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](#_Toc208227385)

[Información General 3](#_Toc208227386)

[Descripción General 3](#_Toc208227387)

[CARACTERISTICAS TECNICAS 4](#_Toc208227388)

[HARDWARE COMPATIBLE 5](#_Toc208227389)

[REQUERIMIENTOS DE INSTALACIÓN Y CONFIGURACIÓN 6](#_Toc208227390)

[Requisitos del Sistema 6](#_Toc208227391)

[Dependencias de Software 6](#_Toc208227392)

[Proceso de Instalación 6](#_Toc208227393)

[INICIO RÁPIDO 7](#_Toc208227394)

[Inicio del Sistema 7](#_Toc208227395)

[Operación Paso a Paso 7](#_Toc208227396)

[ALGORITMOS IMPLEMENTADOS 8](#_Toc208227397)

[APLICACIONES EDUCATIVAS 10](#_Toc208227398)

[APÉNDICES 11](#_Toc208227399)

# INTRODUCCIÓN

## Información General

**Nombre del Software:**SENSORA\_CAPACITIVE

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

## Descripción General

El módulo SENSORA\_CAPACITIVE es un sistema especializado de detección de proximidad capacitiva que utiliza un sensor capacitivo tubular industrial para monitoreo sin contacto en tiempo real. El sistema implementa una arquitectura cliente-servidor basada en TCP que permite la comunicación entre un microcontrolador ESP32 y una aplicación de escritorio desarrollada en PySide6, proporcionando detección digital ON/OFF con retroalimentación visual inmediata y características educativas avanzadas.

# CARACTERISTICAS TECNICAS

**Arquitectura del Software**

* **Patrón Cliente-Servidor TCP**: Comunicación especializada para sensores digitales
* **Threading Asíncrono**: QThread para operaciones de red sin bloqueo de interfaz
* **Retroalimentación Visual Inmediata**: Cambios de estado en tiempo real
* **Sistema de Tolerancia a Fallos**: Recuperación automática ante pérdidas de conexión
* **Interfaz Educativa**: Información técnica integrada del sensor industrial

**Componentes Principales**

* **CapasitiveThread**: Hilo de comunicación TCP especializado para señales digitales
* **CapasitiveLogic**: Controlador principal con gestión de estados visuales
* **Sistema de Visualización**: Indicadores LED virtuales con códigos de color
* **Monitor de Estado**: Seguimiento continuo de conectividad y funcionamiento
* **Información Técnica**: Base de datos integrada de especificaciones del sensor

**Funcionalidades Específicas**

* Detección digital binaria (ON/OFF) de proximidad capacitiva
* Visualización en tiempo real con códigos de color intuitivos
* Información técnica completa del sensor Tecnotron capacitivo tubular
* Diagrama de conexiones con códigos de color de cables
* Sistema de tolerancia a errores con reconexión automática

# HARDWARE COMPATIBLE

**Microcontrolador Principal**

**Modelo**: ESP32 DevKit V1

**Conectividad**: WiFi integrado para comunicación TCP

* **GPIO Digital**: Pin D25 para entrada digital del sensor

**Sensor Capacitivo Requerido**

**Sensor Capacitivo Tubular Tecnotron:**

* Tipo: Sensor capacitivo tubular industrial
* Distancia Sensora: 15 mm de rango de detección
* Capacidad de Carga: 10 a 500 mA de corriente de salida
* Frecuencia de Conmutación: 5 Hz máxima
* Salida: VCA, lógica normalmente cerrada (NF)
* Protección: IP65 (resistente al polvo y chorros de agua)
* Voltaje de Operación: 5V a 12V DC (requiere fuente externa)

**Especificaciones de Cables del Sensor**

* Cable Marrón: Alimentación positiva (+) 5V-12V DC
* Cable Azul: Tierra/GND (-)
* Cable Negro: Señal de salida digital (D)

Diagrama de Conexiones

┌─────────────────────────────────┐

│ ESP32 DevKit V1 │

│ 3V3 ○ ← N/A (no usado) │

│ GND ○ ← Cable Azul (GND) │

│ D25 ○ ← Cable Negro (OUT) │

│ LED integrado: GPIO 2 │

└─────────────────────────────────┘

┌─────────────────────────────────┐

│ Fuente de Voltaje Externa │

│ 5V ○ ← Cable Marrón (+) │

│ GND ○ ← Cable Azul (-) │

└─────────────────────────────────┘

Sensor Capacitivo Tubular Tecnotron:

• Cable Marrón: +5V-12V (fuente externa)

• Cable Azul: GND común (ESP32 + fuente)

• Cable Negro: Señal digital (Pin D25)

