Esercitazione N° 1

Arduino e sensore di temperatura TMP36

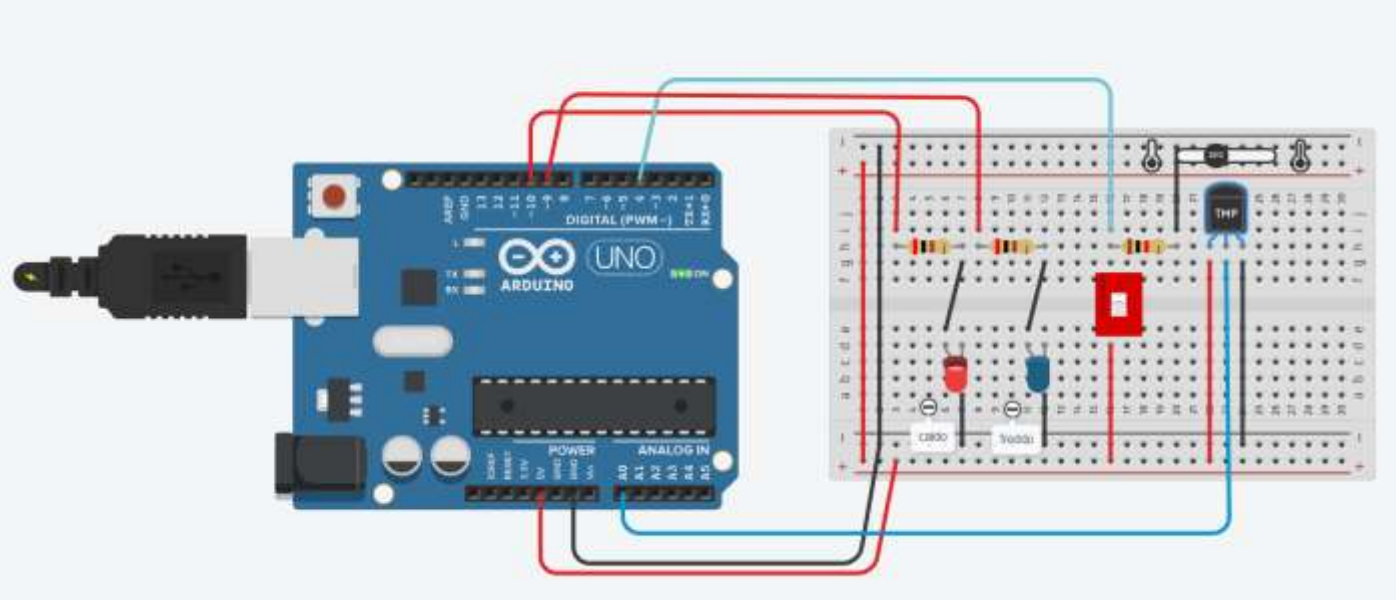
Realizzare un sistema di controllo della temperatura tramite il sensore TMP36 che rilevi la temperatura ogni 2s. in un ambiente.

Se la temperatura è superiore a 20°C viene acceso un LED ROSSO mentre se è inferiore viene acceso un LED BLU.

Il sistema deve funzionare solo se è stato attivato il pulsante START.

Prevedere successivamente la lettura della temperatura media (4 letture ogni 2s) in modo da ridurre gli errori del sensore.

