ANEMÓMETRO TÉRMICO

WINDQX - SA.01

SIN PARTES MÓVILES

PATENTE U202332221

Modelo SA.01

El SA01 es un anemómetro de estado sólido diseñado para la medición digital de la velocidad del aire. No compromete la precisión ni la exactitud, lo que lo convierte en una excelente opción para la monitorización, resolución de proble en sistemas HVAC y protección ante catástrofes medioambientales. Además, su diseño robusto, sin partes con desgastes, permite la instalación permanente, lo que habilita la monitorización continua de la velocidad del aire a lo largo del ti

Características y Beneficios

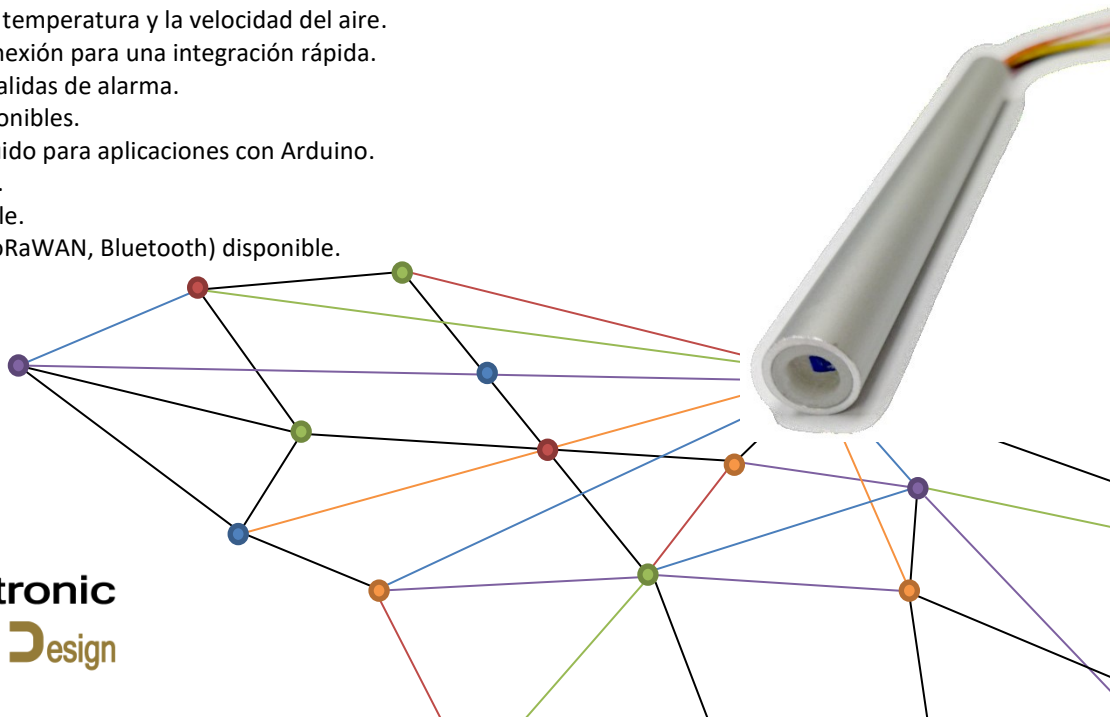
+ Medición precisa de la velocidad del aire y la temperatura.
+ Fácil integración.
+ Sistema de medición sin orificios de entrada y salida de aire.
+ Lectura omnidireccional.
+ Certificado de calibración de nuestro laboratorio incluido.

Aplicaciones

+ Rendimiento de sistemas HVAC.
+ Puesta en marcha y monitorización.
+ Mantenimiento de plantas.
+ Verificación de ambientes críticos.
+ Medición transversal en conductos.

Características Adicionales y Beneficios

+ Medición simultánea de la temperatura y la velocidad del aire.
+ Múltiples interfaces de conexión para una integración rápida.
+ Posibilidad de configurar salidas de alarma.
+ Accesorios de fijación disponibles.
+ Software de descarga incluido para aplicaciones con Arduino.
+ Salida de datos SERIAL TTL.
+ Adaptador RS485 disponible.
+ Adaptador radios (LoRa, LoRaWAN, Bluetooth) disponible.



