A lo largo del desarrollo, se consolidaron habilidades críticas en programación de sistemas embebidos, gestión de redes inalámbricas y el uso de protocolos de mensajería para el Internet de las Cosas (IoT). Asimismo, se profundizó en la interpretación de datos biológicos y en el manejo de bibliotecas de Python para la automatización de procesos y la interacción con APIs de mensajería instantánea. El mayor aprendizaje radica en la capacidad de integrar múltiples capas tecnológicas (Hardware, Software y Redes) para resolver un problema complejo.

Desde una perspectiva social, este prototipo representa una respuesta tangible a la deficiencia en la atención primaria mencionada en la introducción. Al ser un sistema portátil y de bajo costo, tiene el potencial de democratizar el acceso al monitoreo médico para personas de la tercera edad o sectores vulnerables en países de ingresos medios. En el ámbito de la salud, la capacidad de detectar taquicardias, bradicardias o hipoxemia de forma remota permite una intervención temprana, reduciendo la saturación en servicios de urgencias y, lo más importante, salvando vidas mediante la prevención y la vigilancia constante.

10 ANEXOS

10.1 CÓDIGO USADO EN EL ESP32

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include "MAX30105.h"
#include "spo2_algorithm.h"

// --- CONFIGURACIÓN WIFI Y MQTT ---
const char* ssid = "IZZI-7661";
const char* password = "My.Net2xx2";
const char* mqtt_broker = "broker.emqx.io"; // Servidor público
const char* topic_publish = " proyecto/salud/equipo_adri/datos/oxigeno";
// Tópico único
const int mqtt_port = 1883;

WiFiClient espClient;
PubSubClient client(espClient);
MAX30105 particleSensor;

// --- VARIABLES SENSOR ---
uint32_t irBuffer[100];
uint32_t redBuffer[100];
int32_t bufferLength, spo2, heartRate;
int8_t validSPO2, validHeartRate;

void setup_wifi() {
    delay(10);
    Serial.print("\nConectando a "); Serial.println(ssid);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500); Serial.print(".");
    }
    Serial.println("\nWiFi conectado. IP: ");
    Serial.println(WiFi.localIP());
}

void reconnect() {
    while (!client.connected()) {
        Serial.print("Intentando conexión MQTT...");
        // Creamos un ID de cliente único para el broker público
        String clientId = "ESP32Client-Salud-" + String(random(0, 999));
        if (client.connect(clientId.c_str())) {
            Serial.println("conectado");
        } else {
            Serial.print("falló, rc=");
            Serial.print(client.state());
            Serial.println(" reintentando en 5 segundos");
            delay(5000);
        }
    }
}

void setup() {
    Serial.begin(115200);
```

```

setup_wifi();
client.setServer(mqtt_broker, mqtt_port);

if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) {
    Serial.println("MAX30102 no encontrado.");
    while (1);
}

// Configuración sensor
particleSensor.setup(60, 4, 2, 100, 411, 4096);
Serial.println("Sensor listo. Coloca tu dedo.");
}

void loop() {
    if (!client.connected()) reconnect();
    client.loop();

    bufferLength = 100;
    for (byte i = 0 ; i < bufferLength ; i++) {
        while (particleSensor.available() == false) particleSensor.check();
        redBuffer[i] = particleSensor.getRed();
        irBuffer[i] = particleSensor.getIR();
        particleSensor.nextSample();
    }

    maxim_heart_rate_and_oxygen_saturation(irBuffer, bufferLength,
redBuffer, &spo2, &validSPO2, &heartRate, &validHeartRate);

    if (validHeartRate == 1 && validSPO2 == 1 && heartRate > 40 &&
heartRate < 130) {
        // Calculamos el promedio (usando tu lógica de suavizado)
        int finalBPM = heartRate;
        int finalSPO2 = spo2;

        Serial.printf("Enviando -> BPM: %d, SpO2: %d%%\n", finalBPM,
finalSPO2);

        // Creamos el mensaje en formato JSON para que sea fácil de leer
después
        String payload = "{\"bpm\":";
        payload += finalBPM;
        payload += ", \"spo2\":";
        payload += finalSPO2;
        payload += "}";

