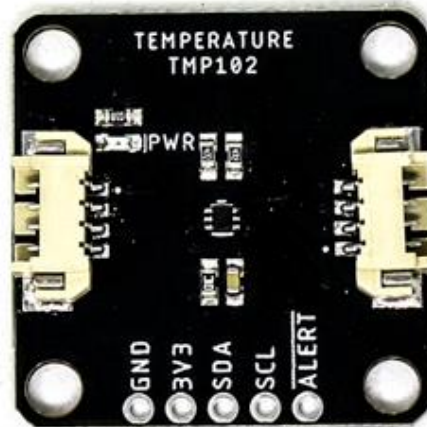




SmartElex Digital Temperature Sensor - TMP102

The Digital Temperature Sensor - TMP102 we've made this just about as easy as it gets. Based off of the original Digital Temperature Sensor Breakout - TMP102, we've added connectors to bring this board into our plug-and-play Ecosystem and added an address jumper instead of breaking out the address pin.



Hardware Overview


Have you heard the phrase "Good things come in small packages"? Well, here's is a prime example! This board centers around Texas Instruments' TMP102 Low-Power Digital Temperature Sensor. This tiny little chip measures 1.6-mm × 1.6-mm and packs quite a nice punch. Here are some of the highlights, but feel free to check out the Datasheet for more information.

Highlights:

- Uses the I²C interface
- 12-bit, 0.0625°C resolution
- Typical temperature accuracy of ±0.5°C
- Supports up to four TMP102 sensors on the I²C bus at a time

Power

Ideally, power will be supplied via the connectors on either side of the board. Alternatively, power can be supplied through the header along the bottom side of the board labeled 3V3 and GND. The input voltage range should be between **1.4-3.6V**.

 **Note:** There is no onboard voltage regulation on this boards. If you choose to provide power via the plated through holes, ensure that your voltage does not exceed the **4V absolute maximum**.

The I²C address of the board is **0x48 by default** , but has 3 other addresses the board can be configured to use.

I²C Pins

The I²C pins break out the functionality of the connectors. Depending on your application, you can connect to these pins via the plated through holes for SDA and SCL.

Alert Pin

The alert pin is an over temperature alert, which has an open-drain and is pulled up through a 10kΩ resistor. The alert can also be read over I²C as shown in the example in the Software Setup and Programming section.

