Stazioni meteo fai-da-te: il prezzo basso dei sensori può costare caro

Negli ultimi anni, nel mondo dell'elettronica hobbistica e dei makers, si sono diffusi in modo massiccio i sensori digitali basati su protocolli bus. Che sia 1-Wire, SPI o, soprattutto, I2C, questi sensori si sono distinti per la facilità d'uso, il bassissimo costo e l'elevata precisione. In pratica, il paradiso dell'hobbista elettronico.

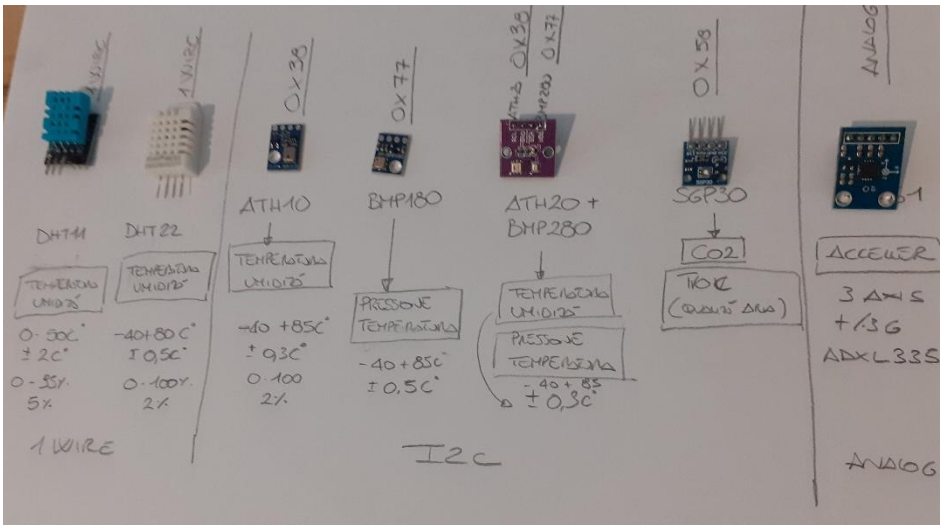

Figura 1 Alcuni sensori che avevo nel cassetto

La misura delle più disparate grandezze fisiche ambientali — temperatura, umidità, CO_2, irraggiamento solare — si riduce praticamente a un unico procedimento: collegare i due fili di alimentazione (0 e 5V) e i due fili del protocollo (SDA e SCL) ai pin dedicati di Arduino, scaricare la libreria, caricare lo sketch di esempio… e voilà, il sensore funziona.

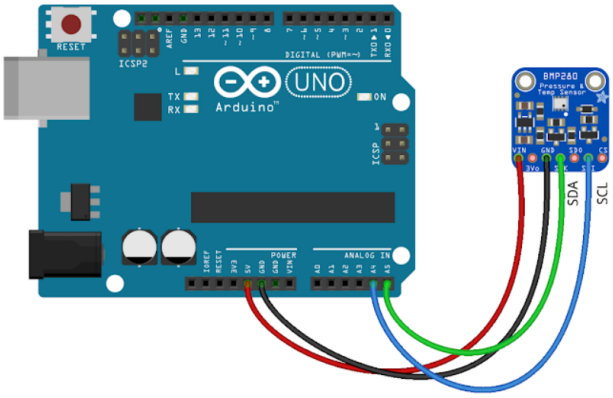

Figura 2 I complessi collegamenti di un sensore I2C ad Arduino

Da qui a trovarsi decine di progetti di stazioni meteo fai-da-te il passo è breve. Addio a stadi amplificatori analogici, tensioni duali, problemi di linearità, tarature di guadagno, offset e a tutte quelle "tristezze" circuitali che un tempo richiedevano nottate di prove, calcoli e compromessi. Oggi, sembrerebbe che tutta questa complessità sia morta e sepolta, resa inutile dai sensori digitali su bus.

Ma è davvero così?