        // Publicamos en el broker
        client.publish(topic_publish, payload.c_str());
    } else {
        Serial.println("Obteniendo señal estable...");
    }
}

```

10.2 SCRIPT DE PYTHON

```
import paho.mqtt.client as mqtt
import telepot
import json
import time
import ssl
import csv
from datetime import datetime

# --- TUS DATOS ---
TOKEN = '8251934467:AAFmLziBdweKuKpkpnqOA4cSvYCK9nPuGiU'
CHAT_ID = 1569528013
TOPIC = "proyecto/salud/equipo_adri/datos/oxigeno"
BROKER = "broker.emqx.io"

ultima_alerta = 0
archivo_excel = "historial_salud.csv"

# Configuración SSL para Telegram
context = ssl._create_unverified_context()
telepot.api._pools['default'] =
telepot.api.urllib3.PoolManager(ssl_context=context)
bot = telepot.Bot(TOKEN)

# Función para guardar en Excel
def guardar_en_excel(bpm, spo2):
    fecha = datetime.now().strftime('%Y-%m-%d')
    hora = datetime.now().strftime('%H:%M:%S')
    try:
        with open(archivo_excel, mode='a', newline='') as file:
            writer = csv.writer(file)
            writer.writerow([fecha, hora, bpm, spo2])
    except PermissionError:
        print("⚠️ No se pudo guardar en Excel: El archivo está abierto.")

# Crear el encabezado del Excel si no existe
try:
    with open(archivo_excel, mode='a', newline='') as file:
        if file.tell() == 0:
            writer = csv.writer(file)
            writer.writerow(["Fecha", "Hora", "BPM", "SpO2"])
except PermissionError:
    print("⚠️ Error inicial: Cierra el Excel antes de empezar.")

def on_connect(client, userdata, flags, rc, properties):
    if rc == 0:
        print("☑ Conectado y listo para monitorear")
        client.subscribe(TOPIC)

def on_message(client, userdata, msg):
    global ultima_alerta
    try:
        payload = msg.payload.decode()
        data = json.loads(payload)
        bpm = data.get("bpm")
```

```

spo2 = data.get("spo2")

guardar_en_excel(bpm, spo2)

tiempo_actual = time.time()

if (tiempo_actual - ultima_alerta) > 20:
    status_emoji = "⚠"
    nota_medica = ""

    # Ajuste de mensajes de alerta
    if bpm < 50:
        status_emoji = "⚠"
        nota_medica = "\n\n⚠*ALERTA: Frecuencia cardíaca\nBAJA.* Por favor, revise sus niveles nuevamente y mantenga la calma."
        nota_medica += "\n\n⚠*Contacto de Emergencia:* [Llamar al 911] (tel:911)"
    elif bpm > 120:
        status_emoji = "⚠"
        nota_medica = "\n\n⚠*ALERTA: Frecuencia cardíaca\nALTA.* Se recomienda reposar y consultar a un médico si persiste."
        nota_medica += "\n\n⚠*Contacto de Emergencia:* [Llamar al 911] (tel:911)"

    if spo2 < 90:
        nota_medica += "\n⚠*Nivel de oxígeno bajo.*"

    texto_alerta = (
        f"\n{status_emoji} *REPORTE DE SALUD* {status_emoji}\n\n"
        f"🕒Pulso: {bpm} BPM\n"
        f"⚡Oxígeno: {spo2}%\n"
        f"⌚ Hora: {datetime.now().strftime('%H:%M:%S')}\n"
        f"\n{nota_medica}"
    )

    bot.sendMessage(CHAT_ID, texto_alerta,
parse_mode='Markdown')
    print(f"⚠ [TELEGRAM] Alerta enviada con {bpm} BPM.")
    ultima_alerta = tiempo_actual
else:
    print(f"⚠ Dato ({bpm} BPM) guardado en Excel. Esperando para Telegram...")

except Exception as e:
    print(f"⚠ Error en procesamiento: {e}")

