

# PROGETTI CON ARDUINO UNO

-Introduzione alla shield Ethernet-

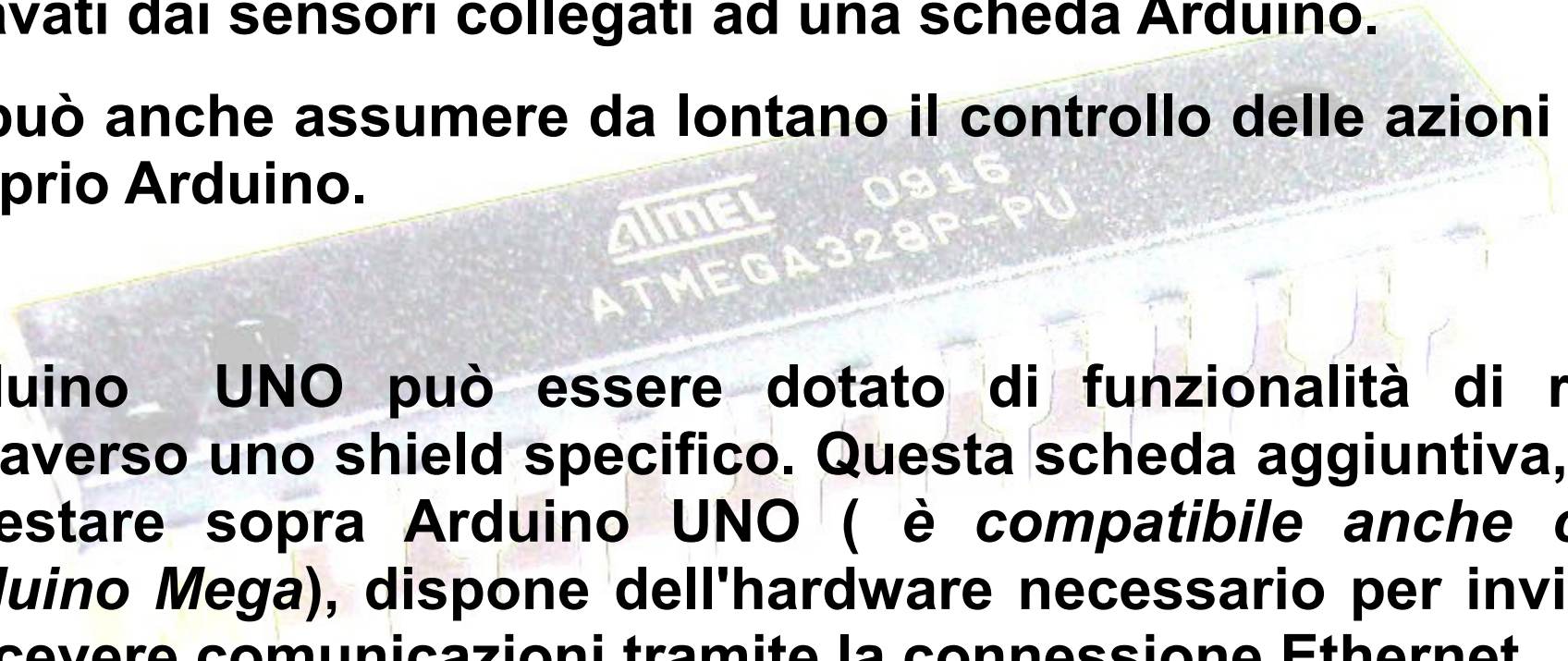


[www.arduino.cc](http://www.arduino.cc)  
[systemisds.altervista.org](http://systemisds.altervista.org)

# ***Ethernet e la comunicazione in rete***

**Con le funzionalità di rete si possono condividere i dati ricavati dai sensori collegati ad una scheda Arduino.**

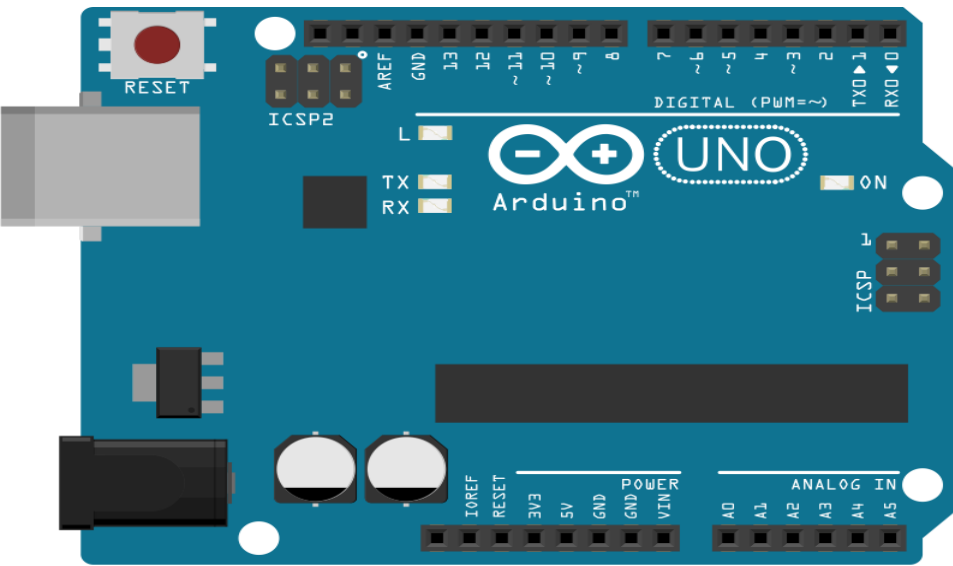
**Si può anche assumere da lontano il controllo delle azioni del proprio Arduino.**



**Arduino UNO può essere dotato di funzionalità di rete attraverso uno shield specifico. Questa scheda aggiuntiva, da innestare sopra Arduino UNO ( è *compatibile anche con Arduino Mega*), dispone dell'hardware necessario per inviare e ricevere comunicazioni tramite la connessione Ethernet.**

**Grazie alla shield Ethernet e alle relative librerie software da utilizzare nell'IDE diventa possibile creare un server per semplici pagine web o leggere dati dalla rete.**

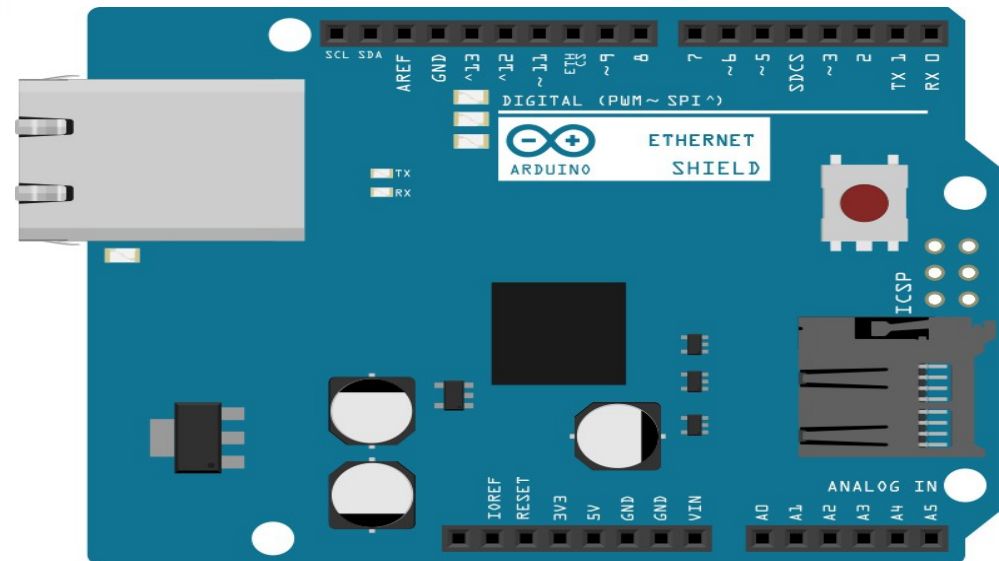
# Arduino UNO e Shield Ethernet



Made with  Fritzing.org

La shield Ethernet viene innestata sopra la scheda Arduino e offre la stessa piedinatura

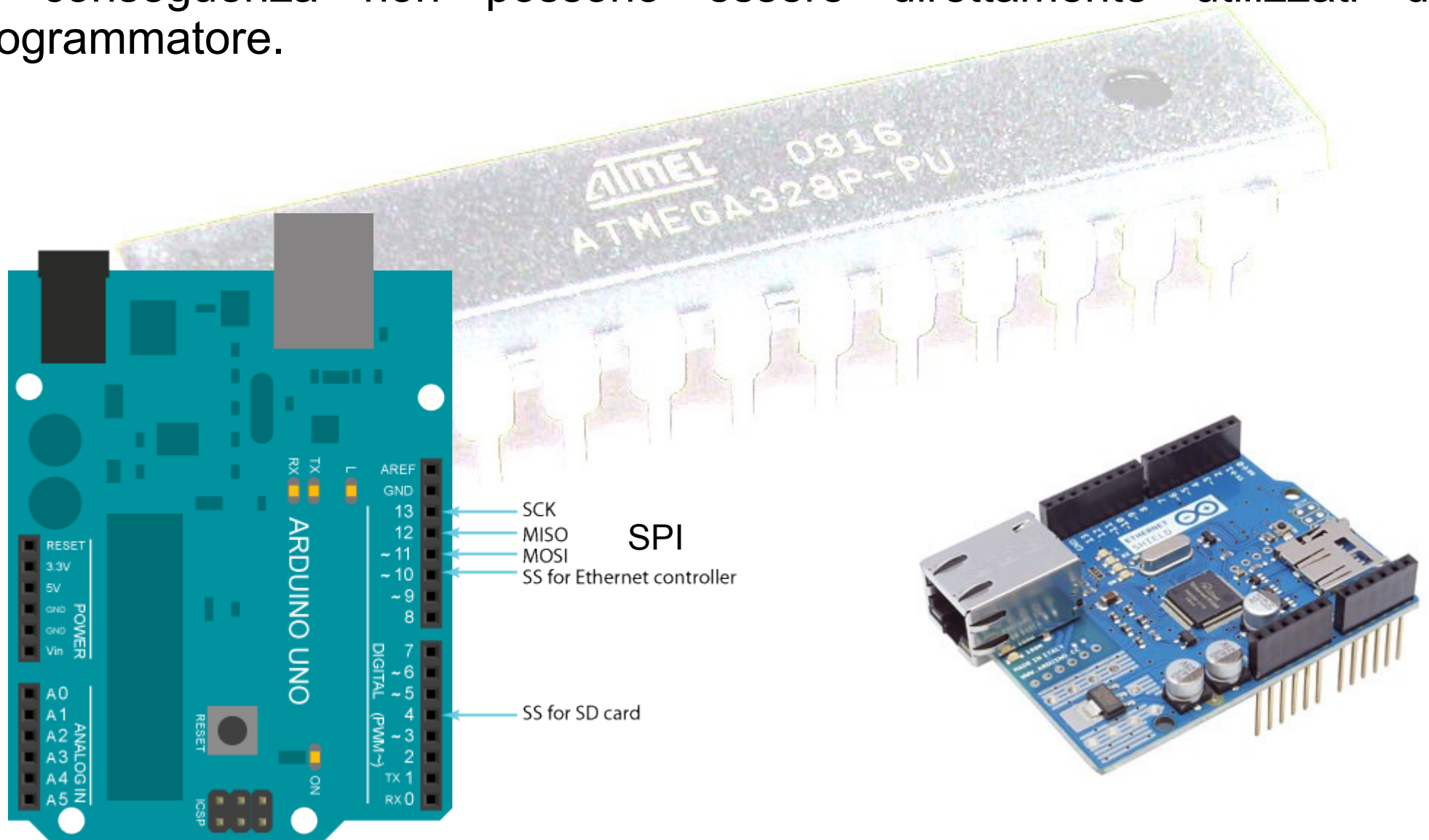
  
Cavo UTP



Made with  Fritzing.org

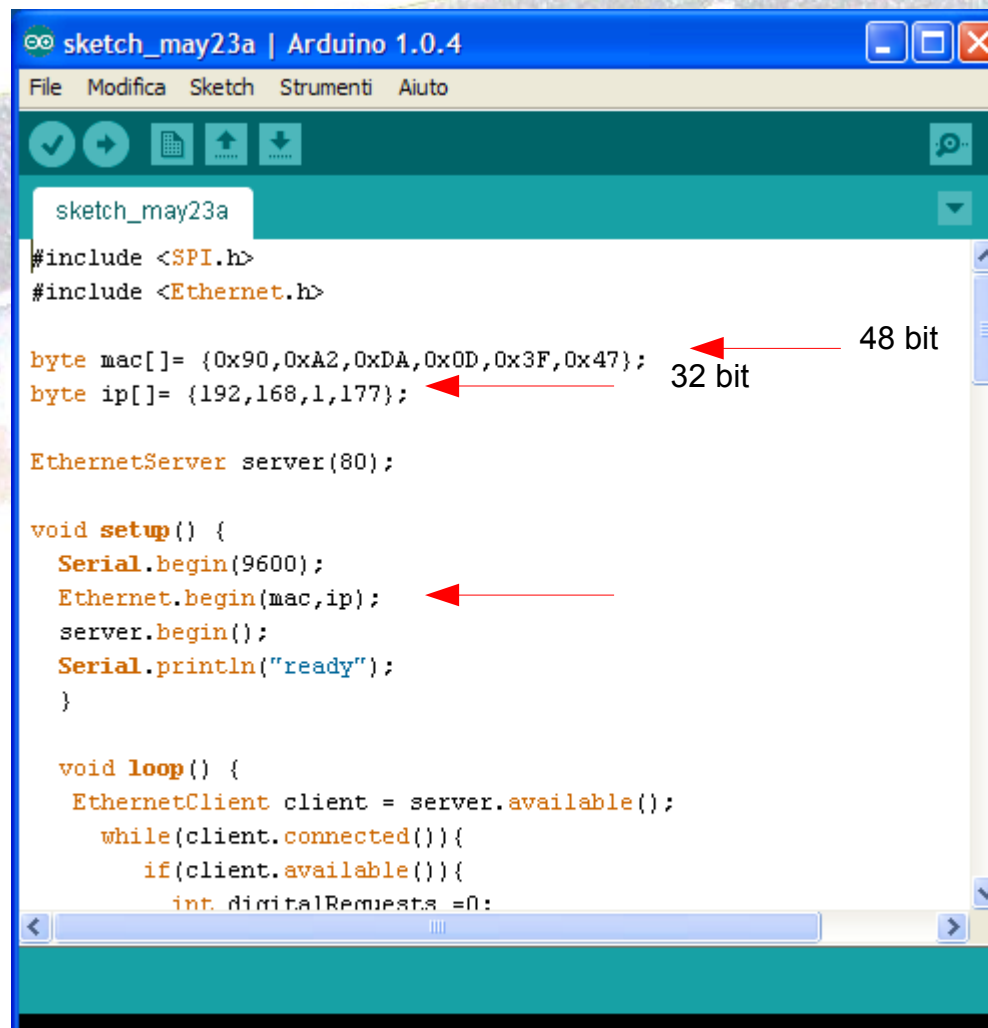
I pin digitali **10, 11, 12 e 13** sono utilizzati per permettere il funzionamento della Ethernet Shield.

Di conseguenza non possono essere direttamente utilizzati dal programmatore.



Le librerie da includere nello sketch sono la `SPI.h` per la *Serial Peripheral Interface* e `Ethernet.h`

E' inoltre necessario impostare un indirizzo fisico MAC address per la scheda e un indirizzo logico IP address, compatibile con la rete alla quale ci si collega, con cavo di rete UTP.

A screenshot of the Arduino IDE interface. The window title is "sketch\_may23a | Arduino 1.0.4". The menu bar includes "File", "Modifica", "Sketch", "Strumenti", and "Aiuto". The toolbar shows icons for check, run, file, upload, and download. The sketch name "sketch\_may23a" is displayed in a dropdown menu. The code editor contains the following code:

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

byte mac[] = {0x90,0xA2,0xDA,0x0D,0x3F,0x47};
byte ip[] = {192,168,1,177};

EthernetServer server(80);

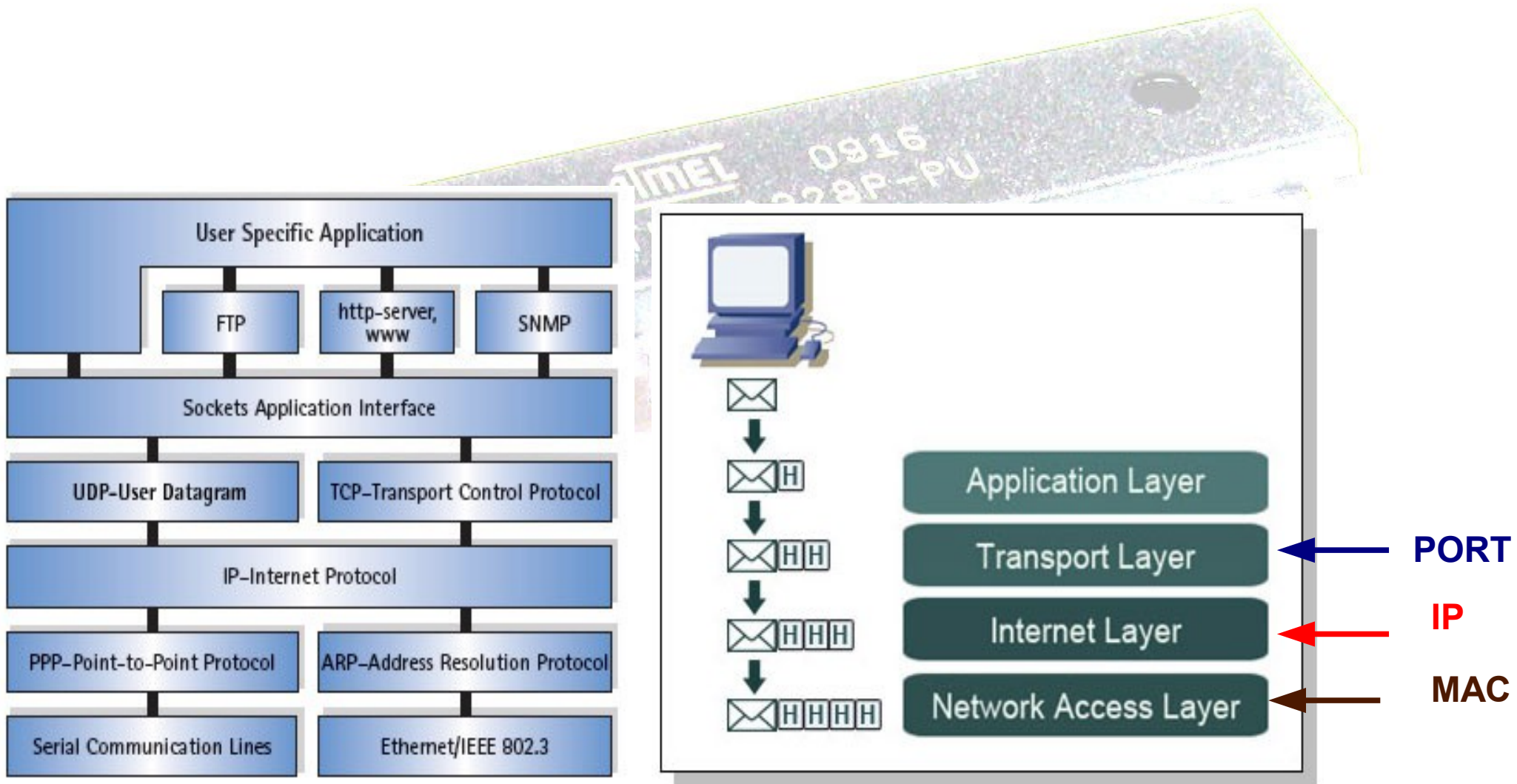
void setup() {
  Serial.begin(9600);
  Ethernet.begin(mac,ip);
  server.begin();
  Serial.println("ready");
}

