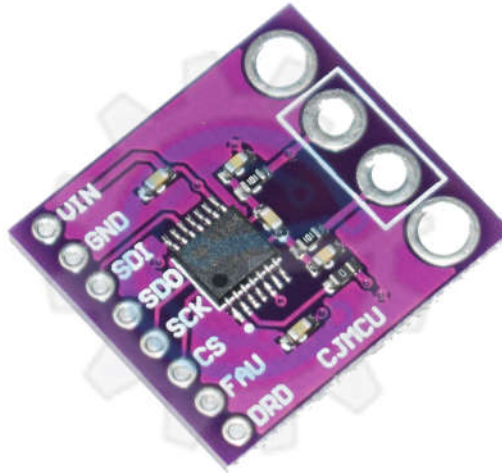


MAX31856 Digital Thermocouple Module



MAX31856 performs cold-junction compensation and digitizes the signal from any type of thermocouple. The output data is formatted in degrees Celsius. This converter resolves temperatures to $0.0078125^{\circ}\text{C}$, allows readings as high as $+1800^{\circ}\text{C}$ and as low as -210°C (depending on thermocouple type), and exhibits thermocouple voltage measurement accuracy of $\pm 0.15\%$. The thermocouple inputs are protected against overvoltage conditions up to $\pm 45\text{V}$.

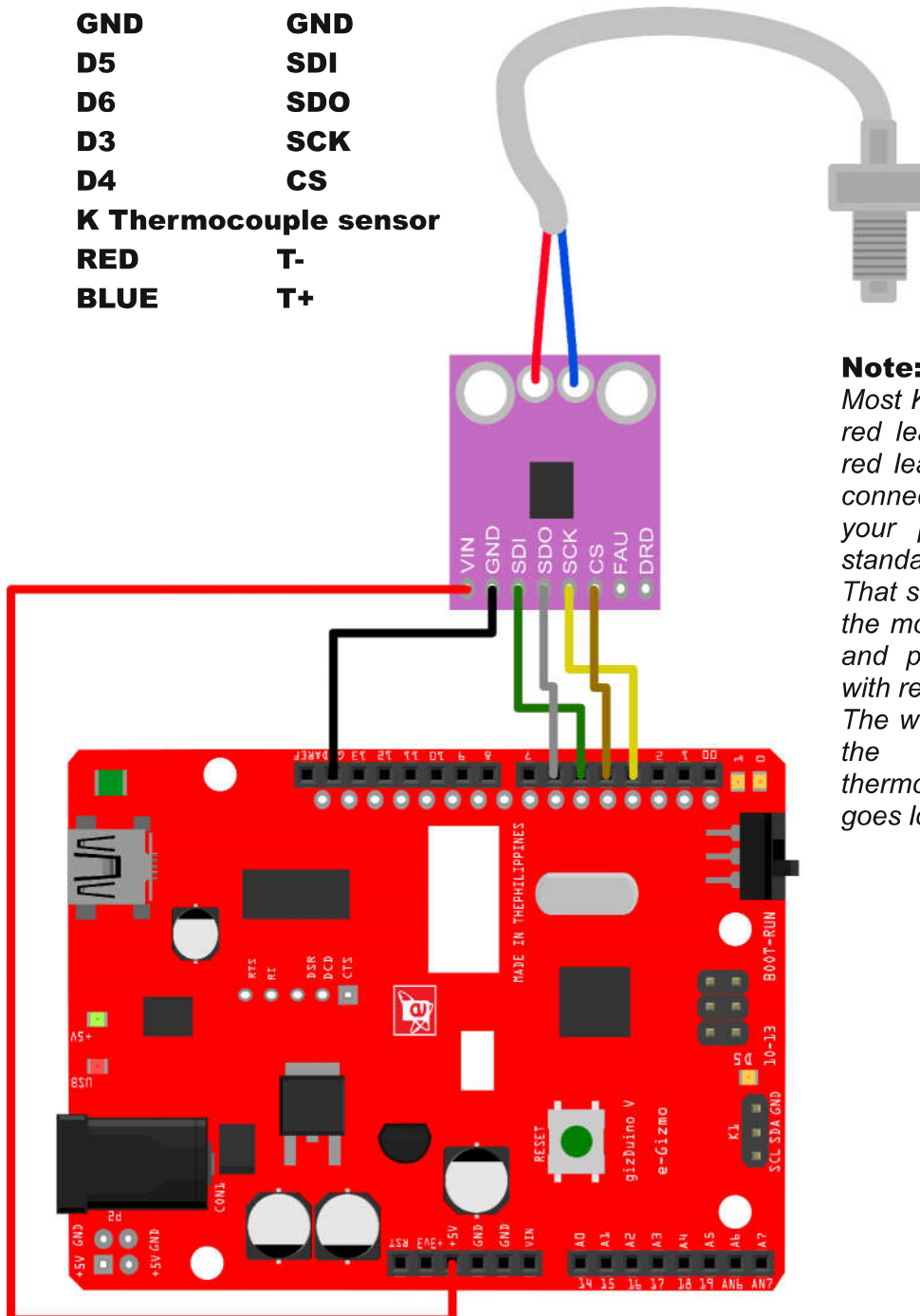
A lookup table (LUT) stores linearity correction data for several types of thermocouples (K, J, N, R, S, T, E, and B). Line frequency filtering of 50Hz and 60Hz is included, as is thermocouple fault detection. A SPI-compatible interface allows selection of thermocouple type and setup of the conversion and fault detection processes.

