

RS-TMP6331-100K Precision Thermal Module

High-Linearity Silicon PTC Temperature Sensor by Ramanu Systems

Architecture

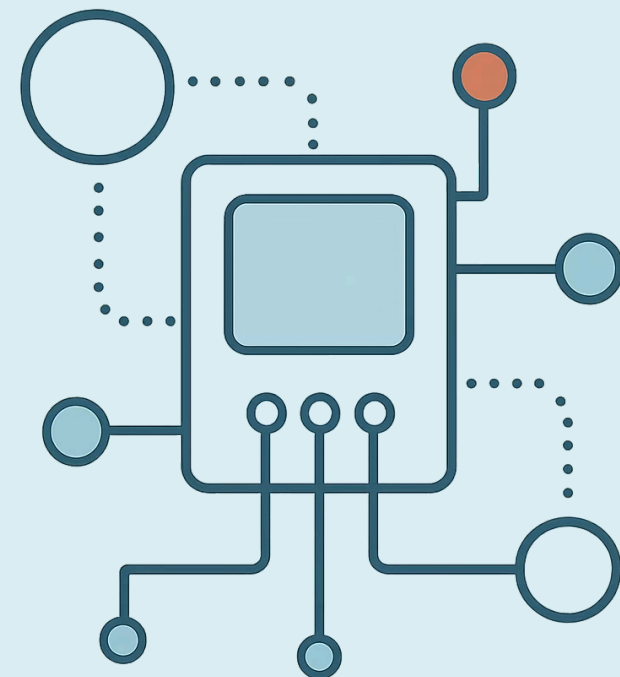
Texas Instruments TMP6331
Silicon Linear PTC with 0.5%
tolerance precision bias resistor

Applications

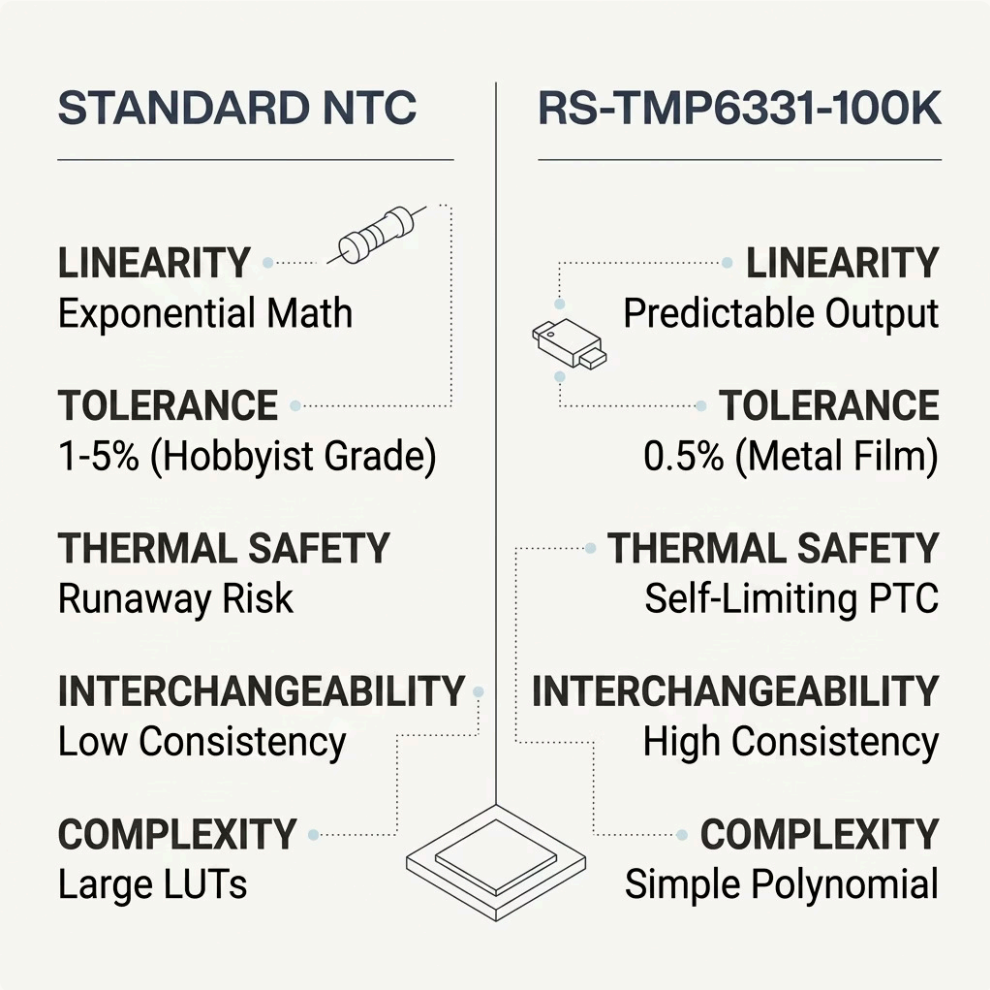
Industrial, automotive, and high-
fidelity electronic systems

Design

High-impedance 100k Ω architecture for minimal self-heating and
simplified software



Key Advantages: Silicon PTC vs. Standard NTC



Ratiometric ADC Measurement

For supply-independent accuracy, implement the module in a **ratiometric** configuration – the ADC uses the same Vcc as its reference, mathematically cancelling supply voltage drift.