Cablaggio sensore con Arduino



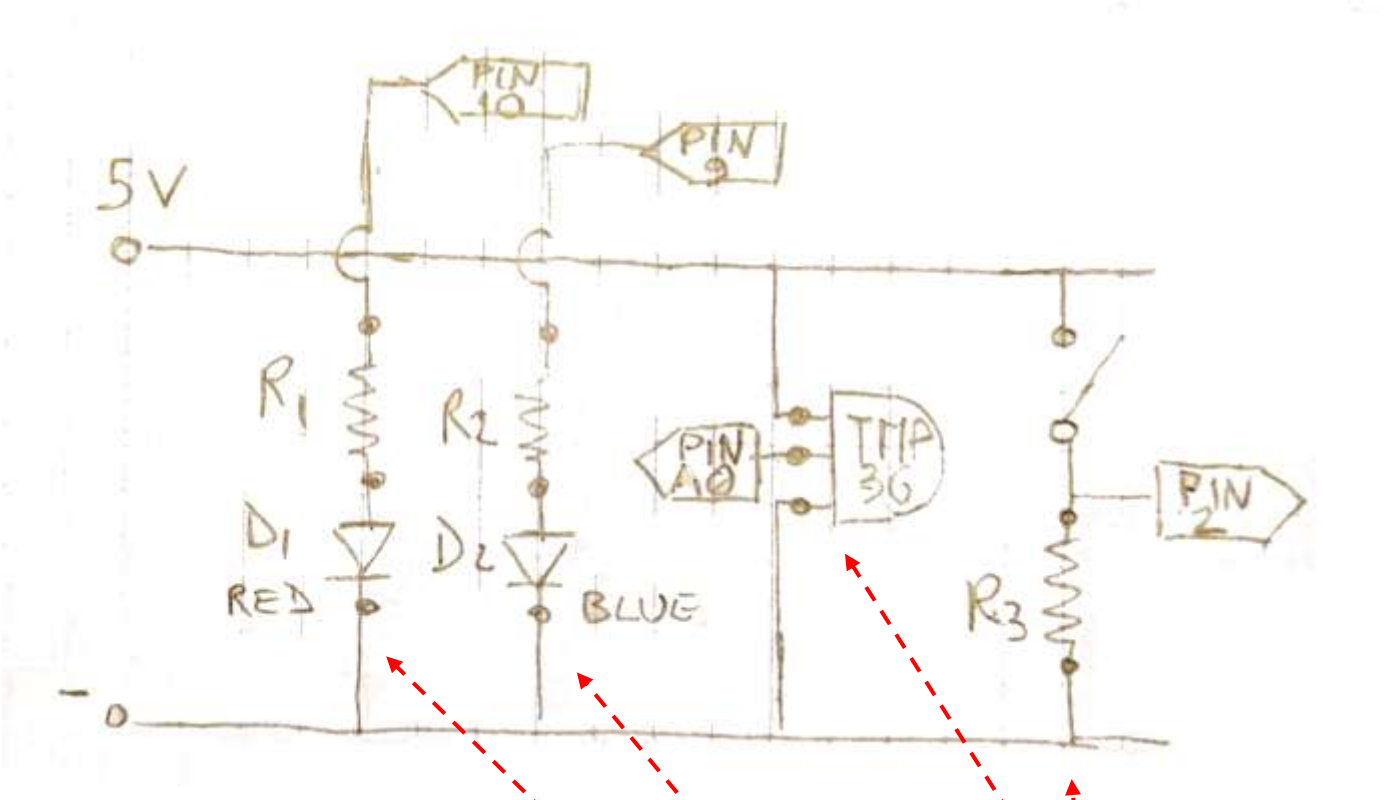

```
// Lettura sensore TMP36
val_ADC = analogRead(tmp36);  // 0-1024
Serial.print("ADC= ");
Serial.println(val_ADC);

// conversione ADC sensore TMP36 in °C
temp = ((val_ADC * 0.00488) - 0.5) / 0.01;
Serial.print("T= ");
Serial.println(temp);
```

