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](#_Toc208142294)

[Información General 3](#_Toc208142295)

[Descripción General 3](#_Toc208142296)

[CARACTERISTICAS TECNICAS 4](#_Toc208142297)

[Arquitectura del Software 4](#_Toc208142298)

[**Componentes Principales** 4](#_Toc208142299)

[Interfaz de Usuario (simpleAngle.ui) 4](#_Toc208142300)

[Sistema de Calibración (Calibración guiada) 4](#_Toc208142301)

[Hilo de Comunicación (SimpleAngleThread) 4](#_Toc208142302)

[Funcionalidades Especificas 5](#_Toc208142303)

[Monitoreo en Tiempo Real 5](#_Toc208142304)

[Sistema de Calibración Avanzado 5](#_Toc208142305)

[Visualización Gráfica 5](#_Toc208142306)

[Exportación de Datos 5](#_Toc208142307)

[HARDWARE COMPATIBLE 6](#_Toc208142308)

[Microcontrolador Principal 6](#_Toc208142309)

[Sensor Requerido 6](#_Toc208142310)

[Diagrama de Conexiones 6](#_Toc208142311)

[REQUERIMIENTOS DE INSTALACIÓN Y CONFIGURACIÓN 7](#_Toc208142312)

[Requisitos del Sistema 7](#_Toc208142313)

[Dependencias de Software 7](#_Toc208142314)

[Proceso de Instalación 7](#_Toc208142315)

[INICIO RÁPIDO 8](#_Toc208142316)

[ALGORITMOS IMPLEMENTADOS 9](#_Toc208142317)

[Calibración por Regresión Lineal 9](#_Toc208142318)

[Conversión ADC a Ángulo 9](#_Toc208142319)

[APLICACIONES EDUCATIVAS 10](#_Toc208142320)

[Niveles Académicos 10](#_Toc208142321)

[Conceptos Didácticos Cubiertos 10](#_Toc208142322)

[APÉNDICES 11](#_Toc208142323)

# INTRODUCCIÓN

## Información General

**Nombre del Software:**SENSORA\_SIMPLE\_ANGLE  
**Fecha de Desarrollo:** 2025  
**Plataforma:** Python 3.x con PySide6  
**Categoría:** Software Educativo/Módulo Didáctico

## Descripción General

SENSORA\_SIMPLE\_ANGLE es un módulo software especializado diseñado para la monitorización en tiempo real de sensores de ángulo mediante potenciómetros conectados a microcontroladores ESP32. Este software forma parte del ecosistema SensoraCore, una plataforma integral de módulos didácticos para el aprendizaje de sistemas embebidos y adquisición de datos.

El software proporciona una interfaz gráfica intuitiva que permite a estudiantes y profesionales visualizar, calibrar y analizar lecturas angulares provenientes de sensores analógicos, facilitando el entendimiento práctico de conceptos como conversión analógico-digital, calibración por regresión lineal y comunicación serie con microcontroladores.

# CARACTERISTICAS TECNICAS

## Arquitectura del Software

Lenguaje de Programación: Python 3.8+.

Framework GUI: PySide6 (Qt6) con elementos personalizados.

Visualización: Matplotlib integrado con FigureCanvasQTAgg.

Comunicación: Sockets TCP (puerto 8080) hacia firmware MicroPython en ESP32.

Procesamiento de Datos: NumPy para cálculos básicos; regresión lineal propia (mínimos cuadrados).

Exportación: openpyxl y pandas para generación de archivos ExcelComponentes principales.

**Componentes Principales**

Interfaz de Usuario (simpleAngle.ui)

def anguloSimple\_UI(self):

"""

Interfaz principal del módulo con:

- Diagrama de conexiones ESP32

- Controles de monitoreo

- Visualización en tiempo real

- Panel de calibración

"""

Sistema de Calibración (Calibración guiada)

class CalibrationDialog(QDialog):

""" Diálogo avanzado para calibración por regresión lineal:

- Entrada manual de puntos de referencia

- Visualización gráfica de la regresión

- Cálculo automático de ecuaciones de calibración

- Estadísticas de calidad de ajuste (R²) """

Hilo de Comunicación (SimpleAngleThread)

class CalibrationDialog(QDialog):

""" Diálogo avanzado para calibración por regresión lineal:

- Entrada manual de puntos de referencia

- Visualización gráfica de la regresión

- Cálculo automático de ecuaciones de calibración

- Estadísticas de calidad de ajuste (R²) """

## Funcionalidades Especificas

Monitoreo en Tiempo Real

* **Frecuencia de Muestreo:** 2 Hz (cada 0.5 s) según firmware actual
* **Rango de Medición:** -135° a +135°
* **Resolución ADC:** 12 bits (0-4095) del ESP32