Resistance Calculation (Raw ADC Counts)

$$R_{th} = 100,000 \times \left(\frac{ADC_{count}}{ADC_{max} - ADC_{count}} \right)$$

Where **ADC_max** = 1024 (10-bit) or 4096 (12-bit). Do **not** convert to volts first.

Fault Diagnostics

ADC Value	Interpretation
0 – 5	Short Circuit
10 – 900	Normal Operation
1020 – 1023	Open Circuit / Wire Break

 Pro Tip: Select the 3.3V or 5V resistance column in your LUT to account for the Self-Heating Delta (e.g. +1,853Ω at 84°C at 5V).

Advanced Tools

TI Thermistor Design Tool

Input R_bias = 100,000Ω and your Vbias to generate custom 4th-order polynomial coefficients for -40°C to +150°C.

TI TMP6331 Datasheet

Power rating charts, long-term stability data, and advanced noise-reduction techniques for industrial environments.

Reliability & Thermal Longevity

Operating Limits

Mode	Max Temp
Intermittent Spikes	150°C
Continuous Operation	125°C
Storage	-65 to +150°C
Supply Voltage	1.8 to 5.5V

 Constant operation above 125°C accelerates ageing of the PCB dielectric laminate. Ensure adequate mechanical fastening for long-term structural integrity.

Mounting Recommendations

01

Direct Contact

Mount the Aluminium PCB back directly onto the target surface (motor casing, heatsink, transformer).

02

Thermal Interface

Apply thermal paste or a thermal pad to eliminate air gaps for maximum accuracy.

03

Mechanical Fastening

Use M3 screws for tight physical contact. Module dimensions: **20mm × 9mm.**

Width: 9mm

Length: 20mm

Ø3.2mm

Protection Diode

C1 R1 100k R2 100k D1 R_{TEMP63}

5V Gnd o

Pinout Configuration

Pin Number	Board Label	Function	Description
1 (Left)	5V	Power Input (V _{BIAS})	+3.3V to +5.0V DC input power.
2 (Center)	Gnd	Ground (0V)	Common ground connection for power and signal.
3 (Right)	o	Analog Output (V _{TEMP})	Analog output voltage proportional to temperature. (Connect to ADC).

5V Input ('5V' labeled on PCB)

Ground Connection ('Gnd' labeled on PCB)

Analog Output ('o' labeled on PCB)

For fault diagnostics, can noise if the end of the module is not connected to ensure the electrical connection with the supply-independent math is primeval.

Fault logic: $0 \leq '0', R_{TMP63} = V'$

Pins are marked with "5V Gnd o" on the PCB surface below the 3-pin header.

Internal circuit diagram showing V_{BIAS}, R_{BIAS}, R_{TEMP63}, and V_{TEMP}.

only 9×20mm with M3 screw hole for fitting on system with maximum thermal interface

