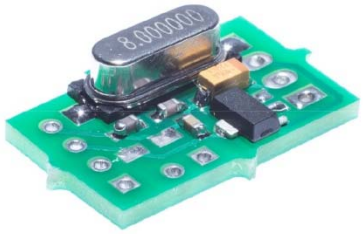
应用

Smartec SMT172 的 IIC 接口电路板

主要特性：



- 整个温度范围内精确测量
- 数字输出的最小增幅为 0.001°C
- 具有 I2C 输出和 PWM 输出（与 SMT160 类似）
- 电路板温度范围 0°C 到 75°C
- 可编程的辅助地址，允许在同一个 I²C 总线上有多个传感器
- 3.3V 稳压器为 SMT172 供电
- 由 I2C 总线本身供电
- 可用于 SMT172 的所有封装

介绍

由于 SMT172 传感器本身的功率很低，传感器的自热可以忽略不计，因而使测量更加准确。片上集成 I²C 接口会产生太多的热量。此微型接口电路板即保留了传感器的测量精度，又提供了 I²C 接口。此微型接口电路板上的微处理器读入 SMT172 的周期调制信号，计算出平均占空比并把它转换成温度以 I²C 形式输出。输出数据的最小增幅为 1mK。
此微型接口电路板也有一个 PWM 模式（而不是 I²C 模式），其 PWM 输出与 Smartec 的前代产品 SMT160-30 完全兼容。对于依旧使用 SMT160-30 的客户，SMT172 与此微型接口电路板结合可以完全取代前代产品 SMT160-30 而无须对现有的测量系统及软件做任何改动。

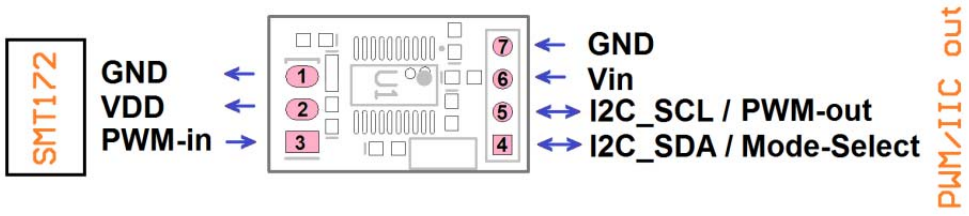
输出信号

输出信号是标准兼容 I²C 信号（从），标准地址是 111（0x6F）。辅助地址的默认值为 115（0x73），用户可以自己定义辅助地址。这个过程将在下面解释。

附加信息

微型接口电路板尺寸：12 x 20 毫米
电压：4V 至 6V
I²C 主地址（7 位）：111
I²C 次地址（7 位）：可编程，0x11 至 0xEF

几何外形图：



应用
Smartec SMT172 的 IIC 接口电路板

模式选择：

SMT172 接口可以在 PWM 模式和 I²C 模式下工作，由管脚 4 控制：

管脚 4	状态	工作模式
H	内部上拉	I²C 输出
L	内部上拉	PMW 输出

在 I²C 总线中，所有主设备和从设备的 SDA 线彼此连接，所有 SCL 也一样彼此相连。
该接口分别有一个 10kΩ的上拉电阻用于 SDA 和 SCL。如果这两个管脚悬空，他们会被上拉至高电平（H）。因此如果此电路板连到 I²C 总线（或让模式选择引脚 4 悬空），电路板将以 I²C 模式工作。 如果需要在 PWM 模式下工作，引脚 4 必须连接到 GND。

从 I²C 接口读取数据：

从 I²C 接口读取必须由能够处理连接线（GND，Vin，SCL 和 SDA）的设备完成。微处理器板卡，例如 Arduino 和 Raspberry Pi 等，对用户来说非常适用，可以很容易地实现温度测量。下面有三个 Python 小程序可以控制 Pi 板和 I²C 接口板的连接。第一个程序从单个 I²C 接口板读取数据。第二个程序显示如何更改该 I²C 接口板的辅助地址。第三个程序显示如何从连接到同一总线的两个 I²C 接口板读取数据，其中之一已经被赋予了另一个辅助地址，如第二个程序所示。