I sensori digitali possiedono davvero un'intrinseca non linearità certa e immutabile, oppure stiamo semplicemente smettendo di guardarla? Da dove nasce questa convinzione?

In realtà, l'adozione dei sensori digitali porta spesso a trascurare aspetti fondamentali come linearità, isteresi, deriva termica e ripetibilità, che nei sensori analogici "lineari" erano invece parametri centrali del progetto. Il fatto che l'uscita sia codificata su un bus non elimina questi fenomeni fisici: li nasconde soltanto dietro un'interfaccia digitale, dando l'illusione che il problema non esista più.

Il problema si manifesta in particolare quando i sensori si discostano dalla logica "pronta all'uso".

In altre parole, per quei dispositivi per cui non esiste una libreria (Arduino) dedicata, oppure per cui la libreria disponibile restituisce valori incoerenti o imprecisi, l'uso diventa estremamente complicato, rendendo il sensore, di fatto, inutilizzabile.

Va detto che per i sensori più comuni esiste quasi sempre almeno una libreria — spesso anche una decina — e questo li rende immediatamente fruibili. Ma provare a implementare un sensore senza libreria è praticamente impossibile, diventano da evitare, relegati a diventare, al massimo, soprammobili elettronici sulla scrivania. La libreria software assume quindi un ruolo funzionale ed essenziale quanto quello del componente hardware stesso, pur rimanendo, in larga parte, un elemento di cui non è sempre chiaro il funzionamento interno.

Proviamo a vedere se, scavando un po', si riesce a capire qualcosa di più. Chissà che quelle conoscenze "classiche" dell'elettronica — quelle che oggi sembrano superate o inutili — non possano tornare utili per capire meglio cosa succede "sotto il cofano" dei dispositivi digitali.

Il nuovo paradigma dei sensori integrati

Innanzitutto, è opportuno analizzare la struttura dei sensori classici, ossia la catena di acquisizione tradizionale a cui siamo abituati. In questa configurazione tipica si distinguono tre elementi fondamentali:

- il sensore vero e proprio,
- il blocco di condizionamento del segnale,
- e il convertitore analogico-digitale (A/D).

(Trascuro il micro che si occupa di convertire il dato digitale in I2C: li l'analogica non c'entra).

Affinché il sensore produca misure attendibili, ciascun componente di questa catena deve possedere caratteristiche di linearità, precisione e accuratezza elevate. Più elevate sono, meglio è.
Proprio per questo motivo, i sistemi di misura tradizionali risultano relativamente costosi: per misure di qualità ci vogliono componenti di qualità, calibrazioni e tarature di qualità e, inutile dirlo, la qualità costa.