void loop() {
  EthernetClient client = server.available();
  while(client.connected()){
    if(client.available()){
      int digitalRequests = 0;
```

Red arrows point to the MAC address array, the IP address array, and the `Ethernet.begin(mac,ip);` line. To the right of the code, there are labels: "48 bit" with an arrow pointing to the MAC array, and "32 bit" with an arrow pointing to the IP array.

# Architettura di rete TCP IP

## livelli e protocolli



# Inizializzazione shield Ethernet

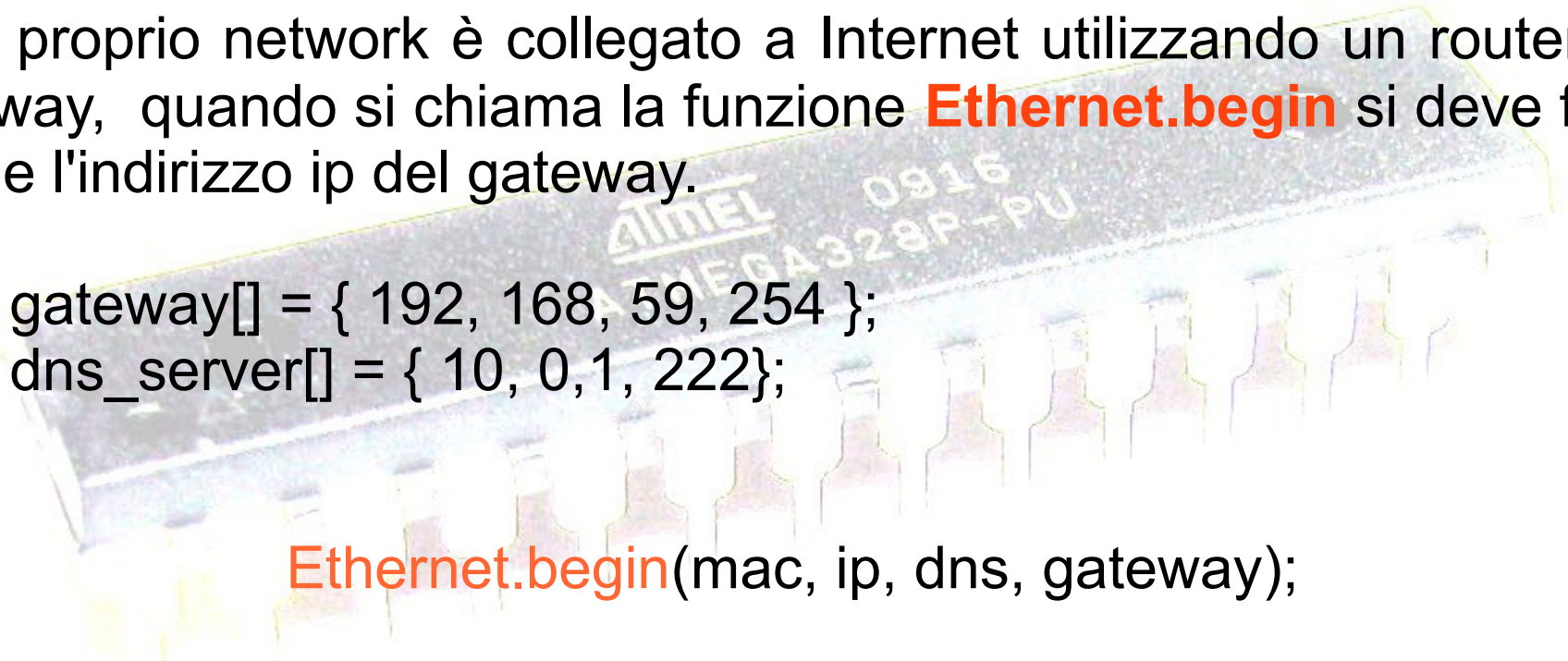
```
Ethernet.begin(mac, ip)
```

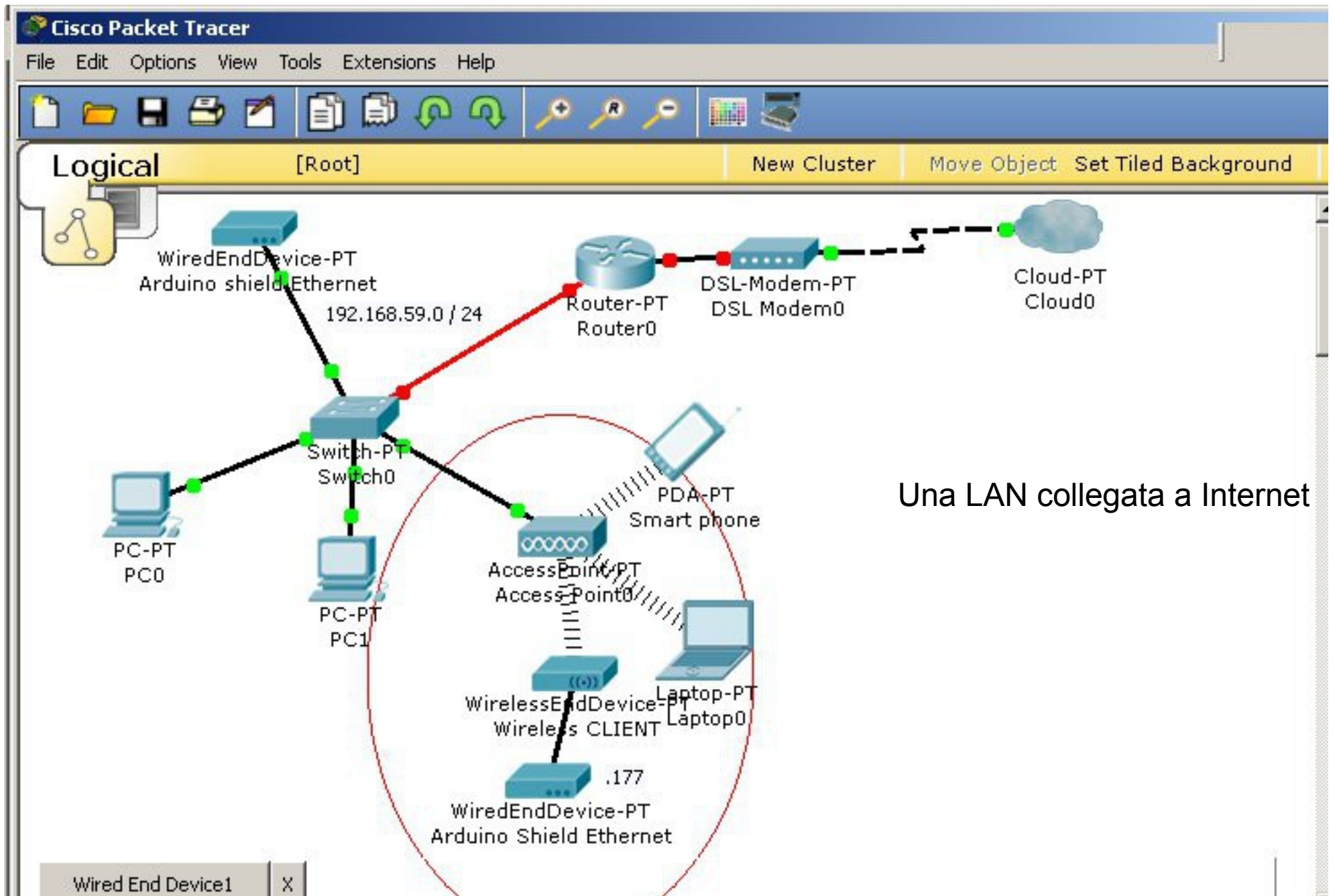
Se il proprio network è collegato a Internet utilizzando un router o un gateway, quando si chiama la funzione **Ethernet.begin** si deve fornire anche l'indirizzo ip del gateway.

```
Byte gateway[] = { 192, 168, 59, 254 };
```

```
Byte dns_server[] = { 10, 0, 1, 222};
```

```
Ethernet.begin(mac, ip, dns, gateway);
```





Una LAN collegata a Internet



## Ethernet

Tecnologia broadcast di rete locale che fornisce le funzionalità di base per trasmettere messaggi tra computer di una LAN. Gli indirizzi di origine e di destinazione di questi messaggi sono identificati da indirizzi **MAC** (Media Access Control).

## TCP e IP

Transmission Control Protocol (**TCP**) e Internet Protocol (**IP**) sono protocolli chiave di Internet .

**TCP** è un protocollo di trasporto end to end e **IP** è un protocollo di instradamento.

I messaggi **TCP/IP** sono consegnati attraverso gli indirizzi IP unici di chi invia e chi riceve.

**IP** lavora sopra **Ethernet** e **TCP** sopra **IP**.

**TCP** introduce l' indirizzo di porta che individua una applicazione in un host.

**TCP** è un protocollo connesso

Per certe applicazioni viene utilizzato **UDP** senza connessione

## Indirizzi IP locali

Se sono presenti più computer e tutti sono collegati a Internet attraverso una LAN, ciascun computer utilizza un indirizzo ip locale che viene fornito dal proprio router attraverso un servizio **DHCP** ( Dynamic Host Configuration Protocol).

Il range di indirizzi a 32 bit da **192.168.0.0** a **192.168.255.255** è assegnato per indirizzi privati (classe C).

La maschera naturale di una classe C è **255.255.255.0**

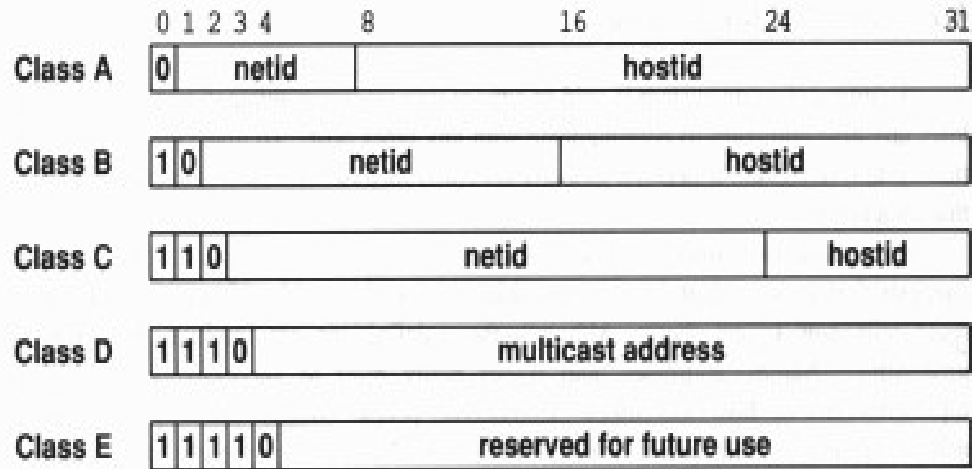
### Indirizzi di rete locale

Sono stati riservati una serie di indirizzi IP come dedicati all'uso su reti locali.

Questi indirizzi sono:

- da **10.0.0.0** a **10.255.255.255** classe A
- da **172.16.0.0** a **172.31.255.255** classe B
- da **192.168.0.0** a **192.168.255.255** classe C

Gli indirizzi **IP** sono indirizzi gerarchici che contengono due informazioni **Rete** e **Host**.



Per individuare l'indirizzo di rete viene eseguita una operazione logica **AND** tra l'indirizzo **IP** e la **maschera**. La maschera naturale di una classe C è **255.255.255.0**

Notazione Decimale Puntata

Forma Binaria

IP address	192.168.5.130	11000000.10101000.00000101.10000010
<b>Subnet Mask</b>	<b>255.255.255.0</b>	<b>11111111. 11111111. 11111111. 00000000</b>
Porzione di rete	192.168.5.0	11000000.10101000.00000101.00000000

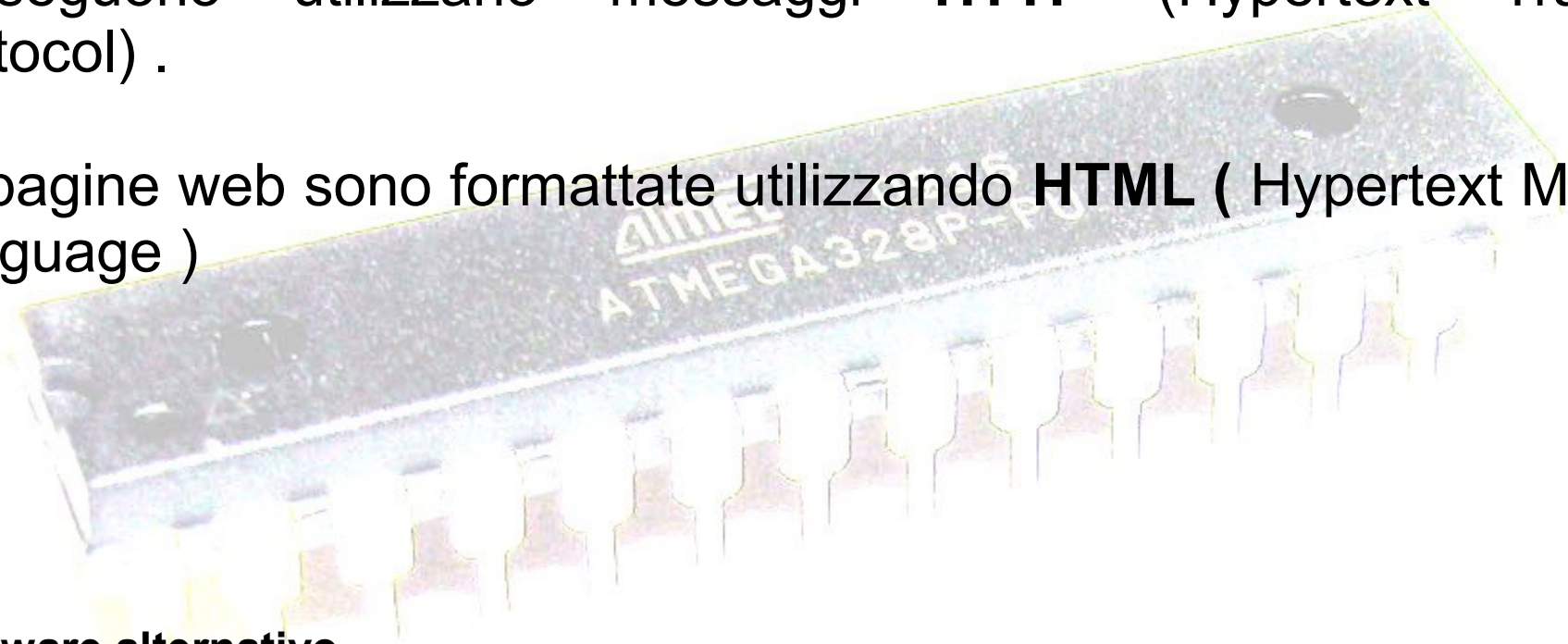
## **HTTP**

Le richieste web inviate dal browser web e le risposte che ne conseguono utilizzano messaggi **HTTP** (Hypertext Transfer Protocol) .

Le pagine web sono formattate utilizzando **HTML** ( Hypertext Markup Language )

### **Hardware alternativo**

In alcune schede il chip Wiznet è sostituito con il dispositivo ENC28J60, più economico. Questo chip utilizza una serie diversa di librerie.



## Configurare Arduino come un server web

### Problema

Si vuole che lo sketch risponda come server web con una pagina web dinamica che contiene i 6 valori degli ingressi analogici della scheda Arduino. La pagina web può essere richiesta da un computer qualsiasi di una rete LAN.

Un browser si connette all'indirizzo assegnato alla scheda Ethernet di Arduino. (per es. 192.168.59.177 per una rete 192.168.59.0/24)

### Soluzione

Alla richiesta di connessione lo sketch su Arduino invia, con un serie di **print**, il codice html di una semplice pagina web.



Richiesta di connessione porta 80

Browser

Server su  
Arduino

Invio pagina web in html

```
/*
```

## Web Server

A simple web server that shows the value of the analog input pins.  
using an Arduino Wiznet Ethernet shield.

```
*/
```

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

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

```
// Enter a MAC address and IP address for your controller
```

```
// The IP address will be dependent on your local network:
```

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
```

```
IPAddress ip(192,168,59,177);
```

```
// (port 80 is default for HTTP)
```

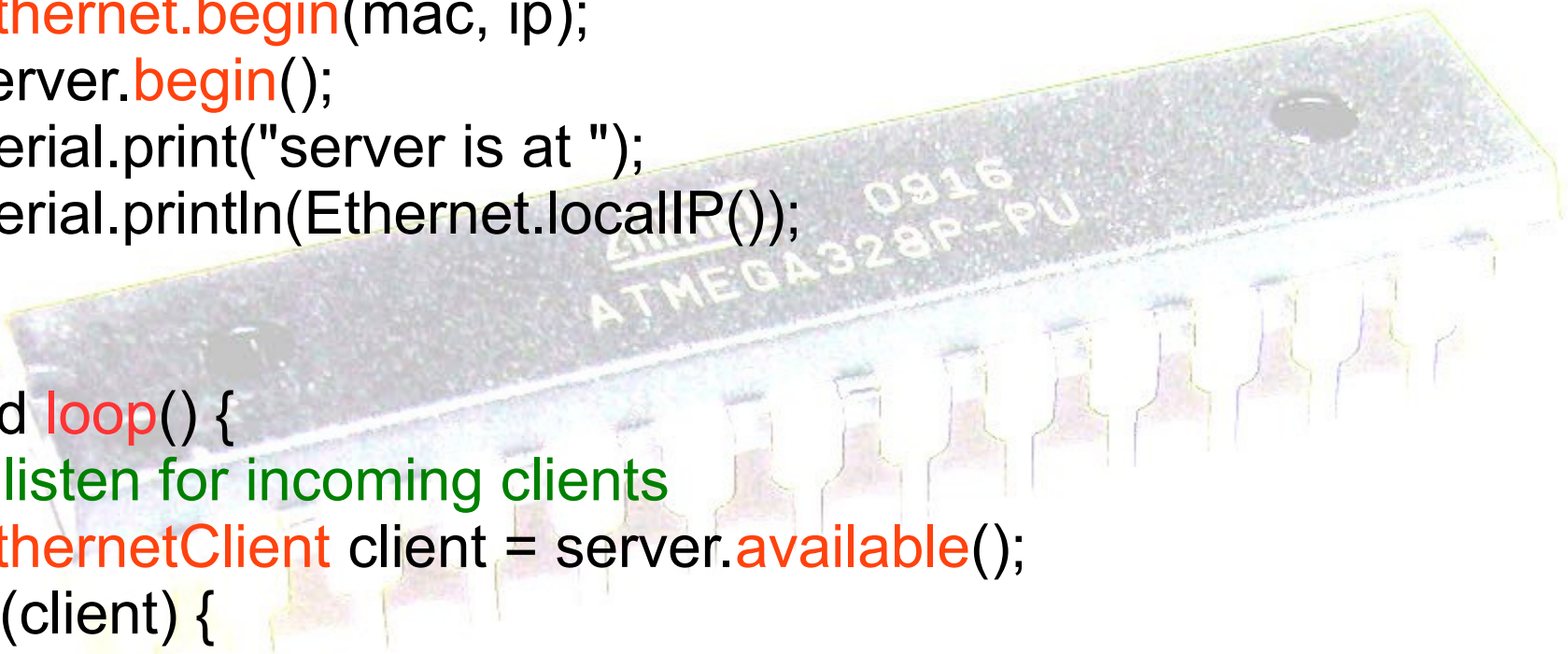
```
EthernetServer server(80);
```





```
void setup() {  
  Serial.begin(9600);  
  
  // start the Ethernet connection and the server  
  Ethernet.begin(mac, ip);  
  server.begin();  
  Serial.print("server is at ");  
  Serial.println(Ethernet.localIP());  
}
```

```
void loop() {  
  // listen for incoming clients  
  EthernetClient client = server.available();  
  if (client) {  
    Serial.println("new client");  
    // an http request ends with a blank line  
    while (client.connected()) {  
      while (client.available()) { // arrivato carattere  
        char c = client.read(); // legge dalla rete  
        Serial.write(c); // scrive sulla seriale  
      }  
    }  
  }  
}
```



# SERIAL MONITOR

server is at 192.168.59.177  
new client

GET / HTTP/1.1



Richiesta del browser

Host: 192.168.59.177

User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:25.0)