NB: 0.00488 =5Vdc /1024 precisione ADC

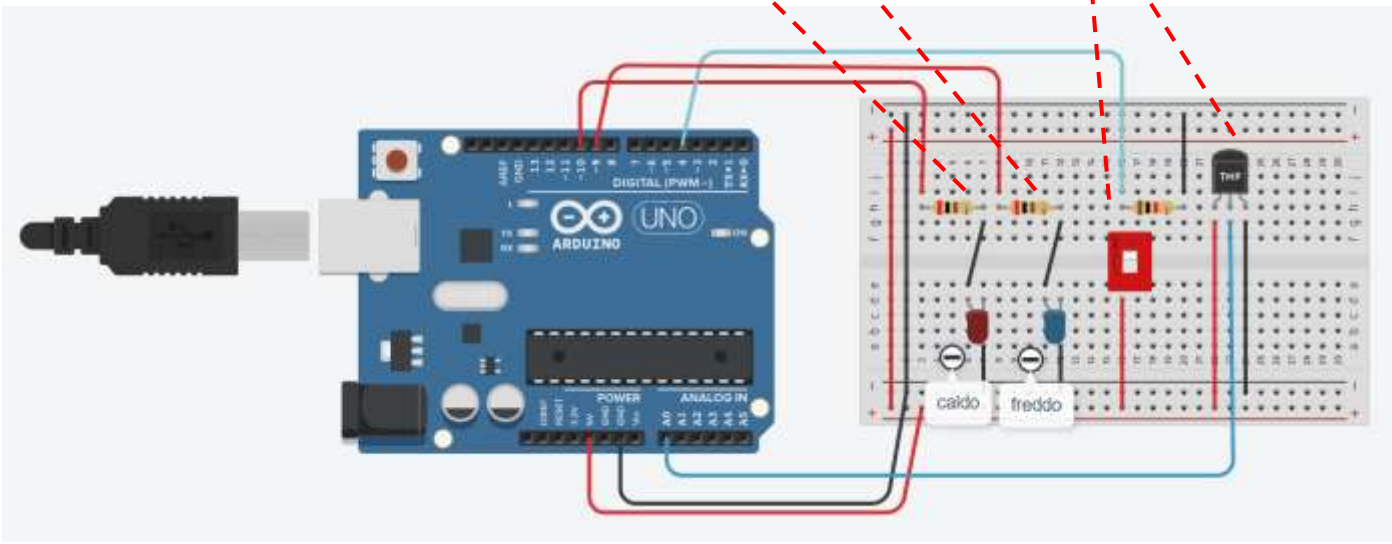
Il TMP36 permette di
acquisire temperature comprese nell'intervallo tra 40°C e +125°C restituendo in uscita valori di tensione lineari tra circa 0.1Vdc e 1.7Vdc.

Una variazione di grado produce una variazione della tensione di uscita pari a 10mV.
Alla temperatura di 0°C il sensore eroga una tensione di 500mV.

SCHEMA ELETTRICO



SCHEMA REALE



```
// Pin Arduino
int led_rosso = 10;
int led_blu = 9;
int start= 4;
int tmp36= 0;

// Stato pulsanti
int  stato_bottone = 0;

// Lettura sensore TMP36
int val_ADC= 0;
float  temp = 0;

void setup()
{
  pinMode(led_rosso, OUTPUT);
  pinMode(led_blu, OUTPUT);
  pinMode(start, INPUT);
  pinMode(tmp36, INPUT);
  Serial.begin(9600);
}

void loop()
{
  Serial.write("AVVIO ...");

  // stato del bottone START
  stato_bottone = digitalRead(start);
  Serial.print("START ");
  Serial.println(stato_bottone);

  if (stato_bottone == 1) {
    val_ADC = analogRead(tmp36);  // Lettura sensore TMP36  → 0-1024
    Serial.print("ADC= ");   Serial.println(val_ADC);
    temp = ((val_ADC * 0.00488) - 0.5) / 0.01;       // conversione ADC sensore TMP36 in °C
    Serial.print("T= ");    Serial.println(temp);

    // Controllo valore temperatura
    if (temp>= 20.0) {
      digitalWrite(led_rosso, HIGH);  digitalWrite(led_blu, LOW);
    }
    else
    {
      digitalWrite(led_rosso, LOW);  digitalWrite(led_blu, HIGH);
     }
  }
  else
  {
     digitalWrite(led_rosso, LOW);
     digitalWrite(led_blu, LOW);
   }
  delay(2000); // Wait for 2000 millisecond(s)
}
```

Come utilizzare il sensore di temperatura TMP36

La conversione di grandezze fisiche in elettriche è un aspetto che prima o poi viene affrontato quando iniziamo a realizzare piccoli esperimenti con Arduino. Oggi la sensoristica è ricca di dispositivi capaci di svolgere questa conversione (come il sensore di temperatura TMP36).

