ETHERNET SHIELD CON ENC28J60 - IN KIT

Prezzo: 19.67 €

Tasse: 4.33 €

Prezzo totale (con tasse): 24.00 €

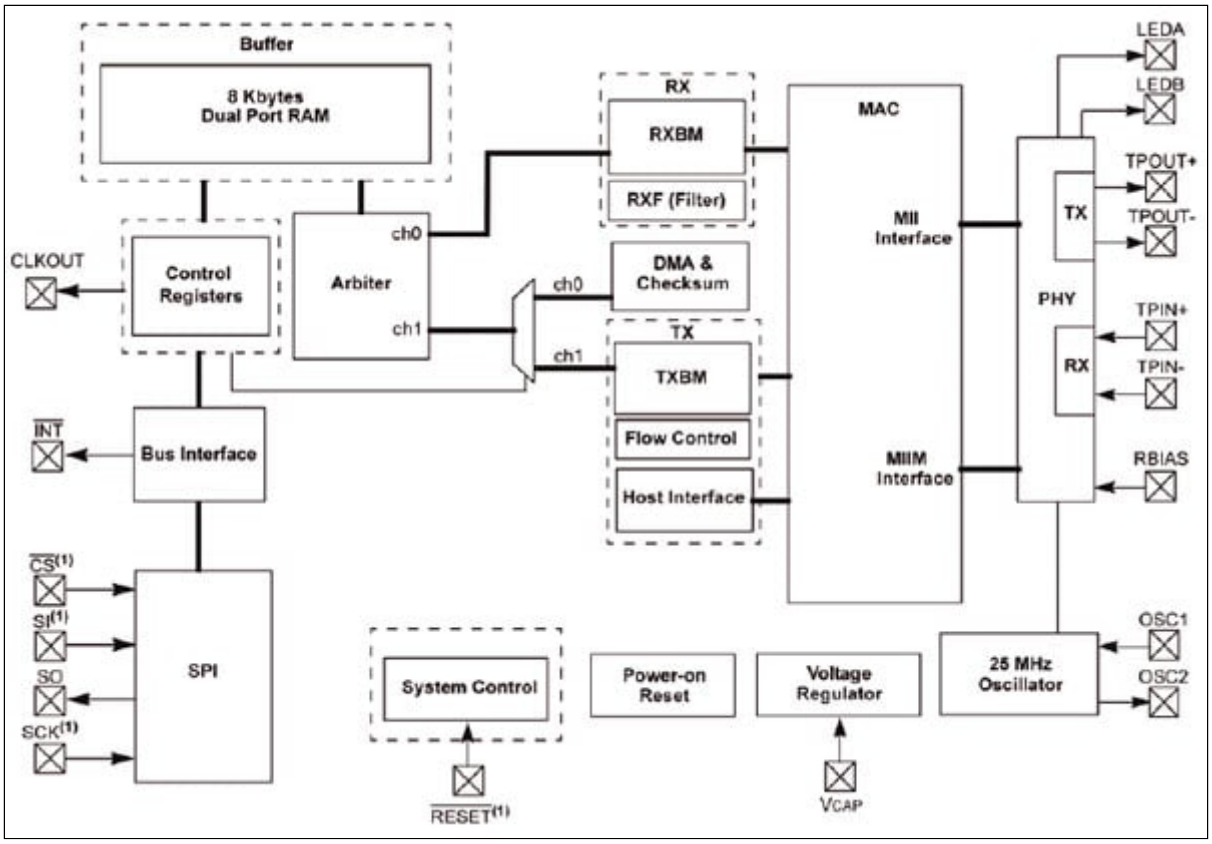
La shield si basa sull'ENC28J60, un microcontrollore della Microchip che svolge insieme le funzioni di interfacciamento con Arduino e di conversione dei dati secondo il protocollo ethernet. Il microcontrollore contiene una completa interfaccia ethernet di tipo 10 baseT, conforme allo standard IEEE 802.3 ed interfacciabile tramite un bus SPI (che implementa) configurabile per un clock massimo di 20 MHz. Utilizza l'ICSP, invece delle linee di I/O laterali, per rendere la shield compatibile a livello hardware con Arduino Duemilanove, UNO e Mega. Alla gestione del micro ethernet provvedono anche le linee digitali D10 e D2 di Arduino, destinate rispettivamente al controllo del CS (Chip Select, attivo a zero logico) e alla lettura dell' INT. La funzionalità ethernet consente di realizzare numerose applicazioni di networking quali il controllo da remoto di sistemi ed utilizzatori, la gestione domotica di abitazioni ed altri luoghi, l'accesso al web per il download o la pubblicazione di dati, e tanto altro ancora; la semplicità di trovare sul web ed integrare le librerie ethernet open-source nell'IDE di Arduino, fa il resto.
L'alimentazione viene prelevata da Arduino mediante i contatti 5V e Vin: il primo fornisce i 5 volt continui stabilizzati ai punti del circuito che li richiedono per funzionare (sostanzialmente al 74HC125 ed alle resistenze di pull-up delle linee di reset e di Chip Select) mentre il secondo alimenta il regolatore integrato U2, il quale provvede a ricavare i 3,3 volt stabilizzati che servono ad alimentare il microcontrollore ed i circuiti di accoppiamento contenuti nella presa RJ45.

