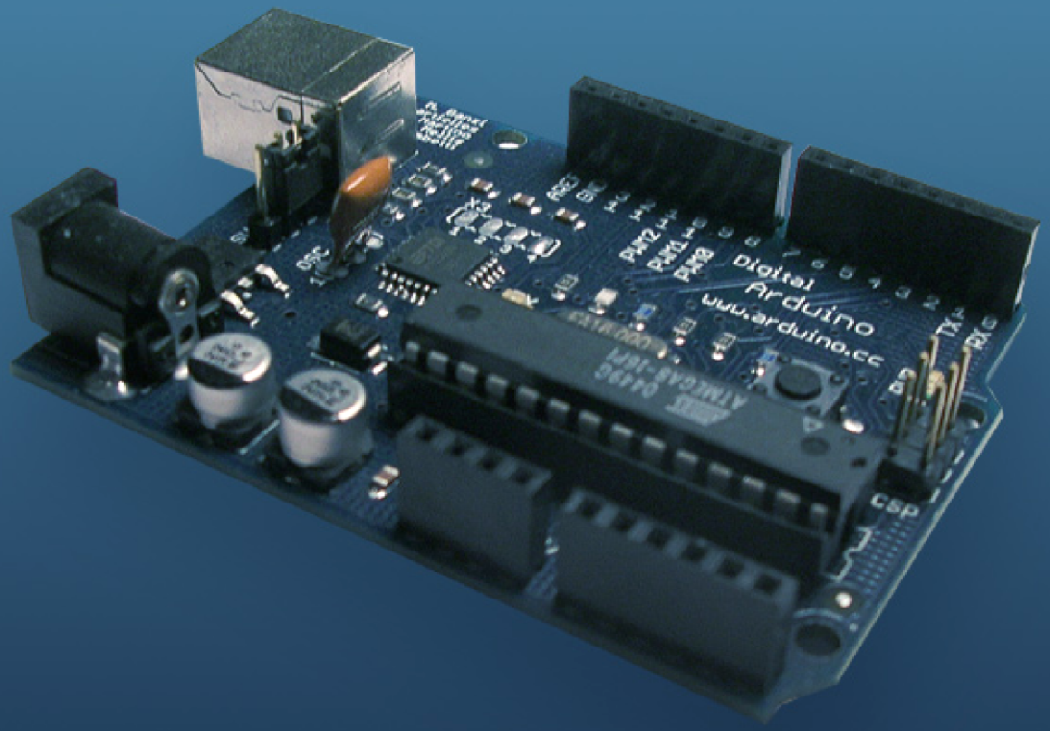
Arduino

Physical Computing I/O board



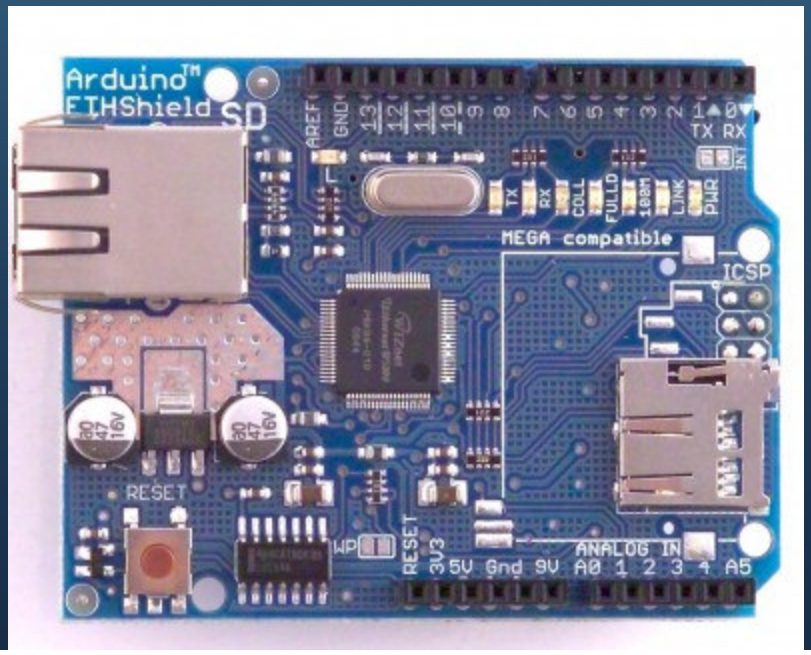
12^ parte : Come gestire la Ethernet Shield