Sistema de Calibración Avanzado

* **Método:** Regresión lineal por mínimos cuadrados
* **Puntos Mínimos:** 2 puntos de referencia
* **Puntos Recomendados:** 5-10 para mejor precisión
* **Estadísticas:** Coeficiente de determinación R²
* **Persistencia:** Guardado/carga de calibraciones en JSON

Visualización Gráfica

* **Tipo de Gráfico:** Línea continua con marcadores
* **Actualización:** Tiempo real con buffer circular
* **Personalización:** Colores y estilos Bootstrap

Exportación de Datos

* **Formato:** Excel (.xlsx) con múltiples hojas; CSV como alternativa
* **Contenido:** Timestamps, lecturas ADC, ángulos calibrados
* **Gráficos:** Gráfico de líneas integrado en Excel
* **Metadatos:** Información de calibración y sesión

# HARDWARE COMPATIBLE

## Microcontrolador Principal

* **Modelo:** ESP32 DevKit V1 o compatible
* **GPIO Utilizado:** Pin 32 (ADC1\_CH4)
* **Alimentación:** 3.3V y GND
* **Comunicación:** WiFi 802.11b/g/n (sockets TCP)

## Sensor Requerido

* **Tipo:** Potenciómetro rotativo
* **Resistencia:** 10kΩ (recomendado)
* **Rango Angular:** -135° 135°
* **Linealidad:** ±1% o mejor
* **Conexión:** 3 pines (VCC, GND, Señal)

## Diagrama de Conexiones

┌─────────────────────────────────┐

│ ESP32 DevKit V1 │

│ 3V3 ○ ←── Potenciómetro (+) │

│ D32 ○ ←── Potenciómetro (S) │

│ GND ○ ←── Potenciómetro (-) │

│ LED integrado: GPIO 2 │

└─────────────────────────────────┘

# 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. Alimentar al ESP32
2. Verificar conexiones del potenciómetro
3. Ejecutar aplicación SensoraCore
4. Seleccionar módulo "Sensor de Ángulo Simple"

Calibración del Sensor

1. Hacer clic en "Calibrar Sensor"
2. Agregar mínimo 2 puntos de referencia:
   * Posicionar potenciómetro en ángulo conocido
   * Anotar lectura ADC mostrada
   * Ingresar ángulo real de referencia
3. Repetir para diferentes ángulos
4. Hacer clic en " Realizar Calibración"
5. Verificar calidad de ajuste (R² > 0.95 recomendado)

Monitoreo en Tiempo Real

1. Hacer clic en " Iniciar Monitoreo"
2. Observar lecturas en tiempo real
3. Analizar gráfica de tendencias
4. Usar "Limpiar Gráfica" para reiniciar

Exportación de Datos

1. Acumular datos durante sesión de monitoreo
2. Hacer clic en " Exportar Excel"
3. Seleccionar ubicación de guardado
4. Abrir archivo generado para análisis

# ALGORITMOS IMPLEMENTADOS

## Calibración por Regresión Lineal

def perform\_calibration(self):

"""

Implementa regresión lineal: y = mx + b

donde:

- x: valores ADC del sensor

- y: ángulos de referencia conocidos

- m: pendiente (sensibilidad)

- b: intersección (offset)

"""

*# xs: lista de valores ADC, ys: lista de ángulos de referencia, n:número de puntos*

mean\_x = sum(xs) / n

mean\_y = sum(ys) / n

numerador = sum((xs[i] - mean\_x) \* (ys[i] - mean\_y) for i in range(n))

denominador = sum((xs[i] - mean\_x) \*\* 2 for i in range(n))

slope = numerador / denominador

intercept = mean\_y - slope \* mean\_x

ss\_res = sum((ys[i] - (slope \* xs[i] + intercept)) \*\* 2 for i in range(n))

ss\_tot = sum((ys[i] - mean\_y) \*\* 2 for i in range(n))

r\_squared = 1 - ss\_res / ss\_tot

return slope, intercept, r\_squared

## Conversión ADC a Ángulo

def apply\_calibration(self, raw\_value):

"""

Aplica calibración lineal a lectura ADC:

angle = slope \* adc\_value + intercept

"""

if self.is\_calibrated:

return self.slope \* raw\_value + self.intercept

*# Conversión por defecto (sin calibrar)*

return (raw\_value / 4095.0) \* 270.0 - 135.0

# APLICACIONES EDUCATIVAS

## Niveles Académicos

* **Educación Media:** Conceptos básicos de sensores
* **Técnico Superior:** Sistemas de adquisición de datos
* **Universitario:** Instrumentación y control
* **Posgrado:** Procesamiento avanzado de señales

## Conceptos Didácticos Cubiertos

* **Electrónica Analógica:** Potenciómetros y divisores de voltaje
* **Conversión A/D:** Cuantización y resolución
* **Calibración:** Regresión lineal y estadística
* **Programación:** Interfaces gráficas y threads
* **Comunicaciones:** Protocolos serie y WiFi

# 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>