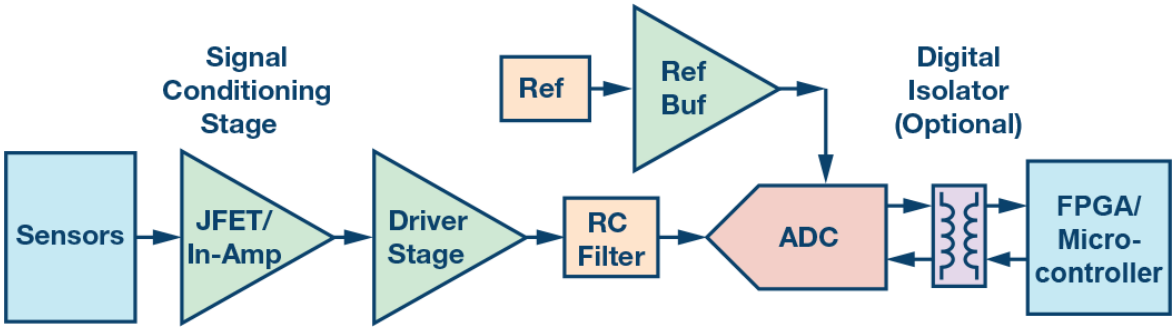

Figura 3 Classica e noiosa catena di acquisizione di precisione

Negli ultimi anni, tuttavia, si è affermato un nuovo paradigma nella progettazione dei sensori, che propone un modo differente di intendere la catena di acquisizione.
A prima vista la struttura può sembrare analoga a quella classica, ma in realtà nasconde una differenza sostanziale. In questa nuova concezione, ogni componente della catena — dal sensore al convertitore A/D fino al microcontrollore — può essere intrinsecamente impreciso, anche di parecchio.
Non si tratta più di sensori fisici tradizionali, come le PT100, la cui linearità è garantita dal comportamento fisico del materiale, bensì di sensori interamente integrati in silicio, nei quali tutti gli elementi (sensore, condizionamento e conversione) sono contenuti in un unico chip.

Figura 4 Il sensore ATH10: è solo il blocchetto metallico in alto a sinistra, il resto serve a non confonderlo con una briciola

Ricordiamo che silicio non è platino, e, pur avendo tante proprietà, non brilla certo per linearità. Questa integrazione comporta inevitabilmente una maggiore imprecisione intrinseca, dovuta alla **diversa natura del principio di misura.** Per compensare tale imprecisione, viene introdotta una memoria interna, all'interno della quale sono memorizzati i fattori di calibrazione necessari a correggere le misure e a garantire prestazioni equivalenti a quelle dei sensori tradizionali.

Non è un caso che tutti i trasduttori moderni, sia di pressione che di umidità **che della grandezza più disparata**, integrino anche un sensore di temperatura. L'acquisizione della temperatura è infatti fondamentale per la compensazione di tutte le grandezze successive — pressione, umidità, massa, intensità luminosa — poiché la temperatura influisce in modo determinante sul comportamento del

sensore stesso.
Essendo realizzati in silicio e condividendo la stessa struttura fisica con il convertitore A/D e il microcontrollore, questi sensori non sono più costituiti da parti separate, come accadeva nei sensori "intelligenti" di precedente generazione (ad esempio gli DHT11 o gli DHT22), ma da un unico blocchetto di pochi millimetri che racchiude l'intero sistema di misura.

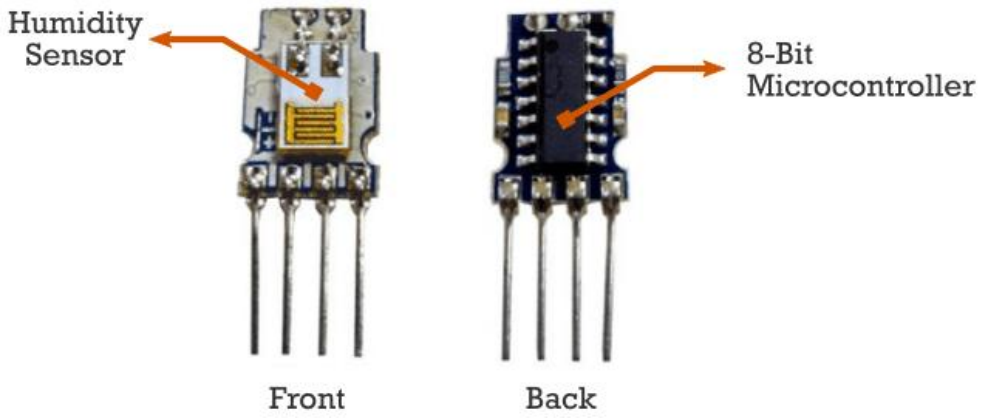

Figura 5 Interno di un "vecchio" sensore DHT22: sono visibili i sensori veri e propri e il micro con A/D sul retro

La catena di acquisizione ottenuta in questo modo — sensore, condizionamento, conversione e controllo — risulta, come detto, intrinsecamente imprecisa, ma la presenza dei fattori di calibrazione memorizzati in EEPROM o memoria non volatile consente di correggere digitalmente gli errori e di ottenere elevati livelli di precisione e linearità.
Tali fattori vengono scritti in fase di fabbricazione per ogni singolo sensore, rendendo possibile la compensazione personalizzata di ciascun dispositivo.
Questa tecnica di calibrazione e compensazione digitale ha reso possibile la produzione di sensori altamente performanti a costi estremamente ridotti. Oggi è infatti possibile reperire sensori integrati con caratteristiche di accuratezza e stabilità comparabili a quelle dei sistemi tradizionali, ma con costi dell'ordine di pochi euro, un risultato che fino a pochi anni fa sarebbe stato considerato impensabile.