IL MICROCONTROLLORE ENC28J60

L’ENC28J60 contiene una completa interfaccia ethernet di tipo 10 baseT, conforme allo standard IEEE 802.3 ed interfacciabile tramite un bus SPI configurabile per un clock massimo di 20 MHz. Il componente integra il controller MAC, dispone di un buffer di 8 kB Transmit/Receive Packet Dual Port Buffer e di un FIFO circolare gestito a livello hardware; permette inoltre la programmazione della ritrasmissione dei dati in caso di collisione.
Per quanto riguarda il Transceiver fisico (PHY) sono previsti un filtro d’uscita sagomatore dei segnali e l’implementazione della modalità Loop back.
Il controllore MAC supporta le modalità Unicast, Multicast e Broadcast packets, ha un pattern programmabile a 64 byte entro un margine consentito all’utente e prevede il programmable wake-up per diversi formati di pacchetti dati (Magic Packet, Unicast, Multicast, Broadcast).
Nel circuito della shield vediamo che il microcontrollore fa tutto da solo e gli servono solo una presa RJ45 standard con integrati i LED, i filtri e i trasformatori di linea ethernet, ed un adattatore di livello per l’interfaccia SPI di cui dispone e che gli occorre, nel nostro caso, per comunicare con Arduino; più esattamente, la comunicazione e la gestione dell’ENC28J60 avvengono mediante le linee dell’SPI-Bus che, nello schema elettrico, fanno capo ai contatti raggruppati sotto l’ICSP.
Quest’ultimo è il connettore che in Arduino UNO e Duemilanove, ma anche in Arduino Mega, viene utilizzato per l’In Circuit Serial Programming (ICSP) ovvero la programmazione in-circuit del microcontrollore ATmega (tipicamente permette di caricare il bootloader nei micro vergini); nel nostro caso, lo sketch li utilizza per gestire l’interfaccia ethernet realizzata dal micro della Microchip.
Abbiamo scelto di utilizzare l’ICSP, invece delle linee di I/O laterali, per rendere lo shield compatibile a livello hardware con il maggior numero possibile di moduli Arduino, ovvero con Arduino Duemilanove, UNO e Mega; se avessimo usato i connettori laterali, avremmo reso lo shield utilizzabile solo su Arduino Duemilanove ed UNO, perché nell’Arduino Mega i contatti sono disposti diversamente, dato che quest’ultimo ha e rende accessibili dall’esterno molti più I/O degli altri. Invece usando l’ICSP, che è disposto ugualmente, possiamo ottenere la compatibilità con tutti e tre i modelli.
Del connettore per l’ICSP, MISO è l’uscita dei dati del dispositivo slave, ossia dell’ENC28J60, e l’input di Arduino, mentre MOSI è il contrario, ossia porta all’ENC28J60 i dati generati da Arduino; SCK è il clock che scandisce la comunicazione bidirezionale sul bus SPI e RESET la linea di reset, cui è collegato anche un pulsante che permette di resettare l’interfaccia ethernet, all’occorrenza, in modo manuale.
Alla gestione del micro ethernet provvedono anche le linee digitali D10 e D2 di Arduino, destinate rispettivamente al controllo del /CS (Chip Select, attivo a zero logico) e alla lettura dell’/ INT, il cui stato viene letto attraverso U3a (74HC125). Quest’ultimo, come U3b (74HC125), serve per adattare i livelli logici cui lavora U1 (che sono a 0/3,3 V) a quelli di Arduino (0/5 V).

