Universidad Autónoma del Caribe

|  |
| --- |
| MANUAL DE USUARIO |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Yamith J. Romero Aldana  Elian D. Ruidiaz Marino | Saul A. Pérez Pérez  Carlos G. Díaz Sáenz  Pablo D. Bonaveri |  |

TABLA DE CONTENIDO

[INTRODUCCIÓN 3](#_Toc208218609)

[Información General 3](#_Toc208218610)

[Descripción General 3](#_Toc208218611)

[CARACTERISTICAS TECNICAS 4](#_Toc208218612)

[Arquitectura del Software 4](#_Toc208218613)

[Especificaciones 4](#_Toc208218614)

[Sensores MQ2 y MQ3 4](#_Toc208218615)

[MQ3 (Sensor de Alcohol): 4](#_Toc208218616)

[HARDWARE COMPATIBLE 5](#_Toc208218617)

[Microcontrolador Principal 5](#_Toc208218618)

[REQUERIMIENTOS DE INSTALACIÓN Y CONFIGURACIÓN 6](#_Toc208218619)

[Requisitos del Sistema 6](#_Toc208218620)

[Dependencias de Software 6](#_Toc208218621)

[Proceso de Instalación 6](#_Toc208218622)

[INICIO RÁPIDO 7](#_Toc208218623)

[ALGORITMOS IMPLEMENTADOS 8](#_Toc208218624)

[APLICACIONES EDUCATIVAS 9](#_Toc208218625)

[APÉNDICES 10](#_Toc208218626)

# INTRODUCCIÓN

## Información General

**Nombre del Software:**SENSORA\_GAS\_REGULATION

**Fecha de Desarrollo:** 2025  
**Plataforma:** Python 3.x con PySide6  
**Categoría:** Software Educativo/Módulo Didáctico

## Descripción General

El módulo **SENSORA\_GAS\_REGULATION** proporciona una solución completa para el monitoreo y análisis de concentraciones de gases utilizando sensores electroquímicos de la serie MQ. Este sistema está diseñado para aplicaciones de seguridad industrial, control ambiental y monitoreo de calidad del aire, ofreciendo mediciones precisas en tiempo real con capacidades de calibración avanzadas.

# CARACTERISTICAS TECNICAS

## Arquitectura del Software

* **Sensores**: MQ2 (humo/gases combustibles) y MQ3 (alcohol/etanol)
* **Medición**: Conversión analógica ADC 12-bit (0-4095) y voltaje (0-3.3V)
* **Calibración**: Sistema matemático basado en ecuaciones logarítmicas Rs/Ro
* **Visualización**: Gráficas PPM en tiempo real con dos canales
* **Exportación**: Datos completos en Excel con hoja de calibración
* **Comunicación**: TCP/IP con ESP32 y selección dinámica de sensor

## Especificaciones

### Sensores MQ2 y MQ3

**MQ2 (Sensor de Humo/Gases Combustibles)**:

* **Gases detectables**: LPG, propano, metano, hidrógeno, humo
* **Rango**: 200-10,000 ppm
* **Temperatura operativa**: -10°C a 50°C
* **Humedad**: 5% a 95% RH (sin condensación)
* **Tiempo de respuesta**: < 10 segundos

### MQ3 (Sensor de Alcohol):

* **Gases detectables**: Etanol, alcohol etílico, vapor de alcohol
* **Rango**: 0.05-10 mg/L (equivalente a 50-10,000 ppm)
* **Selectividad**: Alta sensibilidad al alcohol, baja interferencia
* **Tiempo de respuesta**: < 10 segundos
* **Recuperación**: < 30 segundos

# HARDWARE COMPATIBLE

## Microcontrolador Principal

**Modelo**: ESP32 DevKit V1

**Conectividad**: WiFi integrado para comunicación TCP

**Alimentación**: 5V/3.3V dual para sensor y lógica

Conexiones de Sensor

ESP32 Pin Sensor MQ Función

5V → VCC Alimentación heater

3V3 → VCC\_Logic Alimentación lógica

GND → GND Tierra común

D36/VP ← A0 Salida analógica MQ2

D39/VN ← A0 Salida analógica MQ3

# REQUERIMIENTOS DE INSTALACIÓN Y CONFIGURACIÓN

## Requisitos del Sistema

* **Sistema Operativo:** Windows 10/11
* **Python:** Versión 3.8 o superior
* **RAM:** Mínimo 2GB, recomendado 4GB

