```
#include <Arduino.h>     //libreria base de arduino
#include <ESP8266WiFi.h>  //liberia base para controlar el chip wifi
del esp8266
#include <ESP_Mail_Client.h> //libreria para enviar correos
electronicos
#include <Servo.h>           //libreria para controlar el servomotor

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

// --- DATOS GMAIL ---
#define SMTP_HOST "smtp.gmail.com"      //servidor de google
#define SMTP_PORT 465                  //puerto seguro SSL para enviar
correos (gmail solo funciona con conexiones seguras)
#define AUTHOR_EMAIL "abiiledesma04@gmail.com" //autor del correo
#define AUTHOR_PASSWORD "qzdkbycyubxdxhvy"    //contraseña DE
APLICACIÓN
#define RECIPIENT_EMAIL "cariledesma049@gmail.com" //quien recibirá las
alertas (SE PUEDEN AMPLIAR LOS RECEPTORES)

// --- PINES ---
const int PIN_MQ2 = A0;       //sensor de gas al pin analógico A0
const int PIN_BOTON = 5;     // boton va al pin digital D1 (GPIO 5)
const int PIN_SERVO = 4;     // servomotor va al pin digital D2 (GPIO
4)
const int PIN_BUZZER = 14;   // buzzer va al pin digital D5 (GPIO 14)
const int PIN_LED = 12;      // led y rele van al pin digital D6 (GPIO
12)

// --- AJUSTES ---
int UMBRAL_GAS = 550;        // Nivel de gas necesario para activar la
alarma
const int TIEMPO_CALENTAMIENTO = 20; // Segundos de espera al iniciar
para el calentar y estabilizar al sensor

// --- RELÉ ---
#define RELE_ENCENDIDO HIGH //define que HIGH encienda el relé
#define RELE_APAGADO   LOW  // define que LOW apaga el relé

SMTPSession smtp;  //crea el objeto SMTP para manejar la sesión de
correo
Servo miServo;    //crea el objeto miServo para manejar el servo
```

```
// --- VARIABLES ---
bool alarmaActiva = false;      //inicializo bandera alarmaActiva en
falso
bool correoGasEnviado = false;   // inicializo bandera correoGasEnviado
en falso
unsigned long ultimoTiempoBoton = 0;  //inicializo el cronometro para
el boton de panico
unsigned long tiempoAnteriorParpadeo = 0; //inicializo el cronómetro
para el parpadeo de la luz
int estadoLuz = RELE_APAGADO;  //inicializo la variable estadoLuz como
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); //declaracion de la funcion de
callback para envio de correo. la funcion se define mas abajo

void setup() {
  Serial.begin(115200);   //inicia comunicacion con la PC para ver mjes
en pantalla
  Serial.println("\n\n--- INICIO SISTEMA SENTECH ---"); // imprime mje
de bienvenida

  // configuracion de modos de los pines
  pinMode(PIN_BOTON, INPUT_PULLUP);  //boton como entrada con
resistencia interna
  pinMode(PIN_BUZZER, OUTPUT);  //buzzer como salida
  pinMode(PIN_LED, OUTPUT);    //led o relé como salida


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

  // Inicializar Servo
  miServo.attach(PIN_SERVO); //conecta el software del servo al pin
  miServo.write(0); // Posición cerrada
  delay(500);      //espera medio segundo para que llegue a la posicion
```

```
  miServo.detach(); // desconecta el servo para que no vibre ni gaste
energía

  // --- CONEXIÓN WIFI ---
  WiFi.mode(WIFI_STA);  //define su comportamiento, en este caso es un
cliente, busca un router para tener internet
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD); //inicia el intento de conexion
  Serial.print("Conectando WiFi");

  unsigned long inicioWifi = millis(); //guarda la hora de inicio del
intento wifi
  //bucle mientras no este conectado
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print("."); //imprime puntitos de espera
    delay(500);    //espera medio segundo entre intentos
    // 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;  //rompe el bucle while y sigue con el programa
    }
  }
  if (WiFi.status() == WL_CONNECTED) Serial.println("\n¡WiFi
Conectado!");

  // --- CONFIG HORA Y EMAIL ---