Electronic Circuit Design

Especificaciones Técnicas

	Unidades de Medición	Precisión*	Rango de medición	Resolución
Velocidad del aire	km/h	±(3% del valor + 0.2 km/h) de 0 a 20km/h ±(3% del valor + 0.5 km/h) de 20 a 60km/h ±(3% del valor + 1 km/h) de 60 a 130km/h	De 0 a 130km De 130 a 200km**	0.1Km/h
Temperatura	Celsius	±0.5 °C	De -10 a +60 °C	0.01 °C

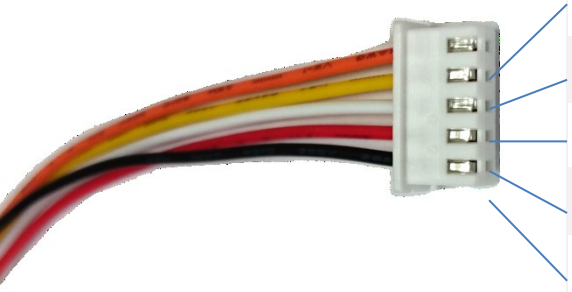
*Todas las precisiones indicadas en esta ficha técnica son en condiciones de laboratorio y pueden garantizarse para mediciones realizadas en las mismas características.
** Interpolado mediante el uso de una curva de calibración obtenida de 0 a 130 km/h.

Características Generales

Elemento de medición	Velocidad del aire: sensor PT100 sobrecalentado. Temperatura ambiente: sensor PT100 combinado con sensor digital.*
Carcasa	ABS-PC y Aluminio, IP67**
Alimentación	De 3.5 a 5.5V (ideal para baterías LiPo de 1 celda o similar).
Consumo	Modo de trabajo: de 1 a 25mA – Promedio de 18mA. Modo de espera: menos de 1mA.
Comunicación	SERIAL TTL @ 3v3 – 115200 baudios – cada 1 segundo.
Dimensiones	Cuerpo: largo 105mm, diámetro 10mm / Cableado: largo 165mm.
Condiciones de operación	De -25 a 50 °C. En condiciones no condensadas.
Peso	12 g.
Tiempo de adquisición	Desde el modo de espera hasta el modo de trabajo: 15 segundos.

*Dependiendo de la versión.
**Solo para el cuerpo del anemómetro.

Conexionado



Sin uso, solo para fines técnicos.
Datos de transmisión: SERIAL-TX / Reloj I2C / Salida de alarma.
Datos de recepción: SERIAL-RX / Datos I2C.
Alimentación: Positivo (3.5 a 5.5V).
Alimentación: Negativo.

Documento técnico — Anemómetro SA01

Operación, configuración, alarmas, verificación de integridad y acceso a certificado (Web)
(Versión web — redactado para usuarios técnicos y soporte)

---

1) Qué salida de datos tiene el SA01 y cómo se usa

El **SA01** es un anemómetro que puede trabajar en distintos modos según la instalación:

- **UART (por defecto, 115200)**: modo recomendado para **configurar** y para diagnóstico.
- **I2C**: modo recomendado para integración en sistemas con microcontrolador.
- **Salida por alarma (paralela)**: el equipo conmuta el nivel de una patilla cuando el viento supera un umbral (Alarma 1).

Además, incluye comandos de diagnóstico para verificar:

- el **estado de integridad** de la configuración (CRC),
- y la **información necesaria para consultar el certificado de calibración** en web.

---

2) Modos de arranque / interfaz (variable USER I00)

La variable **I00** define el modo de funcionamiento al arrancar:

I00	Modo	UART RX	UART TX	I2C
0	**UART full** (default)	Sí	Sí	No
1	**UART solo RX** + TX = Alarma 1	Sí	**Alarma 1**	No
2	**Sin UART RX** + TX = Alarma 1	**No**	**Alarma 1**	No
3	**I2C** (sin UART)	No	No	Sí

Recomendación práctica

- Para instalación y puesta en marcha: **I00=0** (UART full).
- Para salida de alarma y mantener posibilidad de configuración por UART: **I00=1**.
- **I00=2** y **I00=3** restringen la recuperación de comunicación y suelen requerir el **puente 1–5** (ver sección 6).