Author: Ing. Sebastiano Giannitto (ITIS “M.BARTOLO” –PACHINO)

La Ethernet Shield

Una tra le più interessanti shield è la **Ethernet Shield**, una scheda che si innesta direttamente sulla principale e permette di collegare Arduino ad una porta Ethernet, in modo da poterlo controllare via rete locale o via Internet.

La Ethernet Shield è provvista di un connettore RJ45 per la connessione diretta al modem di casa ed inoltre offre la possibilità di utilizzare uno slot microSD per lo scambio diretto di dati con un supporto di memorizzazione dedicato.
La Ethernet Shield è dotata di un chip, il **W5100** che non svolge solamente il ruolo di controller Ethernet, bensì ci permette anche di utilizzare la libreria Ethernet che ad oggi rimane la più intuitiva tra le librerie utilizzabili con i moduli Ethernet.
Wiznet W5100, che è in grado di gestire funzionalità di rete IP con il protocollo TCP , è anche in grado di gestire fino 4 connessioni simultanee.



Per chi fosse interessato al datasheet del dispositivo, è possibile scaricarlo da questo link W5100 Datasheet v1.2.2.
Attraverso questa shield e la libreria Ethernet è possibile è possibile far diventare Arduino un piccolo server web (e lanciare programmi html ad un determinato indirizzo), chatroom, webclient e chi più ne ha più ne metta!

La Ethernet Shield

La scheda si poggia sull'interfaccia SPI sui pin 10,11,12 e 13, con il pin 10 che serve da Slave Select, ovvero da segnale per selezionare la comunicazione con il chip Wiznet.
Sulla scheda si trova anche una interfaccia Secure Digital che condivide i medesimi pi di comunicazione,ma ha sul pin 4 il segnale di Slave Select. Attraverso le opportune librerie si può quindi comunicare sia con il chip Ethernet, sia con la scheda di memoria, ma non contemporaneamente e dovrà essere lo sketch a evitare accessi sovrapposti .
Ovviamente la scheda dispone di un indirizzo IP e di un MAC.

La libreria Ethernet

Importando la libreria ecco cosa viene inserito nello sketch:
#include <Dhcp.h>
#include <Dns.h>
#include <Ethernet.h> ; è la libreria principale
#include <EthernetClient.h>
#include <EthernetServer.h>
#include <EthernetUdp.h>

DHCP riguarda l'acquisizione di un indirizzo IP da un DHCP server,ovvero un server connesso alla rete e preposto all'assegnazione di indirizzi IP ai client che ne fanno richiesta . L'indirizzo può essere in uno spazio di indirizzi privato, ad es. del 192.168.x.x, se assegnato da un Router o da un proxy/firewall, oppure un indirizzo pubblico.
Il DHCP oltre a comunicare l'indirizzo di rete assegnato al dispositivo, definisce anche il tempo di "lease", ovvero di affitto di quello specifico indirizzo al dispositivo.
Trascorso il termine definito, quell'indirizzo sarà nuovamente assegnabile a qualche altro dispositivo. Chiaramente per tutto il tempo di "lease", l'indirizzo non cambia.
DNS permette di interagire con i server in rete che trasformano gli host name in indirizzi IP

La libreria Ethernet

Ethernet è la libreria principale dalla quale dipendono tutte le altre.
Ethernet Client è la libreria grazie alla quale Arduino può collegarsi a server in rete facendo interrogazioni e leggendo dati.
Ethernet Server contiene le funzioni per realizzare server su una qualsiasi porta TCP/IP.
Ethernet UDP implementa le funzioni per la comunicazione con questo protocollo di basso livello.
Util ha il compito di gestire il formato e le conversioni da e per l'esadecimale.