Un esempio: il sensore di pressione e temperatura Bosch BMP180

Il BMP180, prodotto da **Bosch Sensortec**, è un sensore digitale che permette di misurare pressione barometrica, temperatura e altitudine. Supporta misurazioni di pressione da 300 a 1100 hPa (corrispondenti a -500 m ÷ +9000 m rispetto al livello del mare) e temperature da -40°C a +85°C. Il consumo è estremamente basso, appena 5 µA con una lettura al secondo, e il sensore è contenuto in un package LGA di dimensioni 3,6 × 3,8 mm.

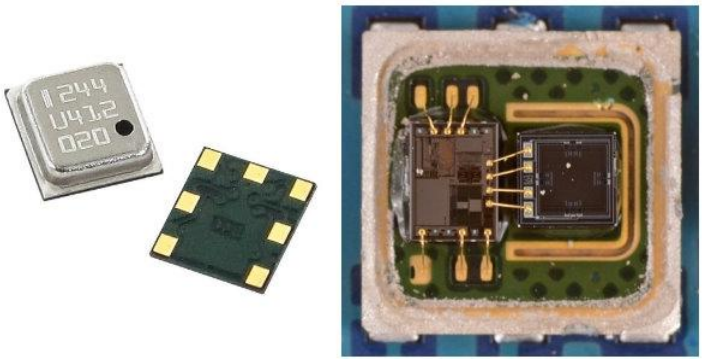

Figura 6 BMP180 esterno e interno

Oltre al valore di pressione, il sensore fornisce anche, e non casualmente, come detto prima, la misurazione della temperatura. I dati vengono trasmessi tramite interfaccia I²C, con indirizzo fisso 0x77. Pur essendo ormai obsoleto e sostituito dal BMP280, il BMP180 è ancora molto usato e ben documentato. La gestione del sensore tramite libreria è estremamente semplice e immediata.

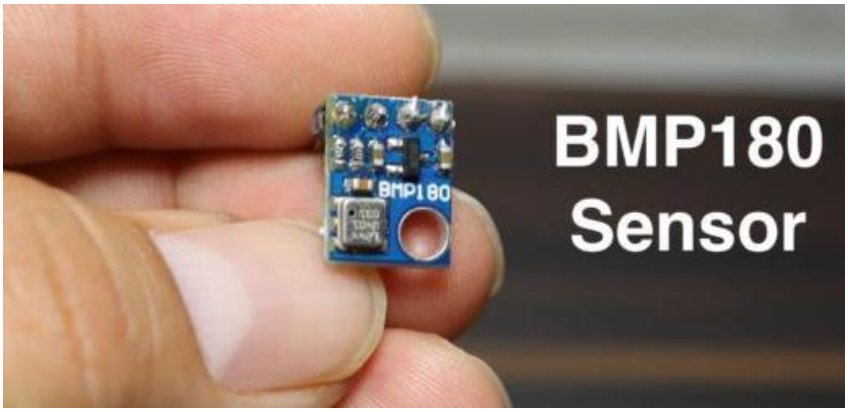

Figura 7 BMP180 montato su una basetta, nelle mani, non troppo pulite, di un tecnico

Ciclo di lettura e ruolo della temperatura

Analizziamo il ciclo di **funzionamento del sensore descritto del datasheet**.
Come già accennato, la rilevazione della temperatura costituisce il punto di partenza di ogni ciclo di misura. Il sensore, infatti, esegue prima la determinazione del valore di temperatura e utilizza successivamente tale informazione per compensare e calcolare le altre grandezze, come la pressione o l'umidità. Nel caso specifico del BMP**1**80, la sequenza di lettura inizia sempre dalla misura della temperatura, che viene poi impiegata nelle funzioni di compensazione dei valori di pressione.