程序 1

```
import smbus
from time import sleep
bus = smbus.SMBus(1)
pda = 0x6f # 111d primary device address
while True:
a = bus.read_byte_data(pda, 0)
b = bus.read_byte_data(pda, 1)
c = bus.read_byte_data(pda, 2)
temp = a*256*256+b*256+c-273150
print temp
sleep(1)
```

实际温度是从三个连续的字节值中计算出来的，字节值储存在地址 111d 处的多个寄存器（寄存器 0 至 2）。 因为 I²C 接口板的输出是所谓的绝对温度（开氏度，单位为毫度），摄氏度结果必须减去数字 273150 才能得到（单位依旧为毫度）。

程序2

```
import smbus
from time import sleep
bus = smbus.SMBus(1)
pda = 0x6F # 111d = primary device address
sda = 0x70 # 112d = secondary device address
sda_write_value = sda * 2 # lsb is skipped!
bus.write_byte_data(pda, 10, sda_write_value) #create secondary address
bus.write_byte_data(pda, 11, sda_write_value) #write copy
print 'sda written'
```

辅助地址从原始出厂默认值 0x73 更改为其他值，上面程序中举例是 0x70。 由于 I²C 芯片只使用 7 个最高有效位，在写入寄存器 pda/10 之前，0x70 必须乘以 2 才能将地址向左移一位。同样，pda/11 也需要再次写入（请不要问我们为什么！）。

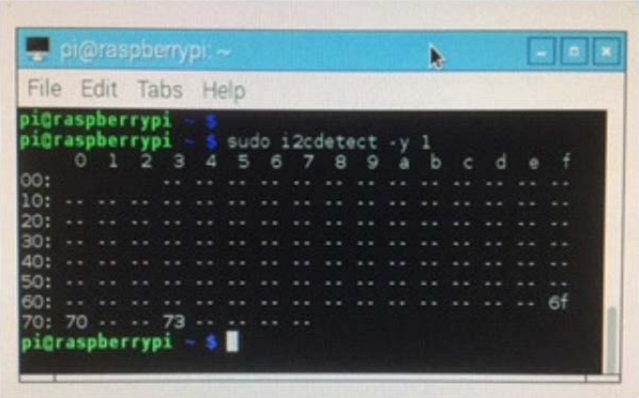
应用

Smartec SMT172 的 IIC 接口电路板

现在，接口板有了新的辅助地址。用户可以将此板与另仪块接口板（出厂默认辅助地址 0x73）一起连接到同一 I²C 总线总线并运行程序 3。在这之前，您可能需要从 Raspberry Pi 运行以下命令命令提示符：

sudo i2cdetect -y 1，

它会给你这样的输出：



该命令列出总线上可用的所有 I²C 地址。在这里你看到 0x6f，0x70 和 0x73。其中 0x6f 是所有 I²C 接口的主地址，0x70 和 0x73 现在是两个接口板的辅助地址。 现在，您可以像程序 3 一样独立读取两个接口板的输出。

程序 3

```
import smbus
from time import sleep
bus = smbus.SMBus(1)
pda = 0x6f # 111d primary device address
sda1 = 0x70 # 112d secondary device address from the first I2C interface
sda2 = 0x73 # 115d secondary device address from the second I2C interface
while True:
a = bus.read_byte_data(sda1, 0)
b = bus.read_byte_data(sda1, 1)
c = bus.read_byte_data(sda1, 2)
temp1 = a*256*256+b*256+c-273150
a = bus.read_byte_data(sda2, 0)
b = bus.read_byte_data(sda2, 1)
c = bus.read_byte_data(sda2, 2)
temp2 = a*256*256+b*256+c-273150
print temp1, temp2
sleep(1)
```

注意：

如果连载 I²C 总线上的接口设备仍然有工厂设置，每次只能对一个接口设备进行重新编程。因为它们不能相互区分。而一旦接口设备具有不同的辅助地址，可以使用这些辅助地址对接口设备任意进行更改，因为可以独立地与每个接口进行通信。还可以通过写入来更改辅助地址。

订单代码:

SMT172TOIIC：SMT172 的 I²C 接口板