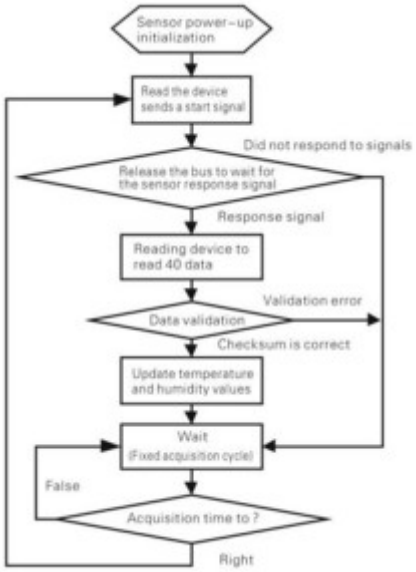



*figura 10*

*In sintesi:*

I bit trasmessi dal sensore DHT22 in totale sono 40 ed hanno la seguente struttura:

Dati(40-bit)= 1° byte parte intera RH%+2° byte parte decimale di RH%+3° byte parte intera T°+ 4° byte parte decimale T°+5° byte di CRC.

L'ultimo byte, il byte di CRC viene inviato per controllare che la trasmissione dei dati sia avvenuta in modo corretto cioè senza errori. Se tutti i quattro byte delle grandezze in misura sono stati trasferiti con successo, in questo caso il byte di CRC deve essere uguale alla somma dei quattro byte ricevuti, cioè:

CRC(8bit)= (1° byte parte intera RH%+2° byte parte decimale di RH%+3° byte parte intera T°+ 4° byte parte decimale T°).

Lo schema di principio dell'interfacciamento del sensore con il microcontrollore PIC18F26K80 è riportato in figura 10.Lo schema è perfettamente conforme con la verifica sperimentale realizzata utilizzando la EASYPIC7 della MikroElektronica. Inoltre il progetto è stato simulato mediante il programma professionale Proteus 8.1 della Labcenter Electronics; ed i risultati ottenuti sono stati in perfetto accordo con quelli sperimentali.

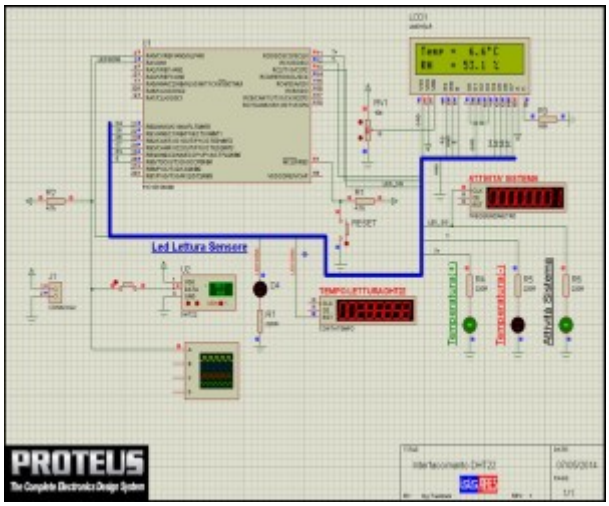


*figura 11*

Vediamo ora le funzioni più importanti scritte in MikroC(rel 6.01) che permettono la gestione del sensore che rispettivamente sono:

1. void StartSensore(void) che permette di fornire il segnale di start al sensore e quindi di iniziare la comunicazione.
2. unsigned char ControlloRisposta(void) questa funzione permette di leggere la risposta del sensore e quindi di stabilire se la comunicazione è presente o meno.
3. Unsigned char LeggiByte(void) consente di leggere i cinque byte, cioè la parte intera e frazionaria della umidità relativa percentuale e la parte intera e frazionaria della tempertaura compreso il byte di CRC

Cominciamo l'analisi della funzione StartSensore(void) il  cui corpo è riportato in figura 12.

```
/*****************************************************
 * Nome Funzione:                                    *
 *   <f_nome>                                        *
 * Descrizione:                                      *
 *   <descrizione>                                   *
 * Parametri Input:                                  *
 *   <lista_parametri>                               *
 * Parametri Restituiti:                             *
 *   <lista_parametri>                               *
 * NOTE:                                             *
 *   <note>                                          *
 *****************************************************/
void StartSensore(void)
{
  DataDir = 0;     // linea dati è in output
  Data    = 0;     // linea dati "0"
  Delay_ms(20);    // master start signal(valore Massimo)
  Data    = 1;     // lana dati "1"
  Delay_us(30);    // rilascio del bus (valore tipico)
  DataDir = 1;     // linea dati e in input risposta sensore
}
```

*figura 12*

Come si nota la linea dati(RA0) viene messa in uscita e successivamente posta a livello logico "0" per un tempo pari a 20ms,dopo tale tempo la linea dati viene posta al valore logico "1" per 30us per poi essere impostata come ingresso aspettando la risposta del sensore, in relazione alle specifiche del costruttore (vedi tabella 1).