Gecko/20100101 Firefox/25.0

Accept:

text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8

Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3

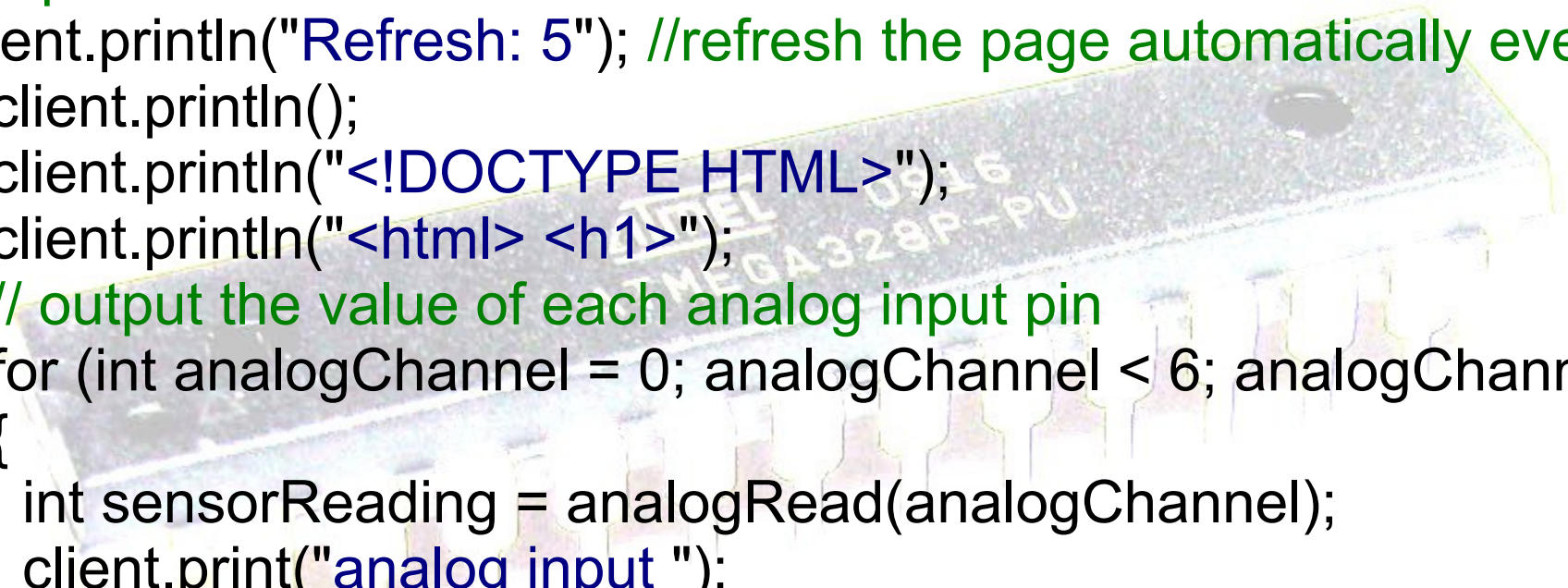
Accept-Encoding: gzip, deflate

Connection: keep-alive

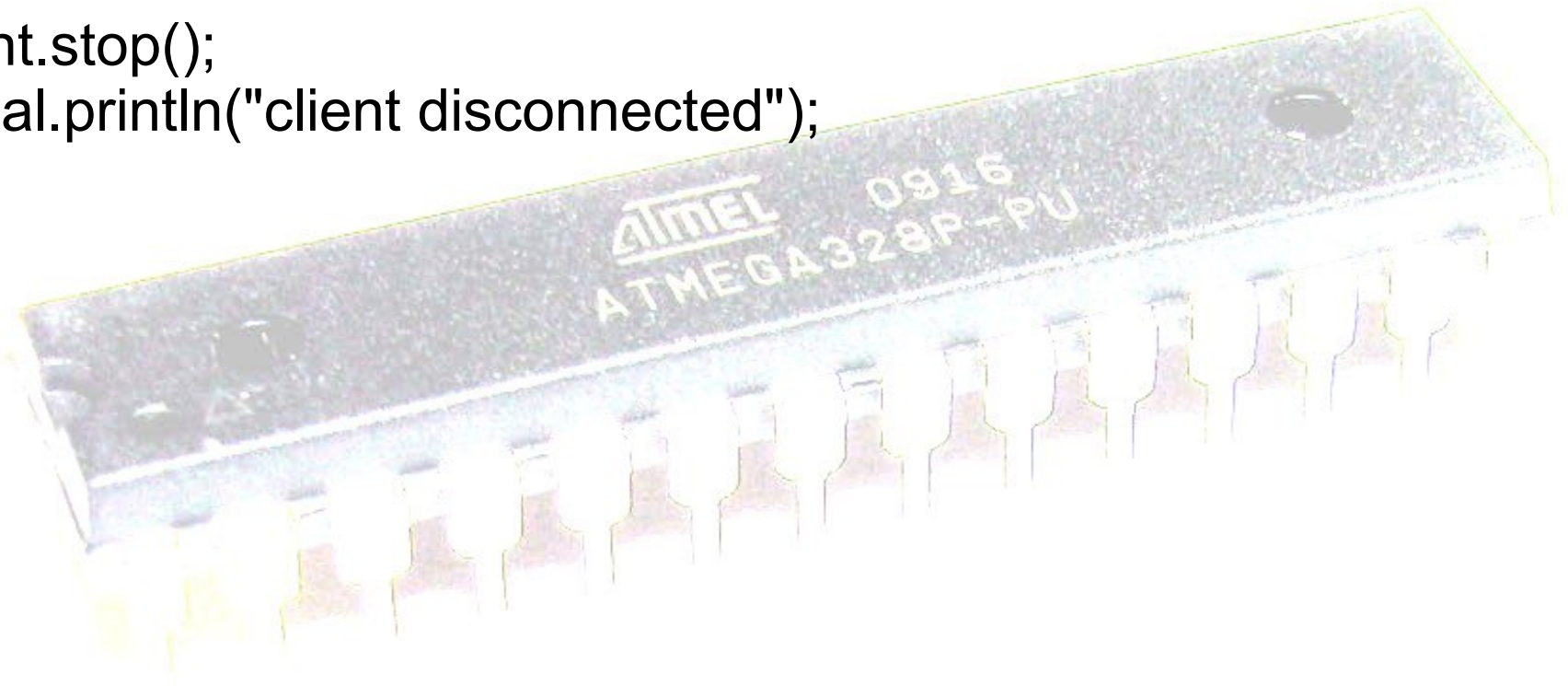


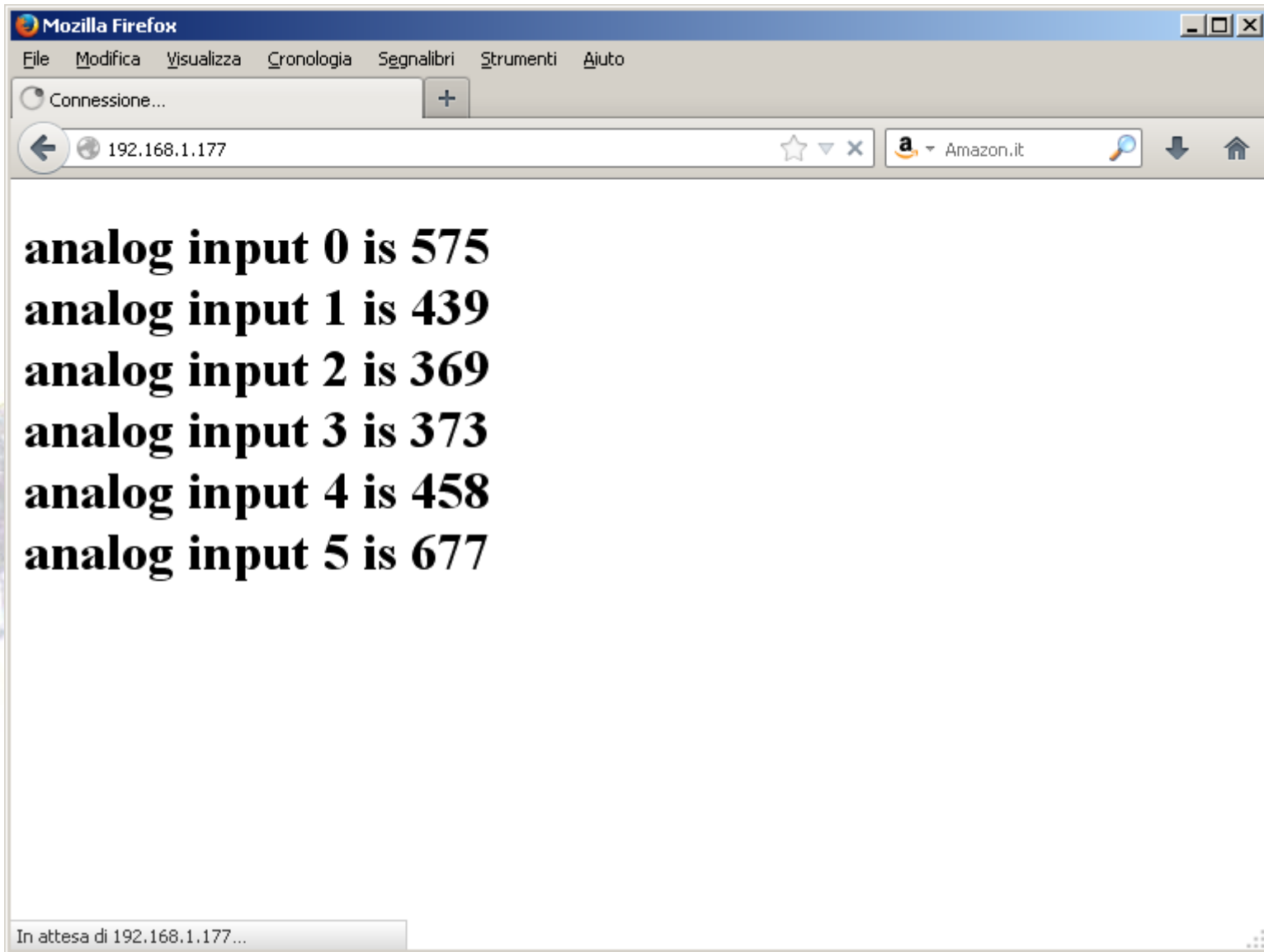
Riga vuota

```
// send a standard http response header
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println("Connection: close"); // the connection will be closed
after response
client.println("Refresh: 5"); //refresh the page automatically every 5 s
client.println();
client.println("<!DOCTYPE HTML>");
client.println("<html> <h1>");
// output the value of each analog input pin
for (int analogChannel = 0; analogChannel < 6; analogChannel++)
{
  int sensorReading = analogRead(analogChannel);
  client.print("analog input ");
  client.print(analogChannel);
  client.print(" is ");
  client.print(sensorReading);
  client.println("<br />");
}
client.println(" </h1> </html>");
break;
}
```



```
// give the web browser time to receive the data
delay(1);
// close the connection
client.stop();
Serial.println("client disconnected");
}
}
```

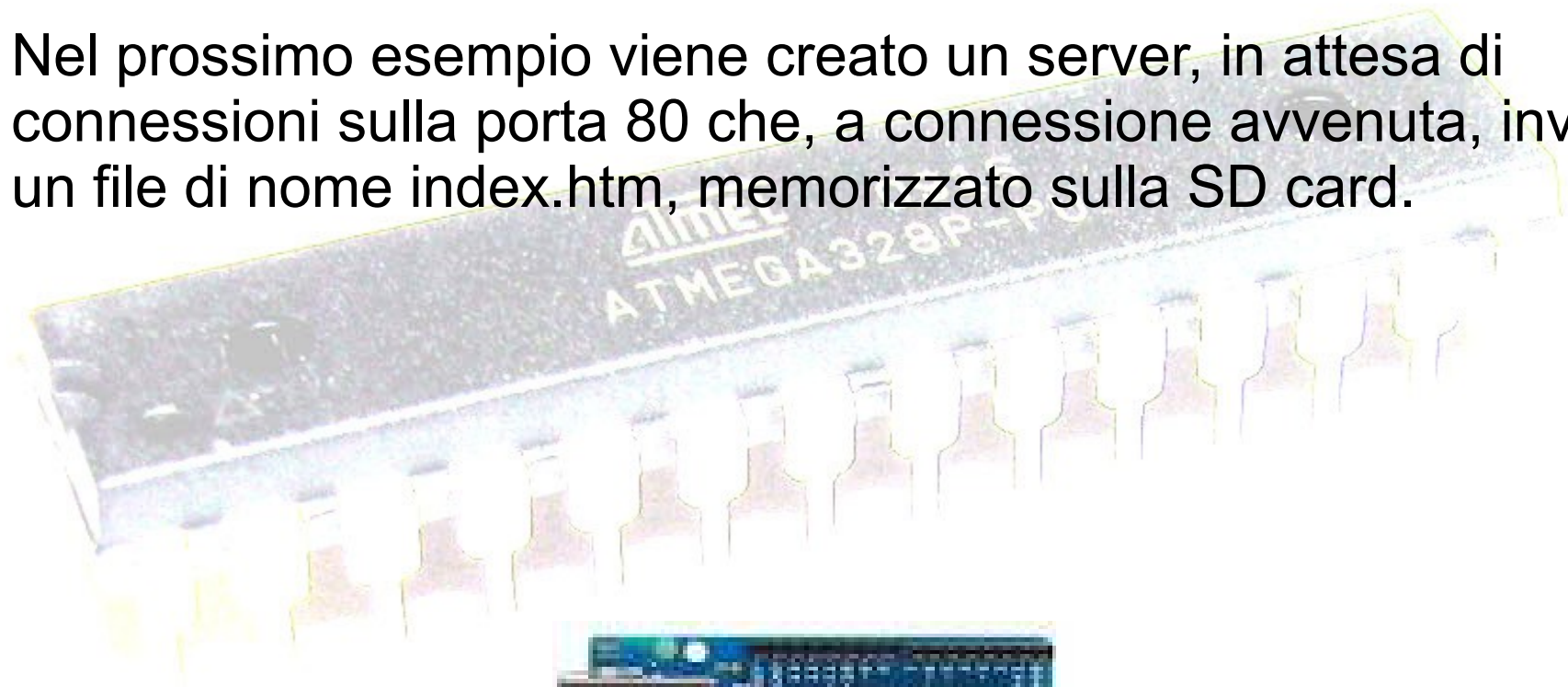




## SD card

La shield ethernet permette l'utilizzo di una scheda di memoria di tipo SD cards.

Nel prossimo esempio viene creato un server, in attesa di connessioni sulla porta 80 che, a connessione avvenuta, invia un file di nome index.htm, memorizzato sulla SD card.



← SD card

```
#include <SD.h>
```

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

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

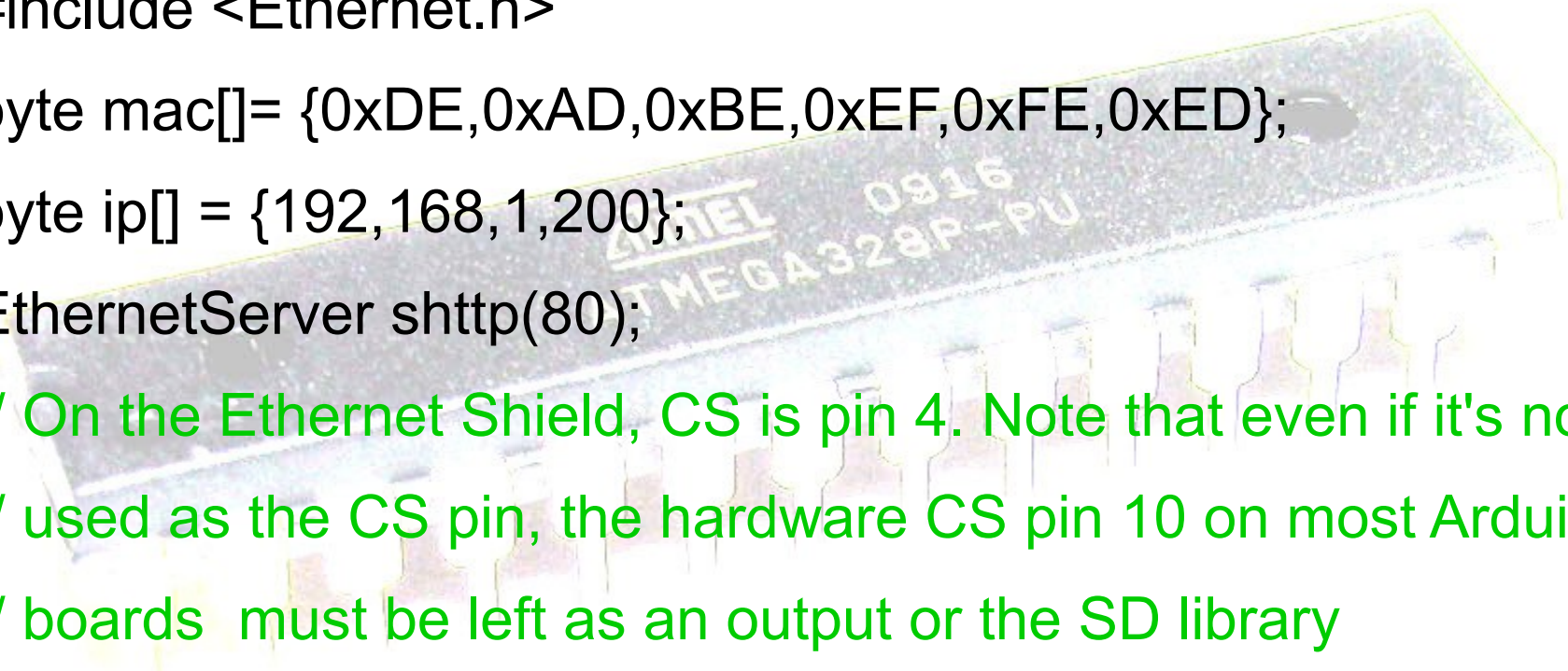
```
byte mac[] = {0xDE,0xAD,0xBE,0xEF,0xFE,0xED};
```

```
byte ip[] = {192,168,1,200};
```

```
EthernetServer shttp(80);
```

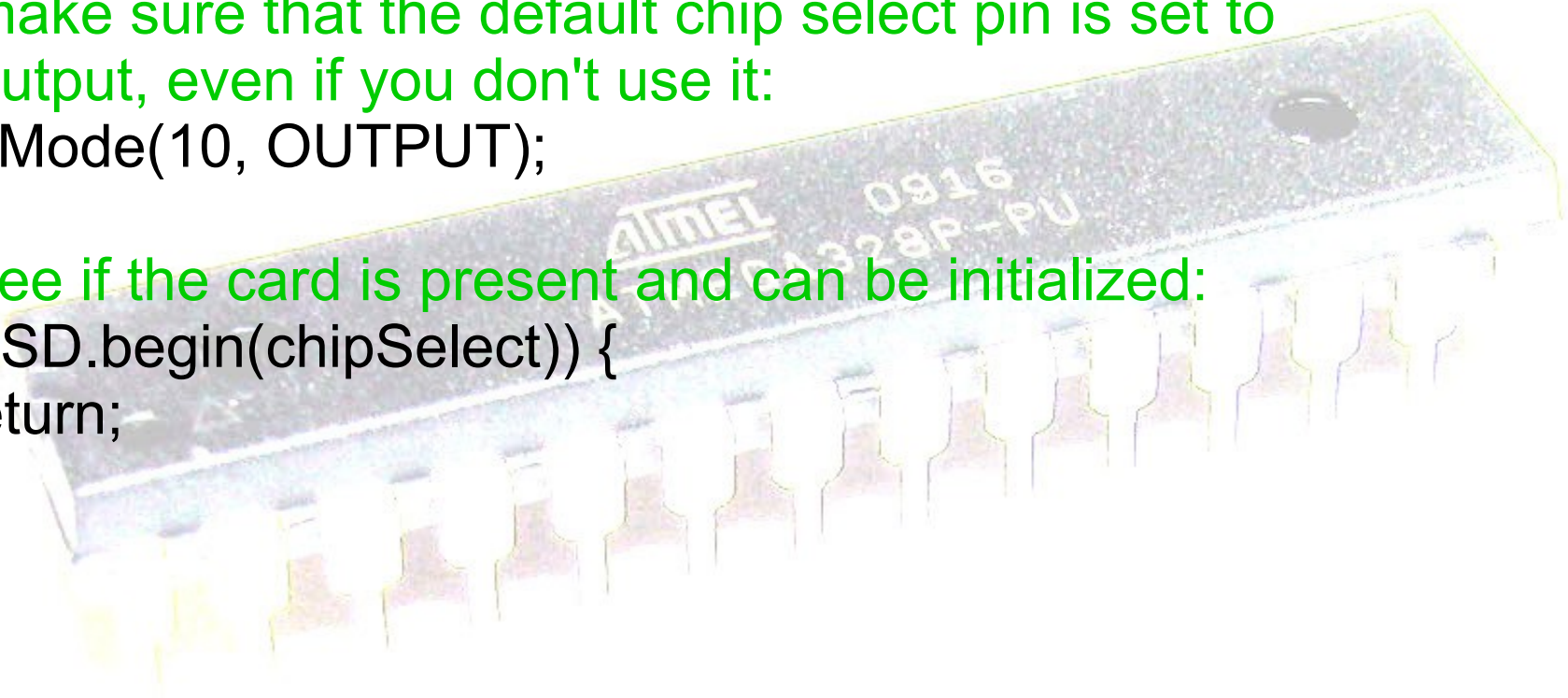
```
// On the Ethernet Shield, CS is pin 4. Note that even if it's not  
// used as the CS pin, the hardware CS pin 10 on most Arduino  
// boards must be left as an output or the SD library  
// functions will not work.
```

```
const int chipSelect = 4;
```