## Dependencias de Software

*# requirements.txt*

PyQt5>=5.15.0

matplotlib>=3.5.0

numpy>=1.21.0

scipy>=1.7.0

openpyxl>=3.0.9

pyserial>=3.5

pandas>=1.3.0

## Proceso de Instalación

1. **Clonar Repositorio:** Descargar código fuente
2. **Instalar Dependencias:** pip install -r requirements.txt
3. **Configurar ESP32:** Flashear firmware compatible
4. **Ejecutar Software:** python main.py

# INICIO RÁPIDO

**Inicio del Sistema**

1. **Abrir aplicación** SENSORA CORE
2. **Seleccionar módulo** "GAS REGULATION"
3. **Precalentar sensores** MQ2/MQ3 (mínimo 20 segundos)
4. **Introducir IP** del ESP32 en campo principal
5. **Verificar conexión** antes de iniciar monitoreo

**Interfaz de Usuario**

* **Panel Superior**: Título y descripción del módulo
* **Panel Izquierdo**: Selector de sensores y datos en tiempo real
* **Panel Central**: Gráfica concentración PPM temporal
* **Controles**: Iniciar/pausar, limpiar, exportar, calibrar

**Selección de Sensor**

1. **MQ2**: Presionar botón "MQ2" (activación exclusiva)
2. **MQ3**: Presionar botón "MQ3" (desactiva MQ2)
3. **Indicador visual**: Botón seleccionado se resalta
4. **Cambio dinámico**: Posible durante monitoreo activo

**Proceso de Calibración**

**Calibración Básica (Solo Ro)**:

1. **Ambiente limpio**: Colocar sensor en aire puro
2. **Presionar "Calibrar"**: Introducir valor Ro en Ω
3. **Confirmar**: Valor se guarda automáticamente

**Calibración Completa (Ecuación PPM)**:

1. **Iniciar calibración avanzada**: Especificar 2+ puntos
2. **Punto 1**: Introducir Rs actual y PPM de referencia
3. **Punto 2**: Repetir con concentración diferente
4. **Puntos adicionales**: Opcional para mejor precisión
5. **Ajuste automático**: Sistema calcula m y b de ecuación logarítmica

**Monitoreo en Tiempo Real**

1. **Iniciar streaming**: Presionar "Iniciar Monitoreo"
2. **Observar datos**: ADC crudo, voltaje y PPM calibrado
3. **Analizar gráfica**: Líneas MQ2 y MQ3 diferenciadas
4. **Cambiar sensor**: Selección dinámica durante operación
5. **Pausar**: Detener streaming manteniendo datos históricos

# ALGORITMOS IMPLEMENTADOS

Ecuación Fundamental de Sensores MQ

Rs = RL \* (Vcc - Vout) / Vout

Donde:

* **Rs**: Resistencia sensor variable con concentración
* **RL**: Resistencia de carga (10kΩ típico)
* **Vcc**: Voltaje alimentación (3.3V)
* **Vout**: Voltaje medido por ADC

Modelo Logarítmico de Calibración

log10(PPM) = m \* log10(Rs/Ro) + b

Donde:

* **Ro**: Resistencia en aire limpio (referencia)
* **m, b**: Parámetros determinados por regresión lineal
* **Rs/Ro**: Ratio normalizado para compensar deriva

Algoritmo de Calibración

def calibrate\_sensor(points):

xs = [log10(rs/ro) for rs, ppm in points]

ys = [log10(ppm) for rs, ppm in points]

m, b = linear\_regression(xs, ys)

return m, b

def compute\_ppm(adc\_value, calibration):

volt = (adc\_value / 4095.0) \* 3.3

Rs = RL \* (3.3 - volt) / volt

ratio = Rs / calibration.Ro

return 10 \*\* (calibration.m \* log10(ratio) + calibration.b)