ADDR Jumpers

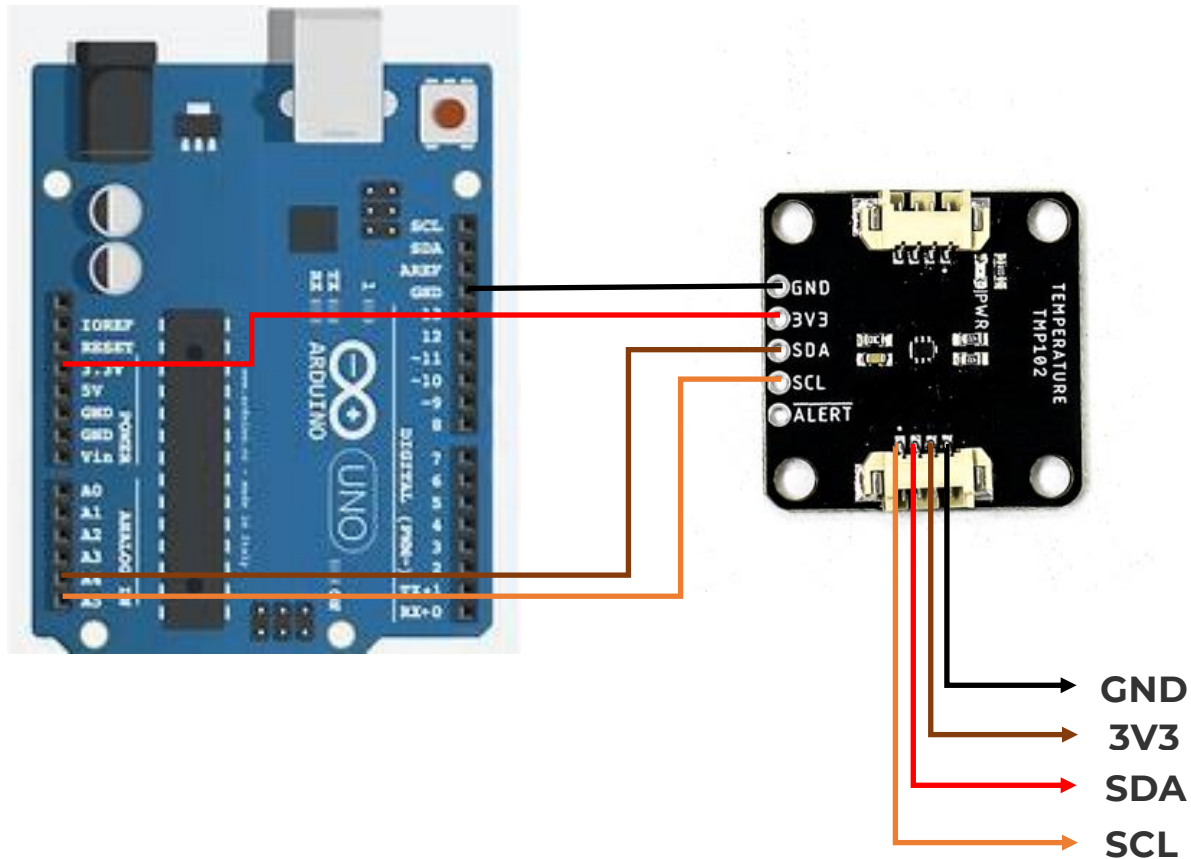
The default I²C address of the board is **0x48**. To change the address, cut the jumper connecting the two pads closest to the 0x48 label. Soldering one of the center pads to one of the outer most pads will change the boards address to the matching label. The TMP102's address is determined by connecting the address pin directly to one of the following:

ADDR	Address
GND	0x48
3V3	0x49
SDA	0x4A
SCL	0x4B

LED Jumpers

Cutting this jumper will disable the Power LED on the front of the board.

Wiring



Arduino	TMP102
SCL(A5)	SCL
SDA(A4)	SDA
5v OR 3.3v	VIN
GND	GND

Software Setup and Programming

You can obtain this library through the Arduino Library Manager by searching for **TMP102**.

Once the library is installed, open Arduino, and expand the examples menu. You should see the TMP102 example.

TMP102 Library Overview

Main functions

These are functions used to read settings and temperatures from the sensor.

- **bool begin(uint8_t deviceAddress, TwoWire &wirePort)** - Takes the device address and I²C bus as optional inputs. If left blank, this function uses the default address 0x48, and uses the Wire bus.
- **float readTempC(void)** - Returns the current temperature in Celsius.
- **float readTempF(void)** - Returns the current temperature in Fahrenheit.
- **float readLowTempC(void)** - Reads T_LOW register in Celsius.
- **float readHighTempC(void)** - Reads T_HIGH register in Celsius.
- **float readLowTempF(void)** - Reads T_LOW register in Fahrenheit.
- **float readHighTempF(void)** - Reads T_HIGH register in Fahrenheit.
- **void sleep(void)** - Put TMP102 in low power mode (<0.5 uA).
- **void wakeup(void)** - Return to normal power mode (~10 uA). When the sensor powers up, it is automatically running in normal power mode, and only needs to be used after * **sleep()** is used.
- **bool alert(void)** - Returns the state of the Alert register. The state of the register is the **same as the alert pin**.
- **void setLowTempC(float temperature)** - Sets T_LOW (in Celsius) alert threshold.
- **void setHighTempC(float temperature)** - Sets T_HIGH (in Celsius) alert threshold.
- **void setLowTempF(float temperature)** - Sets T_LOW (in Fahrenheit) alert threshold.
- **void setHighTempF(float temperature)** - Sets T_HIGH (in Fahrenheit) alert threshold.
- **void setConversionRate(uint8_t rate)** - Sets the temperature reading conversion rate. 0: 0.25Hz, 1: 1Hz, 2: 4Hz (default), 3: 8Hz.
- **void setExtendedMode(bool mode)** - Enable or disable extended mode. 0: disabled (-55C to +128C), 1: enabled (-55C to +150C).
- **void setAlertPolarity(bool polarity)** - Sets the polarity of the alert. 0: active LOW, 1: active HIGH
- **void setFault(uint8_t faultSetting)** - Sets the number of consecutive faults before triggering alert. 0: 1 fault, 1: 2 faults, 2: 4 faults, 3: 6 faults.
- **void setAlertMode(bool mode)** - Sets the type of alert. 0: Comparator Mode (Active from when temperature > T_HIGH until temperature < T_LOW), 1:

Thermostat mode (Active from when temperature > T_HIGH until any read operation occurs.