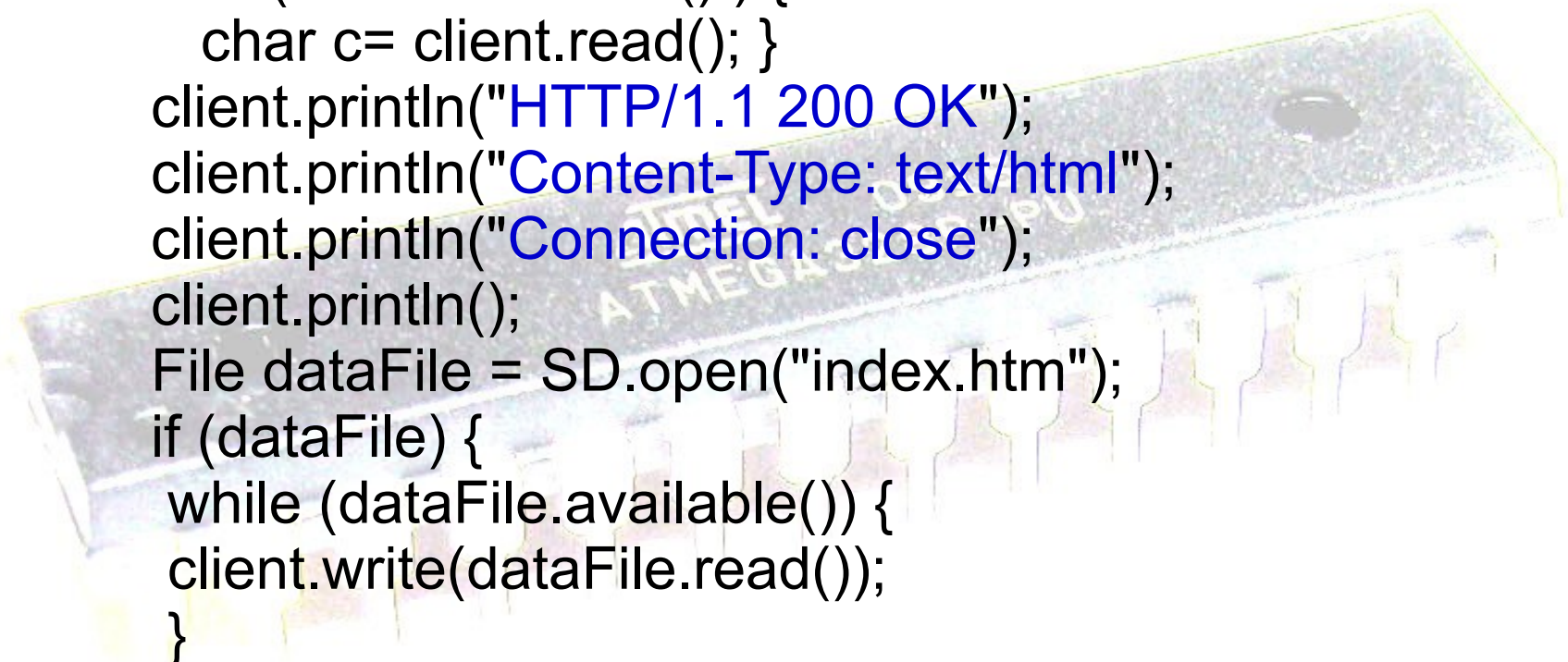
```
void setup()
{
  Ethernet.begin(mac,ip);
  shttp.begin();
  // make sure that the default chip select pin is set to
  // output, even if you don't use it:
  pinMode(10, OUTPUT);

  // see if the card is present and can be initialized:
  if (!SD.begin(chipSelect)) {
    return;
  }
}
```



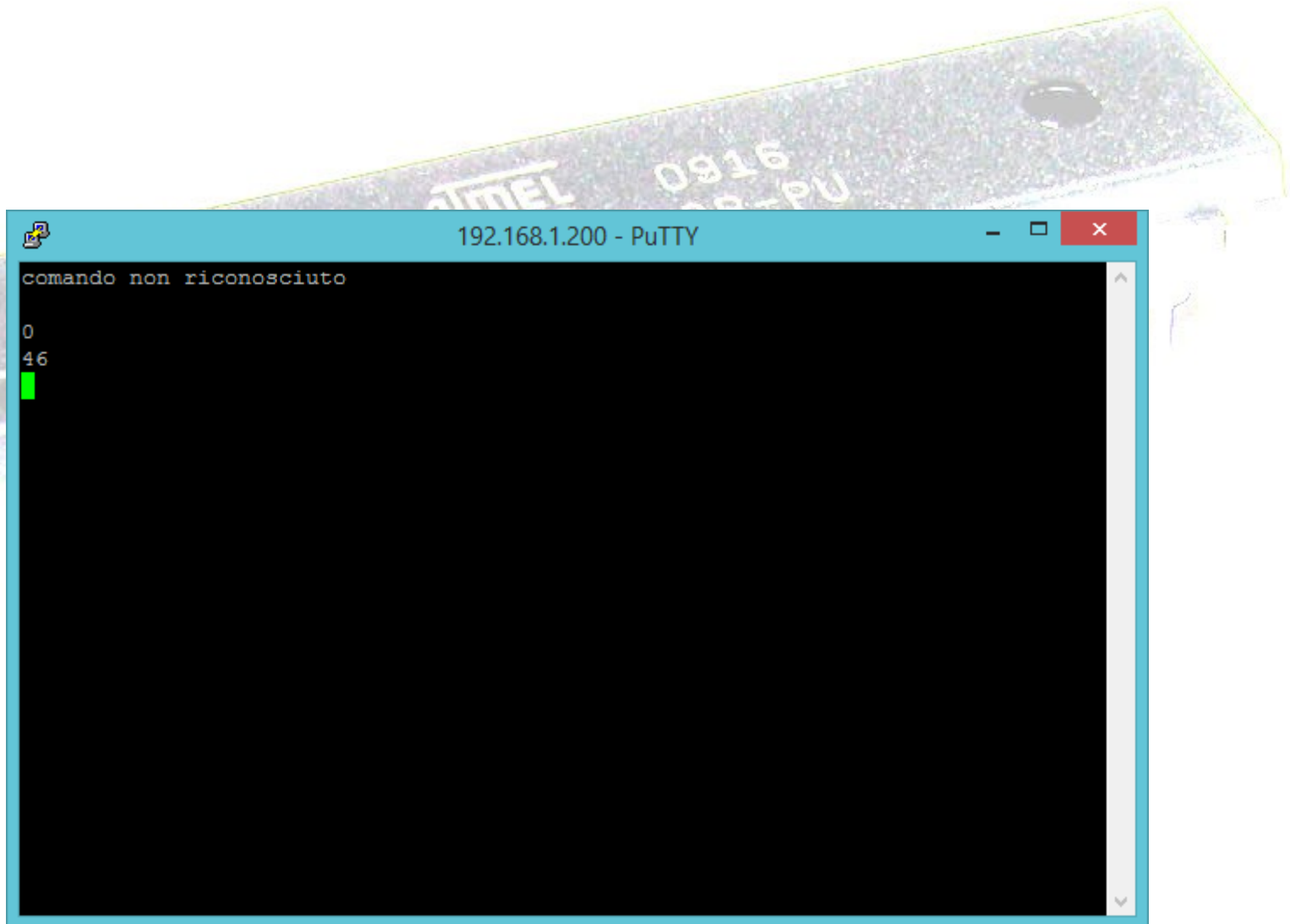


```
void loop() {
  EthernetClient client = shttp.available(); // in ascolto
  if (client) {
    while(client.connected() ){
      while(client.available() ) { // arrivato carattere
        char c= client.read(); }
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println("Connection: close");
        client.println();
        File dataFile = SD.open("index.htm");
        if (dataFile) {
          while (dataFile.available()) {
            client.write(dataFile.read());
          }
          dataFile.close();
        } // end if
        client.stop();
      }
    } // if
  }
}
```



# Comunicazione in telnet sulla porta 23

Viene riconosciuto il comando '0' che restituisce il valore letto dall'ingresso analogico A0.

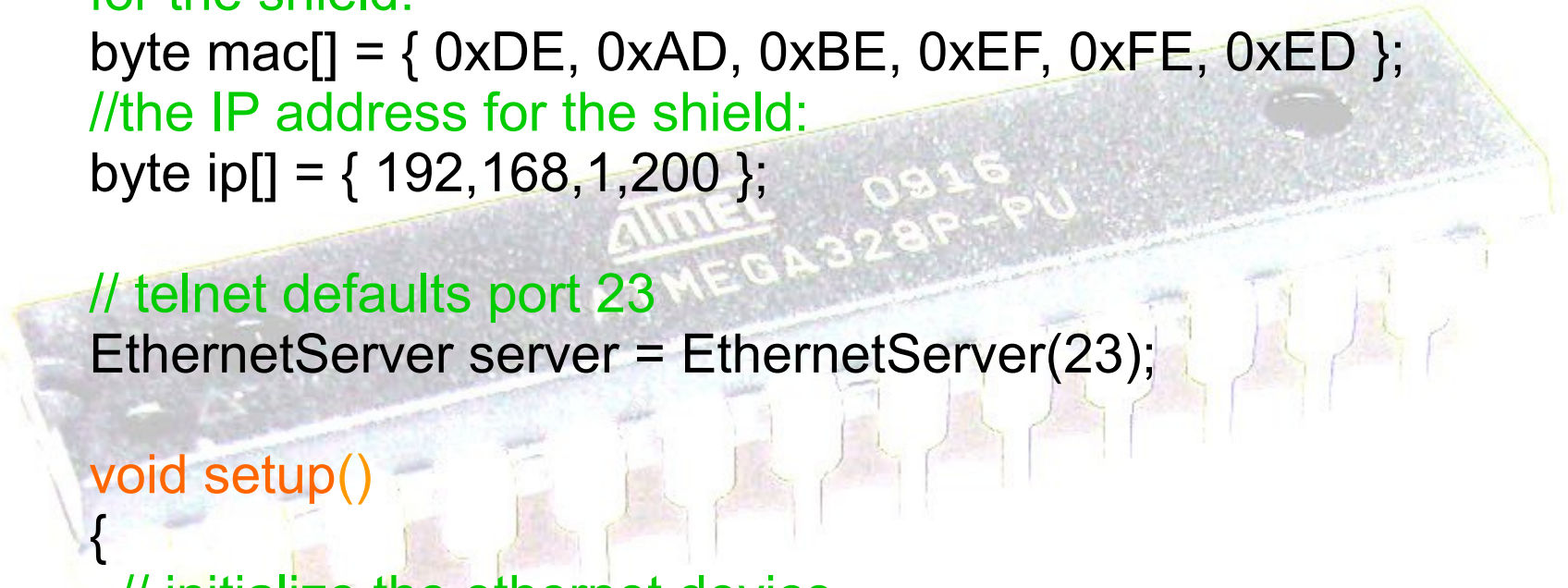


```
#include <SPI.h>
#include <Ethernet.h>
// the media access control (ethernet hardware) address
for the shield:
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
//the IP address for the shield:
byte ip[] = { 192,168,1,200 };

// telnet defaults port 23
EthernetServer server = EthernetServer(23);

void setup()
{
  // initialize the ethernet device
  Ethernet.begin(mac, ip); //gateway and subnet are
optional.

  // start listening for clients
  server.begin();
}
```



```
void loop()
```

```
{
```

```
  // if an incoming client connects
```

```
  EthernetClient client = server.available();
```

```
  if (client == true) {
```

```
    // read bytes from the incoming client
```

```
    if (client.available()) {
```

```
      char c = client.read(); // primo carattere
```

```
      while(client.available())
```

```
        char cc = client.read();
```

```
        //non considero altri caratteri
```

```
        if (c=='0'){
```

```
          client.println(analogRead(A0)); // invio  
lettura su A0
```

```
        }
```

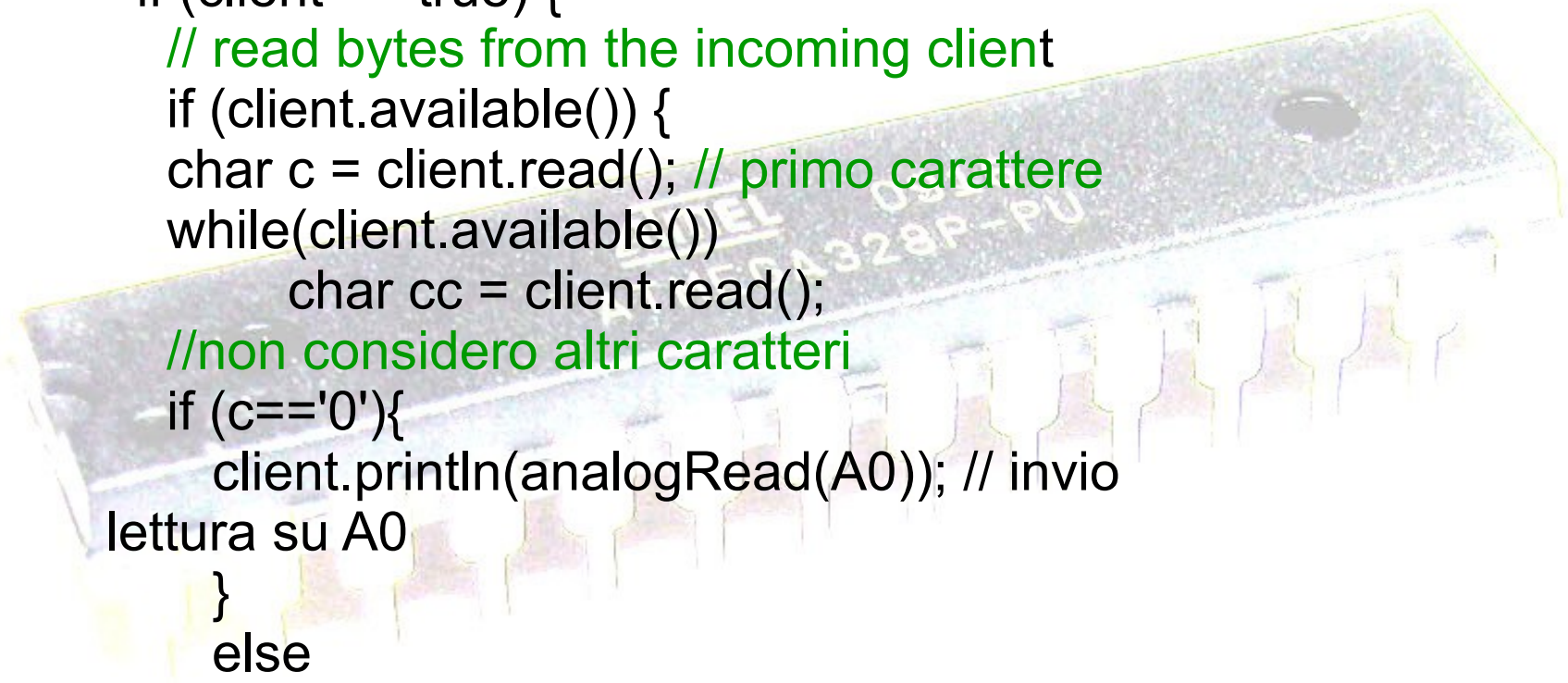
```
        else
```

```
          client.println("comando non  
riconosciuto\n");
```

```
        } //client.available > 0
```

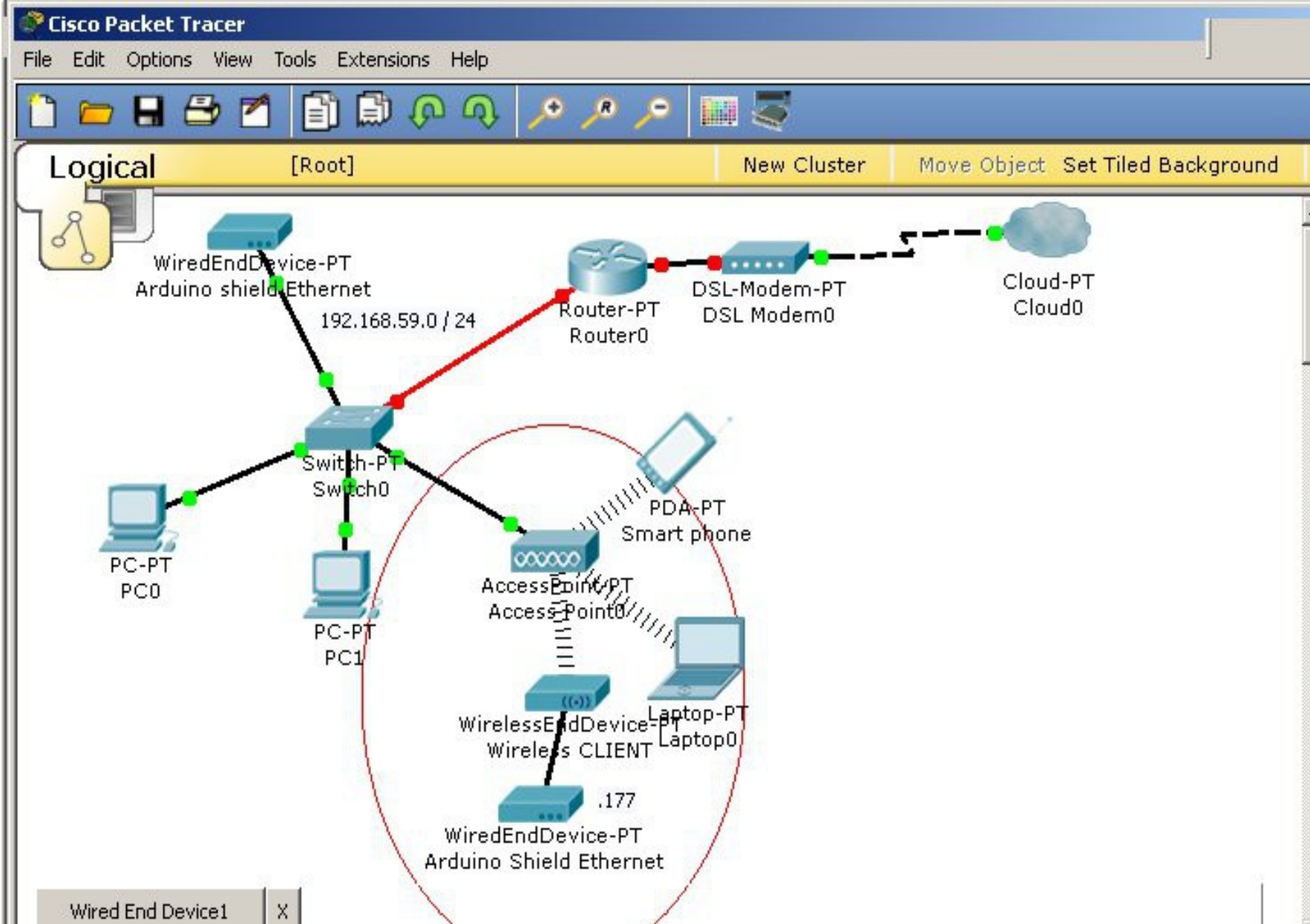
```
    } //client true
```

```
  } // loop
```



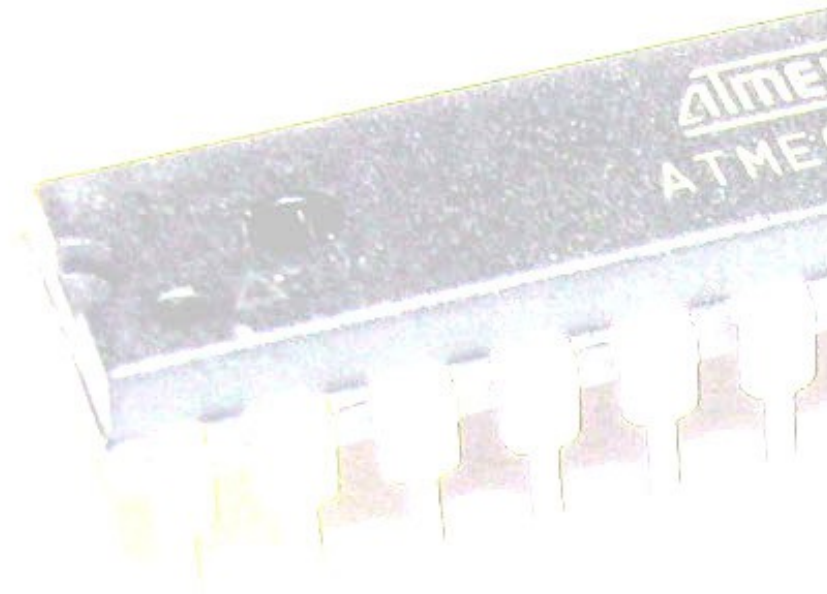
La shield Ethernet può essere collegata direttamente con cavo di rete **UTP** allo switch del laboratorio oppure si può sfruttare un collegamento **WiFi** se la scheda lo consente.





