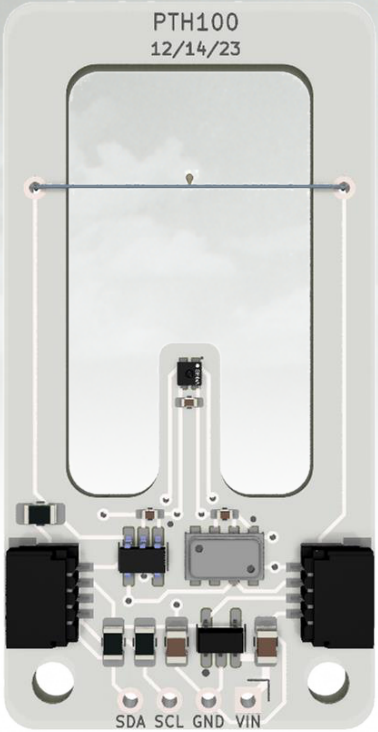
PTH100

Pressure · Temperature · Humidity

Sensor Breakout — Datasheet & Integration Guide



TracerLab

Document revision 1.1 • July 15, 2026

Contents

CHAPTER 1

Overview

The **PTH100** is a high-performance, research-grade breakout board that measures **P**ressure, **T**emperature, and **H**umidity — the same sensing suite that flies in the Tracer Lab **PTHSonde** radiosonde. A fast bead thermistor is paired with a 12-bit ADC-to-I^{2}C converter, so every channel reads over a single I^{2}C bus at 3.3 V logic.

A built-in regulator accepts input voltages from **3.3 to 16.0 V**, adapting the board to a wide range of power configurations, and dual **Qwiic** connectors offer plug-and-play integration with most microcontrollers. The **PTH100** has been successfully deployed across diverse applications, including surface instrumentation, high-altitude balloons, drones, mobile mesonets, and aircraft systems.

1.1 Measurement specifications

The table below gives the rated performance of each measurement over the operating envelope. Values match the sensing suite as flown in the PTHSonde.

Parameter	Range	Accuracy	Resolution	Response (τ_{63})
Air temperature	−70 to +50 °C	±0.3 °C	0.01 °C	< 2 s
Relative humidity	0–100 %RH	±1.8 %RH	0.1 %RH	∼ 4 s
Barometric pressure	10–1200 hPa	±1.5 hPa	0.01 hPa	< 20 ms

Table 1.1: Rated measurement performance of the **PTH100** sensor suite.

1.2 Components & bus addresses

Label	Measurement	Component	I^{2}C address
A	Pressure	MS5611-01BA03	`0x77`
B	Temperature	B57541G1103F005 bead thermistor	—
C	ADC to I^{2}C	MCP3221 (12-bit)	`0x4D`
D	Humidity	SHT41A-AD1B-R2	`0x44`

Table 1.2: **PTH100** components and bus addresses.

Note

Temperature is available from three sources: the bead thermistor (primary, true air temperature), the SHT41, and the MS5611. Use the thermistor for atmospheric work — it is directly exposed to the airflow and responds fastest.

CHAPTER 2

Hardware

2.1 Board & dimensions

The board measures **1.84 in** × **0.94 in** (35.6 mm × 23.9 mm) with 0.10 in (2.54 mm) mounting holes. The bead thermistor is suspended in an open window so it samples free-stream air, away from board self-heating.

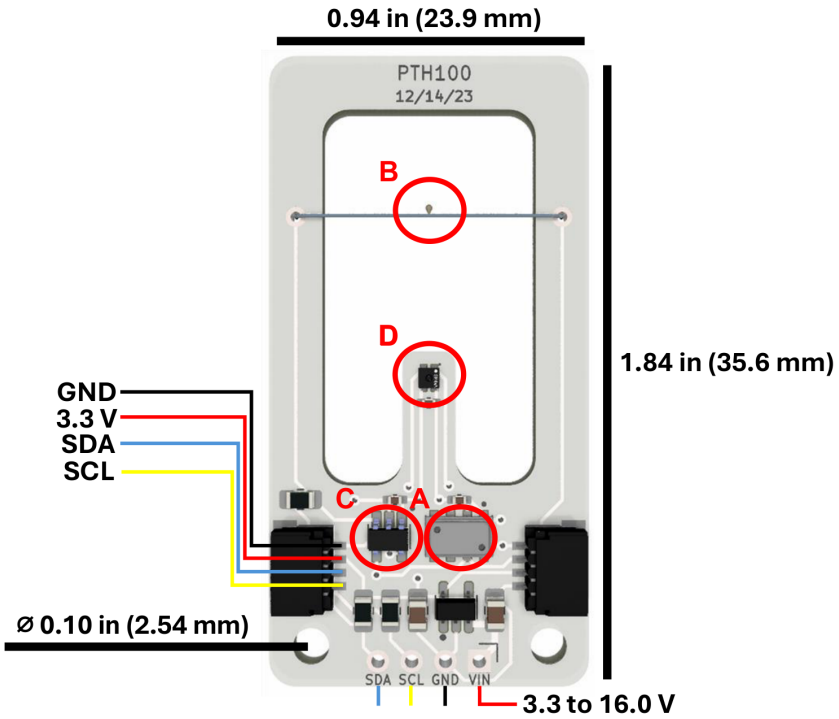


Figure 2.1: **PTH100** pinout and dimensions. Labels A–D correspond to the components in Table 1.2.

2.2 Power & connections

Input	Specification
VIN (header)	3.3 to 16.0 V, on-board regulated
Qwiic	3.3 V
Logic	3.3 V I^2C (SDA / SCL)

Table 2.1: Power inputs and logic levels.