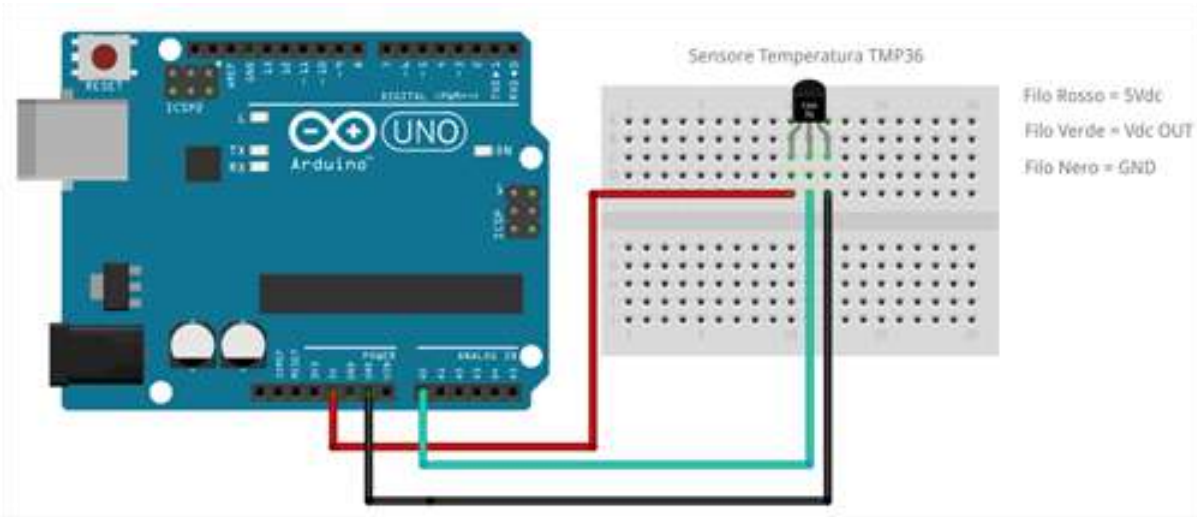
Nello specifico, per quanto riguarda la misura della temperatura abbiamo in commercio dispositivi di ogni sorta, da quelli ultrasensibili a quelli con precisioni dell'ordine del centesimo di grado, dai costi contenuti ai costi esorbitanti. A tal proposito ho in precedenza utilizzato sia il sensore di temperatura analogico MCP9700 sia il sensore digitale MCP9803.

Il segnale di conversione di questi sensori può essere sia analogico (variazione di tensione in funzione della variazione della temperatura) sia digitale (con convertitore analogico digitale ed invio dei dati su linea I2C, SPI o 1Wire).

Per i nostri esperimenti ci accontenteremmo di un sensore molto diffuso, dal costo contenuto e dal semplice utilizzo, il **sensore TMP36** prodotto da Analog Device.

Il **TMP36** permette di acquisire **temperature** comprese nell'**intervallo** tra **-40°C e +125°C** restituendo in uscita valori di tensione lineari tra circa **0.1Vdc e 1.7Vdc**.
Una variazione di grado produce una variazione della tensione di uscita pari a 10mV; alla temperatura di **0°C** il sensore eroga una tensione di **500mV**.

Il circuito che ho realizzato è molto banale, si limita a collegare il sensore direttamente ad **Arduino UNO** tramite la porta analogica **A0**:



guardando frontalmente il **TMP36** troviamo sul lato sinistro il **pin di alimentazione**, sul pin centrale il **segnale in uscita** e sul pin destro il **collegamento a massa**.