---

3) Lectura por I2C: trama de 4 bytes (medida “en vivo”)

El SA01 entrega **4 bytes**:

- **Bytes 0–1**: Viento → rawWind = (b0<<8) | b1
- **Bytes 2–3**: Temperatura → rawTemp = (b2<<8) | b3

Conversión

Viento (km/h)
El viento ya viene en km/h y está escalado ×10 para conservar un decimal sin usar flotantes internamente:

W_km/h=rawWind10.0W_{km/h} = \frac{rawWind}{10.0}W_km/h=10.0rawWind

Temperatura (°C)

T_°C=rawTemp100.0−40.0T_{°C} = \frac{rawTemp}{100.0} - 40.0T_°C=100.0rawTemp−40.0

Ejemplo

- Si rawWind = 253 → W = 25.3 km/h
- Si rawTemp = 6086 → T = 20.86 °C

---

4) Dirección I2C (variable USER I11)

- Dirección por defecto habitual: **0x36** (en decimal: **54**).
- La dirección puede ser modificada por configuración (USER I11).

Recomendación para herramientas de diagnóstico (cuando no se conoce la dirección):

1. Probar primero la dirección “esperada” (por ejemplo 0x36 / 54).
2. Si no responde, hacer **scan** del bus I2C (0x08..0x77).
3. Para cada dirección detectada, **validar** que realmente es el SA01:
    - lectura correcta de **4 bytes**,
    - valores plausibles (viento/temperatura coherentes),
    - para evitar confundirlo con “otro dispositivo I2C” conectado al bus.

---

5) Configuración USER: lectura, escritura y guardado

La configuración del usuario se gestiona por comandos ASCII.
Para poder leer/escribir variables USER, hay un “desbloqueo” previo.

5.1 Desbloqueo (obligatorio)

1. Enviar: SA01 + **CRLF** (\r\n)
2. Verificar eco: SA01

El eco confirma que el SA01 ha habilitado el acceso a lectura/escritura de variables USER.

5.2 Lectura del estado actual

Enviar: a0 + \r\n
El SA01 devuelve el listado:

```
INT User Vars
I00+000000000
I01+000000020
...
I11+000000054
...
---------------------
```

5.3 Escritura (formato estricto y CRLF)

Para que un cambio sea interpretado correctamente (y como medida de seguridad), **la escritura debe respetar el formato exacto**:

- **Formato:** INN+000000XYZ
- **Final obligatorio:** \r\n

Ejemplos correctos:

- I00+000000000\r\n → pone modo de arranque UART full
- I01+000000035\r\n → umbral 35 km/h (alarma 1)

Importante: el "padding" (ceros) y el \r\n forman parte del formato esperado.
Si no se respeta, el dispositivo puede ignorar el comando o no aplicarlo.

5.4 Guardar cambios en EEPROM

Después de ajustar los valores deseados, para que queden persistentes:

- Enviar: COMMIT\r\n
- Respuesta esperada: EEPROM USER SAVE DONE!

---

6) Recuperación de "full UART" (cuando no hay comunicación)

Según el modo en que esté el SA01, existen dos rutas:

6.1 Recuperación por software (si el equipo escucha UART)

Si el SA01 está en un modo donde UART RX está activo (por ejemplo I00=0 o I00=1):

- enviar SA01 repetidamente hasta recibir eco SA01.

6.2 Recuperación por hardware (si NO escucha UART o está en I2C)

Si el SA01 está en:

- I00=2 (sin UART RX) o
- I00=3 (I2C sin UART),

no se podrá recuperar por comandos.

Procedimiento:

1. Desconectar alimentación del SA01
2. Colocar **puente entre pines 1 y 5 antes de alimentar**
3. Alimentar/conectar el SA01 con el puente colocado
4. Recuperada la UART, **retirar el puente** (no es necesario mantenerlo)

---

7) Variables USER utilizadas en esta versión (v1)

En esta versión del SA01 se usan:

7.1 Modo / interfaz

- **I00**: modo de arranque (0..3, ver sección 2)

7.2 Alarma 1 (salida por patilla)

- **I01**: **umbral de viento** en **km/h sin decimales** (entero)
  Ej: I01=35 → 35 km/h
- **I02**: **tiempo requerido por encima del umbral** para disparar (segundos)
  Ej: I02=2 → debe mantenerse la condición durante 2 s para disparar.
- **I03**: **tiempo de espera (anti-reentrada)** tras un disparo (segundos)
  No es "bajo umbral": el firmware **no reevalúa el umbral durante este tiempo**. Su función es evitar disparos sucesivos no deseados: tras disparar, el sistema espera I03 segundos antes de volver a monitorizar para un nuevo disparo.
- **I04**: **ancho de pulso** (segundos)
  - mínimo: **1 s**
  - máximo: **32768 s**
- **I05**: configuración del **nivel de reposo** de la patilla de alarma (sin alarma)
  Define si el estado normal (sin alarma) es:
  - **0 V** o
  - **3.3 V**
    (Equivalente práctico a elegir la polaridad "reposo" de la salida.)