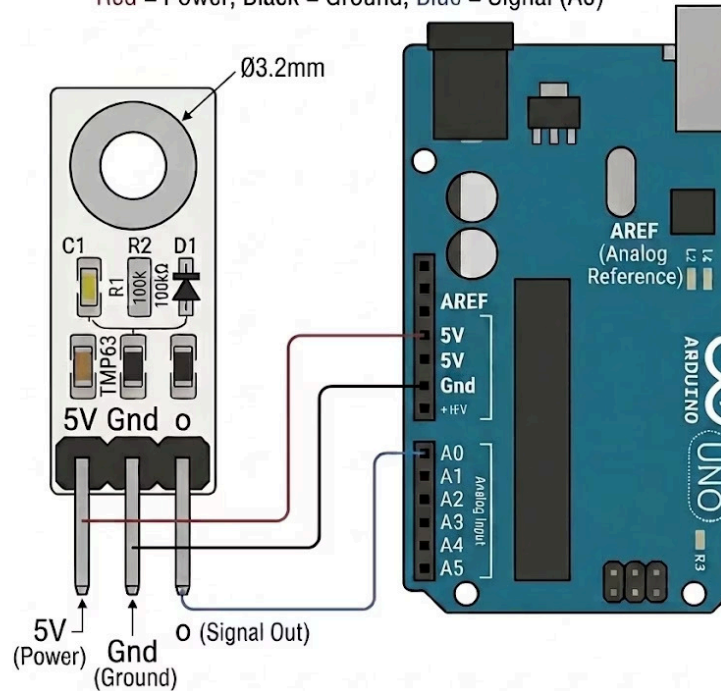
Interfacing Module with Arduino

Hardware Interfacing Diagram

Module Pin	Board Label	Arduino Pin	Description
Pin 1 (Left)	5V	5V	Supply voltage for the divider.
Pin 2 (Center)	Gnd	Gnd	Common Ground.
Pin 3 (Right)	o	A0	Analog Signal Output

SIMPLE CONNECTION DIAGRAM FOR RS-TMP6331-100K

Red = Power, Black = Ground, Blue = Signal (A0)



Many fault diagnostics, can raise if the end
depends to ensure the electrical connection with
supply-independent math is printable.

Fault logic: $0 \leq R_{TMP63} = V$

Connection diagram

Fault Diagnostics & Safety Logic

Professional firmware should use the ADC values to detect hardware failures. Due to the physical limits of the TMP6331, certain readings indicate a fault, not a temperature.

ADC Value (10-bit)	Interpretation	Physical Meaning
0 to 5	SHORT CIRCUIT	Sensor or signal line shorted to Ground.
10 to 900	NORMAL	Sensor is operating within valid range.
1020 to 1023	OPEN CIRCUIT	Sensor disconnected or wire broken.

```
/* * PROJECT: Ramanu Systems RS-TMP6331-100K Integration
 * VERSION: 1.0 (Production Ready)
 * MODULE SIZE: 9x20mm Aluminum MCPCB
 */

// --- Configuration ---
const int SENSOR_PIN = A0;
const float R_FIXED = 100000.0; // 100k 0.5% Precision Resistor
const int ADC_RES = 1024;      // 10-bit resolution
const int SAMPLES = 10;       // Number of samples for averaging

// --- Look-Up Table (5V Characterization) ---
const int LUT_SIZE = 11;
float lut_temp[] = {-40, -20, 0, 25, 40, 60, 80, 84, 100, 120, 150};
float lut_res[] = {66489, 75662, 86019, 100656, 110357, 124383, 139680, 142895, 156278, 174209, 203678};

void setup() {
  Serial.begin(9600);
  // analogReference(EXTERNAL); // Uncomment if 5V is connected to AREF pin
  Serial.println("--- Ramanu Systems: RS-TMP6331 Initialized ---");
}

void loop() {
  // 1. Averaging Filter (Reduces Electrical Noise)
  long adc_sum = 0;
  for(int i = 0; i < SAMPLES; i++) {
    adc_sum += analogRead(SENSOR_PIN);
    delay(5);
  }
  float raw_adc = (float)adc_sum / SAMPLES;

  // 2. Fault Diagnostics (QA Safety Layer)
  if (raw_adc >= ADC_RES - 3) {
    Serial.println("ALARM: SENSOR DISCONNECTED / OPEN CIRCUIT");
  }
  else if (raw_adc <= 3) {
    Serial.println("ALARM: SENSOR SHORT TO GROUND");
  }
  else {
    // 3. Ratiometric Calculation (Supply Independence)
    // Rth = Rfixed * (ADC / (ADC_max - ADC))
    float r_th = R_FIXED * (raw_adc / ((float)ADC_RES - raw_adc));

    // 4. Temperature Interpolation
    float temp_c = getTemperature(r_th);

    // 5. Output Data
    Serial.print("Resistance: "); Serial.print(r_th/1000.0); Serial.print(" kOhm | ");
    Serial.print("Temperature: "); Serial.print(temp_c); Serial.println(" C");
  }

  delay(1000); // 1-second update rate
}

float getTemperature(float res) {
  // Boundary Protection
  if (res <= lut_res[0]) return lut_temp[0];
  if (res >= lut_res[LUT_SIZE - 1]) return lut_temp[LUT_SIZE - 1];

  // Table Scanning & Interpolation
  for (int i = 0; i < LUT_SIZE - 1; i++) {
    if (res >= lut_res[i] && res <= lut_res[i+1]) {
      // Linear slope calculation
      float res_diff = lut_res[i+1] - lut_res[i];
      float temp_diff = lut_temp[i+1] - lut_temp[i];
      float factor = (res - lut_res[i]) / res_diff;
      return lut_temp[i] + (factor * temp_diff);
    }
  }
  return -99.0; // Fallback error
}
```