  configTime(0, 0, "pool.ntp.org", "time.nist.gov");  //sincroniza la
hora con internet (necesario para SSL)
  smtp.debug(1); // muestra errores detallados de correo si es que los
hay
  smtp.callback(smtpCallback); //le avisa al sistema que ejecute la
funcion smtpCallBack para que yo me entere si se envió o no el correo

  // --- 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);

  //bucle de espera simple para calentar el sensor
  for(int i = 0; i < TIEMPO_CALENTAMIENTO; i++) {
```

```
    Serial.print(" " + String(TIEMPO_CALENTAMIENTO - i)); // Cuenta
regresiva en el monitor
    delay(1000); //espera 1 segundo
  }

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

void enviarCorreo(String asunto, String mensaje) {
  // ------------------------------------------------------
  // 1. VERIFICACIÓN Y RECONEXIÓN
  // ------------------------------------------------------
  if (WiFi.status() != WL_CONNECTED) { //pregunta si estan DESCONECTADO
del wifi
    Serial.println("¡WiFi perdido! Buscando reconexión (El servo se
cerrará a tiempo)..."); //avisa que hay un problema

    WiFi.disconnect(); //desconecta cualquier intento fallido para
empezar desde cero
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD); //inicia proceso de conexión

    // BUCLE DE ESPERA "OBSTINADA"
    // Este 'while' es un bucle que se repite infinitamente MIENTRAS NO
haya conexión.
    // El programa NO avanza de aquí hasta que recupere el internet.
    while (WiFi.status() != WL_CONNECTED) {
      delay(500); //espera medio segundo entre cada verificacion para
no saturar al chip
      Serial.print("."); //imprime un puntito para que veamos que esta
trabajando

      // COMANDO CRÍTICO: yield().
      // Como este bucle puede durar mucho tiempo, 'yield()' le dice al
procesador:
      // "Haz tus tareas de fondo (como mantener el WiFi vivo) para no
crashear".
      // Sin esto, el ESP8266 se reiniciaría solo por seguridad
(Watchdog Timer).
      yield();

      // --- LÓGICA DE SEGURIDAD DEL SERVO (DENTRO DEL BUCLE DE WIFI)
---
```

```
      // Problema: Si el WiFi tarda 5 minutos en volver, el código estaría atrapado en este 'while'.
      // Sin este bloque, la ventana (servo) se quedaría abierta 5 minutos, ignorando el límite de 1 min.
      // Aquí revisamos el tiempo MANUALMENTE mientras esperamos el internet.

      if (servoAbierto) { //pregunto, la ventana esta abierta?
        if (millis() - tiempoInicioServo >= TIEMPO_APERTURA) {// ... y ya pasó el tiempo límite (1 minuto)
          Serial.println("\n[AVISO] Se cumplió el minuto mientras esperábamos WiFi. Cerrando Servo...");

          miServo.attach(PIN_SERVO); //conecta el servomotor al pin
          miServo.write(0);  // Cierra la ventana (mueve el servo a 0grados)
          delay(500);        // Espera segura
          miServo.detach();  // desconecta el servo para que deje de hacer fuerza

          servoAbierto = false; //avisa que la ventana cerró y sigue tratando de conectar al wifi
          Serial.println("Ventilación cerrada. Siguiendo con la búsqueda de WiFi...");
        }
      }
      // ------------------------------------------------------------
    } //terminó el while, sali de aqui pq ya se pudo conectar al wifi

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

  // ------------------------------------------------------
  // 2. ENVÍO DE CORREO
  // ------------------------------------------------------
  ESP_Mail_Session session; //crea objeto termporal "session" para guardar configuracion tecnica de gmail
  session.server.host_name = SMTP_HOST; //asigna la dirección del servidor
  session.server.port = SMTP_PORT; //asigna el puerto seguro
  session.login.email = AUTHOR_EMAIL; //asigna mi correo (remitente) para iniciar sesion
```

```
  session.login.password = AUTHOR_PASSWORD; //asigna mi contraseña de aplicacion
