



---

Miguel Jaime Adrián David  
Hernández Razo Juan Daniel

6CM3

Prof. Alemán Arce Miguel Ángel

12/12/2025

---

## 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	Propuesta de solución .....	3
3	Plan de acción.....	4
	Fase 1. Análisis y Diseño .....	4
	Fase 2. Desarrollo de Prototipo de Hardware .....	5
	Fase 3. Plataforma en la Nube y Backend .....	5
	Fase 4. Desarrollo de Aplicación de Usuario.....	5
	Fase 5. Pruebas Piloto .....	5
	Fase 6. Optimización y Escalabilidad.....	5
4	Desarrollo y fases .....	6
5	Resultados y pruebas .....	8
6	Conclusiones .....	10
7	Anexos.....	11
7.1	Código usado en el ESP32 .....	11
7.2	Script de Python.....	16
7.3	Configuración del Broker.....	21
7.4	Visualización del bot y alertas en Telegram.....	22
7.5	Documento de Excel generado.....	23

## 1 PROBLEMA DETECTADO

---

En los últimos años hemos enfrentado emergencias sanitarias a nivel global como lo ha sido la pandemia de COVID-19 y el sistema de salud se vio saturado, lo que provocó tanto detecciones como atenciones erráticas dando como resultado múltiples decesos. Además de tener una atención primaria deficiente, esto cobra relevancia ya que la atención primaria suele ser el primer punto de contacto de las personas con su sistema sanitario, y lo ideal sería que fuese una atención integral, asequible y basada en la comunidad a lo largo de toda la vida.

La atención primaria de la salud puede satisfacer la mayoría de las necesidades de una persona a lo largo de su vida. Se necesitan sistemas sanitarios con una sólida atención primaria de la salud para lograr la cobertura sanitaria universal.

Sin embargo, muchos países no cuentan con establecimientos adecuados de atención primaria de la salud. Ello puede deberse a la falta de recursos en los países de ingresos bajos o medianos.

Pero gracias al avance tecnológico también se han desarrollado sistemas de monitoreo remotos, los cuales son una solución tecnológica avanzada que recopila datos en tiempo real sobre los parámetros de salud de una persona. Estos sistemas están diseñados para rastrear información médica clave, como la frecuencia cardíaca la temperatura, la saturación de oxígeno y la presión arterial, ayudando tanto a médicos como a pacientes.

Ahora, el IoT se ha vuelto una pieza clave para estos sistemas, monitoreando con sensores y wearables, para que esos datos sean procesados por instituciones médicas o doctores y así ofrecer mejor atención médica.

## 2 PROPUESTA DE SOLUCIÓN

---

Desarrollar un sistema de monitoreo remoto de signos vitales basado en tecnologías de Internet de las Cosas (IoT), que permita recopilar, procesar y analizar en tiempo real variables fisiológicas y de contexto de las personas adultas mayores, facilitando la detección temprana de anomalías y mejorando la calidad de vida mediante alertas preventivas.

El sistema propuesto permitirá:

- Monitorear variables clave: frecuencia cardíaca y saturación de oxígeno
- Transmitir los datos de manera inalámbrica hacia una **plataforma en la nube** para su almacenamiento y análisis.
- Generar **alertas automáticas** (SMS, notificación en aplicaciones de mensajería como Telegram o Whatsapp) en caso de valores fuera de rango o eventos de emergencia.
- Brindar acceso a cuidadores y familiares mediante una **aplicación web o móvil**, donde podrán consultar reportes históricos y el estado de salud en tiempo real.

### Propuesta Técnica

- **Sensores biomédicos:**
  - Frecuencia cardíaca y SpO<sub>2</sub>: sensor óptico (ej. MAX30102).