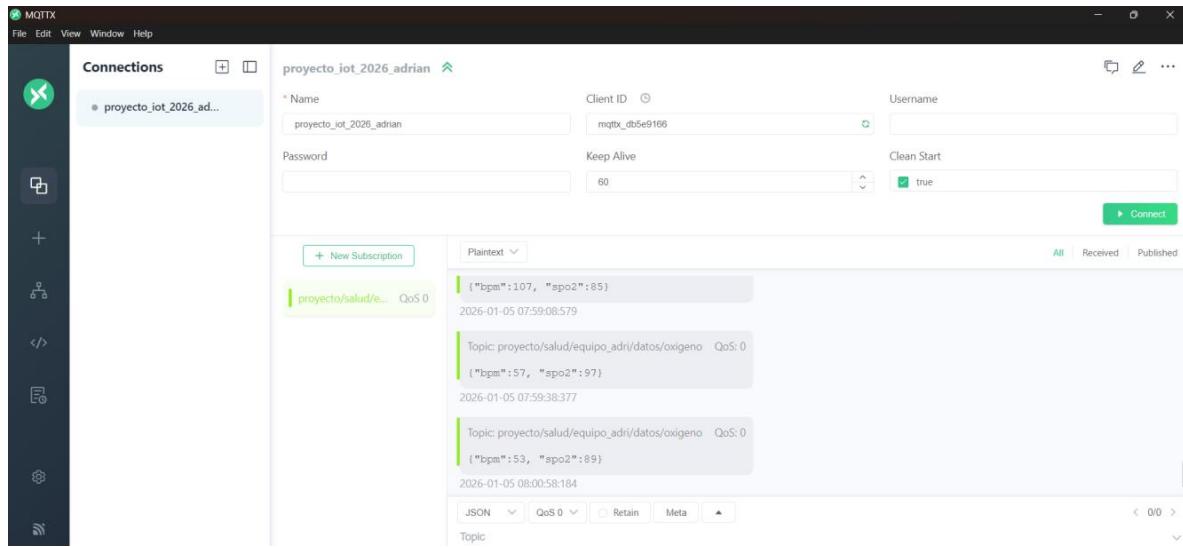
# Configuración MQTT
client = mqtt.Client(mqtt.CallbackAPIVersion.VERSION2)
client.on_connect = on_connect
client.on_message = on_message

print("⚠Puente activado. Iniciando monitoreo...")