  session.login.user_domain = ""; //deja el dominio de usuario vacio (no necesario para gmail)

  SMTP_Message message; //crea objeto message para guardar el texto y los destinatarios
  message.sender.name = "Sentech Alarma"; //nombre que aparecerá como remitente
  message.sender.email = AUTHOR_EMAIL; //confirma el correo desde donde sale
  message.subject = asunto; //asigna el asunto del correo
  //añade al destinatario (etiqueta que aparecera en la bandeja de entrada) y a donde se enviara el mje
  message.addRecipient("Usuario", RECIPIENT_EMAIL);
  message.text.content = mensaje; //cuerpo del correo

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

  if (!smtp.connect(&session)) { //si la conexion falla, sale
    Serial.println("Error conectando a Gmail.");
    return;
  }

  if (!MailClient.sendMail(&smtp, &message)) {
    Serial.println("Error enviando: " + smtp.errorReason()); //si no puede enviar, nos avisa
  } else {
    Serial.println("¡CORREO ENVIADO CON ÉXITO!"); //si pudo enviar el correo
  }

  smtp.sendingResult.clear(); //libera la memoria que se usó para guardar el reporte del envío
}

void activarAlarma() {

  if (!alarmaActiva) { //solo si la alarma ya estaba activa
    alarmaActiva = true; //marcla la alarama activa
    Serial.println("!!! ALARMA DE GAS ACTIVADA !!!");

    // --- 1. SERVO (ABRIR Y CRONOMETRAR) ---
```

```
    if (!servoAbierto) {
      miServo.attach(PIN_SERVO); //conecta a servo
      miServo.write(180); // Abrir ventana/válvula

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

    // --- 2. RUIDO ---
    tone(PIN_BUZZER, 1000); //enciende el buzzer a 1000hz

    // --- 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."); //llama a la funcion de correo
      correoGasEnviado = true; //bloquea futuros envios de correos por
gas
    }
  }
}

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...");
  //cuenta regresiva
  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 ventilación...");

  miServo.attach(PIN_SERVO); //conecta el servo al pin
  miServo.write(0);  // cierra la ventana
  delay(1000);       //espera 1 segundo
  miServo.detach();  //desconecta el servo para que no vibre ni gaste
energia
  servoAbierto = false; //avisa que la ventana esta cerrada

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

void loop() {
  // ============================================================
  // 1. LECTURA Y MONITOREO DEL SENSOR DE GAS (MQ-2)
  // ============================================================
  int lecturaGas = analogRead(PIN_MQ2); //lee el valor del sensor
(señal analogica entre 0 y 1024)

  // 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) { //si el valor de la lectura
del sensor es mayor al umbral calibrado
      activarAlarma(); //activamos la alarma
    }
  }

  // ============================================================
  // 2. EFECTOS VISUALES (PARPADEO RELÉ)
```

```
  // ============================================================
  if (alarmaActiva) {
    // Parpadeo cada 1 segundo
    if (millis() - tiempoAnteriorParpadeo >= 1000) {
      tiempoAnteriorParpadeo = millis();
      estadoLuz = (estadoLuz == RELE_APAGADO) ? RELE_ENCENDIDO :
RELE_APAGADO; //operador ternario. si la luz estaba apagada, prendela,
y viceversa
      digitalWrite(PIN_LED, estadoLuz); //aplica el cambio
    }
  } 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) { //si se presiona el boton
    delay(50); // Antirrebote

    if (digitalRead(PIN_BOTON) == LOW) { //confirma presion del boton
      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 suelte el botón (Evita repeticiones)
      while(digitalRead(PIN_BOTON) == LOW) {
        delay(10);
      }
    }
  }
}

void smtpCallback(SMTP_Status status) {} //aqui se verifica si el
correo salio o no bien
```