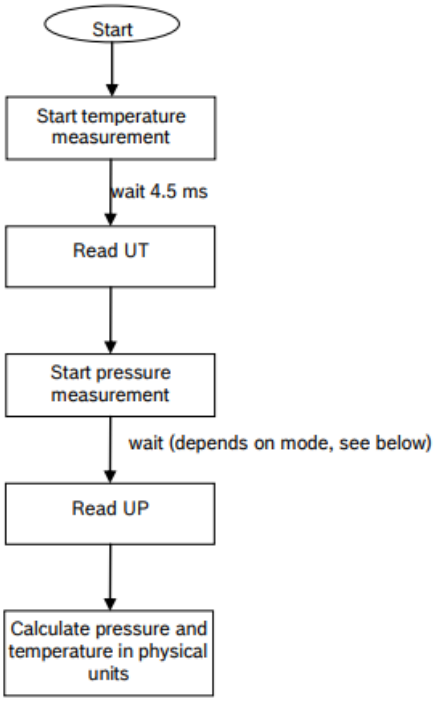

Figura 8 Ciclo di lettura del BMP180

La temperatura, dunque, rappresenta una **grandezza di riferimento essenziale**, poiché influisce in modo diretto sul comportamento del sensore e sulla qualità delle letture successive.
Nel contesto di questa analisi, ci concentreremo esclusivamente sull'estrazione della misura di temperatura, poiché le altre misure, pur basandosi su principi analoghi, risultano più complesse e dipendono strettamente da questo valore per il loro calcolo finale.

Lettura diretta del sensore senza libreria dedicata

Vediamo ora come effettuare la lettura diretta del sensore **senza ricorrere a librerie dedicate**. Il collegamento hardware rimane invariato rispetto alla configurazione standard:

- alimentazione a 3,3–5 V,
- collegamento a GND,
- linee SDA e SCL connesse ai pin dedicati di Arduino.

Per la comunicazione utilizzeremo la libreria Wire, che consente di inviare e ricevere dati tramite il protocollo I²C. Tuttavia, a differenza dell'approccio con librerie ufficiali (come quella fornita **dagli sviluppatori di terze parti**), dovremo gestire manualmente la lettura dei registri, l'interpretazione dei dati grezzi e la conversione dei valori in grandezze fisiche.

Struttura interna del sensore

Lo schema a blocchi del sensore è quello che ci aspettavamo: sensore ADC, micro e memoria.