Il primo sketch di esempio legge dalla **porta analogica A0** il valore di tensione che eroga il **TMP36**:

```
void setup()
{
//Init Seriale
Serial.begin(9600)
}
void loop()
{
delay(500);
int val_ADC = analogRead(0);
//invio il dato acquisito al pc
Serial.println(val_ADC);
}
```

Il codice precedente non fa altro che convertire in forma digitale la tensione presente sul piedino A0, generata dal TMP36. Questo valore deve essere ora interpretato in modo da ottenere direttamente il valore in °C. Per far ciò abbiamo bisogno del datasheet del componente, e precisamente il grafico di conversione °C/Vdc, rappresentato anche nella figura seguente:

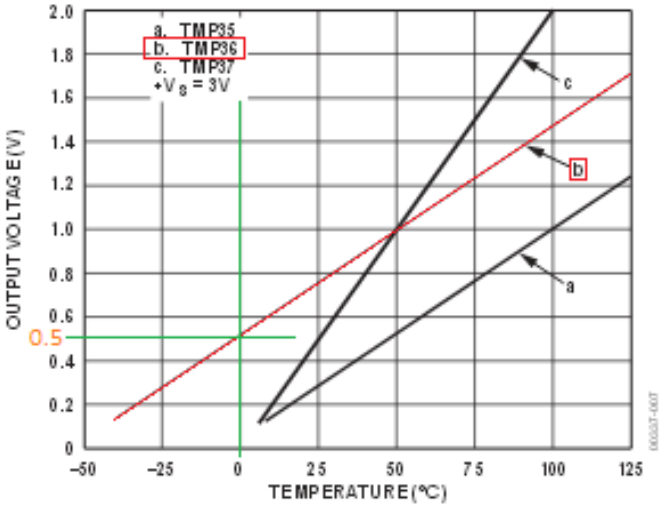


Figure6. Output Voltagevs. Temperature

come possiamo osservare dal grafico (per il **TMP36** la linea b evidenziata in rosso) per una tensione di uscita di 0.5Vdc il sensore rileva la temperatura di 0°C.
Questo dato ci permette subito di intuire che tensioni inferiori a 0.5Vdc indicano una temperatura sotto lo zero.

Inoltre sappiamo che una variazione di grado si ripercuote con una variazione di tensione di 10mV.
Quindi, passando agli esempi, se sul pin A0 sono presenti 510mV significa che il sensore sta rilevando una temperatura di 1°C (510mV – 500mV = 10 mV variazione di 1°C).

Non ci resta che interpretare i dati a nostra disposizione e scrivere uno sketch che restituisca il valore in °C:

```
//variabili globali
int val_Adc = 0;
float temp = 0;
void setup()
{
//init seriale
Serial.begin(9600);
}
void loop()
{
//leggo dalla porta A0
val_Adc = analogRead(0);
//converto il segnale acquisito in un valore
//espresso in gradi centigradi
temp = ((val_Adc * 0.00488) - 0.5) / 0.01;
//invio il dato sulla seriale
Serial.println(temp);
//ritardo di mezzo secondo
delay(500);
}
```