7.3 Dirección I2C

- **I11**: dirección I2C efectiva 7-bit, en **decimal**
  Ej: I11=54 → 0x36

7.4 Variables no utilizadas (reservadas)

- I06..I10 (Alarma 2): **no se usa** en esta versión.
- I12 y I13: **no se usan** (reservadas para futuro).

---

8) Ejemplo de configuración típica de Alarma 1

Objetivo: activar una salida de alarma cuando el viento supere 40 km/h, evitando falsos disparos y evitando reentradas inmediatas.

- I01 = 40 (umbral 40 km/h)
- I02 = 2 (la condición debe mantenerse 2 s)
- I03 = 10 (tras disparo, espera 10 s antes de volver a monitorizar)
- I04 = 5 (pulso de 5 s; si la salida se usa como pulso)
- I05 según polaridad de reposo deseada (0V o 3.3V)

Luego:

- COMMIT para guardar.

---

9) INFO: certificado, integridad y estado (DATA_STAT)

El comando INFO devuelve información del dispositivo para:

- construir el enlace al certificado web (precargando datos),
- y evaluar si la configuración actual coincide con la grabada originalmente (CRC).

Salida típica:

```
----------------------
Device INFO
----------------------
Serial:    001310723000655362
WEBPass:   00010005
CRC_STORED: 0xC303
CRC_CALC:  0x8882
DATA_STAT:  CORRUPT
----------------------
```

Interpretación

- **DATA_STAT: ORIGINAL**
  CRC_STORED == CRC_CALC → la configuración actual coincide con la almacenada originalmente.
- **DATA_STAT: CORRUPT**
  CRC_STORED != CRC_CALC → hay discrepancia respecto al original.
  Puede ocurrir por alteración/corrupción de EEPROM o cambios no previstos.

Nota: “CORRUPT” indica mismatch de datos, no afirma automáticamente que el sensor mida mal; indica que lo actual no coincide con lo registrado originalmente.

10) URL del certificado (precarga para web)

Para facilitar el acceso a la calibración y permitir validación, la herramienta puede generar un enlace que **precarga** el formulario web (sin acceso directo automático).

Formato oficial propuesto:
https://ecdsl.com/certificates/sa01?sn=<Serial>&wp=<WEBPass>&crc=<CRC_CALC_sin_0x>

- sn: Serial (18 dígitos)
- wp: WEBPass (8 hex)
- crc: CRC_CALC (4 hex, sin 0x)

Ejemplo:
https://ecdsl.com/certificates/sa01?sn=001310723000655362&wp=00010005&crc=8882

En la web se completarán manualmente datos de usuario (nombre, email, registro/login), para evitar automatización por robots y mantener control de acceso.

---

11) Recomendaciones I2C (frecuencia realista)

- En pruebas con **pull-up de 10 kΩ**, el máximo estable observado es aproximadamente **650 kHz**.
- Recomendación general:
  - **100 kHz**: máxima compatibilidad
  - **400 kHz**: buen equilibrio
  - por encima: depende de bus corto, capacitancia y calidad de pull-ups

---

12) Guía rápida (checklist) para configurar sin sorpresas

1. Mantén el SA01 en **I00=0** mientras configuras y verificas.
2. Envía SA01\r\n y confirma eco SA01.
3. Envía a0\r\n y guarda el listado actual.
4. Ajusta I01..I05 (alarma 1) y/o I11 (I2C addr) usando formato **INN+000000XYZ\r\n**.
5. Envía COMMIT\r\n y espera EEPROM USER SAVE DONE!.
6. Envía INFO\r\n y revisa DATA_STAT:
   - ORIGINAL → OK
   - CORRUPT → investigar y, si procede, reconfigurar/guardar de nuevo
7. Si terminas en I00=2 o I00=3 y pierdes comunicación, usa recuperación con **puente 1–5** antes de alimentar.

Anexo — Protocolo de salida por UART (trama de datos) + verificación CRC

Cuando el SA01 está configurado para emitir datos por **UART**, genera líneas ASCII con valores de medida y un campo de verificación para validar la integridad de la trama.