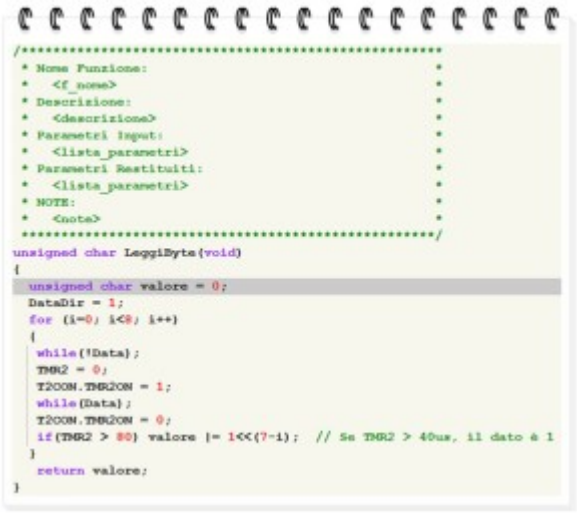
La funzione che controlla la presenza della risposta del sensore è forse più complessa in quanto deve valutare i tempi che costituiscono il segnale di risposta, per questo si avvale del timer2.

Il timer2 viene programmato per avere un timeout pari a 255us che ci permetterà di stabilire se il dispositivo sta  rispondendo o meno. Il corpo di tale funzione è riportato in figura 13.

```
/*****************************************************
 * Nome Funzione:                                    *
 *   <f_nome>                                        *
 * Descrizione:                                      *
 *   <descrizione>                                   *
 * Parametri Input:                                  *
 *   <lista_parametri>                               *
 * Parametri Restituiti:                             *
 *   <lista_parametri>                               *
 * NOTE:                                             *
 *   <note>                                          *
 *****************************************************/
unsigned char ControlloRisposta(void)
{
  Time_Out = 0;
  TMR2 = 0;
  T2CON.TMR2ON = 1;          // start timer2
  while(!Data && !Time_Out);
  if (Time_Out) return 0;
  else
  {
    TMR2 = 0;
    while(Data && !Time_Out);
    if (Time_Out) return 0;
    else
    {
      T2CON.TMR2ON = 0;
      return 1;
    }
  }
}
```

*figura 13*

La variabile di Time_Out tiene conto del tempo di risposta del dispositivo (l'interrupt del timer2 la setta se sono trascorsi 255us) infatti se la risposta del sensore arriva, successivamente al segnale di start, e dura a livello logico basso 80us,passa ad analizzare la durata della parte alta di tale segnale, che deve durare anch'essa 80us. Sotto queste condizioni restituisce il valore logico "1" indicando che il dispositivo ha risposto al segnale di inizializzazione. In caso contrario restituisce il valore logico "0" indicando la non risposta del sensore. Infine la funzione che permette la lettura delle grandezze temperatura e umidità è riportata in figura 14.



```
/*******************************************************
 * Nome Funzione:                                      *
 *   <f_nome>                                          *
 * Descrizione:                                        *
 *   <descrizione>                                     *
 * Parametri Input:                                    *
 *   <lista_parametri>                                 *
 * Parametri Restituiti:                               *
 *   <lista_parametri>                                 *
 * NOTE:                                               *
 *   <note>                                            *
 *******************************************************/
unsigned char LeggiByte(void)
{
  unsigned char valore = 0;
  DataDir = 1;
  for (i=0; i<8; i++)
  {
   while(!Data);
   TMR2 = 0;
   T2CON.TMR2ON = 1;
   while(Data);
   T2CON.TMR2ON = 0;
   if(TMR2 > 80) valore |= 1<<(7-i);  // Se TMR2 > 40us, il dato è 1
  }
   return valore;
}
```

*figura 14*

La lettura sia della temperatura che dell'umidità avviene bit per bit(ciclo for) e grazie al timer2 possiamo stabilire quando il bit inviato è di valore logico "0"(26-28us) o "1"(70us). I bit vengono inseriti nella variabile valore che verrà  restituita dalla funzione a lettura ultimata. Il firmware completo potrà essere scaricato dal sito dell'editore.

