```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP_Mail_Client.h>
#include <Servo.h>

// --- DATOS WIFI ---
const char* WIFI_SSID = "XXXX";
const char* WIFI_PASSWORD = "XXXX";

// --- DATOS GMAIL ---
#define SMTP_HOST "smtp.gmail.com"
#define SMTP_PORT 465
#define AUTHOR_EMAIL "abiiledesma04@gmail.com"
#define AUTHOR_PASSWORD "qzdkbycyubxdxhvy"
#define RECIPIENT_EMAIL "cariledesma049@gmail.com"

// --- PINES ---
const int PIN_MQ2 = A0;
const int PIN_BOTON = 5;      // D1 (GPIO 5) -> Botón debe cerrar a GND
const int PIN_SERVO = 4;      // D2
const int PIN_BUZZER = 14;    // D5
const int PIN_LED = 12;       // D6 (Relé)

// --- AJUSTES ---
int UMBRAL_GAS = 550;         // Este valor lo ajustaremos viendo el
Monitor Serie
const int TIEMPO_CALENTAMIENTO = 20; // Segundos de espera al iniciar
para el sensor

// --- RELÉ ---
#define RELE_ENCENDIDO HIGH
#define RELE_APAGADO   LOW

SMTPSession smtp;
Servo miServo;

// --- VARIABLES ---
bool alarmaActiva = false;
bool correoGasEnviado = false;
unsigned long ultimoTiempoBoton = 0;
unsigned long tiempoAnteriorParpadeo = 0;
int estadoLuz = RELE_APAGADO;
```

```
bool servoAbierto = false;           // Para saber si la ventana está
abierta
unsigned long tiempoInicioServo = 0; // Para guardar la hora en que se
abrió
const unsigned long TIEMPO_APERTURA = 60000; // 1 minuto (en
milisegundos

void smtpCallback(SMTP_Status status);

void setup() {
  Serial.begin(115200);
  Serial.println("\n\n--- INICIO SISTEMA SENTECH ---");

  pinMode(PIN_BOTON, INPUT_PULLUP);
  pinMode(PIN_BUZZER, OUTPUT);
  pinMode(PIN_LED, OUTPUT);

  // Aseguramos que el relé arranque APAGADO
  digitalWrite(PIN_LED, RELE_APAGADO);

  // Inicializar Servo
  miServo.attach(PIN_SERVO);
  miServo.write(0); // Posición cerrada
  delay(500);
  miServo.detach();

  // --- CONEXIÓN WIFI ---
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Conectando WiFi");

  unsigned long inicioWifi = millis();
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
    // Si tarda más de 15 seg, seguimos igual para que la alarma
funcione offline
    if (millis() - inicioWifi > 15000) {
      Serial.println("\n(WiFi no conectado, continuando offline...)");
      break;
    }
  }
```

```
  if (WiFi.status() == WL_CONNECTED) Serial.println("\n¡WiFi
Conectado!");

  // --- CONFIG HORA Y EMAIL ---
  configTime(0, 0, "pool.ntp.org", "time.nist.gov");
  smtp.debug(1); // Modo debug activado para ver errores
  smtp.callback(smtpCallback);

  // --- CALENTAMIENTO DEL SENSOR (SILENCIOSO) ---
  Serial.println("Calentando sensor MQ-2 (Espere " +
String(TIEMPO_CALENTAMIENTO) + " seg)...");

  // Mantenemos la luz APAGADA fijamente mientras esperamos
  digitalWrite(PIN_LED, RELE_APAGADO);

  for(int i = 0; i < TIEMPO_CALENTAMIENTO; i++) {
    Serial.print(" " + String(TIEMPO_CALENTAMIENTO - i)); // Cuenta
regresiva en el monitor
    delay(1000);
  }

  Serial.println("\nSensor Listo y Calibrado.");
}

void enviarCorreo(String asunto, String mensaje) {
  // ------------------------------------------------------
  // 1. VERIFICACIÓN Y RECONEXIÓN (CON CONTROL DE SERVO)
  // ------------------------------------------------------
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("¡WiFi perdido! Buscando reconexión (El servo se
cerrará a tiempo)...");