Example Code

Once the library is installed, open the example code to get started! Make sure to select your board and COM port before hitting upload to begin experimenting with the temperature sensor.

```
#include <Wire.h> // Used to established serial communication on the I2C bus

#include <SparkFunTMP102.h> // Used to send and recieve specific information from
our sensor

// Connections

// VCC = 3.3V

// GND = GND

// SDA = A4

// SCL = A5

const int ALERT_PIN = A3;

TMP102 sensor0;

// Sensor address can be changed with an external jumper to:

// ADD0 - Address

// VCC - 0x49

// SDA - 0x4A

// SCL - 0x4B

void setup() {

  Serial.begin(115200);
```

```
Wire.begin(); //Join I2C Bus

pinMode(ALERT_PIN,INPUT); // Declare alertPin as an input

/* The TMP102 uses the default settings with the address 0x48 using Wire.

   Optionally, if the address jumpers are modified, or using a different I2C bus,
   these parameters can be changed here. E.g. sensor0.begin(0x49,Wire1)

   It will return true on success or false on failure to communicate. */
if(!sensor0.begin())
{
  Serial.println("Cannot connect to TMP102.");

  Serial.println("Is the board connected? Is the device ID correct?");

  while(1);
}

Serial.println("Connected to TMP102!");

delay(100);

// Initialize sensor0 settings

// These settings are saved in the sensor, even if it loses power

// set the number of consecutive faults before triggering alarm.

// 0-3: 0:1 fault, 1:2 faults, 2:4 faults, 3:6 faults.

sensor0.setFault(0); // Trigger alarm immediately

// set the polarity of the Alarm. (0:Active LOW, 1:Active HIGH).

sensor0.setAlertPolarity(1); // Active HIGH

// set the sensor in Comparator Mode (0) or Interrupt Mode (1).
```

```
sensor0.setAlertMode(0); // Comparator Mode.

// set the Conversion Rate (how quickly the sensor gets a new reading)

//0-3: 0:0.25Hz, 1:1Hz, 2:4Hz, 3:8Hz

sensor0.setConversionRate(2);

//set Extended Mode.

//0:12-bit Temperature(-55C to +128C) 1:13-bit Temperature(-55C to +150C)

sensor0.setExtendedMode(0);

//set T_HIGH, the upper limit to trigger the alert on

sensor0.setHighTempF(85.0); // set T_HIGH in F

//sensor0.setHighTempC(29.4); // set T_HIGH in C

//set T_LOW, the lower limit to shut turn off the alert

sensor0.setLowTempF(84.0); // set T_LOW in F

//sensor0.setLowTempC(26.67); // set T_LOW in C

}

void loop()

{

float temperature;

boolean alertPinState, alertRegisterState;

// Turn sensor on to start temperature measurement.

// Current consumption typically ~10uA.

sensor0.wakeup();

// read temperature data
```

```
temperature = sensor0.readTempF();  
//temperature = sensor0.readTempC();  
  
// Check for Alert  
alertPinState = digitalRead(ALERT_PIN); // read the Alert from pin  
alertRegisterState = sensor0.alert(); // read the Alert from register  
// Place sensor in sleep mode to save power.  
// Current consumption typically <0.5uA.  
sensor0.sleep();  
// Print temperature and alarm state  
Serial.print("Temperature: ");  
Serial.print(temperature);  
Serial.print("\tAlert Pin: ");  
Serial.print(alertPinState);  
Serial.print("\tAlert Register: ");  
Serial.println(alertRegisterState);  
delay(1000); // Wait 1000ms  
}
```