L'indirizzo MAC

Quando si inizializza la libremac della scheda con la funzioneria viene anche inizializzato l'hardware e la scheda acquisisce l'indirizzo IP dal DHCP server se è stato specificato solo l'indirizzo della scheda con la funzione

Ethernet.begin(mac);

e l'indirizzo ottenuto al termine dell'inizializzazione si può leggere con la funzione

Ethernet.localIP();

Se si specifica oltre al MAC address anche un indirizzo IP, un gateway e una maschera di sottorete, la scheda non utilizza il meccanismo DHCP ma si presenta in rete con i valori definiti dalla funzione:

Ethernet.begin(mac,ip,gateway,subnet);

Il MAC (Media Accessa Control)
è un indirizzo di sei byte definito per tutti i dispositivi in fase di produzione che qualifica ciascun device in modo univoco.
Nella scheda Ethernet ReV.3 l'indirizzo MAC si trova su un adesivo applicato nella parte inferiore della scheda, mentre se la scheda non riporta nulla, si può assegnare un indirizzo del tipo 90-A2-DA-xx-xx-xx.
Si può recuperare l'indirizzo MAC di una vecchia scheda Ethernet non più in uso ed essere sicuri così di non avere un indirizzo utilizzato da un altro dispositivo in rete.
Dopo la fase di configurazione e inizializzazione il sistema sarà pronto ad eseguire le funzioni della libreria per far funzionare Arduino da Server, Client o comunicazione e bidirezionale **UDP** (User Datagram Protocol) .
Per definire un indirizzo IP in una variabile ip si usa la funzione
IPAddress var(byte, byte, byte, byte) ;
ad es. **IPAddress(192,168,1,1);**
Arduino è capace di gestire solo 4 connessioni e il chip Wiznet 5100 dispone di 16 KB di buffer di memoria.

Esempio di configurazione e inizializzazione della scheda Ethernet

```
#include <SPI.h>
#include <Ethernet.h>


byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
char server[] = "www.spazdiprova.altervista.org";


IPAddress ip(192,168,0,177);


EthernetClient client;
```

```
int temp;
int sensorpin= A0;
```

La prime variabili che vengono specificate sono quelle relative all'indirizzo MAC , e all'indirizzo IP del server, anche se in questo caso è stato scritto in maniera testuale perché a trasformarlo in un indirizzo fisico ci pensa il DNS e l'indirizzo IP della Shield.

A stabilire l'indirizzo IP della scheda ci pensa il DHCP del Router. Se l'indirizzo non viene assegnato in automatico, verrà utilizzato quello specificato da programma.

Arduino come web server

Quando un dispositivo deve poter rispondere a delle interrogazioni relative a dati raccolti o rilevati tramite sensori, assume il tipico ruolo da server.
Praticamente mentre rileva dei dati, resta in attesa di interrogazioni su una porta e risponde inviando dati. Il tutto inserito in pagine HTML più o meno complesse memorizzate in memorie SD o micro SD da inviare ai client connessi.
Questo è possibile tramite la classe "**EthernetServer()**" e le seguenti funzioni:
EthernetServer(porta) classe utilizzata per creare un server sulla porta specificata;
Es:
EthernetServer MioArduino = Server(80); //crea un server dinome MioArduino sulla porta 80
MioArduino.begin(); // il software inizia ad ascoltare sulla porta 80 per il collegamento di un client;

begin() //attiva il server in termini di monitoraggio della porta definita come attiva per collegamenti in entrata da parte di client
available() // la funzione restituisce un oggetto client con dei dati leggibili,altrimenti un valore false
write(dato) // scrive i dati forniti come argomento (byte o char) a tutti i client connessi al server
print(dati, base) // la funzione trasferisce ai client connessi dei dati (numeri o stringhe). Come presentare il numero convertito in caratteri dipende anche dal parametro "base" (BIN,DEC,HEX,OCT). La funzione restituisce il numero di byte inviati al client.
println(dati, base) // aggiunge un ritorno a capo (CR LF) al termine del trasferimento dei dati.

Arduino come client

La classe "**EthernetClient()**" contiene le funzioni necessarie a scrivere un client internet capace di interagire con server accessibili su una rete o su Internet.