**Visualización en Tiempo Real**

* **Líneas diferenciadas**: MQ2 y MQ3 con colores únicos
* **Eje Y**: Concentración en PPM (escala logarítmica opcional)
* **Eje X**: Tiempo relativo en segundos
* **Actualización**: Tiempo real con optimización draw\_idle()

# APLICACIONES EDUCATIVAS

**Instrumentación Científica**

* **Transductores químicos**: Conversión química-eléctrica
* **Acondicionamiento señales**: Divisores voltaje y amplificación
* **Procesamiento digital**: ADC y algoritmos compensación
* **Sistemas de adquisición**: Comunicación tiempo real

**Proyectos de Ingeniería**

* **Sistemas de alarma**: Detección automática y notificaciones
* **Control industrial**: Integración con PLCs y SCADA
* **Monitoreo ambiental**: Redes sensores distribuidos
* **Instrumentación virtual**: Interfaz usuario profesional

**Seguridad e Higiene Industrial**

* **LEL (Lower Explosive Limit)**: Detección concentraciones peligrosas
* **Ventilación controlada**: Activación automática extractores
* **Registro histórico**: Cumplimiento normativas ambientales
* **Capacitación personal**: Interpretación mediciones gases

# APÉNDICES

**Apéndice A: Glosario de Términos**

**ADC (Analog-to-Digital Converter)**: Convertidor analógico-digital que transforma señales analógicas en valores digitales.

**API (Application Programming Interface)**: Interfaz de programación que define métodos de comunicación entre componentes de software.

**ESP32**: Microcontrolador de Espressif Systems con WiFi y Bluetooth integrados, base del sistema de instrumentación.

**GPIO (General Purpose Input/Output)**: Pines de propósito general configurables como entrada o salida digital.

**GUI (Graphical User Interface)**: Interfaz gráfica de usuario que permite interacción visual con la aplicación.

**Handshake**: Proceso de establecimiento de comunicación entre dos dispositivos para verificar conectividad.

**I2C (Inter-Integrated Circuit)**: Protocolo de comunicación serie síncrono para comunicación entre microcontroladores y sensores.

**IoT (Internet of Things)**: Red de dispositivos físicos conectados que pueden intercambiar datos.

**Latencia**: Tiempo de retardo entre el envío de un comando y la recepción de su respuesta.

**LDR (Light Dependent Resistor)**: Resistor dependiente de luz, sensor básico de luminosidad.

**PID (Proportional-Integral-Derivative)**: Algoritmo de control automático usado en sistemas de regulación.

**PWM (Pulse Width Modulation)**: Modulación por ancho de pulso, técnica para controlar potencia de salida.

**Qt**: Framework multiplataforma para desarrollo de interfaces gráficas de usuario.

**SensoraCore**: Nombre del sistema completo de instrumentación científica distribuida.

**TCP/IP (Transmission Control Protocol/Internet Protocol)**: Conjunto de protocolos de comunicación para redes.

**Timeout**: Tiempo máximo de espera antes de considerar una operación como fallida.

**UART (Universal Asynchronous Receiver-Transmitter)**: Protocolo de comunicación serie asíncrono.

**UI (User Interface)**: Interfaz de usuario, punto de interacción entre humano y máquina.

**WiFi**: Tecnología de comunicación inalámbrica basada en estándares IEEE 802.11.

**Apéndice B: Referencias y Documentación Adicional**

**Documentación Técnica**

* **Qt6 Documentation**: <https://doc.qt.io/qt-6/>
* **Python Official Documentation**: <https://docs.python.org/3/>
* **ESP32 Technical Reference**: <https://docs.espressif.com/projects/esp-idf/>
* **PySide6 Reference**: <https://doc.qt.io/qtforpython/>

**Recursos de Desarrollo**

* **GitHub Repository**: <https://github.com/YamithR/SensoraCore>

**Contacto y Soporte**

* **Email de Soporte**: [yamith.romero@uac.edu.co](mailto:yamith.romero@uac.edu.co)

Copyright © 2025 SensoraCore Project. **Versión del documento: 3.2 | Fecha: Agosto 2025** Para la versión más actualizada de este manual, visite: <https://github.com/YamithR/SensoraCore.git>