Sistema Domótico Hospitalario Para Habitacion Aislada

Anny Juliana Acosta
Juana Valentina Monsalve
Luisa Castaño Sepulveda

noviembre de 2025

Índice

1. Resumen

Este proyecto consiste en la implementación de un sistema de domótica hospitalaria basado en el microcontrolador **ESP32** programado en **MicroPython**.

El sistema monitorea en tiempo real una habitación hospitalaria mediante sensores de:

- Temperatura y humedad (DHT11)
- Calidad del aire / gas MQ135
- LDR para luminosidad ambiental
- Sensor MPU6050 como detector de apertura de puerta
- Botón físico de pánico

Con base en las lecturas, el sistema controla actuadores como lámparas, ventilación, calefactor, alarma audible y un LED RGB de estado.

Además, el sistema cuenta con:

- Servidor web embebido para visualización y control
- Comunicación bidireccional mediante **Telegram**
- Modos automático y manual seleccionables por el usuario

El objetivo es replicar funciones reales de una habitación hospitalaria inteligente.

2. Descripción del hardware

Componentes principales

- **ESP32**: microcontrolador principal.
- **Sensor DHT11**: lectura de temperatura y humedad.
- **MQ135**: detección de gases.
- **LDR**: medición de nivel de luz ambiental.
- **MPU6050**: detección de movimiento para apertura de puerta mediante acelerómetro.
- **LED RGB**: muestra el estado del sistema (normal, alerta, información).
- **Buzzer**: señal sonoro ante alarmas o condición de pánico.
- **Botón de pánico**: activa alarmas críticas.
- **Relés o drivers**: activan actuadores como ventilación, calefacción y luces.

Configuración de pines del ESP32

Componente	Pin
DHT11	25
MQ135	34 (ADC)
LDR	35 (ADC)
Botón de pánico	18
MPU6050 (I2C)	SDA 21 / SCL 22
Buzzer	16
RGB LEDs	R=14, G=12, B=15
Ventilación	27 / 13
Luz techo	32 / 33
Calefactor	26 / 4

Circuito final

3. Descripción del software

Lenguaje: MicroPython **IDE :** Thonny

Librerías utilizadas

- `machine`
- `network`
- `socket`
- `urequests`
- `struct`
- `json`
- `dht`
- `gc` y `micropython` para optimización de memoria

Flujo de ejecución

1. Conexión automática al Wi-Fi configurado.
2. Inicialización de sensores, actuadores y variables globales.
3. Apertura de servidor web en el ESP32.
4. Comprobación constante del bot de Telegram:
    - comandos de estado
    - cambio de modo
    - activación de actuadores