Benefits and Features:

- ~ Provides High-Accuracy Thermocouple Temperature Readings
- ~ Includes Automatic Linearization Correction for 8 Thermocouple Types
- ~ $\pm 0.15\%$ (max, -20°C to $+85^{\circ}\text{C}$) Thermocouple Full-Scale and Linearity Error
- ~ 19-Bit, $0.0078125^{\circ}\text{C}$ Thermocouple Temperature Resolution
- ~ Internal Cold-Junction Compensation Minimizes System Components
- ~ $\pm 0.7^{\circ}\text{C}$ (max, -20°C to $+85^{\circ}\text{C}$) Cold-Junction Accuracy
- ~ $\pm 45\text{V}$ Input Protection Provides Robust System Performance
- ~ Simplifies System Fault Management and Troubleshooting
- ~ Detects Open Thermocouples
- ~ Over- and Undertemperature Fault Detection
- ~ 50Hz/60Hz Noise Rejection Filtering Improves System Performance

Wiring Connections:**GizduinoV to MAX31856 module**

+5V	VIN
GND	GND
D5	SDI
D6	SDO
D3	SCK
D4	CS
K Thermocouple sensor	
RED	T-
BLUE	T+

**Note:**

Most K thermocouples come with a red lead and a yellow lead. The red lead is normally your negative connection and the yellow lead is your positive. That is industry standard.

That said, some of the suppliers for the module will in fact jack this up and provide you a thermocouple with red indicating positive.

The way to know is if you increase the temperature at the thermocouple tip and the indication goes lowerd

Figure 1. Sample Wiring Diagram with Gizduino V ATmega328P.