Viene aggiunto allo switch un **Access Point** per collegare Arduino in WiFi. Per aggiungere la funzionalità WiFi alla scheda Ethernet viene utilizzato un dispositivo TP-link 702 in modalità **Client**.

Sarà ora possibile collegare Arduino e raggiungerlo tramite la rete Wi-fi



# Renderere *wireless* l'*ethernet shield*

Esistono diversi *shield* Wi-Fi per Arduino: partendo da quella ufficiale, fino a quelle basate su integrati Microchip, “cugini” dell’enc28j60.

A volte può essere più comodo collegare Arduino alla rete Wi-Fi sfruttando la *shield ethernet* abbinata ad un dispositivo **wireless**.

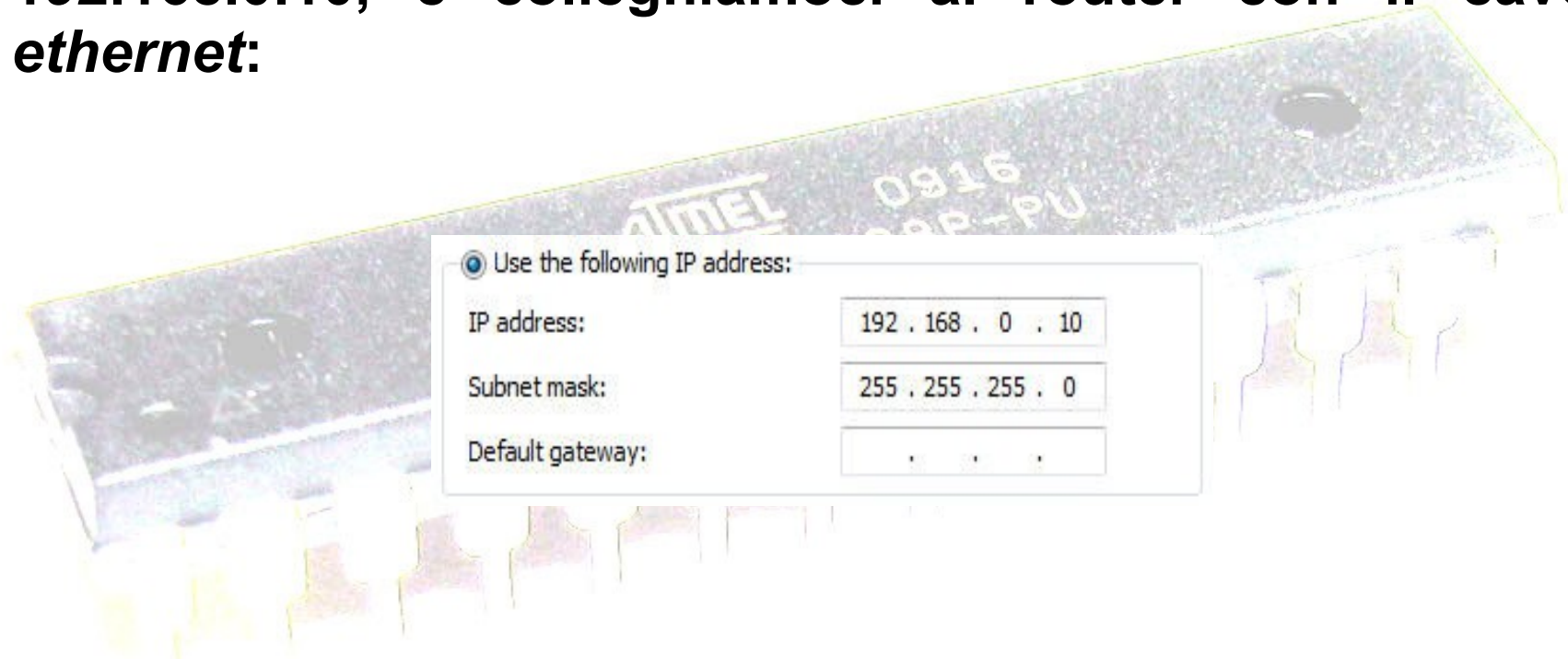
In particolare il *nano router* di TP-LINK ha tra le proprie funzionalità anche la modalità **client** nella quale il router fa da “ponte” tra un dispositivo connesso alla sua porta *ethernet* e la rete Wi-fi.

Il router TL-WR702N inoltre ha dimensioni davvero ridotte e può essere alimentato tramite un connettore **micro USB**.



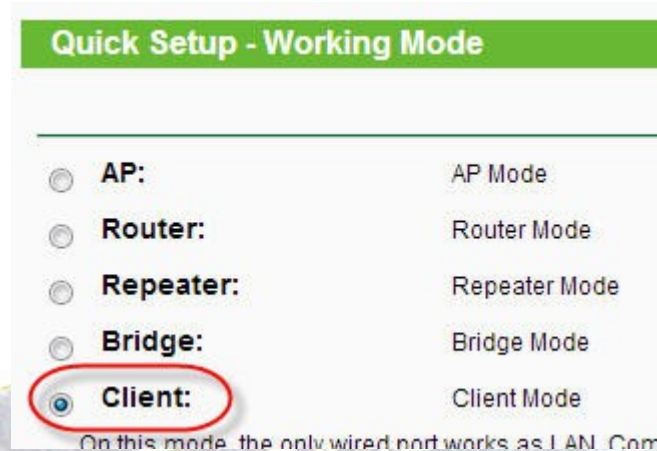
## Configurarlo in modalità client è molto semplice

Per prima cosa diamo al nostro PC un indirizzo fisso, 192.168.0.10, e colleghiamoci al router con il cavo *ethernet*:



Apriamo il browser e digitiamo **http://192.168.0.254**, accedendo con utente e password **admin**.

Selezioniamo **Client** mode:

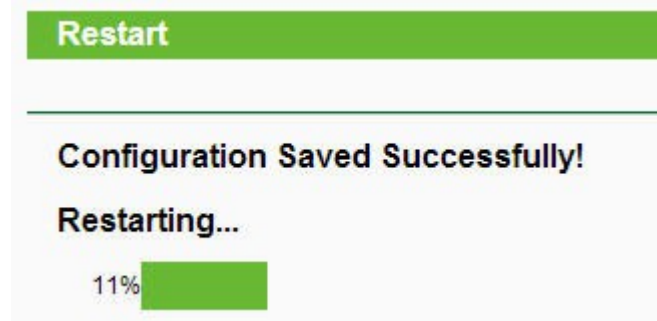


Quick Setup - Working Mode	
<input type="radio"/> AP:	AP Mode
<input type="radio"/> Router:	Router Mode
<input type="radio"/> Repeater:	Repeater Mode
<input type="radio"/> Bridge:	Bridge Mode
<input checked="" type="radio"/> Client:	Client Mode

On this mode, the only wired port works as LAN. Comp

Clicchiamo su **Survey** per cercare la nostra rete Wi-fi, quindi inseriamo gli eventuali parametri di sicurezza.

Attendiamo il restart del router:



Restart

Configuration Saved Successfully!

Restarting...

11%

# Ricevere un insieme di valori analogici dal convertitore ADC del controllore e visualizzarli su un grafico

## Problema

Si vuole che lo sketch risponda con una pagina web che crea sul browser una visualizzazione simile a quella di un oscilloscopio.

## Soluzione

Viene utilizzato un **canvas** di **html5**.

La funzione **analogRead** impiega circa 120 us per acquisire un valore. 400 campioni vengono acquisiti con un ciclo in circa 48 ms.

La rappresentazione viene fatta con una risoluzione di 800x600, 2pixel per campione.

I campioni, numeri da 0 a 1023, vengono mappati nel range da 0 a 599.

Attraverso delle print lo sketch invia al browser una pagina web utilizzando un canvas.

Mozilla Firefox

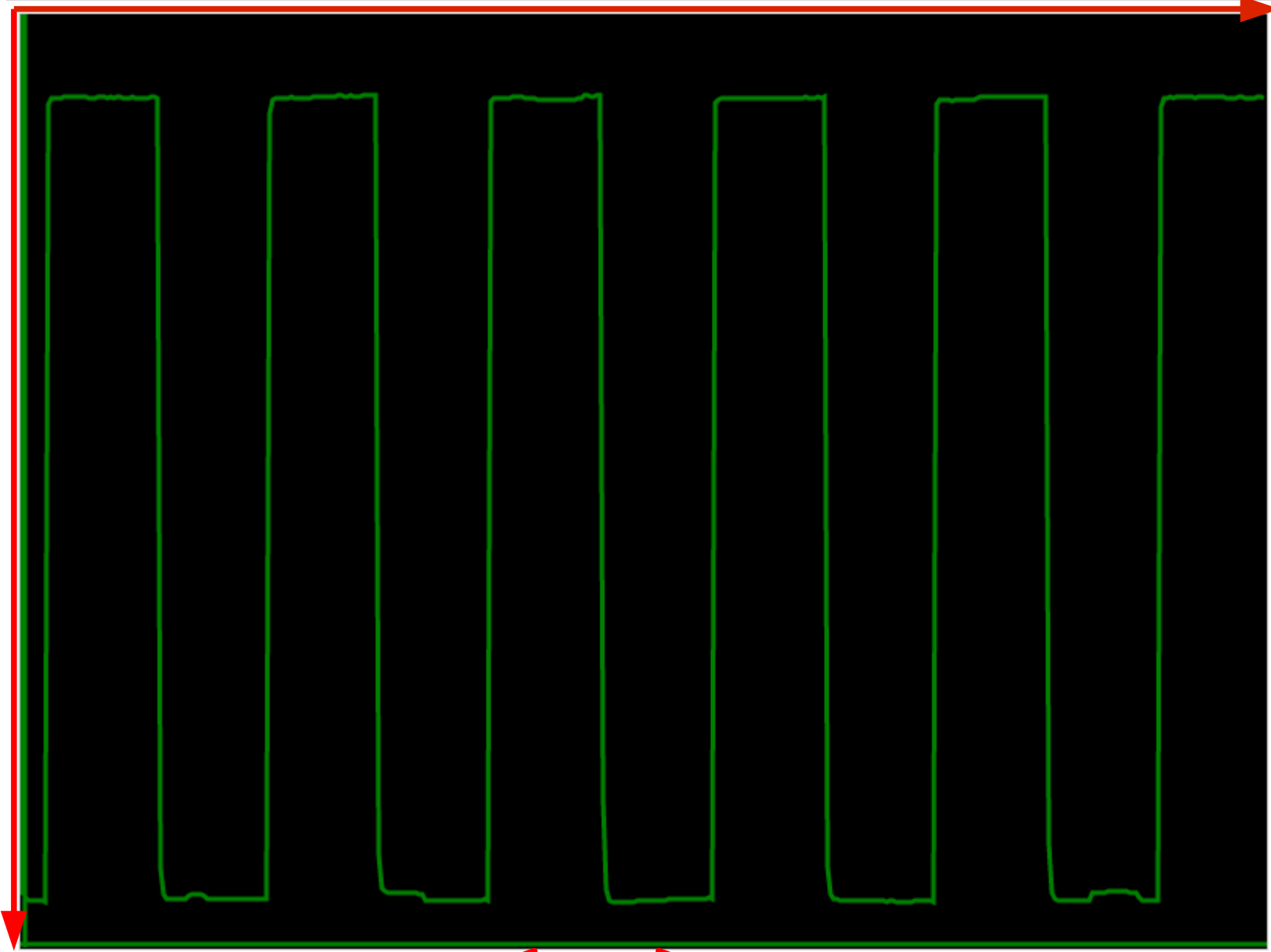
File Modifica Visualizza Cronologia Segnalibri Strumenti Aiuto

http://192.168.1.177/

192.168.1.177

Italiano (Italia)

Amazon.it

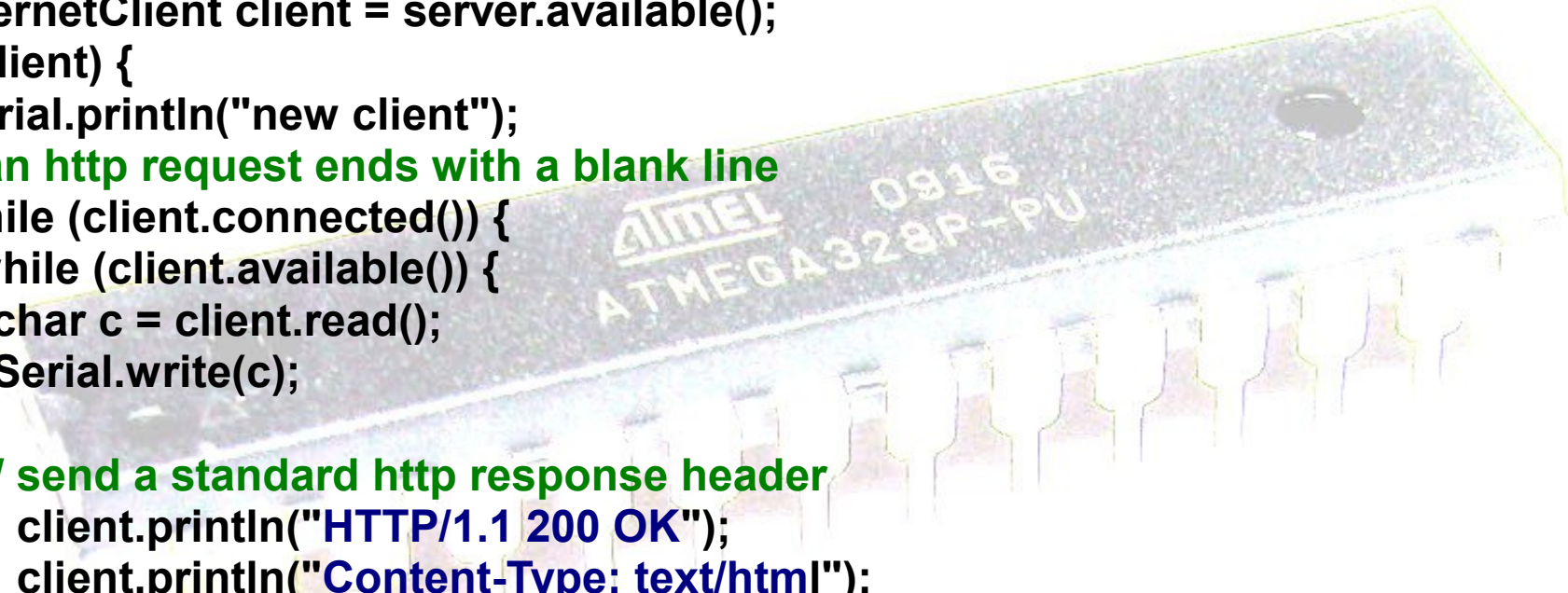


X

y

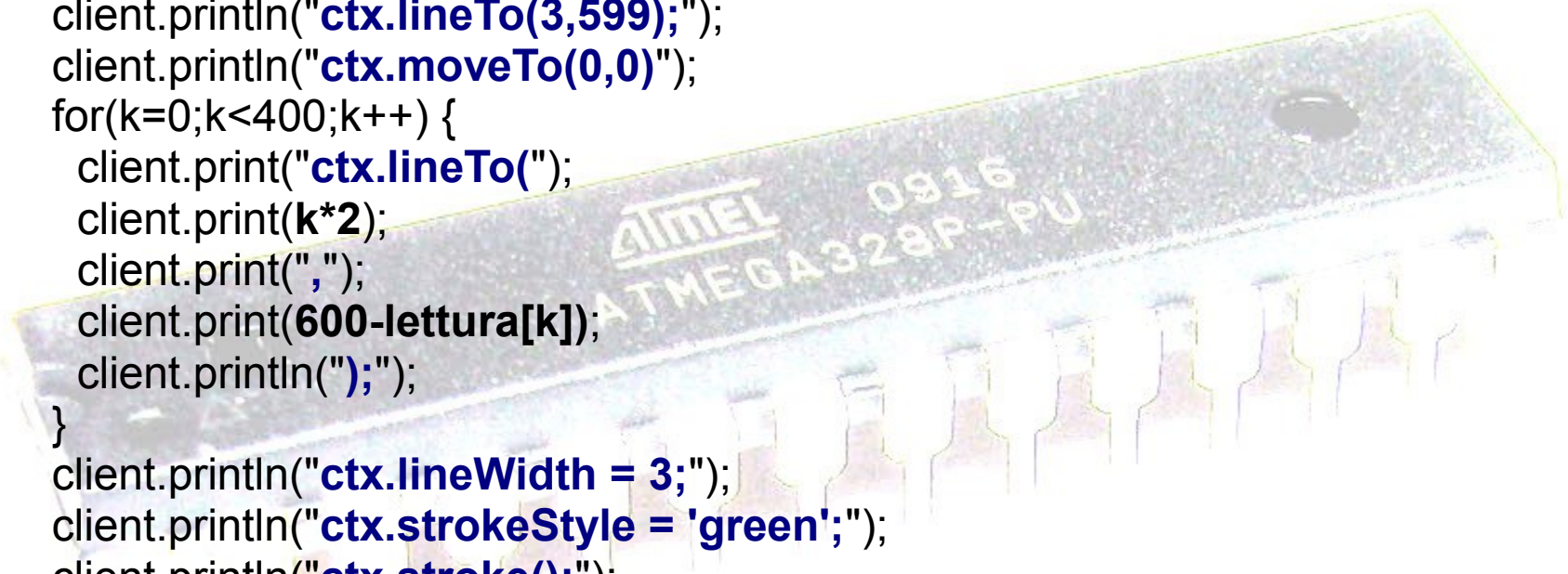
8 ms

```
void loop() {
  for(k=0;k<400;k++)
    lettura[k]=analogRead(A0);
  for(k=0;k<400;k++)
    lettura[k]=map(lettura[k],0,1023,0,599);
  // listen for incoming clients
  EthernetClient client = server.available();
  if (client) {
    Serial.println("new client");
    // an http request ends with a blank line
    while (client.connected()) {
      while (client.available()) {
        char c = client.read();
        Serial.write(c);
      }
      // send a standard http response header
      client.println("HTTP/1.1 200 OK");
      client.println("Content-Type: text/html");
      client.println("Connection: close"); // the connection will be closed after
      // completion of the response
      client.println("Refresh: 5"); // refresh the page automatically every 5 sec
      client.println();
      client.println("<!DOCTYPE HTML>");
      client.println("<html>");
      client.println("<body>");
      client.println("<canvas id='myCanvas' width='800' height='600'
      style='border:1px solid #d3d3d3;background:black'>");
    }
  }
}
```

