WeatherShield 1

Introduzione

Lo WeatherShield1 e' uno shield per Arduino o Netduino che permette di implementare una piccola stazione metereologica in grado di rilevare temperatura, pressione ed umidita' relativa, in maniera semplice e veloce grazie al microcontrollore di cui e' dotato. Esso, infatti, si prende in carico di eseguire tutte le operazioni necessarie alla misurazione, alla media ed al condizionamento dei risultati, nonche' alla conversione dei risultati in valori compatibili con le unita' di misurazione normalmente utilizzate.

Tutti i risultati possono essere trasferiti dal microcontrollore, presente sullo WeatherShield1, all'unita' che lo ospita (Arduino UNO o Netduino), attraverso una connessione seriale sincrona bidirezionale. L'interfacciamento avviene sfruttando le linee digitali D2 (piedino dati, bidirezionale) e D7 (piedino clock, unidirezionale uscente da Arduino). E' possibile, tuttavia, modificare l'assegnazione dei piedini di input/output per poterlo adattare a differenti esigenze.

Ogni WeatherShield1e' dotata di un indirizzo, riprogrammabile via software, che le permette di essere identificata nel caso in cui venissero montate piu' WeatherShield1 su uno stesso Arduino/Netduino.

Utilizzo

L'utilizzo dello WeatherShield1 e' molto semplice ed e' ancora piu' semplificato dalla presenza di librerie ed esempi liberamente scaricabili dal sito di EtherMania.

Una volta entrati in possesso delle librerie, per dialogare con lo WeatherShield1 bastera' richiamare le funzioni che mettono a disposizione. Qui di seguito si fornisce un esempio di utilizzo ed una guida esaustiva delle API fornite dalle librerie.

Il primo passo da eseguire e' quello di includere le librerie all'interno del proprio sketch:

```
#include <WeatherShield1.h>
```

dopodiche' sara' necessario instanziare un oggetto globale di tipo WeatherShield.

```
WeatherShield1 weatherShield;
```

Nel caso in cui si voglia far uso di piedini specifici di input/output, invece che quelli di default, o di indirizzare le librerie verso una WeatherShield1 con un differente indirizzo, al posto della precedente riga si potra' utilizzare il codice seguente:

```
#define IODATA_PIN    2
#define CLOCK_PIN     7
#define MY_WEATHERSHIELD_ADDRESS   10
WeatherShield1 weatherShield(CLOCK_PIN, IODATA_PIN, MY_WEATHERSHIELD_ADDRESS);
```

modificando i valori di IODATA_PIN, CLOCK_PIN o MY_WEATHERSHIELD_ADDRESS in modo che corrispondano alle proprie esigenze.

A questo punto tutto e' pronto per poter impartire comandi allo WeatherShield1 e/o leggere i valori di pressione, temperatura ed umidita'.

Come descritto in precedenza, tutte le comunicazioni da e verso lo WeatherShield1 avvengono attraverso un protocollo seriale. La comunicazione viene implementata dalle librerie le quali si occupano di decodificare i valori ricevuti dallo WeatherShield1 in valori utilizzabili all'interno del programma su Arduino. Per far questo le librerie necessitano di una piccola area di memoria di 4 byte, o buffer, nel quale memorizzare i risultati temporanei. Negli esempi riportati, quest'area di memoria viene definita all'interno della funzione “loop()” come qui sotto riportato:

```
void loop() {
   unsigned char ucBuffer[4];
...
```

Tale area di memoria dovra' essere sempre passata come parametro alle chiamate della libreria. L'idea di base e'

1. la richiesta di un comando fa si che il buffer venga popolato con la risposta proveniente dallo WeatherShield1
2. l'area di memoria viene passata ad un comando di decodifica che, leggendo il valore presente nel buffer, rilascia il risultato nel formato voluto (intero, float etc.).

La sequenza qui sopra riportata deve essere ripetuta per ogni comando, con eccezione di alcune richieste nelle quali non si attende un valore di ritorno.

Invio di un comando

L'invio di un comando allo WeatherShield1 viene gestito attraverso la chiamata alla funzione SendCommand. Tale funzione si aspetta tre parametri in ingresso, rispettivamente:

1. L'identificativo del comando
2. Un parametro il cui significato dipende dal tipo di comando richiesto
3. Il puntatore al buffer temporaneo nel quale inserire il risultato.