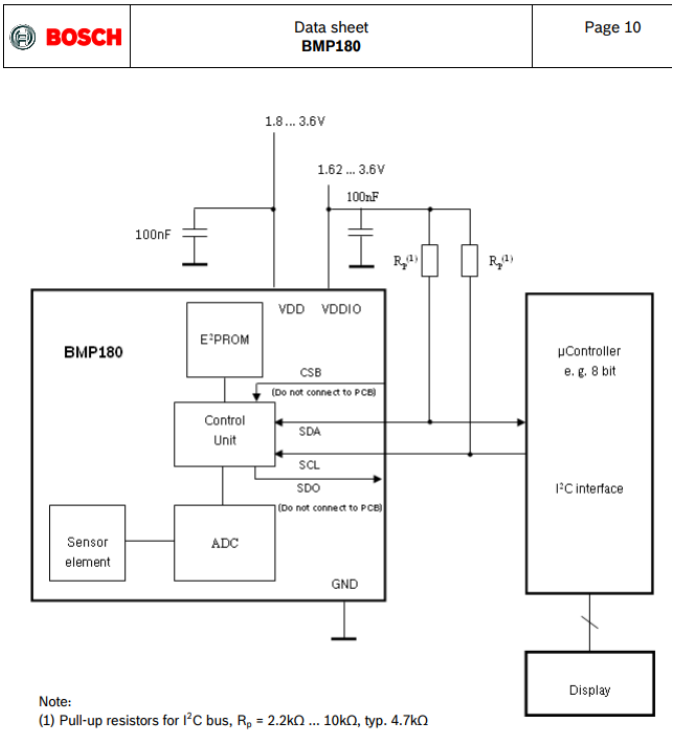

Figura 9 Schema del BMP180 direttamente dal datasheet. Notare la EEPROM in alto

Analizzando nel dettaglio la memoria interna del sensore, si osserva che essa è composta da 176 byte, organizzati in 11 registri da 16 bit ciascuno **(2 celle da 8 bit per ogni registro, da 0x00 a 0xFF per ogni cella).** All'interno di questi registri sono memorizzati i fattori di calibrazione che definiscono il comportamento del sensore e risultano unici per ogni dispositivo, poiché determinati individualmente durante la fase di fabbricazione. Questi dati vengono utilizzati dal microcontrollore interno per correggere le misure grezze provenienti dagli elementi sensibili, compensando gli effetti di non linearità, offset e dipendenza termica. In questo modo, il valore finale restituito dal sensore risulta già corretto e calibrato, pronto per essere interpretato come temperatura, pressione, umidità, o altra grandezza fisica misurata. Per comprendere meglio come tali informazioni influenzino la misura, è possibile leggere direttamente il contenuto dei registri. **P**ossiamo dialogare direttamente con il sensore tramite la libreria Wire, inviando manualmente i comandi che normalmente verrebbero generati in modo automatico**, leggendo la memoria interna del sensore come se si trattasse di una comune memoria EEprom.**
Questo approccio consente di analizzare in dettaglio il protocollo di comunicazione I²C e di osservare come i dati grezzi e i fattori di calibrazione vengano impiegati nel processo di acquisizione e compensazione della misura.

	BMP180 reg adr	
Parameter	**MSB**	**LSB**
AC1	0xAA	0xAB
AC2	0xAC	0xAD
AC3	0xAE	0xAF
AC4	0xB0	0xB1
AC5	0xB2	0xB3
AC6	0xB4	0xB5
B1	0xB6	0xB7
B2	0xB8	0xB9
MB	0xBA	0xBB
MC	0xBC	0xBD
MD	0xBE	0xBF

Figura 10 Registri di calibrazione interni alla memoria EEprom

Suddivisione logica dei coefficienti

Gruppo	Coefficienti	Tipo	Ruolo principale
A… (AC1–AC6)	AC1, AC2, AC3, AC4, AC5, AC6	"Assembly Coefficients" (o "calibrazione principale")	Descrivono **come il sensore reagisce alla temperatura e alla pressione**. Sono legati alla struttura fisica del chip (resistenze, sensori, amplificatori).
B… (B1, B2)	B1, B2	"Bridge" coefficients	Termini di **correzione secondaria** della pressione, per modellare la **non-linearità** del ponte piezoresistivo interno.

Gruppo	Coefficienti	Tipo	Ruolo principale
M… (MB, MC, MD)	MB, MC, MD	“Microcontroller” coefficients	Coefficienti specifici per la **compensazione della temperatura**, usati solo nella formula della temperatura (X2).

I coefficienti di calibrazione (AC1…MD) sono memorizzati nella EEPROM del BMP180 e devono essere letti dal microcontrollore una volta all’avvio. Questi coefficienti servono per calcolare la vera temperatura e pressione compensata dal valore ADC grezzo.

Essendo che noi andiamo ad estrapolare la sola temperatura **utilizziamo solo 4 registri da 16 bit (8 celle da 8 bit) AC5, AC6, MC e MD.** Gli altri servono per la pressione, che per ora tralasciamo.