EthernetClient() classe utilizzata per creare un'istanza definendone il nome
Es. ***Ethernet Client mioclient;***

Connected(); //questa funzione restituisce TRUE se l'istanza del client è connessa a un server o se ci sono ancora dati nel buffer e la connessione è stata chiusa.

Connect (ip,porta); // avvia la connessione a livello di indirizzo IP e di porta specificati, restituendo TRUE se il collegamento va a buon fine e FALSE se fallisce;

write(dato) // scrive i dati forniti come argomento (byte o char) sul collegamento aperto verso un server;

print(dati, base) // la funzione trasferisce dati ai server connessi dei dati (numeri o stringhe). Come presentare il numero convertito in caratteri dipende anche dal parametro "base" (BIN,DEC,HEX,OCT). La funzione restituisce il numero di byte inviati al server.

println(dati, base) // aggiunge un ritorno a capo (CR LF) al termine del trasferimento dei dati.

available() restituisce il numero di byte in attesa di essere letti e già ricevuti dal server col quale il client è connesso;

read() legge il byte successivo disponibile nel buffer di ricevimento. Ad ogni chiamata read() il buffer viene decrementato di un carattere. Se non ci sono più caratteri nel buffer, la funzione restituisce -1.

flush() rimuove dal buffer di ricezione tutti i caratteri non ancora letti

stop () disconnette dal server con cui ci si è connessi con la funzione connect().

I trasferimenti UDP

Questo protocollo permette di scambiare messaggi in modo rapido fra dispositivi collegati in rete senza dover realizzare una connessione di tipo client o server.
E'un protocollo più leggero e non ha particolari meccanismi di verifica e ordinamento dei pacchetti e non garantisce che tutte le informazioni ricetrasmesse vadano a buon fine.

EthernetUdp() // classe utilizzata per creare un'istanza definendone il nome
Es. ***EthernetUdp mioudp;***

begin(porta); // inizializzazione per la trasmissione Udp. L'indirizzo IP dipende dalla funzione ***Ethernet.begin(mac,ip)*** inserita nella parte di setup dello sketch.

read() // legge il byte successivo disponibile nel buffer di ricevimento assegnato all'istanza Udp. Ad ogni chiamata read() il buffer viene decrementato di un carattere. Poiché la funzione restituisca il byte , è necessario che sia chiamata successivamente alla funzione parserPacket()
Con la sintassi read(packetBuffer,Maxsize) la funzione definisce il buffer dei caratteri in ingresso e la sua dimensione massima

write(messaggio) // viene inviato il messaggio Udp ed è utilizzata all'interno di un costrutto beginPacket() / EndPacket().La prima serve a definire indirizzo e porta del destinatario del pacchetto Udp, mentre la seconda avvia la trasmissione del pacchetto. Il valore restituito dalla funzione è il numero dei caratteri scritti.

I trasferimenti UDP

Es:

```
Void loop ()
{
Udp.beginPacket(Udp.remoteIP(),Udp.remotePort()); // inizializza e prepara il
collegamento Udp;
Udp.write("Messaggio");
Udp.endPacket();  // chiude la procedura di preparazione, scrittura dei dati nel buffer e invio
                     del pacchetto Udp
}
```

parsePacket() *// controlla la presenza di un pacchetto Udp in ingresso e ne conta il numero di byte*

available() *// utilizzabile solo dopo la funzione parsePacket() e restituisce il numero di caratteri ancora da leggere nel buffer di ricezione.*

remoteIP() *// serve a conoscere l'indirizzo IP mittente del pacchetto ricevuto.E' necessario aver usato prima la funzione parsePacket(). Il valore restituito è composto da 4 byte e va quindi memorizzato in una variabile definita con la funzione IPAddress*

remotePort() *// resituisce la porta utilizzatadal computer per l'invio del pacchetto*

Progetto n°1

Servendoci di una fotoresistenza (da 10KΩ) e di un sensore di temperatura (NTC da10KΩ o LM35) si vuole monitorare una piccola serra, scoprendo via internet se c'è abbastanza luce e se la temperatura è giusta. Si vogliono poi memorizzare i valori minimi e massimi di entrambe le grandezze per capire se ad es. sono state superate certe soglie.