    WiFi.disconnect();
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

    // BUCLE DE ESPERA "OBSTINADA"
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");


      yield(); //PAUSAR EL CÓDIGO, PERO VOLVER INMEDIATAMENTE
```

```
      if (servoAbierto) {
        if (millis() - tiempoInicioServo >= TIEMPO_APERTURA) {
          Serial.println("\n[AVISO] Se cumplió el minuto mientras
esperábamos WiFi. Cerrando Servo...");

          miServo.attach(PIN_SERVO);
          miServo.write(0);  // Cerrar
          delay(500);        // Espera segura
          miServo.detach();  // Apagar

          servoAbierto = false; // Marcar como cerrado
          Serial.println("Ventilación cerrada. Siguiendo con la
búsqueda de WiFi...");
        }
      }
      // ------------------------------------------------------------
    }

    Serial.println("\n¡Conexión recuperada! Procediendo a enviar...");
  }

  // ------------------------------------------------------
  // 2. ENVÍO DE CORREO
  // ------------------------------------------------------
  ESP_Mail_Session session;
  session.server.host_name = SMTP_HOST;
  session.server.port = SMTP_PORT;
  session.login.email = AUTHOR_EMAIL;
  session.login.password = AUTHOR_PASSWORD;
  session.login.user_domain = "";

  SMTP_Message message;
  message.sender.name = "Sentech Alarma";
  message.sender.email = AUTHOR_EMAIL;
  message.subject = asunto;
  message.addRecipient("Usuario", RECIPIENT_EMAIL);
  message.text.content = mensaje;

  Serial.println("Conectando con Gmail...");

  if (!smtp.connect(&session)) {
    Serial.println("Error conectando a Gmail.");
```

```
    return;
  }

  if (!MailClient.sendMail(&smtp, &message)) {
    Serial.println("Error enviando: " + smtp.errorReason());
  } else {
    Serial.println("¡CORREO ENVIADO CON ÉXITO!");
  }

  smtp.sendingResult.clear();
}

void activarAlarma() {
  // Solo entramos si la alarma no estaba ya activa para no repetir
acciones
  if (!alarmaActiva) {
    alarmaActiva = true;
    Serial.println("!!! ALARMA DE GAS ACTIVADA !!!");

    // --- 1. SERVO (ABRIR Y CRONOMETRAR) ---
    if (!servoAbierto) {
      miServo.attach(PIN_SERVO);
      miServo.write(180); // Abrir ventana/válvula

      servoAbierto = true;
      tiempoInicioServo = millis(); // Guardamos la hora actual
      Serial.println("Ventilación abierta. Se cerrará automáticamente
en 1 minuto.");
    }

    // --- 2. RUIDO ---
    tone(PIN_BUZZER, 1000);

    // --- 3. CORREO ---
    // Enviamos solo si no se ha enviado antes en este ciclo
    if (!correoGasEnviado) {
      enviarCorreo("ALERTA GAS", "Nivel critico detectado por sensor
MQ-2. Ventilación activada por 1 min.");
      correoGasEnviado = true;
    }
  }
}
```

```
void desactivarAlarma() {
  Serial.println("\n>>> ALARMA DETENIDA POR USUARIO <<<");

  // 1. APAGADO INMEDIATO DE MOLESTIAS (RUIDO Y LUZ)
  noTone(PIN_BUZZER);
  digitalWrite(PIN_LED, RELE_APAGADO); // Luz apagada inmediatamente

  // 2. ESPERA DE 1 MINUTO (SOLO VENTILACIÓN)
  Serial.println("Ruido y luz apagados. Manteniendo ventilación 60
segundos...");

  for (int i = 60; i > 0; i--) {
    Serial.print("Apagando sistema en: ");
    Serial.print(i);
    Serial.println(" s");


    // Solo esperamos 1 segundo en silencio y oscuridad.