SCHEMA A BLOCCHI DEL MICROCHIP ENC28J60

il componente che realizza l’interfacciamento sull’ethernet per la nostra shield è composto sostanzialmente da un controllore MAC interfacciato tramite bus SPI.



LA RETE VIAGGIA...SULLA RETE

Per disporre della connessione ad Internet in vari punti della casa ci sono due soluzioni: realizzare una LAN cablata o una rete wireless; la prima impone tutta una serie di cavi in giro o la realizzazione delle tracce nel muro, mentre la seconda, oltre a porre problemi di sicurezza, nelle case a più piani potrebbe non fornire la necessaria copertura.
Da qualche tempo si profila all'orizzonte una terza possibilità, ossia la LAN su PowerLine: in pratica un sistema ad onde convogliate che permette di realizzare una LAN sfruttando i cavi della rete elettrica a 220 volt di casa, naturalmente senza prendere la scossa!
Questa tecnologia si basa su apparecchi "homeplug" che sono dei transceiver ethernet con trasformatore di accoppiamento alla linea elettrica (si trovano in vendita in coppie a poche decine di euro): basta inserire uno dei dispositivi nella presa di corrente più vicina al router con cui si è collegati ad Internet e l'altro in una qualsiasi presa del locale dove vogliamo collegarci con il computer.
Ogni adattatore dispone di una presa RJ45 cui collegare il cavo LAN e può essere collegato a svariati dispositivi, anche solo per realizzare una rete locale generica con cui far dialogare un PC e una stampante o altri dispositivi.
La LAN su PowerLine può anche essere utilizzata in combinazione con altre reti, ovviamente anche con le wireless, in quanto gli adattatori funzionano sostanzialmente da ponti tra reti. La tecnica oggi è in grado di offrire adattatori capaci di superare i 100 Mbps.

SKETCH ETHERNET...

Per poter utilizzare la connettività ethernet caricate in Arduino lo sketch che vi serve, ricordando che dal nostro sito ***WWW.ELETTRONICAIN.IT*** potete scaricare la libreria specifica per la gestione dell'ENC28J60 Microchip e quindi del nostro shield. La libreria originale da cui abbiamo ricavato quella che proponiamo nel nostro sito Internet può essere scaricata dal sito ***https://github.com/jcw/ethercard***; dal nostro sito è possibile scaricare la stessa libreria, però con a contorno un numero superiore di esempi applicativi.

Nel Listato 1 trovate uno sketch di esempio che permette di realizzare un Web Server; in particolare, nella pagina web visualizzata dal nostro sistema si vedrà il tempo, nel formato ore, minuti e secondi, trascorso dall'accensione di Arduino.

Listato 1

```
// This is a demo of the RBBB running as webserver with the Ether Card
// 2010-05-28 <jc@wippler.nl> http://opensource.org/licenses/mit-license.php

#include <EtherCard.h>

// ethernet interface mac address
static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0x31 };
// ethernet interface ip address
static byte myip[] = { 192,168,0,188 };
// gateway ip address
static byte gwip[] = { 192,168,0,1 };

byte Ethernet::buffer[500];
BufferFiller bfill;

void setup () {
  if (ether.begin(sizeof Ethernet::buffer, mymac) == 0)
    Serial.println( "Failed to access Ethernet controller");
  ether.staticSetup(myip);
}

static word homePage() {
  long t = millis() / 1000;
  word h = t / 3600;
  byte m = (t / 60) % 60;
  byte s = t % 60;
  bfill = ether.tcpOffset();
  bfill.emit_p(PSTR(
    "HTTP/1.0 200 OK\r\n"
    "Content-Type: text/html\r\n"
    "Pragma: no-cache\r\n"
    "\r\n"
    "<meta http-equiv='refresh' content='1'/>"
    "<title>RBBB server</title>"
    "<h1>$D$D:$D$D:$D$D</h1>"),
      h/10, h%10, m/10, m%10, s/10, s%10);
  return bfill.position();
}

void loop () {
  word len = ether.packetReceive();
  word pos = ether.packetLoop(len);

  if (pos)  // check if valid tcp data is received
    ether.httpServerReply(homePage()); // send web page data
}
```