Add the MAX31856 library to My Documents> Arduino> libraries

```
#include <MAX31856.h>

// This sample code works with this breakout board:
// http://www.ebay.com/itm/301671408961 (5V)
// http://www.ebay.com/itm/301671398870 (3.3V)
//
// The power requirement for the board is less than 2mA. Most microcontrollers can source
// or sink a lot more
// than that one each I/O pin. For example, the ATmega328 supports up to 20mA. So it is
// possible to power the
// board using I/O pins for power - so you can turn the board on and off (if you want to).
// FAULT and DRDY are not used by the library (see above)
#define SCK 3
#define CS 4
#define SDI 5
#define SDO 6

// MAX31856 Initial settings (see MAX31856.h and the MAX31856 datasheet)
// The default noise filter is 60Hz, suitable for the USA
#define CR0_INIT (CR0_AUTOMATIC_CONVERSION +
CR0_OPEN_CIRCUIT_FAULT_TYPE_K /* + CR0_NOISE_FILTER_50HZ */)
#define CR1_INIT (CR1_AVERAGE_2_SAMPLES + CR1_THERMOCOUPLE_TYPE_K)
#define MASK_INIT (~(MASK_VOLTAGE_UNDER_OVER_FAULT +
MASK_THERMOCOUPLE_OPEN_FAULT))

MAX31856 *temperature;

void setup() {
  // Display temperatures using the serial port
  Serial.begin(9600);
  delay(3000);
  Serial.println("MAX31856 Sample application");

  // Define the pins used to communicate with the MAX31856
  temperature = new MAX31856(SDI, SDO, CS, SCK);

  // Initializing the MAX31855's registers
  temperature->writeRegister(REGISTER_CR0, CR0_INIT);
  temperature->writeRegister(REGISTER_CR1, CR1_INIT);
  temperature->writeRegister(REGISTER_MASK, MASK_INIT);

  // Wait for the first sample to be taken
  delay(200);
}
```

```
void loop () {
  float t;

  // Display the junction (IC) temperature
  // Sometimes the junction temperature is not provided until a thermocouple is attached
  t = temperature->readJunction(CELSIUS);
  Serial.print("Junction (IC) temperature =");
  printTemperature(t);

  // Display the thermocouple temperature
  t = temperature->readThermocouple(CELSIUS);
  Serial.print(" Thermocouple temperature = ");
  printTemperature(t);

  Serial.println();
  delay(1000);
}

// Print the temperature, or the type of fault
void printTemperature(double temperature) {
  switch ((int) temperature) {
    case FAULT_OPEN:
      Serial.print("FAULT_OPEN");
      break;
    case FAULT_VOLTAGE:
      Serial.print("FAULT_VOLTAGE");
      break;
    case NO_MAX31856:
      Serial.print("NO_MAX31856");
      break;
    default:
      Serial.print(temperature);
      break;
  }
  Serial.print(" ");
}
```

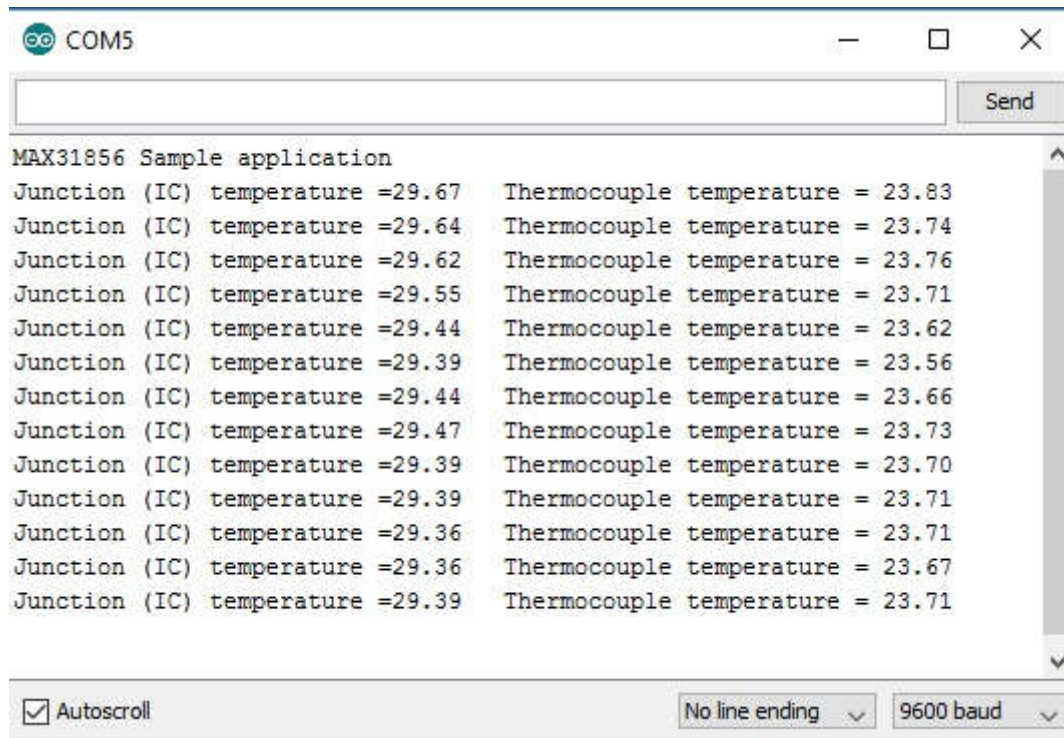


Figure 2. On the Serial monitor you can see the output of the Thermistor temperature sensor.