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](#_Toc208224725)

[Información General 3](#_Toc208224726)

[Descripción General 3](#_Toc208224727)

[CARACTERISTICAS TECNICAS 4](#_Toc208224728)

[Especificaciones 4](#_Toc208224729)

[Sensor TCS3200 4](#_Toc208224730)

[Sistema de Comunicación 4](#_Toc208224731)

[Algoritmo de Calibración 4](#_Toc208224732)

[HARDWARE COMPATIBLE 5](#_Toc208224733)

[Microcontrolador Principal 5](#_Toc208224734)

[REQUERIMIENTOS DE INSTALACIÓN Y CONFIGURACIÓN 6](#_Toc208224735)

[Requisitos del Sistema 6](#_Toc208224736)

[Dependencias de Software 6](#_Toc208224737)

[Proceso de Instalación 6](#_Toc208224738)

[INICIO RÁPIDO 7](#_Toc208224739)

[ALGORITMOS IMPLEMENTADOS 8](#_Toc208224740)

[APLICACIONES EDUCATIVAS 9](#_Toc208224741)

[APÉNDICES 10](#_Toc208224742)

# INTRODUCCIÓN

## Información General

**Nombre del Software:**SENSORA\_COLOR\_TCS

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

## Descripción General

El módulo **SENSORA\_COLOR\_TCS** representa una solución avanzada para la detección y análisis de color basada en el sensor **TCS3200**. Aprovecha la conversión luz frecuencia del TCS3200 para obtener señales digitales proporcionales a la intensidad en cada canal (R, G, B), facilitando mediciones reproducibles en entornos controlados e industriales. El diseño del módulo prioriza la estabilidad y la integración: incluye rutinas de calibración multipunto (negro/blanco y colores puros), compensación de ganancia por canal y exportación de datos con metadatos para trazabilidad.

# CARACTERISTICAS TECNICAS

* **Sensor**: TCS3200 con fotodiodos y filtros RGB integrados
* **Medición**: Conversión luz-frecuencia con salida digital
* **Calibración**: Sistema de 5 puntos (blanco, negro, rojo, verde, azul)
* **Visualización**: Gráficas RGB en tiempo real (0-255)
* **Exportación**: Datos completos en formato Excel con metadatos
* **Comunicación**: TCP/IP con ESP32 para streaming continuo

## Especificaciones

### Sensor TCS3200

* **Rango espectral**: 700-1000 nm
* **Resolución**: 16 fotodiodos con filtros RGB y clear
* **Salida**: Frecuencia proporcional a intensidad lumínica
* **Alimentación**: 3.3V DC
* **Consumo**: < 10 mA
* **Temperatura operativa**: -25°C a +70°C

### Sistema de Comunicación

* **Protocolo**: TCP/IP over WiFi
* **Comando inicial**: "MODO:COLOR\_TCS"
* **Streaming**: Configurable desde 150ms
* **Formato datos**: "SENSOR:TCS3200,R:,G:,B:"
* **Puerto**: 8080 (configurable)

### Algoritmo de Calibración

* **Mapeo dinámico**: Min/max por sesión (modo no calibrado)
* **Calibración completa**: Escalado 0-255 con referencias de color
* **Compensación**: Ganancia por canal basada en colores puros
* **Persistencia**: Calibración guardada en JSON local

# HARDWARE COMPATIBLE

## Microcontrolador Principal

**Modelo**: ESP32 DevKit V1

**Conectividad**: WiFi integrado para comunicación TCP

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

**Pines GPIO:** 6 pines digitales

Conexiones de TCS3200

ESP32 Pin TCS3200 Pin Función

3V3 → VCC Alimentación

GND → GND Tierra

D36 ↔ S0 Escala frecuencia

D39 ↔ S1 Escala frecuencia

D34 ↔ S2 Selección filtro

D35 ↔ S3 Selección filtro

D32 ← OUT Salida frecuencia

# 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** "COLOR TCS"
3. **Verificar conexión** del sensor TCS3200
4. **Introducir IP** del ESP32 en campo correspondiente

**Interfaz de Usuario**

* **Panel Superior**: Título y descripción del módulo
* **Panel Controles**: Botones de operación y datos RGB
* **Panel Diagrama**: Esquema de conexiones
* **Panel Gráfica**: Visualización temporal RGB

**Proceso de Calibración**

1. **Iniciar calibración**: Presionar "No Calibrado"
2. **Blanco de referencia**: Colocar objeto blanco y capturar
3. **Negro de referencia**: Colocar objeto negro y capturar
4. **Rojo puro**: Colocar objeto rojo y capturar
5. **Verde puro**: Colocar objeto verde y capturar
6. **Azul puro**: Colocar objeto azul y capturar
7. **Completar**: Calibración guardada automáticamente

**Monitoreo en Tiempo Real**

1. **Iniciar streaming**: Presionar "Iniciar Monitoreo"
2. **Observar datos**: Valores Hz (crudo) y 0-255 (calibrado)
3. **Analizar gráfica**: Líneas RGB con ventana deslizante 25s
4. **Pausar**: Detener streaming manteniendo datos

**Exportación de Datos**

1. **Recopilar datos**: Ejecutar sesión de monitoreo
2. **Exportar Excel**: Presionar "Exportar a Excel"
3. **Seleccionar ubicación**: Elegir carpeta de destino
4. **Archivo generado**: ColorTCS\_YYYYMMDD\_HHMMSS.xlsx

# ALGORITMOS IMPLEMENTADOS

Conversión a Escala 0-255

Modo No Calibrado:

def dyn\_map(v, ch):

lo = min\_sesion[ch]

hi = max\_sesion[ch]

x = (v - lo) / (hi - lo)

return x \* 255.0

Modo Calibrado:

def to\_255\_calibrated(r\_hz, g\_hz, b\_hz):

*# 1. Escalado base [negro, blanco]*

r\_base = (r\_hz - black\_r) / (white\_r - black\_r)

g\_base = (g\_hz - black\_g) / (white\_g - black\_g)

b\_base = (b\_hz - black\_b) / (white\_b - black\_b)

*# 2. Ganancia por color puro*

gain\_r = 1.0 / red\_reference\_intensity

gain\_g = 1.0 / green\_reference\_intensity

gain\_b = 1.0 / blue\_reference\_intensity

*# 3. Aplicar compensación*

return 255 \* r\_base \* gain\_r, 255 \* g\_base \* gain\_g, 255 \* b\_base \* gain\_b

**Visualización Matplotlib**

* **Tres líneas**: Rojo (#ff0000), Verde (#00aa00), Azul (#0000ff)
* **Eje X**: Tiempo relativo en segundos
* **Eje Y**: Valores 0-255 (fijo)
* **Ventana**: 25 segundos deslizante
* **Actualización**: Tiempo real con draw\_idle()

# APLICACIONES EDUCATIVAS

**Laboratorio de Óptica**

* **Espectro visible**: Análisis de componentes RGB
* **Ley Beer-Lambert**: Absorción vs concentración
* **Temperatura color**: Caracterización de fuentes lumínicas
* **Metamerismo**: Colores perceptualmente iguales

**Instrumentación Científica**

* **Colorimetría**: Medición objetiva de color
* **Fotometría**: Intensidad lumínica relativa
* **Espectrofotometría**: Análisis espectral básico
* **Sensores**: Principios de transducción

**Proyectos Aplicados**

* **Control calidad**: Clasificación automática
* **Robótica**: Seguimiento por color
* **Proceso industrial**: Monitoreo continuo
* **Arte digital**: Análisis de pigmentos

# 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>