- **Plataforma de procesamiento local:**
  - Microcontrolador ESP32 con conectividad WiFi/Bluetooth para recolectar datos y transmitirlos.
- **Comunicación y almacenamiento:**
  - Envío de datos a una base en la nube mediante protocolos IoT (MQTT/HTTP).
  - Almacenamiento en un documento de Excel
- **Aplicación de usuario:**
  - Bot que recopile la información y haga reportes mensuales o semanales que son enviados al cuidador o familiar cercano
  - Alertas en tiempo real ante emergencias.

### **Beneficios esperados**

- **Prevención:** detección temprana de anomalías en signos vitales.
- **Seguridad:** alertas automáticas en casos de emergencias o registros no atendidos.
- **Accesibilidad:** monitoreo remoto desde cualquier lugar con internet.
- **Soporte a cuidadores y médicos:** registro histórico de datos para mejorar el diagnóstico y tratamiento.

El sistema combina sensores biomédicos de bajo costo con una arquitectura IoT escalable, permitiendo un monitoreo continuo, no invasivo y accesible para adultos mayores, a diferencia de los equipos médicos convencionales. Además, su integración con servicios en la nube permite aplicar algoritmos de análisis predictivo y aprendizaje automático en etapas futuras.

## **3 PLAN DE ACCIÓN**

---

### **FASE 1. ANÁLISIS Y DISEÑO**

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

**Actividades:**

- Levantamiento de requerimientos.
- Selección de variables fisiológicas y ambientales prioritarias ( $\text{SpO}_2$ , 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.

**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.

**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.

**Actividades:**

- Implementación de bot para generar reportes semanales/mensuales.
- Pruebas de usabilidad con usuarios finales.

## **FASE 5. PRUEBAS PILOTO**

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

**Actividades:**

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

## **FASE 6. OPTIMIZACIÓN Y ESCALABILIDAD**

**Objetivo:** Ajustar el sistema para mayor robustez y crecimiento.

**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.

## 4 DESARROLLO Y FASES

---

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

En esta etapa, el sensor MAX30102 interactúa con el ESP32 mediante el protocolo I2C.

Algoritmo de filtrado: Se implementó un buffer circular (RATE\_SIZE) para promediar las lecturas de BPM y descartar valores fuera de rango (menos de 50 o más de 120 BPM en reposo), garantizando la estabilidad de la información.

Conectividad: El ESP32 actúa como un cliente WiFi que utiliza la librería PubSubClient para empaquetar los datos en formato JSON.

### Fase 2: Protocolo de Comunicación y Broker en la Nube

Para la transmisión de datos se eligió el protocolo MQTT (Message Queuing Telemetry Transport) por su ligereza:

Broker: Se utilizó el servidor público broker.emqx.io.

Puerto: 1883 (Estándar para IoT).

Tópico de publicación: proyecto/salud/equipo\_adri/datos/oxigeno. Esto permite que el sistema sea escalable, ya que cualquier suscriptor autorizado puede recibir los datos en tiempo real desde cualquier parte del mundo.

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

Se desarrolló un script en Python que actúa como el "cerebro" del sistema fuera del dispositivo físico. Sus funciones principales son:

- Persistencia de datos: Cada lectura recibida se almacena automáticamente en un archivo CSV (historial\_salud.csv), permitiendo generar un historial médico para análisis posterior.
- Lógica de Salud: El script analiza si el BPM es menor a 50 (Bradícardia) o mayor a 120 (Taquicardia).

- Control de Saturación: Implementa un filtro de tiempo de 20 segundos para evitar el bloqueo del bot de Telegram por spam, asegurando una comunicación fluida.

#### Fase 4: Interfaz de Usuario y Notificaciones de Emergencia

La última capa es la integración con la API de Telegram:

Bot de Monitoreo: Se creó un bot personalizado que envía reportes con formato Markdown y emojis visuales (  ,  ,  ).

Acceso a Emergencias: En caso de detectar rangos peligrosos, el bot añade automáticamente un enlace directo de llamada al 911, facilitando la reacción inmediata del cuidador o familiar.