A photograph of an ATMEGA328P-PU microcontroller chip, a common component used in Arduino Uno boards. The chip is a small, rectangular integrated circuit with a black plastic package and gold-plated pins. The text 'ATMEL 0916 ATMEGA328P-PU' is visible on the top surface of the chip.

```
client.println("<script>");
  client.println("var c=document.getElementById('myCanvas');");
  client.println("var ctx=c.getContext('2d');");
  client.println("ctx.moveTo(0,597);");
  client.println("ctx.lineTo(800,597);");
  client.println("ctx.moveTo(3,0);");
  client.println("ctx.lineTo(3,599);");
  client.println("ctx.moveTo(0,0)");
  for(k=0;k<400;k++) {
    client.print("ctx.lineTo(");
    client.print(k*2);
    client.print(",");
    client.print(600-lettura[k]);
    client.println(");");
  }
  client.println("ctx.lineWidth = 3;");
  client.println("ctx.strokeStyle = 'green';");
  client.println("ctx.stroke();");
  client.println("</script>");
  client.println("</body>");

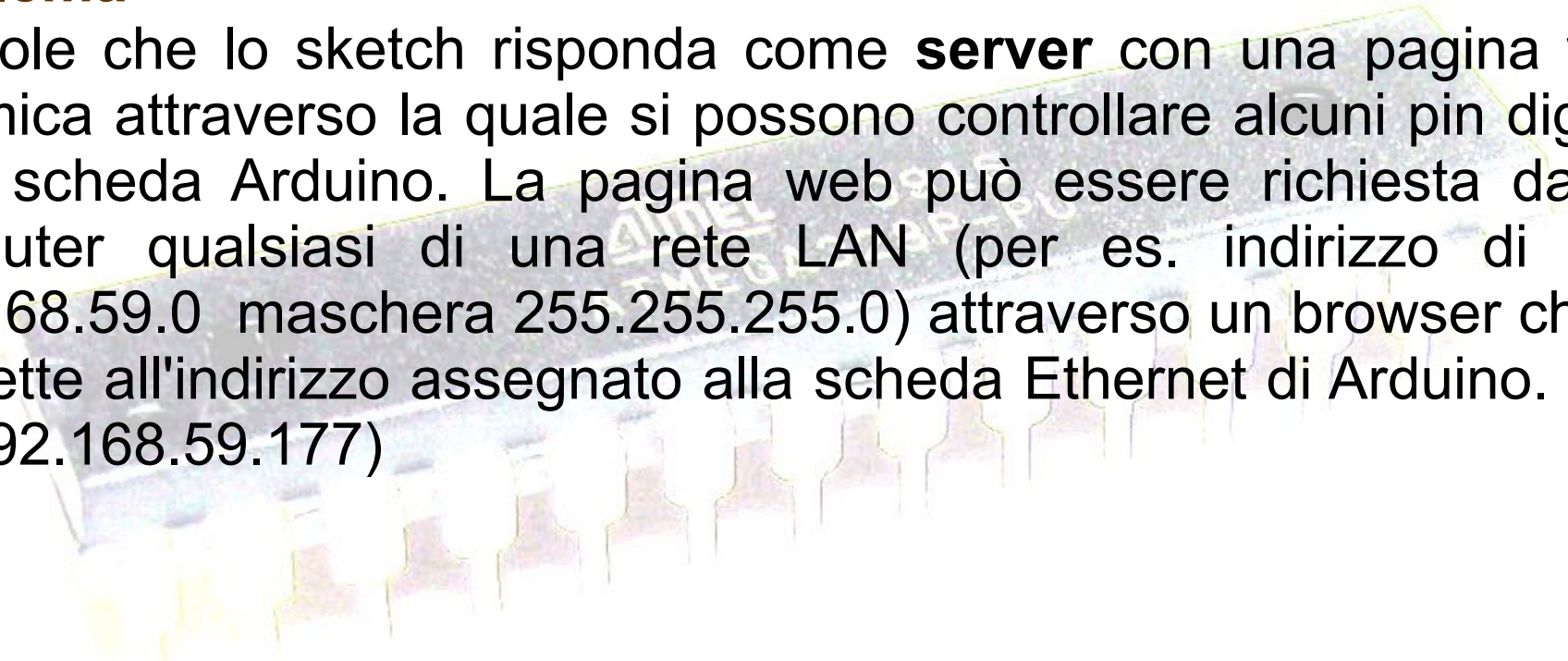
  client.println("</html>");
  break;
}
delay(1); // give the web browser time to receive the data
client.stop(); // close the connection:
Serial.println("client disconnected");
}
```



# Controllare alcuni pin digitali attraverso una pagina web

## Problema

Si vuole che lo sketch risponda come **server** con una pagina web dinamica attraverso la quale si possono controllare alcuni pin digitali della scheda Arduino. La pagina web può essere richiesta da un computer qualsiasi di una rete LAN (per es. indirizzo di rete 192.168.59.0 maschera 255.255.255.0) attraverso un browser che si connette all'indirizzo assegnato alla scheda Ethernet di Arduino. (per es. 192.168.59.177)



**Alla prima connessione la stringa inviata dal browser che si connette è del tipo  
GET / HTTP1.1**

In questo caso il programma su Arduino passa direttamente alla serie di print che inviano al browser il codice **html** che costituisce la pagina web. Da notare le prime due print e la riga vuota prima della pagina html.

```
client.println("HTTP/1.1 200 OK");  
client.println("Content-Type: text/html");  
client.println();
```

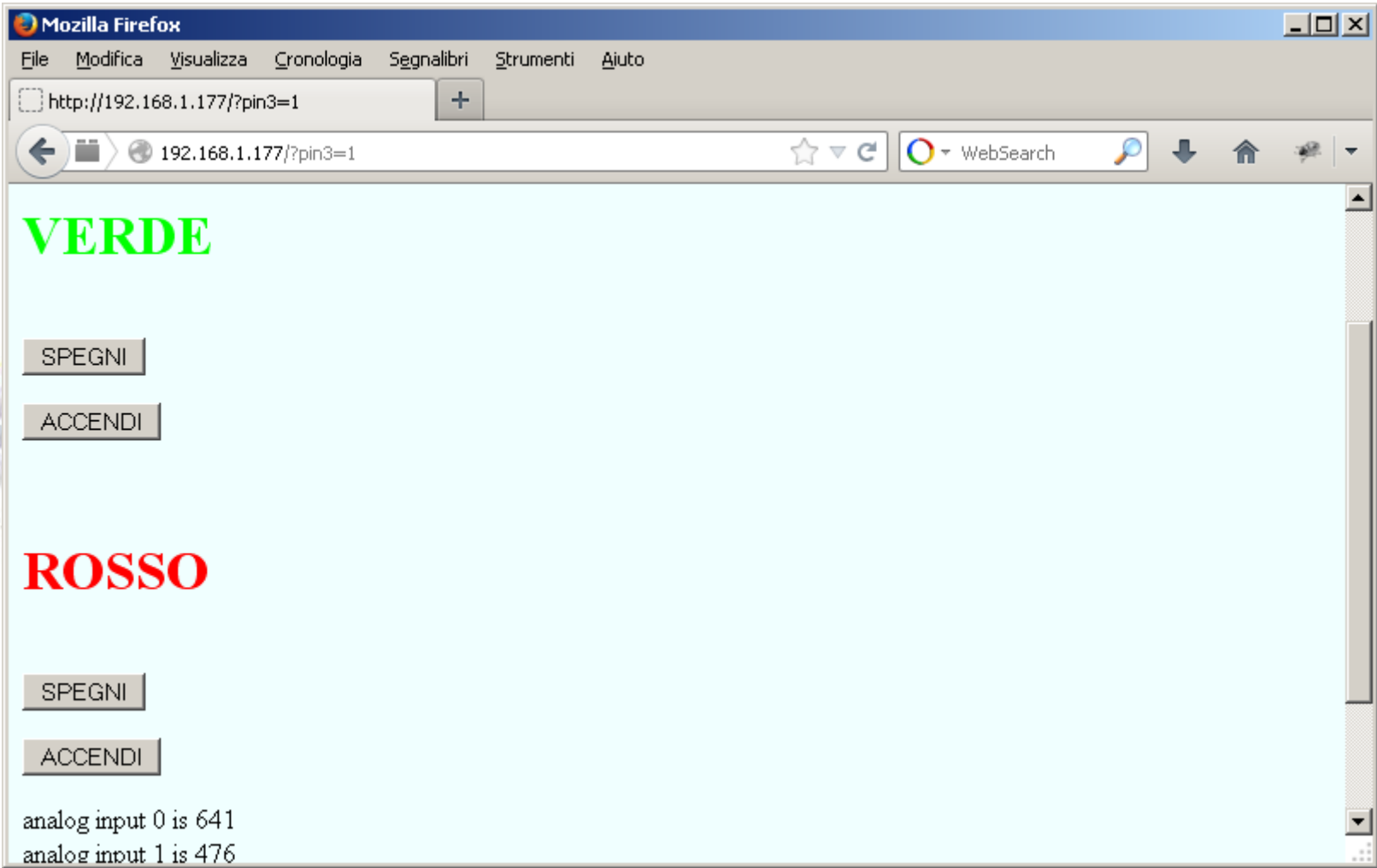
segue la pagina html che contiene dei form

... form

... form

**Nelle altre connessioni la stringa inviata con dei form è del tipo  
GET /?pin3=1 HTTP1.1**





**VERDE**

SPEGNI

ACCENDI

**ROSSO**

SPEGNI

ACCENDI

analog input 0 is 641  
analog input 1 is 476

```
EthernetServer server(80);  
server.begin();  
//viene creato un server che rimane in attesa sulla porta 80.
```

```
EthernetClient client = server.available();  
//quando un client si connette si ottiene l'oggetto client che  
rappresenta il browser che si collega e con il quale si dialoga.
```

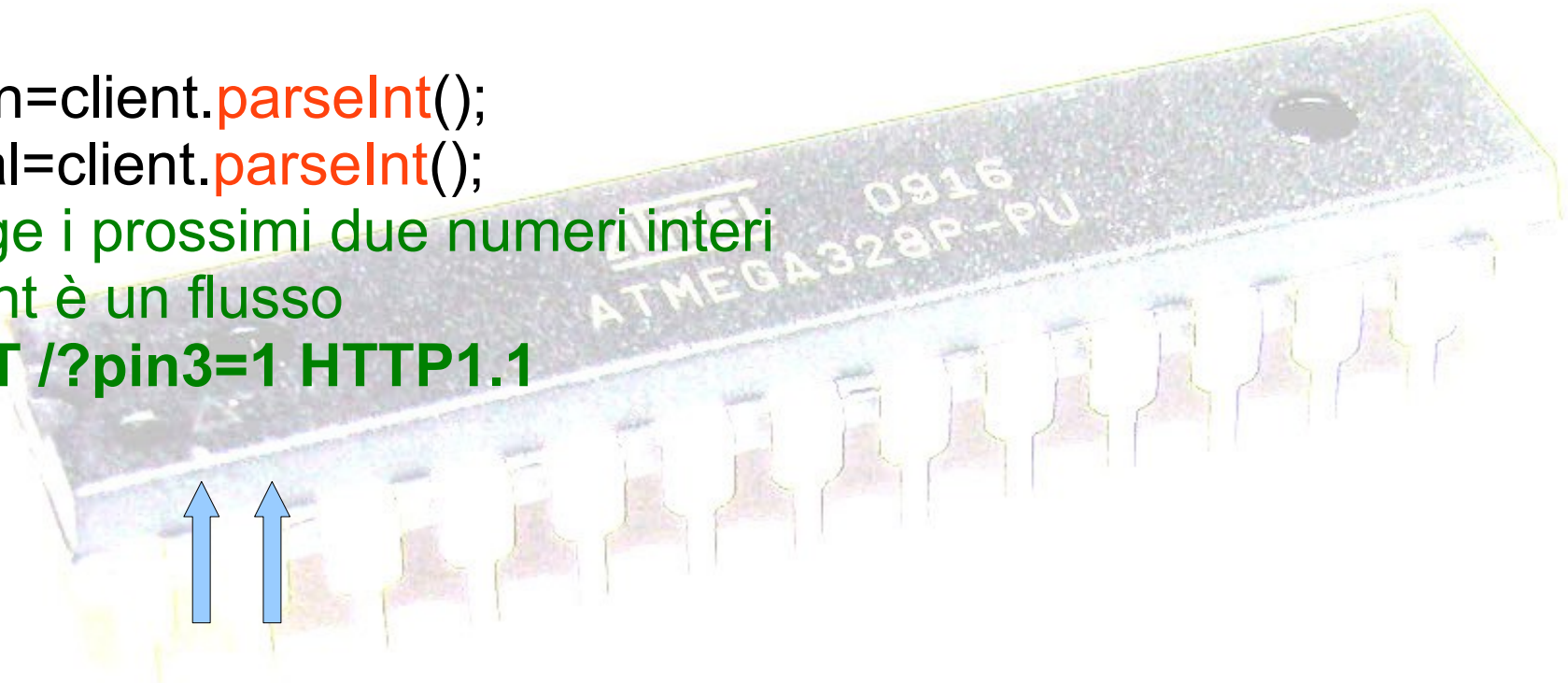
```
while(client.connected())  
// controlla che il server web sia connesso al client
```

```
if(client.available())  
controlla se sono disponibili dati
```

```
if(client.find("GET /" )  
//cerca nel flusso ricevuto la stringa "GET /"
```

```
while(client.findUntil("pin","\r\n")) {  
//cerca la stringa "pin" , termina con una riga vuota
```

```
int pin=client.parseInt();  
int val=client.parseInt();  
// legge i prossimi due numeri interi  
client è un flusso  
GET /?pin3=1 HTTP1.1
```



*due interi*

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

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

```
byte mac[]= {0x90,0xA2,0xDA,0x00,0xF9,0xA5};
```

```
byte ip[]= {192,168,1,177};
```

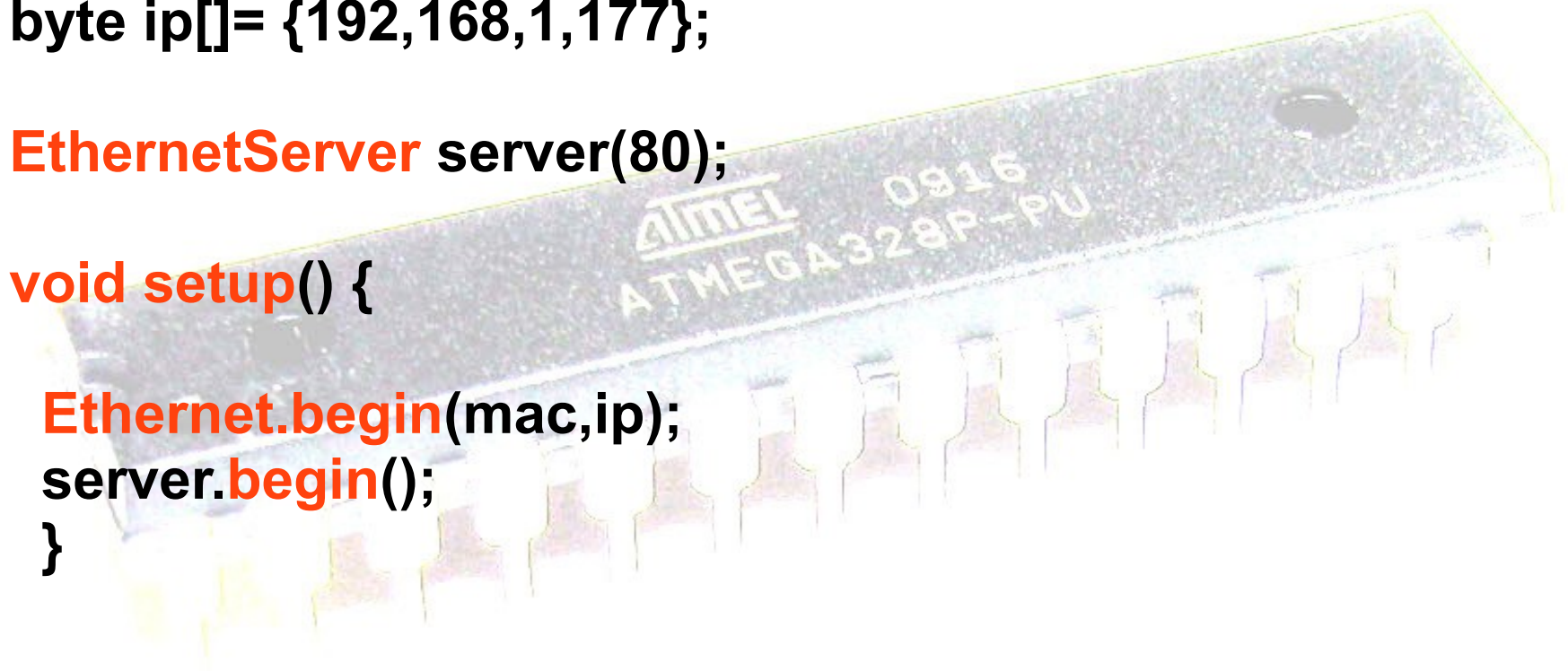
```
EthernetServer server(80);
```

```
void setup() {
```

```
  Ethernet.begin(mac,ip);
```

```
  server.begin();
```

```
}
```



```
void loop() {  
  EthernetClient client = server.available();  
  while(client.connected()){  
    if(client.available()){  
      if(client.find("GET /") ) {  
        while(client.findUntil("pin", "\r\n")) {  
          int pin=client.parseInt();  
          int valore=client.parseInt();  
          pinMode(pin,OUTPUT);  
          digitalWrite(pin,valore);  
        } //end while findUntil  
      } // end if client.find GET  
    }  
  }  
}
```

esecuzione della richiesta



## FORM e CSS

```
<head>
<style>
#rosso {color: #FF0000}
#verde {color: #00FF00}
</style>
</head>
```

```
<br>
<h1 id='rosso'> ROSSO </h1> <br>
<form action='/' >
<input type='hidden' name='pin3' value='0'>
<input type='submit' size='10' value='SPEGNI'>
</form>
```

form




*pulsante*

`action = '...'` indica la pagina web da richiedere quando viene cliccato il pulsante submit  
la `/` indica la “stessa pagina html”