    delay(1000);
  }

  // 3. CERRAR SERVO DESPUÉS DEL MINUTO
  Serial.println("Tiempo cumplido. Cerrando Servo...");

  miServo.attach(PIN_SERVO);
  miServo.write(0);  // Cerrar
  delay(1000);
  miServo.detach();
  servoAbierto = false;

  // 4. REINICIO
  Serial.println("Reiniciando ESP8266...");
  ESP.restart();
}

void loop() {
  // ============================================================
  // 1. LECTURA Y MONITOREO DEL SENSOR DE GAS (MQ-2)
  // ============================================================
  int lecturaGas = analogRead(PIN_MQ2);

  // Imprimir valor en Monitor Serie cada 2 seg para control
  static unsigned long ultimaImpresion = 0;
```

```
  if (millis() - ultimaImpresion > 2000) {
    Serial.print("[MONITOR] Gas: ");
    Serial.println(lecturaGas);
    ultimaImpresion = millis();
  }

  // Si supera el umbral Y la alarma NO está activa aún
  if (lecturaGas > UMBRAL_GAS && !alarmaActiva) {
    delay(100); // Filtro pequeño para evitar falsos positivos
    if (analogRead(PIN_MQ2) > UMBRAL_GAS) {
      activarAlarma();
    }
  }

  // ============================================================
  // 2. EFECTOS VISUALES (PARPADEO RELÉ)
  // ============================================================
  if (alarmaActiva) {
    // Parpadeo cada 1 segundo (sin delay, usando millis)
    if (millis() - tiempoAnteriorParpadeo >= 1000) {
      tiempoAnteriorParpadeo = millis();
      estadoLuz = (estadoLuz == RELE_APAGADO) ? RELE_ENCENDIDO :
RELE_APAGADO;
      digitalWrite(PIN_LED, estadoLuz);
    }
  } else {
    // Si no hay alarma, mantener apagado
    digitalWrite(PIN_LED, RELE_APAGADO);
  }

  // ============================================================
  // 3. CONTROL DE TIEMPO DEL SERVO (Cierre automático)
  // ============================================================
  if (servoAbierto) {
    // Comprobamos si ya pasó 1 minuto (TIEMPO_APERTURA)
    if (millis() - tiempoInicioServo >= TIEMPO_APERTURA) {
      Serial.println("Tiempo de ventilación cumplido. Cerrando...");

      miServo.attach(PIN_SERVO); // Reconectamos por si acaso
      miServo.write(0);          // Cerrar (0 grados)
      delay(500);                // Espera breve para que llegue
      miServo.detach();          // Apagar servo
```

```
      servoAbierto = false;       // Detener cronómetro
    }
  }

  // ============================================================
  // 4. LECTURA DEL BOTÓN (PÁNICO Y RESET)
  // ============================================================
  if (digitalRead(PIN_BOTON) == LOW) {
    delay(50); // Antirrebote

    if (digitalRead(PIN_BOTON) == LOW) {
      Serial.println(">>> BOTÓN PRESIONADO <<<");

      if (alarmaActiva) {
        // --- CASO A: APAGAR ALARMA ---
        desactivarAlarma();
      } else {
        // --- CASO B: BOTÓN DE PÁNICO ---
        // Evitamos enviar mails seguidos (wait 5 seg)
        if (millis() - ultimoTiempoBoton > 5000) {
          Serial.println("Protocolo Pánico iniciado...");



          enviarCorreo("ALERTA BOTON", "Boton pánico presionado. Se
necesita ayuda");

          ultimoTiempoBoton = millis();
        }
      }

      // Esperar a que sueltes el botón (Evita repeticiones)
      while(digitalRead(PIN_BOTON) == LOW) {
        delay(10);
      }
    }
  }
}

void smtpCallback(SMTP_Status status) {}
```