🚨 Proyecto: Alarma Inteligente con LilyGO T-Display v1.1

🔧 Descripción general

Este proyecto usa una placa LilyGO T-Display v1.1 que funciona como el cerebro del sistema. Se conecta a sensores que detectan apertura de puertas, movimiento o rotura de vidrio. Cuando la alarma está activada y alguno de estos sensores detecta algo, la placa envía una señal a un relé que enciende una sirena para alertar de la intrusión.

La LilyGO tiene también una pequeña pantalla que muestra mensajes como “alarma activada” o “alarma desactivada”, e indica qué zona se activó (por ejemplo, “puerta abierta” o “movimiento detectado”). Todo esto funciona gracias a un programa en MicroPython que revisa constantemente los sensores y actualiza la pantalla.

Además, la placa crea una página web a la que se puede acceder desde el celular o la computadora. Desde esa página se puede activar o desactivar la alarma y ver en tiempo real el estado de cada sensor, como si fuera un panel de control.

🛠️ Componentes utilizados

- LilyGO T-Display v1.1 (ESP32)
- Sensor magnético (puerta principal)
- Sensor PIR (movimiento)
- Sensor de rotura de vidrio
- Módulo de relé (para controlar la sirena)
- Fuente externa de 5V para alimentar el módulo de relé
- Fuente externa de 12v para alimentar Pir y Rotura
- Conexión Wi-Fi para el control web

🛠️ Diagrama de conexión

- **Sensor de puerta (magnético):** GPIO27
- **Sensor PIR:** GPIO26
- **Sensor rotura:** GPIO25
- **Relé para sirena:** GPIO2
- **Pantalla LCD (ST7789):** Pines SPI + control:
- SCK: GPIO18
- MOSI: GPIO19
- CS: GPIO5
- DC: GPIO16
- RESET: GPIO23
- Backlight: GPIO4

📲 Funcionamiento

- La placa se conecta al Wi-Fi.
- La pantalla LCD muestra: Estado de la alarma (ACTIVADA o DESACTIVADA)
- Estado de cada zona (puerta, PIR, rotura)
- Un servidor web permite: Activar o desactivar la alarma
- Ver en tiempo real los estados (se actualiza cada 2 segundos)
- Si la alarma está activada y se detecta una intrusión, se activa la sirena a través del relé.

🧩 Explicación del código

1 Importaciones

Se importan módulos de control de pines, red Wi-Fi, servidor web, pantalla y tiempo.

2 Configuración de hardware

Se definen los sensores como entradas con pull-up y el relé como salida.

3 Pantalla LCD

Se inicializa la pantalla ST7789 usando SPI para mostrar información del sistema.

4 Conexión Wi-Fi

Se conecta a la red usando el SSID y contraseña configurados.

5 Estado de la alarma

Se utiliza una variable `alarma_activada` para determinar si la alarma está activa.

6 Actualización de pantalla

Función `actualizar_pantalla()` actualiza solo las partes que cambiaron, evitando parpadeo.

7 Página web

Función `generar_web()` devuelve el HTML que muestra el estado actual y botones para activar/desactivar.

8 Servidor web

Atiende peticiones de los navegadores y actualiza el estado de la alarma según los botones presionados.

🌐 Interfaz web

- Botones "Activar" y "Desactivar"
- Estado de cada zona visible
- Refresco automático cada 2 segundos para ver cambios en vivo

✅ Conclusión

Este proyecto combina hardware (sensores y relé) con software en MicroPython para crear una alarma conectada y fácil de controlar desde el celular o PC. Además, la pantalla LCD permite visualizar rápidamente el estado del sistema.

🧩 1 Importaciones

```
from machine import Pin, SPI

import network

import socket

import st7789

import vga1_8x8 as font

import time
```

✅ Importamos librerías para:

- Pines (`Pin`)
- Comunicación SPI (para pantalla)
- Red Wi-Fi (`network`)
- Servidor web (`socket`)
- Control de pantalla LCD (`st7789`)
- Fuente para escribir texto (`vga1_8x8`)
- Temporización (`time`)

🛠️ 2 Configuración de sensores y relé

```
# Sensores conectados a GPIO con resistencia pull-up

sensor_puerta = Pin(27, Pin.IN, Pin.PULL_UP)

sensor_pir = Pin(26, Pin.IN, Pin.PULL_UP)

sensor_rotura = Pin(25, Pin.IN, Pin.PULL_UP)


# Relé que activa la sirena

rele_sirena = Pin(2, Pin.OUT)

rele_sirena.value(1)  # Inicia apagado (1 = desactivado porque es activo bajo)
```

✅ Detectamos:

- Sensor magnético de puerta
- Sensor PIR (movimiento)

- Sensor de rotura de vidrio

Todos devuelven **0** cuando están "tranquilos" y **1** cuando detectan algo.