Tip

Keep the thermistor window exposed to airflow and shaded from direct sun for the truest air-temperature readings — radiative heating biases any exposed sensor.

CHAPTER 3

Thermistor Calibration

The **PTH100** uses a TDK Electronics bead thermistor for fast-response temperature measurements. To convert resistance to temperature, the Steinhart–Hart equation is fitted to the manufacturer's calibration values:

$$T = \frac{1}{A + B\,\ln(R) + C\,[\ln(R)]^3}$$

where T is the temperature in Kelvin, R is the thermistor resistance in ohms, and A, B, C are the Steinhart–Hart coefficients. The coefficients were determined by fitting the manufacturer data specifically over the atmospheric range of -60 to $+40\,^{\circ}\mathrm{C}$, optimizing the model for typical environmental conditions:

$$A = 8.498083 \times 10^{-4} \qquad B = 2.608224 \times 10^{-4} \qquad C = 1.304578 \times 10^{-7}$$

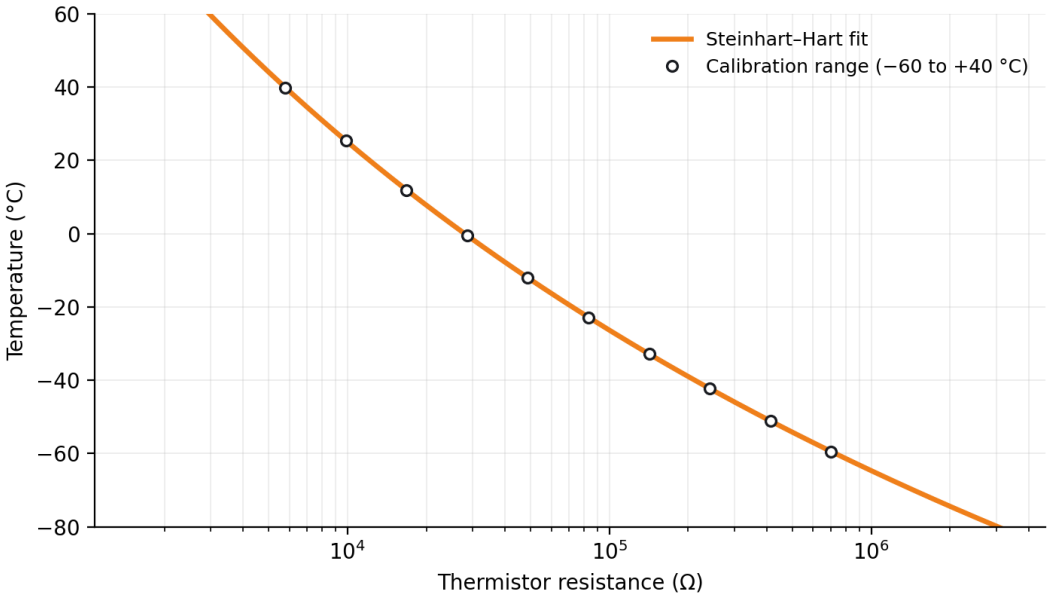


Figure 3.1: Steinhart–Hart fit across the atmospheric calibration range.

CHAPTER 4

Arduino Example

The snippet below reads all **PTH100** channels using standard Arduino IDE libraries (`MS5611` by Rob Tillaart, `Adafruit SHT4x`). Note that temperature is reported by the pressure and humidity sensors in addition to the thermistor.

```
#include <Wire.h>
#include <MS5611.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_SHT4x.h>
#include <math.h>

#define MCP3221_ADDR 0x4D        // thermistor ADC
#define VOLTAGE_REFERENCE 3.3

MS5611 ms5611;
Adafruit_SHT4x sht4 = Adafruit_SHT4x();

void setup() {
  Serial.begin(9600);
  Wire.begin();

  if (!ms5611.begin()) {          // pressure
    Serial.println("ERROR: MS5611 not detected.");
    while (1);
  }
  ms5611.setOversampling(OSR_ULTRA_HIGH);

  if (!sht4.begin()) {            // humidity
    Serial.println("ERROR: SHT4x not detected.");
    while (1);
  }
  sht4.setPrecision(SHT4X_HIGH_PRECISION);
}

void loop() {
  if (ms5611.read() == MS5611_READ_OK) {
    Serial.print("Pressure: ");
    Serial.print(ms5611.getPressure());   // hPa
    Serial.println(" hPa");
  }

  sensors_event_t humidity, temp;
  sht4.getEvent(&humidity, &temp);
  Serial.print("Humidity: ");
  Serial.print(humidity.relative_humidity);
  Serial.println(" %");

  Serial.print("Thermistor: ");
  Serial.print(readThermTemp());          // deg C
  Serial.println(" C");

  delay(2000);
}

// thermistor via MCP3221 + Steinhart-Hart
float readThermTemp() {
```

```
  Wire.requestFrom(MCP3221_ADDR, 2);
  if (Wire.available() >= 2) {
    int adc = ((Wire.read() & 0x0F) << 8) | Wire.read();
    float volts = (adc * VOLTAGE_REFERENCE) / 4095.0;
    float r = 10000 * ((VOLTAGE_REFERENCE / volts) - 1);
    return 1.0 / (0.8498083e-3
                + 2.608224e-4 * log(r)
                + 1.304578e-7 * pow(log(r), 3)) - 273.15;
  }
  return NAN;
}
```

Note

Example output: `Pressure: 954.31 hPa  Humidity: 34.2%  Thermistor: 21.4C` — all three channels on one I²C bus, polled every two seconds.