---

1) Formato de la trama (ASCII)

Cada trama es una línea terminada en **CRLF** (\r\n) con **4 campos** separados por **un espacio**:

AAAA WWWW TTTT CCC\r\n

Ejemplos:

0400 0000 6800 019
0400 0001 6805 210
0401 0002 6799 224

Campos:

- AAAA (4 dígitos): valor analógico "crudo" del sensor (**diagnóstico**, no correlacionado directamente con viento).
- WWWW (4 dígitos): velocidad de viento en **km/h × 10** (1 decimal implícito).
- TTTT (4 dígitos): temperatura escalada, para obtener °C con:
  - T(°C) = (TTTT / 100.0) - 40.0
- CCC (3 dígitos): **LSB (byte menos significativo)** del **CRC-16 Modbus** del payload.

Detalles importantes:

- Los campos van con **ceros a la izquierda** (%04d o %04ld), y el CRC con **3 dígitos** (%03u).
- Separador: **espacio** (0x20).
- Final: \r\n.

---

2) Conversión a unidades físicas

Dada una línea AAAA WWWW TTTT CCC:

- Analógico crudo:
  analog = AAAA
- Viento (km/h):
  W_km/h = WWWW / 10.0
- Temperatura (°C):
  T_°C = (TTTT / 100.0) - 40.0

Ejemplo
Trama:

0401 0253 6086 123

- analog = 401
- W = 253 / 10 = 25.3 km/h
- T = 6086 / 100 - 40 = 20.86 °C

---

3) CRC: qué se calcula exactamente

El SA01 usa **CRC-16 Modbus** con:

- Polinomio reflejado: 0xA001 (equivalente a 0x8005)
- Valor inicial: 0xFFFF
- Reflejo entrada/salida: sí
- XOR final: 0x0000

3.1 Qué bytes entran al CRC (payload)

El CRC se calcula sobre el string ASCII **solo del payload**:

"AAAA WWWW TTTT"

- **Incluye** los espacios entre los campos.
- **No incluye**: el espacio antes de CCC, el CCC, ni \r\n.

Longitud exacta del payload: 14 bytes
(4 dígitos + 1 espacio + 4 dígitos + 1 espacio + 4 dígitos)

3.2 Qué representa CCC

CCC es el **byte menos significativo (LSB)** del CRC-16 calculado sobre el payload:

- crc16 = CRC16_Modbus(payload, init=0xFFFF)
- CCC = crc16 & 0xFF
- Se imprime como decimal con 3 dígitos: 000..255

---

4) Implementación de referencia (CRC-16 Modbus)

Código C de referencia:

```
// CRC-16 Modbus (poly 0xA001, init 0xFFFF)
unsigned int crc16_modbus(char *data, unsigned char inicio, unsigned char length, unsigned int crc) {
   unsigned char pos, i;

   data += inicio;

   for (pos = 0; pos < length; pos++) {
      crc ^= (unsigned int)(unsigned char)data[pos];

      for (i = 8; i != 0; i--) {
         if ((crc & 0x0001) != 0) {
            crc >>= 1;
            crc ^= 0xA001;
         } else {
            crc >>= 1;
         }
      }
   }
   return crc;
}
```

Uso típico en el receptor:

- Construir o extraer exactamente el payload "AAAA WWWW TTTT" (14 bytes).
- Calcular crc16_modbus(payload, 0, 14, 0xFFFF).
- Comparar CCC con (crc16 & 0xFF).

---

5) Ejemplo completo de validación (paso a paso)

Supongamos que recibes:

0400 0001 6805 210\r\n

1. Separas campos:

- AAAA="0400"
- WWWW="0001"
- TTTT="6805"
- CCC="210"

2. Formas el payload exacto:

"0400 0001 6805"

(14 bytes)

3. Calculas CRC16 Modbus con init 0xFFFF:

- crc16 = CRC16_Modbus("0400 0001 6805", 14, 0xFFFF)

4. Extraes LSB:

- lsb = crc16 & 0xFF

5. Compruebas:

- si lsb == 210 → **trama válida**
- si no → **trama corrupta** (descartar y contar error)

---

6) Recomendaciones para implementarlo bien

- Leer por UART **hasta \n**, y comprobar que la línea termina con \r\n.
- Validar formato fijo:
  - 4 dígitos, espacio, 4 dígitos, espacio, 4 dígitos, espacio, 3 dígitos
- Recalcular CRC siempre sobre **14 bytes exactos** del payload.
- Si falla el CRC:
  - descartar la línea,
  - incrementar contador de error,
  - (opcional) mostrar la línea cruda para diagnóstico.