# 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. **Verificación de Hardware**:
   * Confirmar todas las conexiones según diagrama
   * Verificar alimentación 5V-12V al sensor capacitivo
   * Asegurar que ESP32 esté energizado y conectado a WiFi
   * Probar respuesta del sensor manualmente
2. **Configuración de Comunicación**:
   * Introducir IP del ESP32 en campo correspondiente
   * Verificar conectividad de red antes de iniciar
   * Confirmar que puerto 8080 esté disponible
3. **Inicio de Monitoreo**:
   * Hacer clic en "Iniciar Monitoreo"
   * Observar cambio de estado del botón a "Pausar"
   * Confirmar llegada de datos del sensor capacitivo

**Operación Paso a Paso**

1. **Interpretación de Estados**:
   * **ON (Verde)**: Objeto detectado dentro del rango de 15mm
   * **OFF (Rojo)**: No hay objetos en el rango de detección
   * **Estado Visual**: Cambios inmediatos de color y texto
2. **Funcionamiento del Sensor**:
   * **Lógica Normalmente Cerrada (NF)**: Sensor activo cuando no detecta
   * **Rango de Detección**: 15mm de distancia efectiva
   * **Materiales Detectables**: Papel, plástico, vidrio, líquidos, polvo
   * **Frecuencia**: Hasta 5 Hz de conmutación
3. **Información Técnica Integrada**:
   * **Especificaciones Completas**: Datos técnicos del sensor Tecnotron
   * **Diagramas de Conexión**: Códigos de color de cables
   * **Notas de Seguridad**: Voltajes y precauciones de instalación

**Funcionalidades Avanzadas**

* **Monitoreo Continuo**:
  + Seguimiento constante del estado digital
  + Recuperación automática ante pérdidas de comunicación
  + Indicadores visuales de conectividad
* **Información Educativa**:
  + Especificaciones técnicas detalladas
  + Principios de funcionamiento capacitivo
  + Aplicaciones industriales típicas

# ALGORITMOS IMPLEMENTADOS

Comunicación TCP para Sensor Digital

*# Protocolo de comunicación especializado para sensor capacitivo*

def run(self):

self.sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.sock.connect((self.esp32\_ip, self.port))

self.sock.sendall(b'MODO:DISTANCIA\_CAP')

*# Verificación de handshake específico*

resp = self.sock.recv(64).decode('utf-8', errors='ignore').strip()

if 'DISTANCIA\_CAP\_OK' not in resp:

self.status.emit("ESP32 no aceptó modo DISTANCIA\_CAP")

return

*# Loop de recepción de datos digitales*

buffer = ""

while self.\_running:

chunk = self.sock.recv(128)

buffer += chunk.decode('utf-8', errors='ignore')

while '\n' in buffer:

line, buffer = buffer.split('\n', 1)

*# Parseo: CAP\_DIGITAL:True/False*

try:

if ':' in line:

key, val = line.split(':', 1)

if key.strip().upper() == 'CAP\_DIGITAL':

sensor\_on = val.strip().lower() in ('true', '1', 'on')

self.state.emit(sensor\_on)

except Exception:

continue *# Ignorar líneas malformadas*

Gestión de Estados Visuales

*# Sistema de retroalimentación visual inmediata*

def \_set\_state(self, on: bool):

label\_name = "EstadoDeSensorCapatitivo\_ON\_OFF"

if hasattr(self.ui, label\_name) and self.monitoring:

lbl = getattr(self.ui, label\_name)

lbl.setText("ON" if on else "OFF")

*# Estados visuales con códigos de color*

if on:

*# Estado ACTIVO: Verde brillante*

lbl.setStyleSheet("""

background-color: #d4edda;

color: #155724;

font-weight: bold;

padding: 8px;

border-radius: 4px;

border: 2px solid #c3e6cb;

""")

else:

*# Estado INACTIVO: Rojo suave*

lbl.setStyleSheet("""

background-color: #f8d7da;

color: #721c24;

font-weight: bold;

padding: 8px;

border-radius: 4px;

border: 2px solid #f5c6cb;

""")

# APLICACIONES EDUCATIVAS

**Conceptos Físicos**:

* Principios de capacitancia y dieléctricos
* Detección sin contacto físico
* Sensores digitales vs analógicos

**Experimentos Sugeridos**:

* Detección de diferentes materiales
* Medición del rango de detección
* Comparación con otros tipos de sensores

**Conceptos Tecnológicos**:

* Automatización industrial y sensores
* Sistemas de control sin contacto
* Especificaciones técnicas de sensores industriales

**Proyectos Propuestos**:

* Sistema de conteo de objetos automático
* Detector de nivel en contenedores
* Sistema de seguridad por proximidad

**Conceptos Especializados**:

* Tecnología capacitiva en automatización
* Sistemas de detección industrial
* Integración de sensores en IoT

# 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>