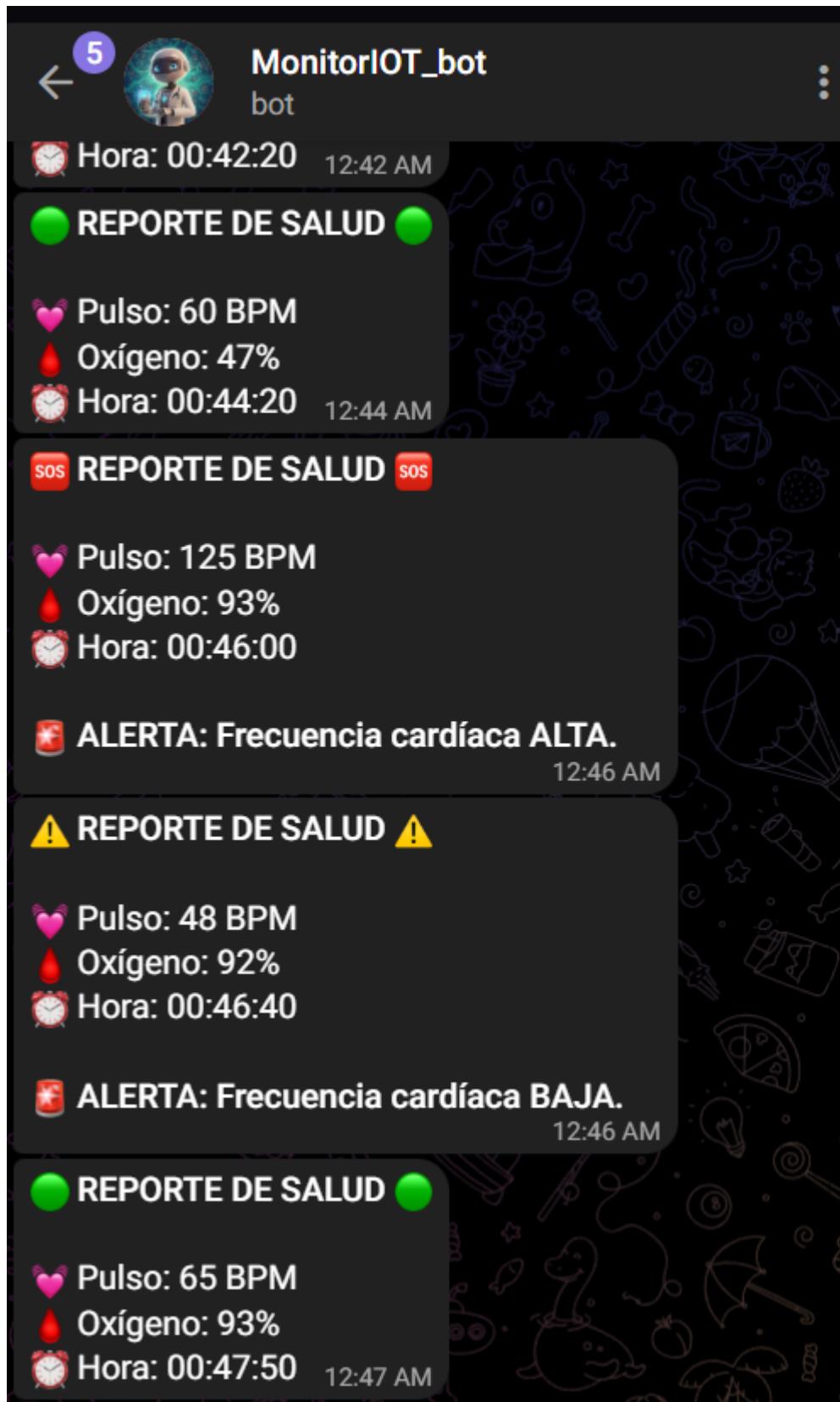
```

```
# Bucle principal con protección contra cierres
try:
    while True:
        try:
            client.connect(BROKER, 1883, 60)
            client.loop_forever()
        except Exception as e:
            print(f"X Conexión perdida: {e}. Reintentando en 5
segundos...")
            time.sleep(5)
except KeyboardInterrupt:
    print("\n□□Sistema apagado por el usuario.")
```

10.3 CONFIGURACIÓN DEL BROKER



10.4 VISUALIZACIÓN DEL BOT Y ALERTAS EN TELEGRAM



10.5 DOCUMENTO DE EXCEL GENERADO

	A	B	C	D	
1	Fecha	Hora	BPM	SpO2	
2	05/01/2026	00:14:43	51	99	
3	05/01/2026	00:15:23	115	96	
4	05/01/2026	00:16:03	93	100	
5	05/01/2026	00:24:12	125	99	
6	05/01/2026	00:24:22	125	99	
7	05/01/2026	00:24:32	68	98	
8	05/01/2026	00:24:42	115	98	
9	05/01/2026	00:24:52	100	100	
10	05/01/2026	00:41:40	115	99	
11	05/01/2026	00:42:20	107	99	
12	05/01/2026	00:44:20	60	47	
13	05/01/2026	00:46:00	125	93	
14	05/01/2026	00:46:40	48	92	
15	05/01/2026	00:47:50	65	93	
16	05/01/2026	00:49:40	88	83	
17	05/01/2026	00:49:59	125	95	
18	05/01/2026	07:49:29	107	99	
19	05/01/2026	07:49:39	53	98	
20	05/01/2026	07:50:49	83	100	
21	05/01/2026	07:51:39	93	100	
22	05/01/2026	07:51:49	125	100	
23	05/01/2026	07:56:08	115	100	
24	05/01/2026	07:56:18	107	100	
25	05/01/2026	07:59:08	107	85	