I possibili comandi sono cosi' definiti:

- CMD_ECHO_PAR: lo WeatherShield1 risponde ripetendo lo stesso carattere specificato come parametro durante la chiamata della funzione.
- CMD_SET_SAMPLETIME: imposta il tempo di campionamento utilizzato dallo WeatherShield1, ovvero il tempo che intercorre tra l'acquisizione di un valore di temperatura, pressione, umidita' relativa, ed il successivo. Il parametro specificato durante la chiamata equivale al numero di secondi a cui il tempo di campionamento verra' impostato (0->1s, 1->2s ... 255->256s). Il valore viene memorizzato da WeatherShield1 in un'area EEPROM e, quindi, rimarra' memorizzato anche in caso di assenza di tensione fino a che non verra' successivamente specificato. N.B. Il valore di default impostato in fabbrica e' pari a 30 secondi.
- CMD_GETTEMP_C_AVG: legge il valore di temperatura calcolato come media delle ultime 8 misurazioni e correzione secondo quanto specificato da datasheet del sensore. Il valore ritornato dallo WeatherShield1 e' misurato in gradi C e viene fornito con precisione di 2 decimali dopo la virgola. Ogni eventuale parametro specificato al momento della chiamata della funzione viene ignorato.
- CMD_GETTEMP_C_RAW: legge un valore a 10bit proveniente dal sensore di temperatura. Il parametro specificato definisce quale valore si vuole leggere. Impostando il parametro al valore PAR_GET_LAST_SAMPLE verra' ritornato l'ultimo campionamento effettuato; impostando il parametro al valore PAR_GET_AVG_SAMPLE verra' ritornato il valore a 10bit corrispondenti alla media delle ultime 8 misurazioni (senza effettuare alcuna correzione); impostando il parametro ad un valore compreso tra 0 e 7 sara' possibile accedere ad uno degli otto campionamenti precedenti.
- CMD_GETPRESS_AVG: legge il valore di pressione calcolato come media delle ultime 8 misurazioni e correzione secondo quanto specificato da datasheet del sensore. Il valore ritornato dallo WeatherShield1 e' misurato in hPa e viene fornito con precisione di 2 decimali dopo la virgola. Ogni eventuale parametro specificato al momento della chiamata della funzione viene ignorato.
- CMD_GETPRESS_RAW: legge un valore a 10bit proveniente dal sensore di pressione. Il parametro specificato definisce quale valore si vuole leggere. Impostando il parametro al valore PAR_GET_LAST_SAMPLE verra' ritornato l'ultimo campionamento effettuato; impostando il parametro al valore PAR_GET_AVG_SAMPLE verra' ritornato il valore a 10bit corrispondenti alla media delle ultime 8 misurazioni (senza effettuare alcuna correzione); impostando il parametro ad un valore compreso tra 0 e 7 sara' possibile accedere ad uno degli otto campionamenti precedenti.
- CMD_GETHUM_AVG: legge il valore di umidita' calcolato come media delle ultime 8 misurazioni e correzione secondo quanto specificato da datasheet del sensore. Il valore ritornato dallo WeatherShield1 e' misurato in % e viene fornito con precisione di 2 decimali dopo la virgola. Ogni eventuale parametro specificato al momento della chiamata della funzione viene ignorato.
- CMD_GETHUM_RAW: legge un valore a 10bit proveniente dal sensore di umidita'. Il parametro specificato definisce quale valore si vuole leggere. Impostando il parametro al valore PAR_GET_LAST_SAMPLE verra' ritornato l'ultimo campionamento effettuato; impostando il parametro al valore PAR_GET_AVG_SAMPLE verra' ritornato il valore a 10bit corrispondenti alla media delle ultime 8 misurazioni (senza effettuare alcuna correzione); impostando il parametro ad un valore compreso tra 0 e 7 sara' possibile accedere ad uno degli otto campionamenti precedenti.
- CMD_SETADDRESS: cambia l'indirizzo a cui lo WeatherShield1 e' associato. Il parametro specificato verra' passato allo WeatherShield1 che lo memorizzera' in un'area EEPROM in cui vi rimarra' anche in assenza di alimentazione. Tale funzione e' utile nel caso in cui si voglia utilizzare piu' di uno WeatherShield1 all'interno dello stesso progetto. N.B. L'indirizzo di default dello WeatherShield1 e' pari al valore 1.