la formula che converte il valore acquisito in gradi centigradi è la seguente

T(°C) = ((valoreADC * PrecisioneADC) – TensioneZeroGradi) / stepGradoTensione

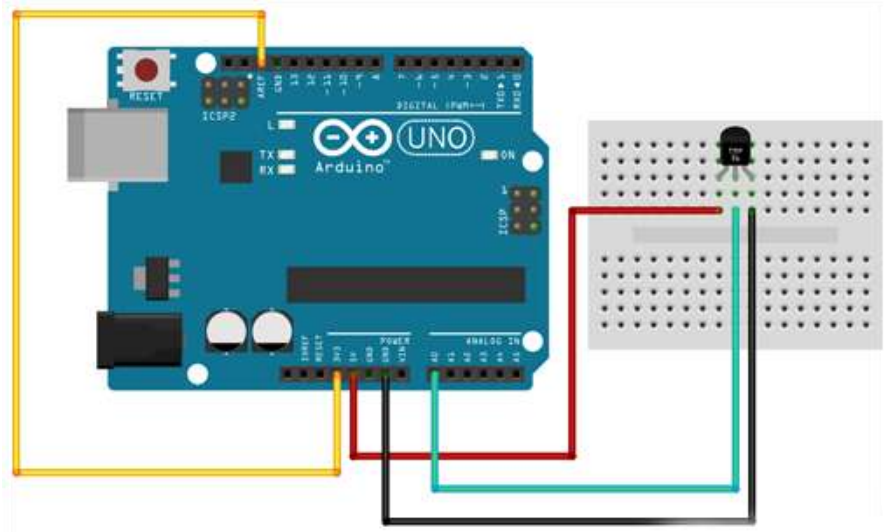
dove

T(°C) = valore della temperatura in gradi centigradi
valoreADC = valore della conversione analogico digitale restituito da analogRead
PrecisioneADC = questo valore è ottenuto dividendo la tensione di riferimento dell'ADC (default 5Vdc) e il numero massimo restituito dalla conversione (1024). (5Vdc /1024 = 0.00488)
TensioneZeroGradi = tensione di uscita dal sensore quando rileva una temperatura di 0°C
stepGradoTensione = variazione di tensione per ogni variazione di grado (0.01 = 10 mV)

Come aumentare la risoluzione nelle letture del TMP36

Eseguendo l'ultimo sketch ci accorgiamo che i valori di temperatura non sono proprio quelli che ci aspettiamo. Considerando la **precisione** del sensore **TMP36** che si attesta a **+/- 2°C** per tutta la scala, otteniamo comunque dei valori che differiscono di circa 5-7°C dal valore reale. Questo 'errore' dipende da diversi fattori tra cui imprecisione dell'ADC, rumore del segnale e approssimazione dei valori nei calcoli.
Per limitare questi errori possiamo adottare due tecniche, la prima è quella di usare il **pin AREF** dell'**Arduino UNO** per dare al convertitore analogico digitale dell'ATmega328 un riferimento di tensione più basso (di default è 5Vdc), l'altra tecnica è quella di effettuare diverse letture ed eseguire una media dei valori letti.

Nello schema seguente viene indicato come collegare il **pin AREF** al **pin power di 3.3Vdc**:



Anche via software dobbiamo indicare al microcontrollore che intendiamo usare il pin AREF per le operazioni di conversione A-D. l'istruzione è la **analogReference()** e viene impiegata in questo modo:

```
//variabili globali
int val_Adc = 0;
float temp = 0;
void setup()
{
//init seriale
Serial.begin(9600);
//utilizzando l'istruzione analogReference
//indico al convertitore AD che deve impiegare
//la tensione presente sul pin AREF come
//valore di riferimento per la conversione
analogReference(EXTERNAL);
}
```

Usando la tensione di 3.3Vdc come riferimento per la conversione analogico digitale otteniamo che il valore **PrecisioneADC** nella formula precedente cambia da 0.00488 a 0.0032 (3.3Vdc / 1024 = 0.0032), aumentando così la precisione della formula.
Impieghiamo anche la lettura multipla di valori per cercare di minimizzare le fluttuazioni, come descritto nel codice seguente:

```
int val_Adc = 0;
float temp = 0;
void setup()
{
//init seriale
Serial.begin(9600);
//utilizzando l istruzione analogReference
//indico al convertitore AD che deve impiegare
//la tensione presente sul pin AREF come
//valore di riferimento per la conversione
analogReference(EXTERNAL);
}
void loop()
{
//ritardo di mezzo secondo
delay(500);
//init variabile
val_Adc = 0;
//eseguo un ciclo
for (byte Ciclo = 0; Ciclo&lt;100; Ciclo++)
{
//acquisisco il valore e lo sommo alla
//variabile
val_Adc += analogRead(0);
//questo ritardo serve per dare il tempo
//all ADC di eseguire correttamente
//la prossima acquisizione
delay(10);
}
//eseguo la media dei 100 valori letti
val_Adc /= 100;
//calcolo la temperatura in °C
temp = ((val_Adc * 0.0032) - 0.5) / 0.01;
//invio i dati al computer
Serial.println(temp);
}
```

in questo modo otterremmo delle misure di temperatura che si avvicineranno molto al valore reale.