Sensore DTH11 con Arduino

Il sensore DHT11 (**fig. 1**) è un dispositivo che, alimentato a 3÷5 V, fornisce un'informazione digitale di temperatura e umidità ambientale, che aggiorna al suo interno ogni un secondo, con accuratezza di qualche percento (**tab. 1**).

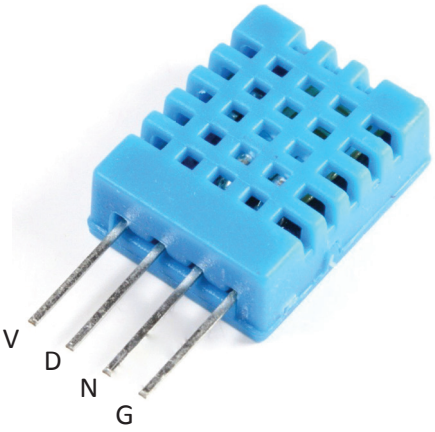

Fig. 1. *DHT11.*

Tab. 1 – Prestazioni del sensore DHT11		
DTH11	**Range**	**Accuratezza**
Temperatura	0 – 50 °C	±2 °C
Umidità	20 – 90% RH	±5% RH

La linea di comunicazione è bidirezionale a un solo filo (*Single-Wire Two-Way*), con il master che chiama e il dispositivo che risponde. La linea è tenuta alta, a riposo, mediante una resistenza di pull-up (**fig. 2**).

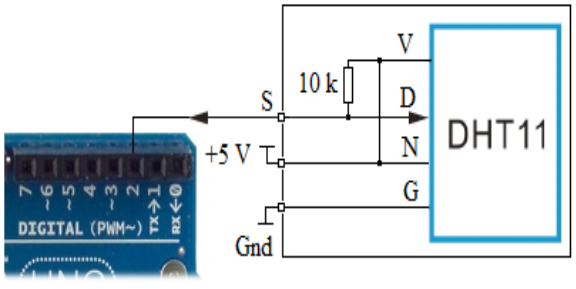

Fig. 2. *Connessione tra Arduino e DHT11.*

Se il DTH11 è fornito su circuito stampato, risulta completo di pull-up e con connettore a 3 pin.
Quando il dispositivo è a riposo assorbe 150 µA, mentre quando è attivo il consumo passa a 1 mA.
Secondo il protocollo, il master attiva la comunicazione inviando un impulso di start attivo basso, che dura almeno 18 ms, al quale il dispositivo risponde entro 20÷40 µs con un impulso di accettazione di 80 µs, seguito, dopo altri 80 µs di pull-up, da una stringa di 40 bit (**fig. 3**).

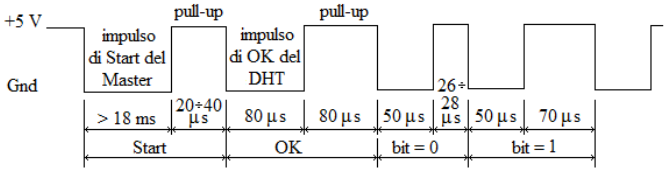

Fig. 3. *Protocollo di comunicazione.*

Se la durata alta è di soli 26÷28 µs (breve) il singolo bit vale 0 e vale 1 se è di 70 µs (lunga).
I 40 bit comprendono:

- 8 bit integral RH data + 8 bit decimal RH data
- 8 bit integral T data + 8 bit decimal T data
- 8 bit check-sum

Il primo bit trasmesso è sempre il più significativo.
Se la trasmissione è corretta, il byte di check-sum corrisponde agli 8 bit meno significativi della somma di tutti i byte precedenti.

Esperienza pratica
Acquisizione con Arduino

Collegare il pin di segnale (S) del dispositivo DHT11 all'ingresso A0 di Arduino, come indicato in **fig. 2**, e collaudare il programma che segue.

```
#define DHT_pin A0
byte Error, DHT_dat[5];

void setup(){
  pinMode(DHT_pin,OUTPUT);
  digitalWrite(DHT_pin,HIGH);
  Serial.begin(9600);
}

void loop(){
  ReadDHT();
  switch (Error){
   case 0:
    Serial.print("Umidita' = ");
    Serial.print(DHT_dat[0], DEC);
    Serial.print(".");
    Serial.print(DHT_dat[1], DEC);
```

```
    Serial.print("% ");
    Serial.print("temp. = ");
    Serial.print(DHT_dat[2], DEC);
    Serial.print(".");
    Serial.print(DHT_dat[3], DEC);
    Serial.println(" C");
    break;
   case 1:
    Serial.println("Error 1: DHT non risponde
    allo Start");
    break;
   case 2:
    Serial.println("Error 2: OK non termina-
    to");
    break;
   case 3:
    Serial.println("Error 3: errore di
    checksum");
    break;
   default:
    Serial.println("Error: protocollo non
    corretto");
   break;
 }
 delay(1000);
}

void ReadDHT(){
 Error=0;
 byte dht_in, i;
 digitalWrite(DHT_pin,LOW);
 delay(20); //Start basso > 18 ms
 digitalWrite(DHT_pin,HIGH);
 pinMode(DHT_pin,INPUT);
 delayMicroseconds(40); //wait pull-up
 dht_in=digitalRead(DHT_pin); // OK =0?
 if(dht_in){Error=1;return;}
 delayMicroseconds(80);
 dht_in=digitalRead(DHT_pin); // fine OK?
 if(!dht_in){Error=2;return;}
 delayMicroseconds(80);
 for (i=0; i<5; i++) // per 5 volte
  DHT_dat[i] = read_DHT();//legge 8 bit
 pinMode(DHT_pin,OUTPUT);
 digitalWrite(DHT_pin,HIGH);
 byte check_sum =
 DHT_dat[0]+DHT_dat[1]+DHT_dat[2]+DHT_
 dat[3];
 if(DHT_dat[4]!= check_sum) Error=3;
};
 byte read_DHT(){ //legge 8 bit di dato
 byte i = 0;
 byte result=0;
 for(i=0; i< 8; i++){
  while(digitalRead(DHT_pin)==LOW);
  delayMicroseconds(30); // se >30, bit = 1
  if (digitalRead(DHT_pin)==HIGH)
   result |=(1<<(7-i));
  while (digitalRead(DHT_pin)==HIGH);
 }
 return result;
}
```

Il *main* chiama ogni 1 s la funzione *readDHT()* e gestisce la comunicazione con la finestra di monitor seriale.
La funzione *readDHT()* abbassa la linea per più di 18 ms, per generare l'impulso di Start, e attende il segnale basso di OK da parte del dispositivo. Successivamente acquisisce i 40 bit, identificandoli in base alla durata del livello alto, e verifica la coerenza del byte di check-sum.

Meccanismo di time-out

Modificare i loop di attesa, del tipo

```
while(digitalRead(DHT_pin)==LOW)
```

presenti nella funzione *readDHT()* inserendo un meccanismo di time-out di uscita.