La funzione ritorna un valore “true” se la comunicazione con lo WeatherShield1 e' stata effettuata correttamente; “false” in caso contrario.

Decodifica del risultato di un comando

Una volta inviata una richiesta allo WeatherShield1 e' possibile leggere il valore da esso trasferito verso Arduino tramite una serie di comandi che interpretano il risultato convertendolo nel formato voluto.

- `float decodeFloatValue(unsigned char *pucBuffer)`: viene utilizzata prevalentemente dopo un comando di richiesta di valori di tipo AVG (CMD_GETTEMP_C_AVG, CMD_GETPRESS_AVG, CMD_GETHUM_AVG). Essa ritorna il valore in formato float. Si aspetta come parametro in ingresso il puntatore all'area di memoria utilizzata come buffer.
- `void decodeFloatAsString(unsigned char *pucBuffer, char *chString)`: viene utilizzata prevalentemente dopo un comando di richiesta di valori di tipo AVG (CMD_GETTEMP_C_AVG, CMD_GETPRESS_AVG, CMD_GETHUM_AVG). Si aspetta, come parametri, il puntatore all'area di memoria utilizzata come buffer, nonche' il puntatore all'area di memoria dove verra' memorizzato il risultato.
- `unsigned short decodeShortValue(unsigned char *pucBuffer)`: viene utilizzata prevalentemente dopo un comando di richiesta di valori di tipo RAW (CMD_GETTEMP_C_RAW, CMD_GETPRESS_RAW, CMD_GETHUM_RAW). Essa ritorna il valore in formato intero (16 bit senza segno). Si aspetta come parametro in ingresso il puntatore all'area di memoria utilizzata come buffer.

Ulteriori funzioni aggiuntive

E' disponibile la funzione resetConnection() che effettua, se chiamata, un reset del sistema di comunicazione seriale. Questo puo' essere utile nel caso in cui, per qualsiasi motivo, non si riuscisse piu' a dialogare con lo WeatherShield1 e la funzione SendCommand ritornasse un valore “false”.

La funzione resetConnection() effettua solo il reset del sottosistema di comunicazione e, anche se chiamata periodicamente, non cancella la cronologia dei campionamenti, ne' i valori mediati calcolati.

Sketch di esempio

Come esempio viene proposto lo sketch necessario ad implementare un termometro, visualizzando il risultato letto dallo WeatherShield1 tramite Serial Monitor.

```
#include <WeatherShield1.h>
#define RXBUFFERLENGTH          4
WeatherShield1 weatherShield;

/* This is the main sketch setup handler */
void setup(){
  /* Initialize the serial port */
  Serial.begin(9600);
}

/* This is the main sketch loop handler */
void loop() {
    /* This is the buffer for the answers */
  unsigned char ucBuffer[RXBUFFERLENGTH];

  /* Check for the weather shield connection */
  if (weatherShield.sendCommand(CMD_ECHO_PAR, 100, ucBuffer)) {
    Serial.println("Connection With Shield Performed OK");
  }

  /* Start reading temperature */
  float fTemperature = 0.0f;
  unsigned short shTemperature = 0;
  if (weatherShield.sendCommand(CMD_GETTEMP_C_AVG, 0, ucBuffer))
    fTemperature = weatherShield.decodeFloatValue(ucBuffer);
  if (weatherShield.sendCommand(CMD_GETTEMP_C_RAW, PAR_GET_LAST_SAMPLE, ucBuffer))
    shTemperature = weatherShield.decodeShortValue(ucBuffer);

  /* Send all data through the serial line */
  Serial.print("Temperature ");
  Serial.print(fTemperature);
  Serial.print(" Celsius (");
  Serial.print(shTemperature);
  Serial.println(")");

  /* Wait some time before running again */
  delay(1000);
}
```