Schema elettrico

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = {0x90,0xA2,0xDA,0x00,0x55,0xA0};
IPAddress ip(192,168,0,33);
```

// Come già detto, è necessario impostare l'inidrizzo MAC della scheda Ethernet, possibilmente rispettando quello proprio o quello di una vecchia scheda in disuso. L'indirizzo IP viene invece definito in modo arbitrario o tramite DHCP. Nel nostro esempio abbiamo scelto un IP fisso appartenente allo spazio degli indirizzi privati.

```
EthernetServer server(80);
```

// Creiamo l'istanza chiamata "server" che risponde al protocollo IP e che dialoga sulla porta 80 (se ne possono creare 5 su porte diverse)

Progetto n°1

Adesso definiamo le variabili e le costanti necessarie alla gestione della temperatura e della luce.

```
Int PinCalore =0; // A0 per la NTC o per l'LM35
Int PinLuce=1; // A1 per la fotoresistenza

int Luce;
double Celsius;
double Tmin=1000;
double Tmax=0;
int Lmin=1000;
int Lmax=0;
double R1=10000.0;
double V_in=5.0;

double A= 1.189148e-3;
double B= 2.34125e-4;
double C= 8.76741e-8;          // parametri da aggiustare sperimentalmente con la propria NTC
double K=6; // fattore di dissipazione dal Datasheet;
```

[Nel caso usassimo un LM35 ricordiamoci della conversione in gradi °C in fase di acquisizione
Tempc= analogRead(PinCalore)*500/1024;]

Progetto n°1

Nella parte di Setup() inizializziamo la porta seriale dalla quale potremo monitorare il traffico, sia in fase di test che di funzionamento normale.

Con la funzione Ethernet.bgin (mac,ip); viene inizializzata la libreria e per essere certi che l'assegnazione dell'IP sia andata a buon fine o per vedere cosa ha assegnato un eventuale DHCP, ne facciamo una stampa con l'istruzione Ethernet.localIP().

```
void setup()
{
Serial.begin(9600);
Ethernet.begin (mac,ip);
server.begin;
Serial.print ("il Server è a questo indirizzo ");
Serial.println(Ethernet.localIP());
}

void loop ()
{
Misura(); // la misurazione viene chiamata come funzione esternamente dal loop che fornirà i valori di luce
          e temperatura; questo per concentrarci sulla gestione del colloquio attraverso la creazione
          dell'istanza client se c'è una connessione che è stata attivata sul server. La ricezione dei dati
          dal client che si è connesso viene considerata terminata solo se viene ricevuta una riga vuota
          ed è questo il test che viene effettuato per continuare ad inserire caratteri in un astringa di
          ricezione "Response" prendendo in considerazione i primi 15 caratteri, dentro ai quali siamo
          sicuri che ci arriverà l'informazione significativa. Quello che ci aspettiamo è una richiesta di
          aggiornamento della pagina o una richiesta di azzeramento dei valori minimi e massimi
          memorizzati.( solo se troviamo il carattere "C" come invio dal client all'interno del messaggio
          http
```

Progetto n°1

```
Ethernet client = server.available(); //Si è collegato qualcuno?
If (client)
{
        Serial.println("Nuovo client");
        Boolean currentLineIsBlank = true; // fine messaggio HTTP= linea blank
        String Response;          // qui ci vanno i caratteri ricevuti
        Response= String(""");    // iniziamo con una stringa vuota
        int RL=1;                 // var che conta i caratteri in arrivo dal client
        while (client.connected()) // adesso che il client è connesso
        {
        if (client.available())
                {
                char c=client.read(); //leggiamo i caratteri provenienti dal client
                Serial.write(c);      // e intanto li scriviamo sulla seriale
                if (RL<=15) {Response+=c; RL+=1;  // incrementiamo RL  sino a 15
                }
        // verifichiamo se è abbiamo ricevuto tutto il messaggio in arrivo dal client
        attraverso la riga vuota. Se la condizione si verifica, possiamo procedere alla
        gestione di quanto ricevuto
        if (c='\n' && currentLineIsBlank)
                Serial.println(Response.substring(7,8)); //quale azione?
                client.println("HTTP/1.1 200 OR");
                client.println("Content-Type: text/html");
```

Progetto n°1