Dal browser viene inviata la stringa

**GET /?pin3=0 HTTP1.1**

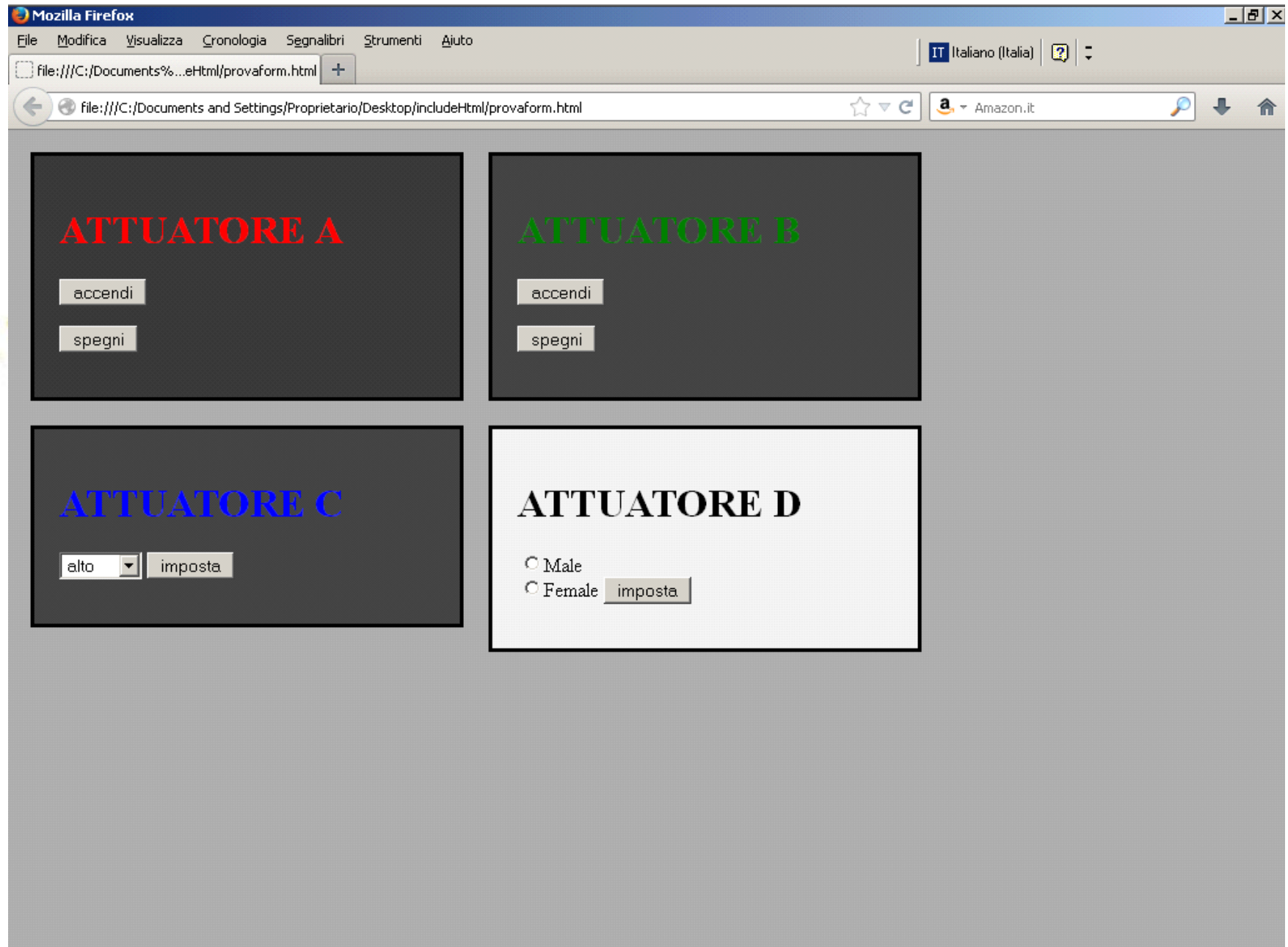


```
delay(1000); // è necessario
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println(); // riga vuota
client.println("<html>"); // pagina richiesta
client.println("<head>");
client.println("<style>");
client.println("#rosso {color: #FF0000}");
client.println("#verde {color: #00FF00}");
client.println("</style>");
client.println("</head>");
client.println("<body bgcolor='azure'>");
client.println("<h2>");
client.println("Viene inviata una richiesta ad Arduino");
client.println("</h2>");
client.println("<br>");
client.println("<h1 id='verde'> VERDE </h1> <br>");
client.println(" <form action='/' >");
client.println("<input type='hidden' name='pin2' value='0'>");
client.println("<input type='submit' size='10' value='SPEGNI'>");
client.println(" </form>");
client.println(" <form action='/' >");
client.println("<input type='hidden' name='pin2' value='1'>");
client.println("<input type='submit' size='10' value='ACCENDI'>");
client.println(" </form>");
```

```
client.println("<br>");
client.println("<h1 id='rosso'> ROSSO </h1> <br>");
client.println(" <form action='/' >");
client.println("<input type='hidden' name='pin3' value='0'>");
client.println("<input type='submit' size='10' value='SPEGNI'>");
client.println(" </form>");
client.println(" <form action='/' >");
client.println("<input type='hidden' name='pin3' value='1'>");
client.println("<input type='submit' size='10' value='ACCENDI'>");
client.println(" </form>");
//fine richiesta
for(int i=0;i< 6 ;i++) {
  client.print("analog input ");
  client.print(i);
  client.print(" is ");
  client.print(analogRead(i));
  client.print("<br />");
}
client.println("</body>");
client.println("</html>");
break; // dalla while principale ,vuole disconnettersi
} //end if client.available
} //end while client.connected
delay(1);
client.stop();
} //loop
```



# ***pagina web per controllo attuatori attraverso: pulsanti, selezione, radio button***



```
<html>
```

```
<head>
```

```
<style>
```

```
body {background: #AAAAAA;}
```

```
#iAon { color: red; }
```

```
#iBon { color: green; }
```

```
#iCon { color: blue; }
```

```
#iDon { color: #0; }
```

```
#IA { width:300px; margin:10px;border:3px solid; padding:20px;float: left;  
background: #333333;}
```

```
#IB { width:300px ;margin:10px; border:3px solid; padding:20px;float: left;  
background: #444444;}
```

```
#IC { width:300px ;margin:10px; border:3px solid; padding:20px;float: left;  
background: #444444;}
```

```
#ID { width:300px ;margin:10px; border:3px solid; padding:20px;float: left;  
background: #EEEEEE;}
```

```
</style>
```

```
</head>
```

CSS



```
<body>
```

```
<div id="IA">
```

```
<form>
```

```
<h1 id=iAon> ATTUATORE A </h1>
```

```
<input type="hidden" name="pin3" value="1">
```

```
<input type="submit" action="provaform.html" value="accendi">
```

```
</form>
```

```
<form>
```

```
<input type="hidden" name="pin3" value="0">
```

```
<input type="submit" action="provaform.html" value="spegni">
```

```
</form>
```

```
</div>
```

```
<div id="IB">
```

```
<form>
```

```
<h1 id=iBon> ATTUATORE B </h1>
```

```
<input type="hidden" name="pin2" value="1">
```

```
<input type="submit" action="provaform.html" value="accendi">
```

```
</form>
```

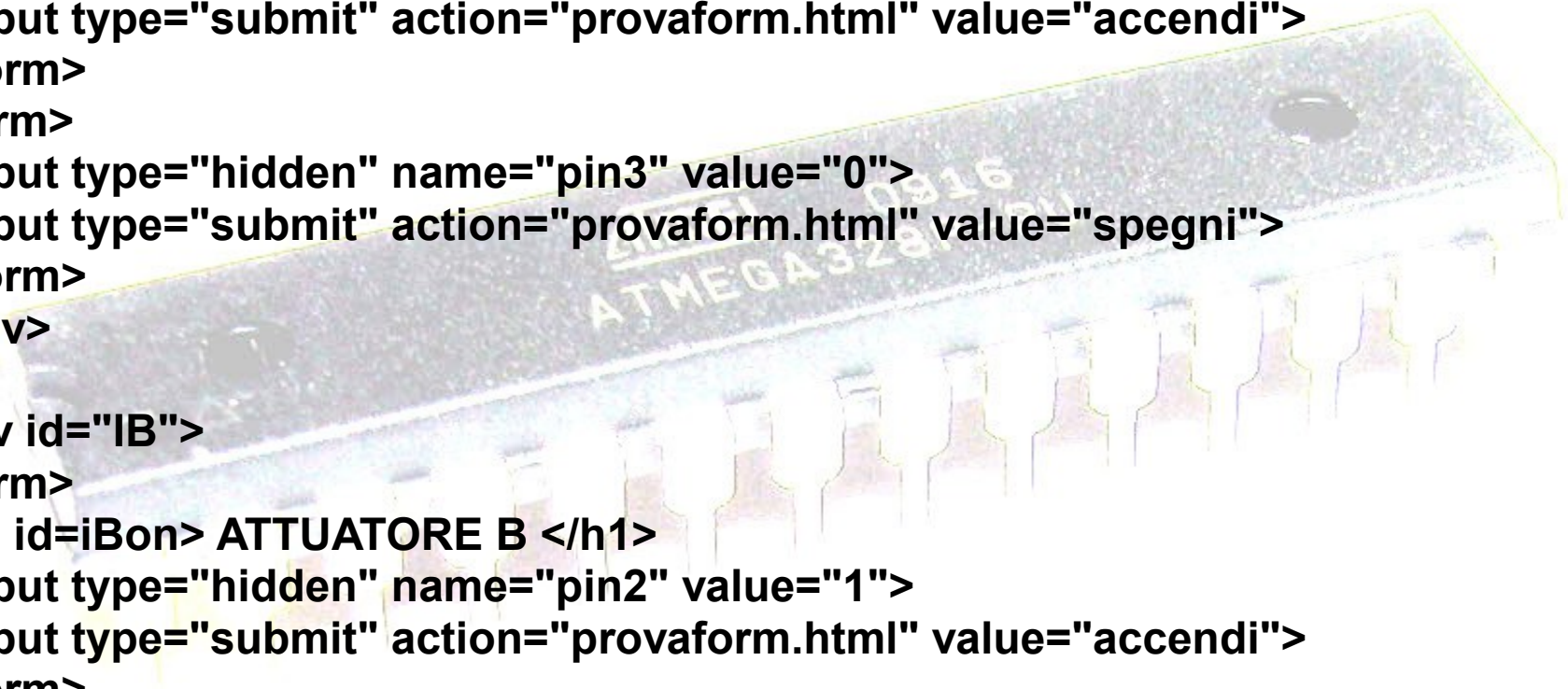
```
<form>
```

```
<input type="hidden" name="pin2" value="0">
```

```
<input type="submit" action="provaform.html" value="spegni">
```

```
</form>
```

```
</div>
```



```
<div id="IC">
<form>
<h1 id=iCon> ATTUATORE C </h1>
<select name="selez" size="1">
<option value="255">alto </option>
<option value="125">medio </option>
<option value="20"> basso </option>
```

**option**



```
</select>
```

```
<input type="submit" action="provaform.html" value="imposta">
</form>
</div>
```

```
<div id="ID">
<form>
<h1 id=iDon> ATTUATORE D </h1>
<input type="radio" name="sex" value="male">Male<br>
<input type="radio" name="sex" value="female">Female
```

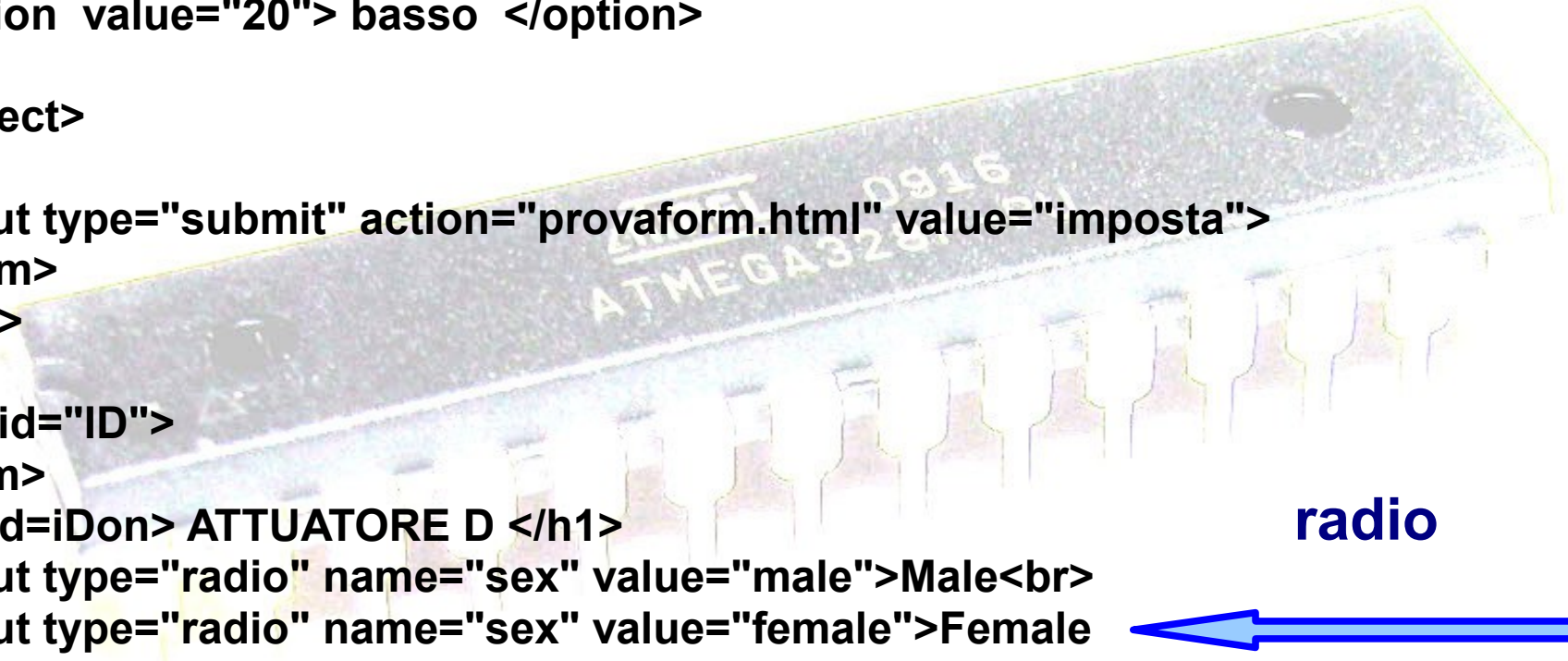
**radio**



```
<input type="submit" action="provaform.html" value="imposta">
</form>
```

```
</div>
</body>
```

```
</html>
```



# Controllo del funzionamento di un semaforo con una pagina web

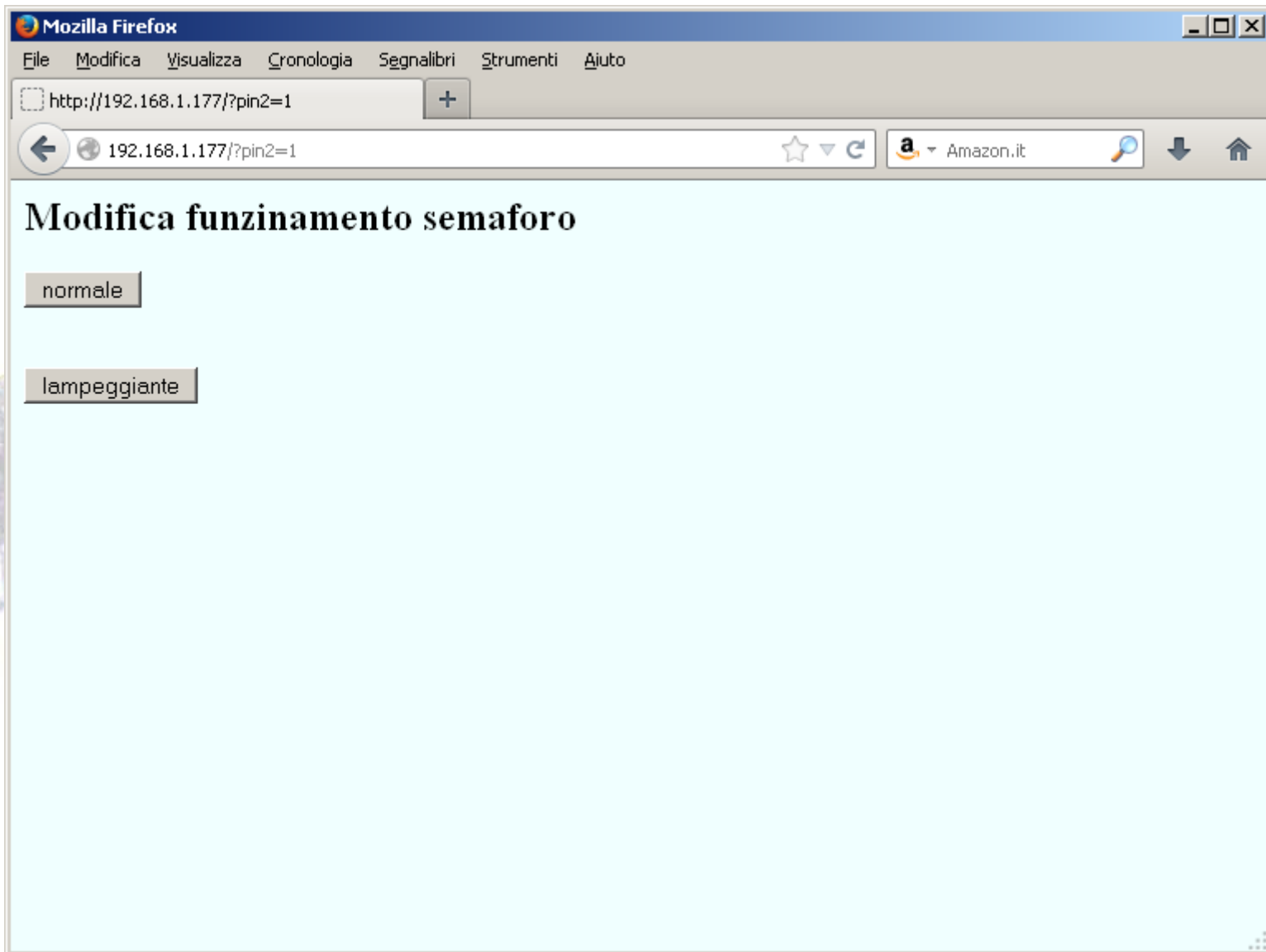
## Problema

Si vuole che lo sketch risponda come server web per il controllo del funzionamento di un semaforo (rosso, verde, giallo o giallo lampeggiante)

## Soluzione

Non è possibile utilizzare la funzione `delay` per tempi lunghi se si vuole rispondere in tempo reale alla richiesta di connessione da parte di un browser per l'eventuale cambiamento del funzionamento. Viene utilizzata la funzione `millis()` associata ad un ciclo `while`.

```
void attendiEcontrolla(int tempo) {  
    long attuali=millis();  
    while(millis() < (attuali+tempo)){  
        EthernetClient client = server.available(); // controllo connessione  
        ...  
    }  
}
```



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

```
const int verde=2;
const int giallo=13;
const int rosso=3;
byte mac[]={0x90,0xA2,0xDA,0x00,0xF9,0xA5};
byte ip[]={192,168,1,177};
boolean rossoVerdeGiallo=true; // stabilisce il tipo di funzionamento
```

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

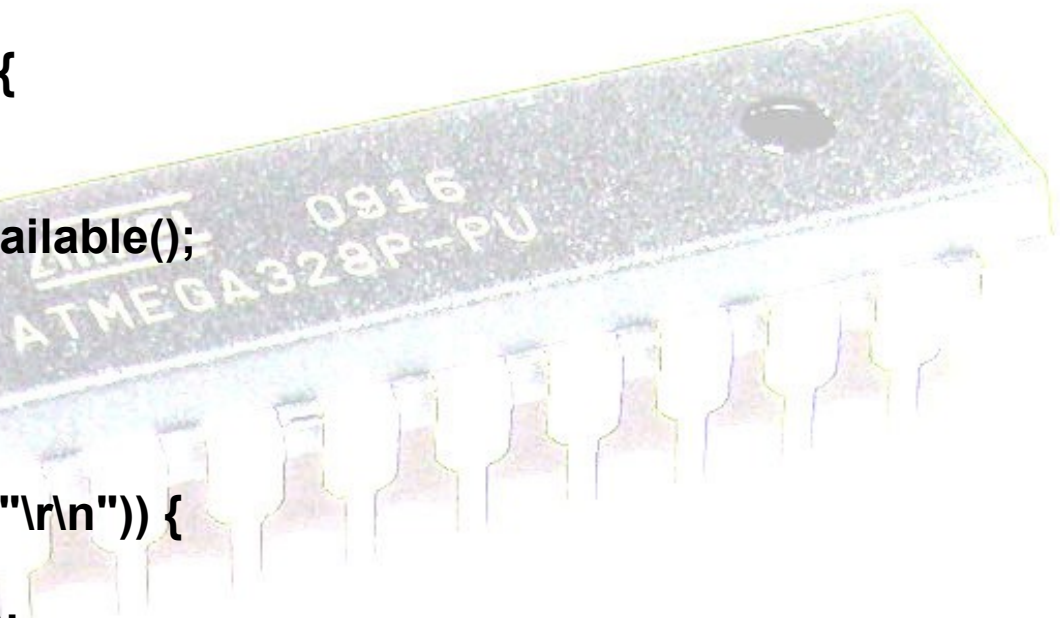
```
void setup() {
  pinMode(verde,OUTPUT);
  pinMode(giallo,OUTPUT);
  pinMode(rosso,OUTPUT);

  Ethernet.begin(mac,ip);
  server.begin();
}
```