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

Incorpora una capa de persistencia de datos mediante la creación de un archivo CSV (**historial\_salud.csv**).

- **Registro Continuo:** Mientras que las alertas de Telegram se filtran por tiempo para no saturar al usuario, el script de Python registra el **100% de las lecturas** recibidas del ESP32 en el archivo local.
- **Estructura de Datos:** El archivo organiza la información en cuatro columnas: **Fecha, Hora, BPM y SpO2**, permitiendo una trazabilidad exacta del estado del paciente en momentos específicos del día.
- **Valor Clínico:** Este historial permite al personal médico observar tendencias (como taquicardias recurrentes o caídas de oxígeno durante la noche) que no serían visibles en una medición aislada, cumpliendo con el objetivo de proporcionar una "atención primaria" eficiente.

## 5 RESULTADOS Y PRUEBAS

---

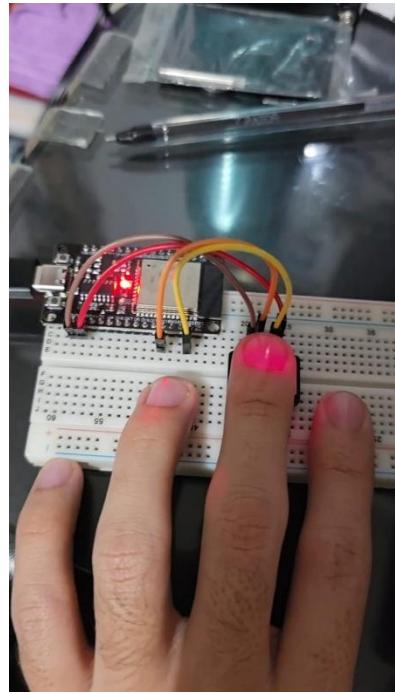


Ilustración 1- Medición de pulso cardiaco con sensor y ESP32

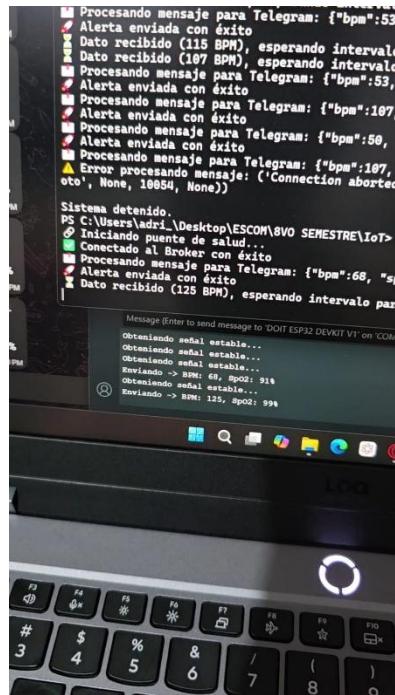


Ilustración 2-Obtención de datos a través de IDE Arduino y envío de datos con script de Python

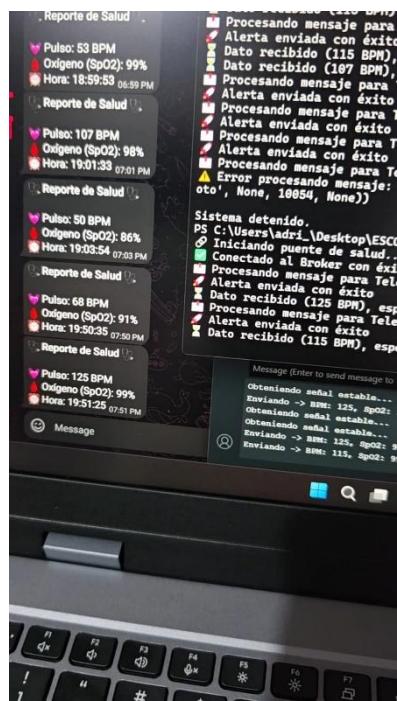


Ilustración 3- Visualización de Bot de Telegram, Terminal de Windows e IDE Arduino