```
client.println("Connection: close");
client.println();
 client.println("<!DOCTYPE HTML>");
 client.println("<html>");
 client.println("Benvenuto al micro server <br /> delle condizioni
                ambientali.<br /><br />");
```

Adesso controlliamo se prima di proseguire , dobbiamo azzerare i valori in quanto il client ci ha trasmesso la lettera "C"

```
if (Response.substring(7,8) =="C"
{ // solo se c'è in una precisa posizione il carattere "C" alla posizione 7,8 troveremo sempre il primo carattere
    inviato dal client
        Tmin=1000;
        Tmax=0;
        Lmin=1000;
        Lmax=0;
        Misura();
        Client.print("Massimi e minimi azzerati come richiesto");
        Client.println("<br /><br />");
}
```

Progetto n°1

```
// altrimenti non modifichiamo i valori minimi e massimi e stampiamo sul client le informazioni vere e proprie
client.print("Temperatura corrente: ");
client.print(Celsius);
client.print("<<br /> Temperatura minima: ");
client.print(Tmin);
client.print("<<br /> Temperatura massima: ");
client.print(Tmax);
client.print(<br /><br />);
client.print("Luminosità corrente: ");
client.print(Luce);
client.print("<br />Luminosità minima: ");
client.print(Lmin);
client.print ("<br />Luminosità massima: ");
client.print(Lmax);
client.print(<br /><br />);
```

Progetto n°1

```
// creazione dei pulsanti Aggiorna e Cancella
client.print("<form action=\"http:// ");
client.print(ip);
client.println("?A\"method=\"post\">");
client.println("<input type=\"submit\" value=\"Aggiorna\"/>");
client.print("</form><br />");

client.print("<form action=\"http:// ");
client.print(ip);
client.println("?C\"method=\"post\">");
client.println("<input type=\"submit\" value=\"Cancella\"/>");
client.print("</form><br />");

client.println("</html>");
break;
}
```

Progetto n°1

```
// Se non si raggiunge la fine della comunicazione la variabile currentLinesIsBlank è false e si prosegue nella
                lettura dei caratteri di ingresso
If  (c==‘\n’) {
                 currentLinesIsBlank=true;
                  }
                else if (c!=‘\r’) {
                                        currentLinesIsBlank=false;
                                        }
      }
}
delay(1);
client.stop();
Serial.println(“Client disconnected”);  // chiude la comunicazione col client
}
}
```

Progetto n°1

```
// Per l'NTC occorre convertire da ohm a °Kelvin secondo la formula sviluppata da Steinhart e Hart
double SteinhartHart (double R)
{
        double logR= log(R);
        double logR3= log(R)*logR*logR;
        return 1.0/(A+B*logR+C*logR3);
}
// Misura() legge i valori dei due sensori collegati alle porte analogiche A0 e A1. Ripetendo la lettura tre volte
   per la temperatura e due volte per la luce per poi farne la media. Sempre in questa funzione abbiamo il
   confronto dei valori memorizzati con quelli correnti per gestire i minimi e i massimi valori.
void Misura()
{
double adc_raw =analogRead(PinCalore);
delay(10);
adc_raw = adc_raw+analogRead(PinCalore);
delay(10);
adc_raw = (adc_raw+analogRead(PinCalore))/3; //media di 3 letture
double V=adc_raw/1024*V_IN;
double R_th= (R1*V) / (V_IN-V);  // calcolo resistenza misurata in ohm
double  kelvin=SteinhartHart(R_th) – V*V/(R*R_th); // otteniamo i ° Kelvin
Celsius = Kelvin – 273.15; // da gradi Kelvin a Celsius
Luce=analogRead(PinLuce);
delay(10);
```

Progetto n°1

```
Luce=(Luce +analogRead (PinLuce))/2;
if (Luce<=Lmin)  {Lmin=Luce;}
if (Lmax<=Luce) {Lmax=Luce;}
if (Celsius<=Tmin) {Tmin=Celsius;}
if (Tmax<=Celsius) {Tmax=Celsius;}
}
```