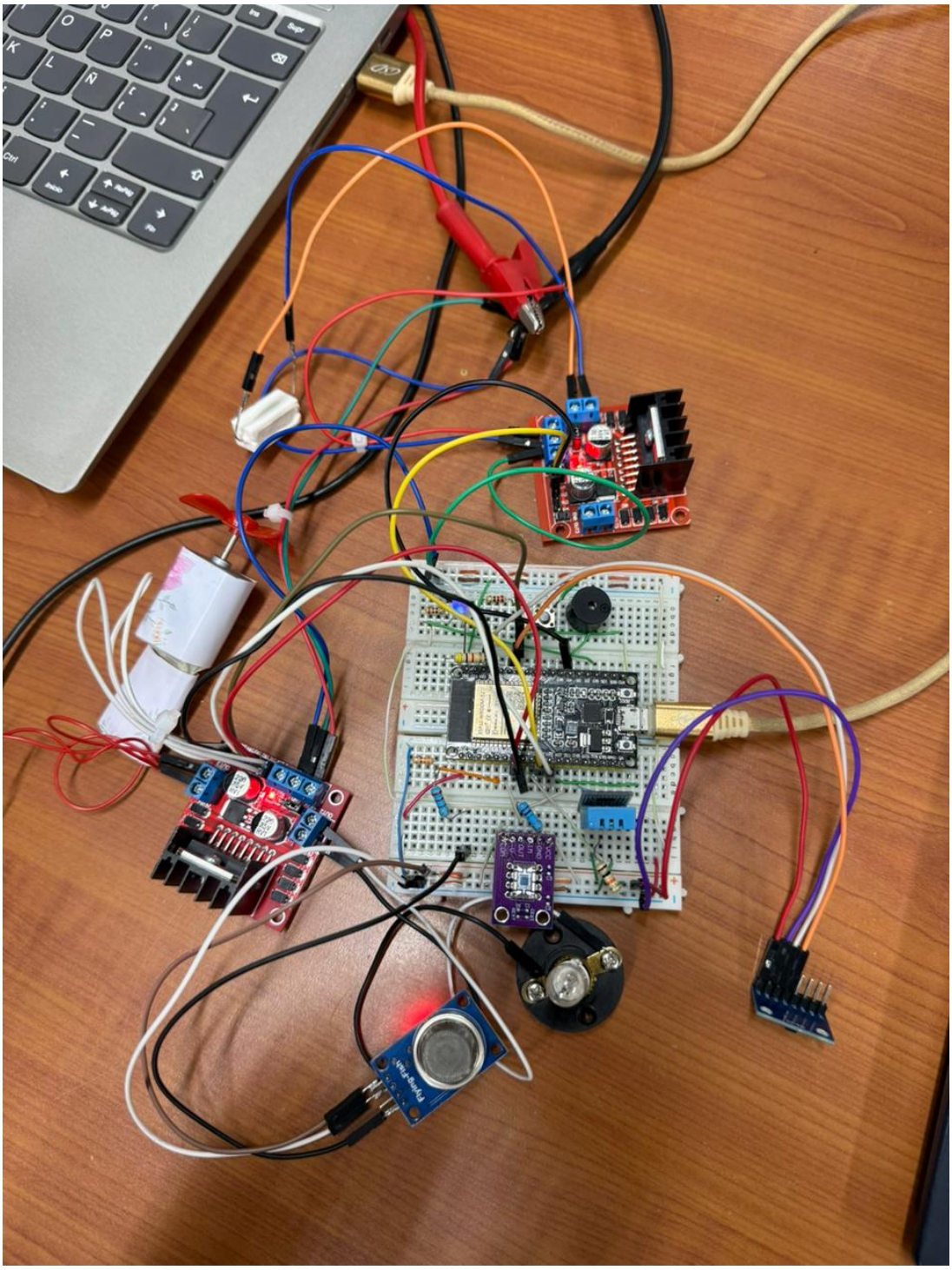

Figura 1: Circuito implementado en protoboard.

5. Lectura de sensores cada 1 segundo.
6. Ejecución automática de acciones según el modo seleccionado.

4. Estructura del código

- **Configuración:** credenciales Wi-Fi, token de Telegram y umbrales.

- **Inicialización de hardware:**
  - Sensores (DHT11, MQ135, LDR, MPU6050)
  - LED RGB y buzzer
  - Salidas para ventilación, calefacción y luces
- **Servidor Web:**
  - Página HTML con botones y lectura en tiempo real
  - Controles utilizando parámetros GET
- **Módulo Telegram:**
  - Envío de mensajes mediante API HTTP
  - Recepción de comandos como:
    - /estado
    - /silenciar
    - /manual
    - /auto
    - /luz on / off
    - /vent on / off
    - /calef on / off
- **Bucle principal:**
  - Lectura de sensores
  - Actualización de variables globales
  - Activación automática de actuadores
  - Gestión de alarmas y pánico

5. Explicación de funcionamiento

Modo Automático

El ESP32 toma decisiones sin intervención del usuario:

- Activa calefacción si la temperatura cae por debajo de `22°C`.
- Activa ventilación si el gas supera el umbral.
- Enciende luces si la habitación está oscura y la puerta está abierta.

Modo Manual

El usuario controla cada dispositivo vía Telegram o web:

- Luz, calefacción y ventilador ON/OFF
- Cambio de modo con un comando

Alarmas

- **Botón físico de pánico** Activa buzzer, cambia el LED y envía un mensaje a Telegram.
- **Gas peligroso** Detecta valores altos y envía alertas periódicamente.
- **Puerta abierta** Cambia estado del sistema.

6. Comunicación

- **Wi-Fi** El ESP32 se conecta como cliente a la red local.
- **Servidor Web embebido** Construido con el módulo `socket`. Permite:
  - Visualizar valores de sensores
  - Conmutar actuadores
  - Cambiar modo de operación
- **Telegram** Comunicación bidireccional mediante API HTTP:
  - Consulta de estado
  - Activación de dispositivos
  - Silenciar buzzer
  - Cambiar entre modo manual / automático

7. Interfaz de usuario

Página Web

La interfaz web incluye:

- Lecturas en tiempo real
- Estado de puerta, gas y sistema
- Botones para luces, ventilación, calefacción
- Autoactualización cada 5 segundos

Telegram

- Reporte en cualquier momento con `/estado`
- Alarmas instantáneas en caso de pánico o gas
- Menú de ayuda `/start`

8. Procedimiento de pruebas

1. Cargar código al ESP32 y reiniciar el microcontrolador.
2. Confirmar conexión Wi-Fi.
3. Acceder desde navegador al servidor web.
4. Verificar lectura de sensores.
5. Probar modo automático elevando gas o bajando luz.
6. Ingresar comandos por Telegram como:
   - /estado
   - /manual
   - /luz on
7. Activar botón de pánico y observar respuesta:
   - buzzer activo
   - reporte inmediato por Telegram
   - LED rojo

9. Manejo de errores y seguridad

- Se usa gestión de excepciones en lectura de sensores.
- Uso de `gc.collect()` para optimizar memoria.
- Comunicación sensible se realiza por HTTPS.
- Control remoto restringido al chat ID configurado.
- Se recomienda implementar watchdog para mayor robustez.