## 6 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.

## 7 ANEXOS

---

### 7.1 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) {

    // Cálculo del promedio del BPM

    int finalBPM = heartRate;

    int finalSPO2 = spo2;

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

    // Creación del mensaje en formato JSON

    String payload = "{\"bpm\":";
    payload += finalBPM;
    payload += ", \"spo2\":";
    payload += finalSPO2;
    payload += "}";
}

```

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

## 7.2 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 BAJA.* Por favor,\nrevise sus niveles nuevamente y mantenga la calma."
                nota_medica += "\n\n📞 *Contacto de Emergencia:* [Llamar al 911](tel:911)"

    except Exception as e:
        print(f"Error: {e}")
```

```
elif bpm > 120:
```

```
    status_emoji = "🆘 "
```

```
    nota_medica = "\n\n⚠ *ALERTA: Frecuencia cardíaca ALTA.* 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" {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")}"
```

```
f" {nota_medica}"
```

```
)
```

```
bot.sendMessage(CHAT_ID, texto_alerta, parse_mode='Markdown')
```

```
print(f"🚀 [TELEGRAM] Alerta enviada con {bpm} BPM.")
```

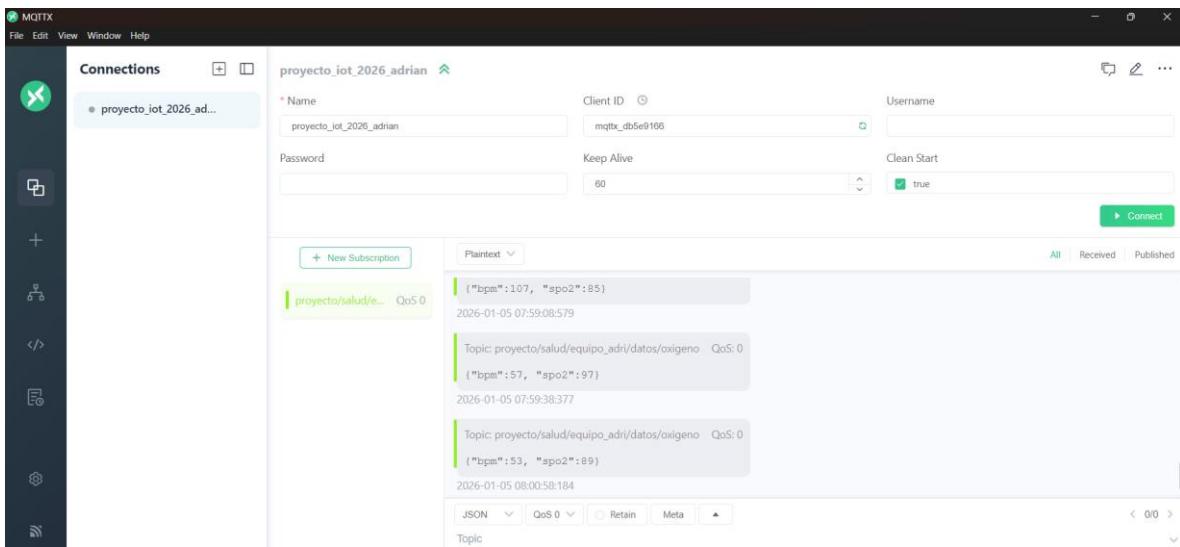
```
ultima_alerta = tiempo_actual
```

```
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"❌ Conexión perdida: {e}. Reintentando en 5 segundos...")  
            time.sleep(5)  
    except KeyboardInterrupt:  
        print("\n🔴 Sistema apagado por el usuario.")
```

### 7.3 CONFIGURACIÓN DEL BROKER



## 7.4 VISUALIZACIÓN DEL BOT Y ALERTAS EN TELEGRAM



## 7.5 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	