**Osservazioni finali:** Il firmware proposto per il progetto, legge il DHT22 ogni due secondi, mediante una temporizzazione (routine di ritardo) ottenuta fruttando il timer0. Riporto per comodità del lettore le configurazioni del timer0 e del timer2.Le figure 15 e 16 esplicitano tali configurazioni ,ricordo di porre molta attenzione alla configurazione dei timer, ed in modo particolare alla configurazione del timer2, per una corretta lettura del dispositivo. Le temporizzazioni dei timer sono state realizzate utilizzando l'oscillatore interno del PIC18F26K80 con una frequenza di lavoro pari ad 8Mhz,ovviamente senza far uso del PLL, inoltre ho usato una resistenza di pull-up di 4.7K sulla linea dati.

Configurazione del Timer0.

```
//---------------Impostazione dei Timer0--------------//
// Timer0 Registers:// 8-Bit Mode;
// Prescaler=1:256;Freq=30.51758Hz; Period=32.768 ms
 T0CON.TMR0ON = 1; // Timer0 Control bit:1=abilito Timer0
 T0CON.T08BIT = 1; // Timer0 8-bit
 T0CON.T0CS   = 0; // TMR0 Clock Interno
 T0CON.T0SE   = 0; // TMR0 low/high
 T0CON.PSA    = 0; // Prescaler assegnato a TMR0
 T0CON.T0PS2  = 1; // Fattore di divisione del Prescaler
 T0CON.T0PS1  = 1;
 T0CON.T0PS0  = 1;
 TMR0H = 0x0;      // Preset Timer0 MSB
 TMR0L = 0x0;      // Preset Timer0 LSB
 INTCON.TMR0IE=1;  // abilita interruzione timer0
 INTCON2.TMR0IP=0; // Interrupt Timer0 bassa priorità
```

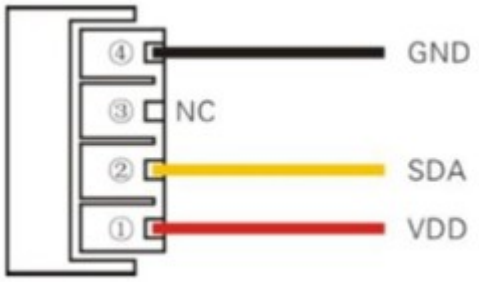
*figura 15*

Configurazione  del Timer2.

```
//---------------Impostazione dei Timer2--------------//
// Prescaler=1:1; TMR2 PostScaler=1:2; PR2=255 -
// Freq = 3.92157kHz - Period = 255.00 µs
  T2CON.T2OUTPS3 = 0;   //Postscaler 1:2
  T2CON.T2OUTPS2 = 0;
  T2CON.T2OUTPS1 = 0;
  T2CON.T2OUTPS0 = 1;
  T2CON.TMR2ON   = 0;   // Timer2 è disattivo;
  T2CON.T2CKPS1  = 0;   // Divisione prescaler 1:1
  T2CON.T2CKPS0  = 0;
  IPR1.TMR2IP=1;        // Abilito interrupt alta priorità
  PIE1.TMR2IE=1;        // Abilito interrupt per in timer
  TMR2=0;               // Resetto il Timer
  PR2=255;              // Configuro registro periodo
```

*figura 16*

In ultimo riporto nelle figure sottostanti, il pinout del dispositivo ed il suo corretto collegamento alla alimentazione.




| **Elenco componenti** | |
|---|---|
| Sensore | DHT22 della AOSONG |
| Microcontrollore | PIC18F26K80 della microchip |
| Resistenza | 4.7K pull-up(1/4w) |

# SUGGERITI DA


## 10 progetti con i PIC

10 progetti con i microntrollori PIC utilizzando mikroBASIC.
I progetti sono completi di codice sorgente.
File PDF di 86 pagine + file zip.

ACQUISTALO ORA!

Secure Payments By PayPal


## Bundle eBook sulla programmazione BASIC per PIC.

L'ebook "Basic per PIC" e l'ebook "10 progetti con PIC usando mikroBASIC" ad un prezzo speciale.

OFFERTA LIMITATA!

ACQUISTALO ORA!

Secure Payments By PayPal


## Pillole di microcontrollori PIC

Per imparare a programmare i PIC utilizzando il linguaggio C.

ACQUISTALO ORA!

Secure Payments By PayPal


## ARDUINO Projects

190 pagine di progetti con Arduino in pdf con spiegazioni dettagliate e codici sorgente.

ACQUISTALO ORA!

Secure Payments By PayPal