La procedura è quindi questa: leggiamo il valore GREZZO in uscita dal ADC di temperatura, che prende il nome di **UT (Uncompensated Temperature)**, a 16 bit. Questo valore è piuttosto impreciso e il datasheet ci fornisce le formule per calibrarlo, introducendo dei parametri intermedi X1, X2 e B5.

$$X1 = \frac{(UT - AC6) \times AC5}{2^{15}}$$

$$X2 = \frac{MC \times 2^{11}}{X1 + MD}$$

$$B5 = X1 + X2$$

$$T = \frac{B5 + 8}{16}$$

Figura 11 Formule per calcolare T da UT, si intuisce che AC6 è un offset e AC5 è un guadagno

La successiva determinazione del valore di pressione P utilizza, oltre a UP (Uncompensated Pressure), altri fattori di calibrazione oltre che T, ottenuta da questo procedimento, ma come abbiamo detto, per ora ci fermiamo a T.

In pratica cosa succede?

Colleghiamo al classico Arduino Uno un sensore BMP180. Realizziamo uno programmino che nell’ordine:

1. **Leggere prima i registri di calibrazione** (AC5, AC6, MC, MD, ecc.)

```
// Legge valori di calibrazione dal sensore sapendo la posizione in memoria
uint16_t AC5 = read16(0xB2);
uint16_t AC6 = read16(0xB4);
int16_t MC  = readS16(0xBC);
int16_t MD  = readS16(0xBE);
```

2. **Avviare la misura della temperatura** scrivendo 0x2E su 0xF4.

```
// Funzione per avviare la misura della temperatura
void startTemperatureMeasurement() {
Wire.beginTransmission(BMP180_ADDR);
Wire.write(0xF4);   // registro controllo
Wire.write(0x2E);   // comando misura temperatura (UT)
Wire.endTransmission();
```

3. **Attendere almeno 5 ms** (tempo necessario alla conversione).

4. **Leggere UT** dai registri 0xF6 e 0xF7.

```
// Legge temperatura grezza (UT)
uint16_t UT = read16(0xF6);
Serial.print("UT: "); Serial.println(UT);
```

5. **Calcolare la temperatura reale** usando AC5, AC6, MC, MD e la formula del datasheet.

```
// Calcolo temperatura reale
int32_t X1 = ((int32_t)(UT - AC6) * AC5) >> 15;
int32_t X2 = ((int32_t)MC << 11) / (X1 + MD);
int32_t B5 = X1 + X2;
float T = (B5 + 8) >> 4;       // Temperatura in 0.1 °C
T = T / 10.0;                   // Temperatura in °C
```

6. **Stampa tutti i valori sulla seriale**

```
// Stampa dei valori intermedi e temperatura
    Serial.println("Valori di calibrazione BMP180:");
    Serial.print("AC5: "); Serial.println(AC5);
    Serial.print("AC6: "); Serial.println(AC6);
    Serial.print("MC: ");  Serial.println(MC);
    Serial.print("MD: ");  Serial.println(MD);
    Serial.println("----------------------");
    Serial.println("Calcolo temperatura:");
    Serial.print("X1: "); Serial.println(X1);
    Serial.print("X2: "); Serial.println(X2);
    Serial.print("B5: "); Serial.println(B5);
    Serial.print("Temperatura: ");
    Serial.print(T);
    Serial.println(" °C");
    Serial.println("----------------------");
```

Prendendo quindi 2 BMP180 apparentemente identici, dovremmo leggere valori di lettura adc e taratura diversi, come in effetti accade. Alla stessa temperatura, alla fine, compare lo stesso valore.

```
----------------------
UT: 25544
Valori di calibrazione BMP180:
AC5: 24787
AC6: 17383
MC: -11786
MD: 2950
----------------------
Calcolo temperatura:
X1: 6173
X2: -2645
B5: 3528
Temperatura: 22.10 °C
----------------------
```

```
UT: 25875
Valori di calibrazione BMP180:
AC5: 25334
AC6: 17790
MC: -11786
MD: 2616
----------------------
Calcolo temperatura:
X1: 6250
X2: -2722
B5: 3528
Temperatura: 22.10 °C
----------------------
```