3 Inicialización de la pantalla LCD

```
spi = SPI(1, baudrate=40000000, sck=Pin(18), mosi=Pin(19))
tft = st7789.ST7789(spi, 135, 240,
                    reset=Pin(23, Pin.OUT),
                    cs=Pin(5, Pin.OUT),
                    dc=Pin(16, Pin.OUT),
                    backlight=Pin(4, Pin.OUT),
                    rotation=1)
tft.init()
```

✅ Configuramos el driver ST7789 para la LilyGO T-Display

- `spi`: define la velocidad y los pines usados
- `tft.init()`: inicializa y limpia la pantalla

🌐 4 Conexión a la red Wi-Fi

```
ssid = "Luzmi"
password = "40696991"
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(ssid, password)
while not wlan.isconnected():
    time.sleep(1)
print("Conectado! IP:", wlan.ifconfig()[0])
```

✅ Nos conectamos a la red y mostramos la IP.

🔁 5 Variables de estado

```
alarma_activada = False


# Estados previos para saber qué cambió y actualizar solo eso en pantalla
prev_estado = {
    "alarma": None,
    "puerta": None,
    "pir": None,
    "rotura": None
}
```

✅ `alarma_activada` indica si está ON o OFF
✅ `prev_estado` evita parpadeo: solo actualiza la pantalla si cambió algo

🖉 6 Función para dibujar texto

```
def draw_text(label, value, y, color=st7789.WHITE):
    tft.fill_rect(10, y, 220, 20, st7789.BLACK)  # borra solo la línea
    tft.text(font, f"{label}: {value}", 10, y, color)
```

✅ Borra la zona y escribe el texto (evita que se "ensucie" la pantalla).

📺 7 Función que actualiza la pantalla

```
def actualizar_pantalla():
    # Estado de la alarma
    if prev_estado["alarma"] != alarma_activada:
        texto = "ACTIVADA" if alarma_activada else "DESACTIVADA"
        color = st7789.GREEN if alarma_activada else st7789.RED
        draw_text("Alarma", texto, 10, color)
        prev_estado["alarma"] = alarma_activada

    # Zona1: puerta
    estado_puerta = sensor_puerta.value()
```

```
    if prev_estado["puerta"] != estado_puerta:
        txt = "Abierta" if estado_puerta else "Cerrada"
        draw_text("Zona1 Puerta", txt, 40)
        prev_estado["puerta"] = estado_puerta

    # Zona2: PIR
    estado_pir = sensor_pir.value()
    if prev_estado["pir"] != estado_pir:
        txt = "Detectado" if estado_pir else "No"
        draw_text("Zona2 PIR", txt, 60)
        prev_estado["pir"] = estado_pir

    # Zona3: rotura
    estado_rotura = sensor_rotura.value()
    if prev_estado["rotura"] != estado_rotura:
        txt = "Detectada" if estado_rotura else "No"
        draw_text("Zona3 Rotura", txt, 80)
        prev_estado["rotura"] = estado_rotura
```

✅ Solo actualiza las partes que cambiaron → más fluido, sin parpadeo.

🌐 8 Página web

```
def generar_web():     html = f"""
    <html>
    <head>
        <title>Alarma</title>
        <meta http-equiv="refresh" content="2">
    </head>
    <body>
    <h1>Alarma {"ACTIVADA" if alarma_activada else "DESACTIVADA"}</h1>
    <p>Zona1 Puerta: {"Abierta" if sensor_puerta.value() else "Cerrada"}</p>
```

```
    <p>Zona2 PIR: {"Detectado" if sensor_pir.value() else "No"}</p>
    <p>Zona3 Rotura: {"Detectada" if sensor_rotura.value() else "No"}</p>
    <a href="/activar"><button>Activar</button></a>
    <a href="/desactivar"><button>Desactivar</button></a>
    </body></html>"""
    return html
```

✅ Muestra estado y tiene botones para activar/desactivar

✅ `<meta refresh>` hace que se recargue cada 2 seg.

✒️ 9 Servidor web

```
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
s = socket.socket()
s.bind(addr)
s.listen(1)
s.setblocking(False)  # para que no bloquee el loop principal
print('Servidor web listo')
```

✅ Escucha en el puerto 80 todas las conexiones locales

10 Loop principal

```
last_screen_update = 0

while True:
    # Lógica de la alarma
    if alarma_activada:
        if sensor_puerta.value()==1 or sensor_pir.value()==1 or
sensor_rotura.value()==1:
            rele_sirena.value(0)  # activa sirena
        else:
            rele_sirena.value(1)
```

```
    else:
        rele_sirena.value(1)  # siempre apagada

    # Actualiza la pantalla cada 200 ms
    now = time.ticks_ms()
    if time.ticks_diff(now, last_screen_update) > 200:
        actualizar_pantalla()
        last_screen_update = now

    # Atiende web rápidamente
    try:
        conn, addr = s.accept()
        request = conn.recv(1024)
        request = str(request)
        if '/activar' in request:
            alarma_activada = True
        if '/desactivar' in request:
            alarma_activada = False
        response = generar_web()
        conn.send('HTTP/1.1 200 OK\nContent-Type: text/html\nConnection:
close\n\n')
        conn.sendall(response)
        conn.close()
    except:
        pass

    time.sleep(0.05)  # loop rápido, no bloquea
```