Figura 12 Valori di UN BMP180 paragonati a quelli di un altro BMP180 alla stessa temperatura

Prendiamo ad esempio che il valore GREZZO in uscita da ADC a 16 bit (da 0 a 65536) a 22.10C° equivale a 25544 ad uno e a 25875 all'altro. Usando le formule del BMP180 (stesso procedimento che abbiamo visto):

- $X1 = \frac{(UT - AC6) \cdot AC5}{2^{15}}$
- $X2 = \frac{MC \cdot 2^{11}}{X1 + MD}$
- $B5 = X1 + X2$
- $T = \frac{B5+8}{2^4} \rightarrow$ valore in **0.1 °C**

Con i valori (AC5=24787, AC6=17383, MC=-11786, MD=2950):

- Per **UT = 25 544** ottenevamo (come mostrato):
  X1 ≈ 6173, X2 ≈ −2646, B5 ≈ 3528 → **T ≈ 22.10 °C**
- Per **UT = 25 875** si ottiene:
  X1 ≈ 6424
  X2 ≈ −2575
  B5 ≈ 3849
  Temperatura ≈ 24.10 °C

Quindi con UT = 25 875 la temperatura calcolata sale a circa **24,10 °C** (circa +1,99 °C rispetto ai 22,10 °C precedenti). Che non è proprio poco. Se poi partiamo da questo valore per calcolare la pressione, allora vai con la propagazione di errori.

Alcune considerazioni sul comportamento del BMP180

1. **Lettura dei registri di calibrazione all'avvio**
   All'avvio del microcontrollore leggiamo i registri di calibrazione dal BMP180. Questi valori sono fondamentali per calcolare correttamente la temperatura.

2. **Sensore scollegato durante il funzionamento**
    - Se durante il funzionamento il sensore viene scollegato, il valore di temperatura grezzo `UT` diventa `1111111111111111` (0xFFFF o 65535).
    - Con questo valore, la temperatura calcolata risulta **223,8 °C**, un valore chiaramente fuori dall'intervallo realistico e quindi facilmente scartabile.
    -
3. **Avvio senza sensore collegato**
    - Se invece si avvia il programma senza che il sensore sia collegato, i registri di calibrazione stessi risultano tutti a `0xFFFF` (65535 per i valori unsigned, -1 per i signed).
    - Esempio di output tipico:
    - `UT: 65535`
    - `Valori di calibrazione BMP180:`
    - `AC5: 65535`
    - `AC6: 65535`
    - `MC: -1`
    - `MD: -1`
    - `Temperatura calcolata: 12,8 °C`

In questo scenario, la temperatura risultante è **diversa da quella con sensore scollegato dopo l'avvio**, anche se il sensore non è presente.

4. **Curiosità sul range dei valori**
    - Se il sensore non è collegato o viene scollegato, il valore di temperatura dipende dai valori di calibrazione letti all'avvio. Questo significa che, a seconda della combinazione di parametri di calibrazione, si può ottenere una temperatura "di default" variabile, ma comunque in un range **plausibile** per il sensore. Che non è proprio il massimo, soprattutto per misure ambientali.

Conclusioni

Speriamo che questo articolo e qualche piccolo esperimento abbiano fatto un po' di luce sui moderni sensori in protocollo digitale. In apparenza sembrano semplici, quasi "amichevoli", ma sotto la superficie nascondono una certa complessità… un po' come iceberg tecnologici: quello che si vede è solo la punta!

Ci siamo concentrati sul BMP180, ma i sensori più moderni sono molto più sofisticati e pieni di funzionalità… e insidie. La matematica dietro la correzione dei dati e la linearità è un vero oceano da navigare: formule, coefficienti, bit nascosti… insomma, una crociera tra dati con qualche iceberg invisibile sotto la superficie.

La regola d'oro? Usarli con consapevolezza, riconoscendo la loro complessità e fragilità. Se li si sottovaluta, si rischia un "disastro tecnologico": valori improbabili, calcoli sballati e un sacco di grattacapi.

Parodi Marino