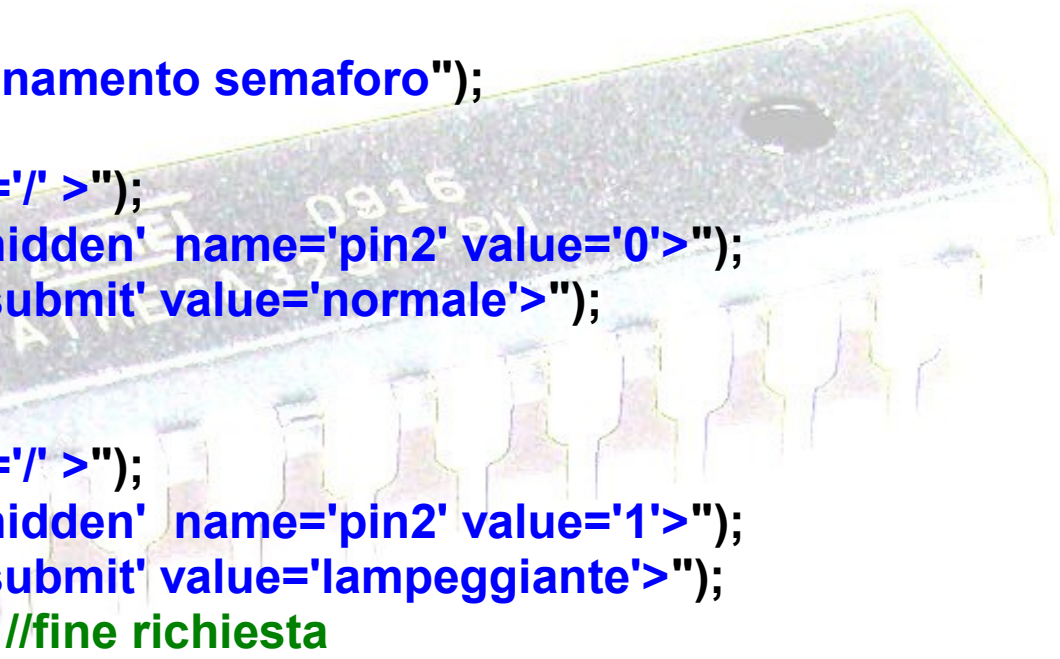
```
void loop() {
  if(rossoVerdeGiallo)
    normale();
  else
    gialloL();
}

void attendiEcontrolla(int tempo) {
  long attuali=millis();
  while(millis() < (attuali+tempo)){
    EthernetClient client = server.available();
    if (client == true) {
      while(client.connected()){
        if(client.available()){
          if(client.find("GET /") ) {
            while(client.findUntil("pin","\r\n")) {
              int pin=client.parseInt();
              int valore=client.parseInt();
              if(valore == 0) {
                rossoVerdeGiallo=true;
                break;
              }
            }
            else{
              rossoVerdeGiallo=false;
              break;
            }
          } //end while findUntil
        } // end client.find GET
      }
    }
  }
}
```



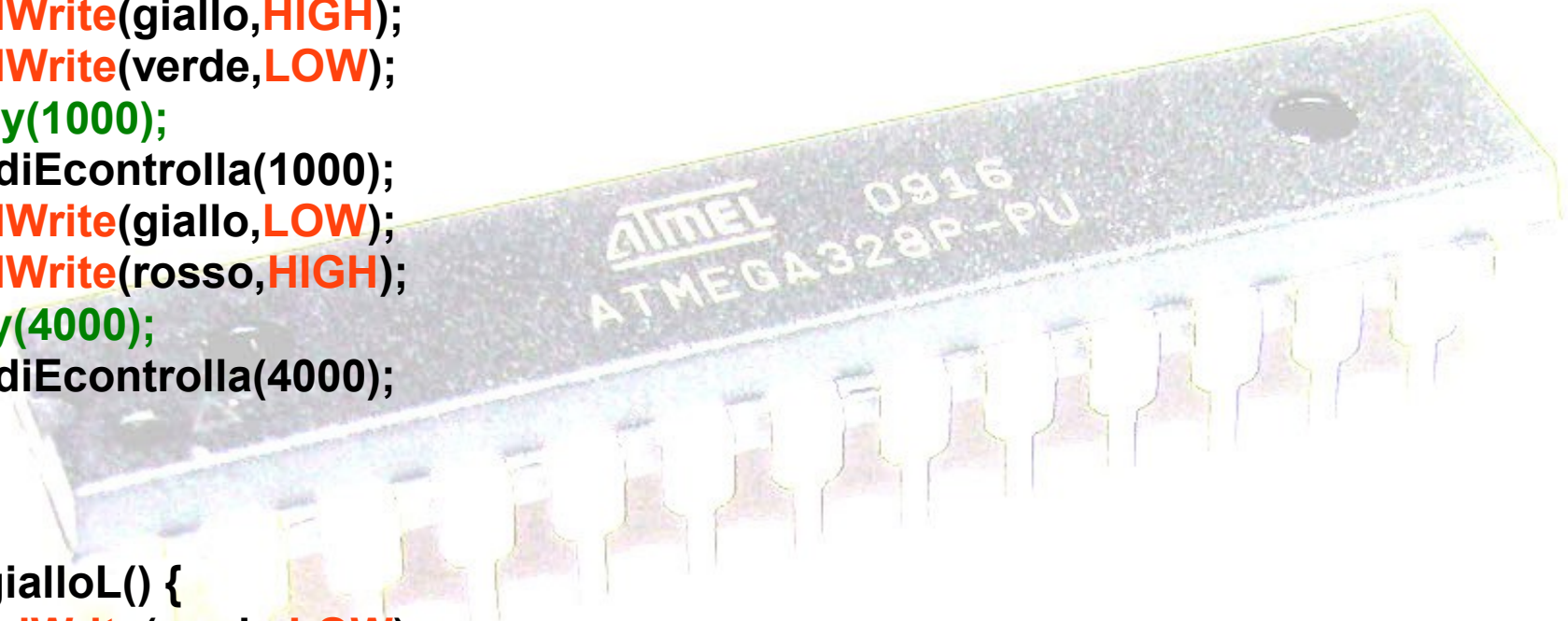


```
client.println("HTTP/1.1 200 OK");
client.println("Content-Type: text/html");
client.println();
//pagina richiesta
client.println("<html>");
client.println("<body bgcolor='azure'>");
client.println("<h2>");
client.println("Modifica funzinamento semaforo");
client.println("</h2>");
client.println(" <form action='/' >");
client.println("<input type='hidden' name='pin2' value='0'>");
client.println("<input type='submit' value='normale'>");
client.println("</form>");
client.println("<br>");
client.println(" <form action='/' >");
client.println("<input type='hidden' name='pin2' value='1'>");
client.println("<input type='submit' value='lampeggiante'>");
client.println("</form>"); //fine richiesta
client.println("</body>");
client.println("</html>");
break; // dalla while principale vuole disconnettersi
} //end if client.available
} //end while client.connected
delay(1);
client.stop();
```



```
void normale(){
  digitalWrite(rosso,LOW);
  digitalWrite(verde,HIGH);
  digitalWrite(giallo,LOW);
  // delay(4000);
  attendiEcontrolla(4000);
  digitalWrite(giallo,HIGH);
  digitalWrite(verde,LOW);
  // delay(1000);
  attendiEcontrolla(1000);
  digitalWrite(giallo,LOW);
  digitalWrite(rosso,HIGH);
  //delay(4000);
  attendiEcontrolla(4000);
}
```

```
void gialloL() {
  digitalWrite(verde,LOW);
  digitalWrite(rosso,LOW);
  digitalWrite(giallo,HIGH);
  //delay(500);
  attendiEcontrolla(4000);
  digitalWrite(giallo,LOW);
  //delay(500);
  attendiEcontrolla(4000);
}
```



# Comunicazione in rete tra Arduino e PC

## Problema

Si vuole che lo sketch su Arduino rilevi la posizione di un potenziometro attraverso un ingresso analogico e, funzionando da server, lo invii su richiesta ad uno sketch di processing su Pc.

La richiesta di connessione viene fatta da un computer qualsiasi di una rete LAN ( per esempio indirizzo di rete 192.168.59.0 maschera 255.255.255.0) sul quale gira uno sketch di processing che si connette all'indirizzo assegnato alla scheda Ethernet di Arduino. (per es. 192.168.59.177 porta 2300).

La posizione del potenziometro, letta da Arduino, determina l'inclinazione di un segmento, gestito da processing su pc, che simula le ali di un aereo.

## Soluzione

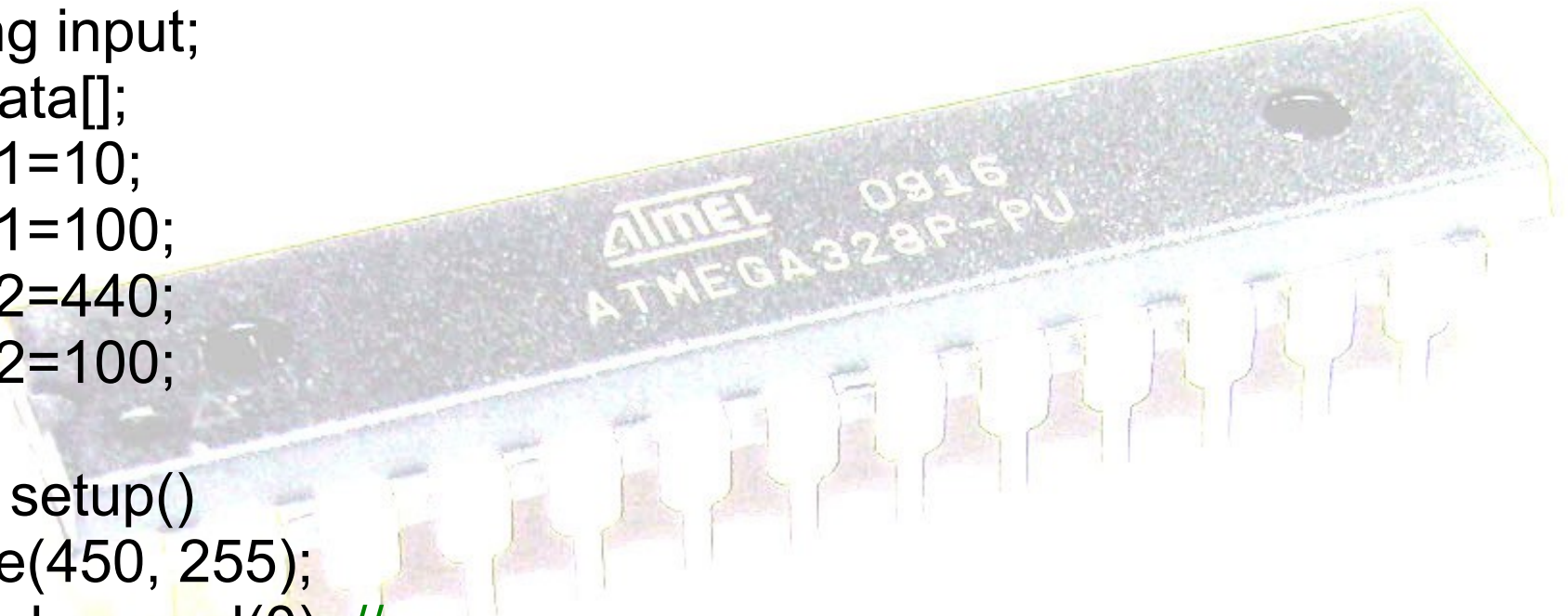
Processing si connette con:

```
c = new Client(this, "192.168.59.177", 2300);
```

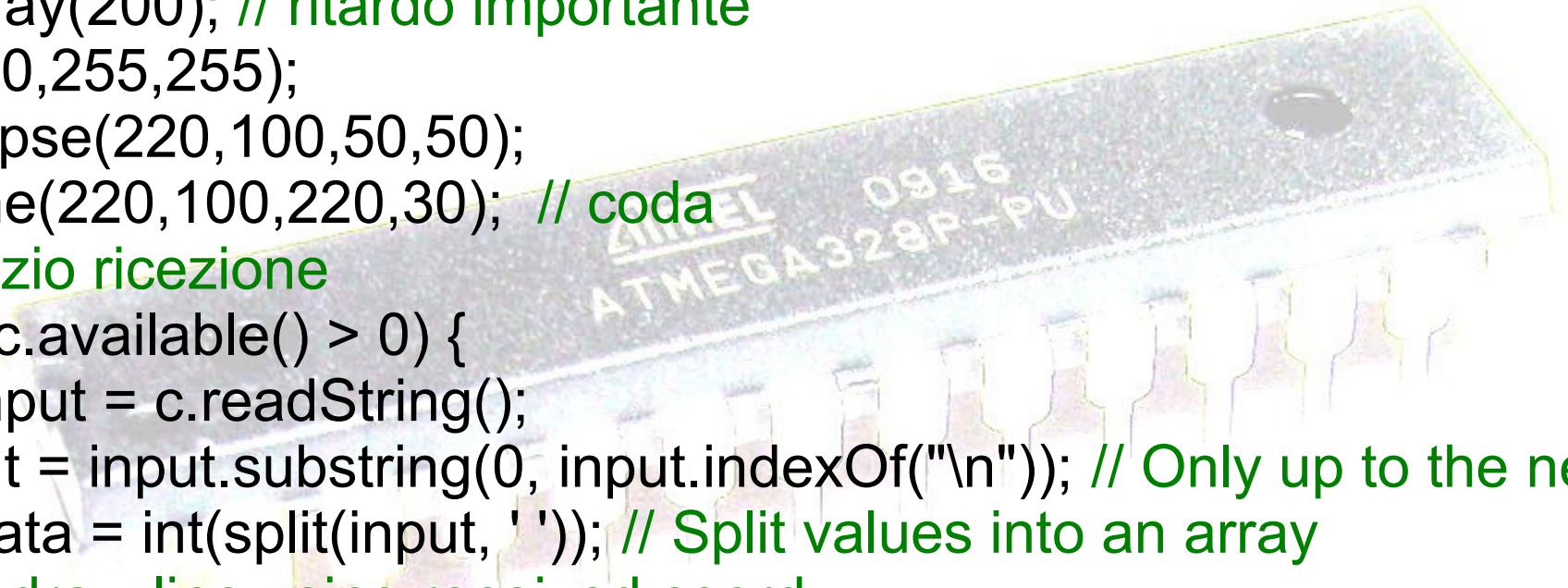
# Codice per processing (client)

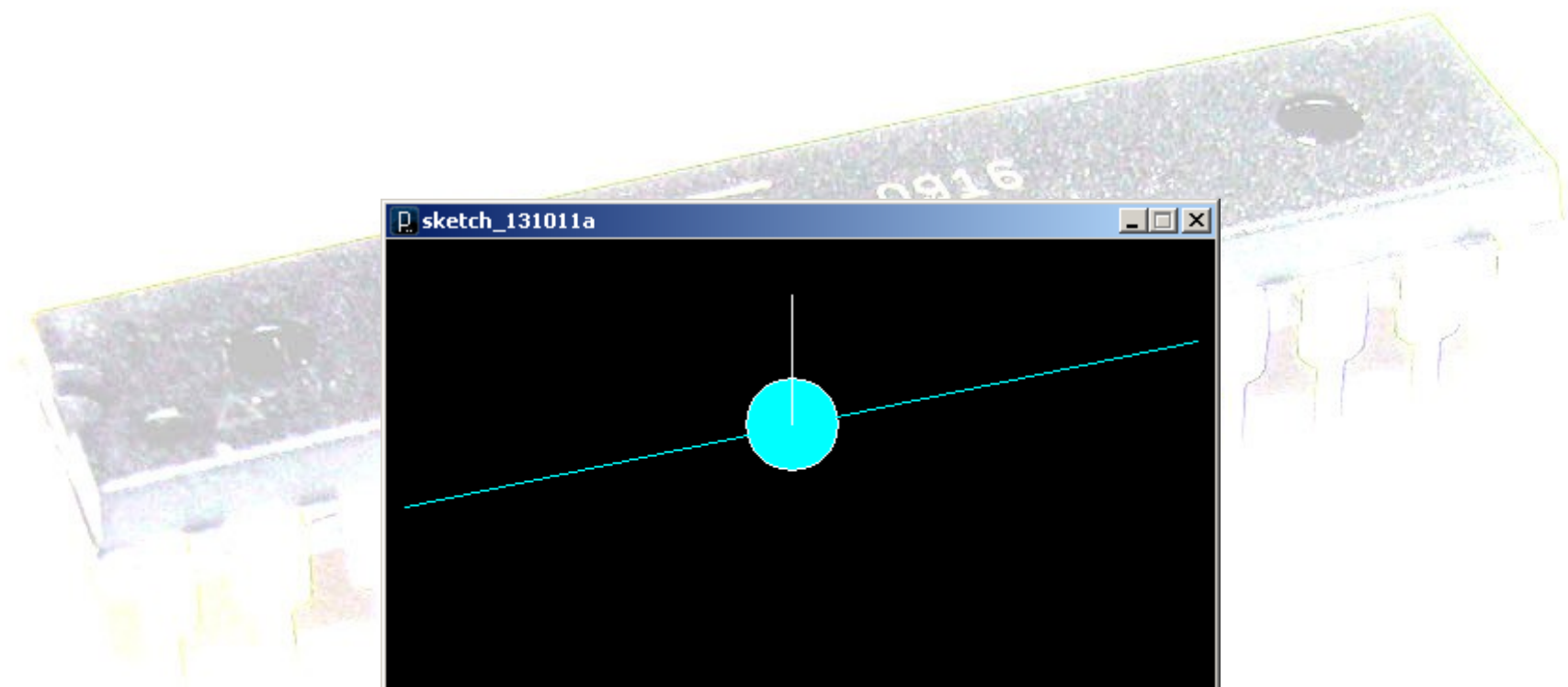
```
import processing.net.*;
Client c;
String input;
int data[];
int x1=10;
int y1=100;
int x2=440;
int y2=100;

void setup()
{ size(450, 255);
  background(0); // nero
  stroke(0);
  // Connect to the server's IP address and port
  c = new Client(this, "192.168.59.177", 2300);
}
```



```
void draw()
{ stroke(255);
  c.write(x1+" "+y1+" "+x2+" "+y2+"\n"); // invia al server
  background(0);
  delay(200); // ritardo importante
  fill(0,255,255);
  ellipse(220,100,50,50);
  Line(220,100,220,30); // coda
// inizio ricezione
  if (c.available() > 0) {
    input = c.readString();
input = input.substring(0, input.indexOf("\n")); // Only up to the newline
    data = int(split(input, ' ')); // Split values into an array
    // draw line using received coord
    stroke(0,255,255);
    line(data[0], data[1], data[2], data[3]);
    // print(input);
    print(" ");
  }
}
```

A photograph of an ATMEGA328P-PU microcontroller chip, which is a common component used in Arduino Uno boards. The chip is a small, rectangular integrated circuit with a black surface and gold-colored pins. The text 'ATMEGA328P-PU' and '0916' are visible on the top surface of the chip.



## Codice per Arduino

**/\* A simple server Using an Arduino Wiznet Ethernet  
Ethernet shield attached to pins 10, 11, 12, 13**

**Analog inputs attached to pins A0**

**\*/**

**#include <SPI.h>**

**#include <Ethernet.h>**

**// Enter a MAC address and IP address for your controller  
below.**

**// The IP address will be dependent on your local network.**

**// gateway and subnet are optional:**

**byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };**

**IPAddress ip(192,168,59, 177);**

**IPAddress gateway(192,168,59, 254);**

**IPAddress subnet(255, 255, 255, 0);**

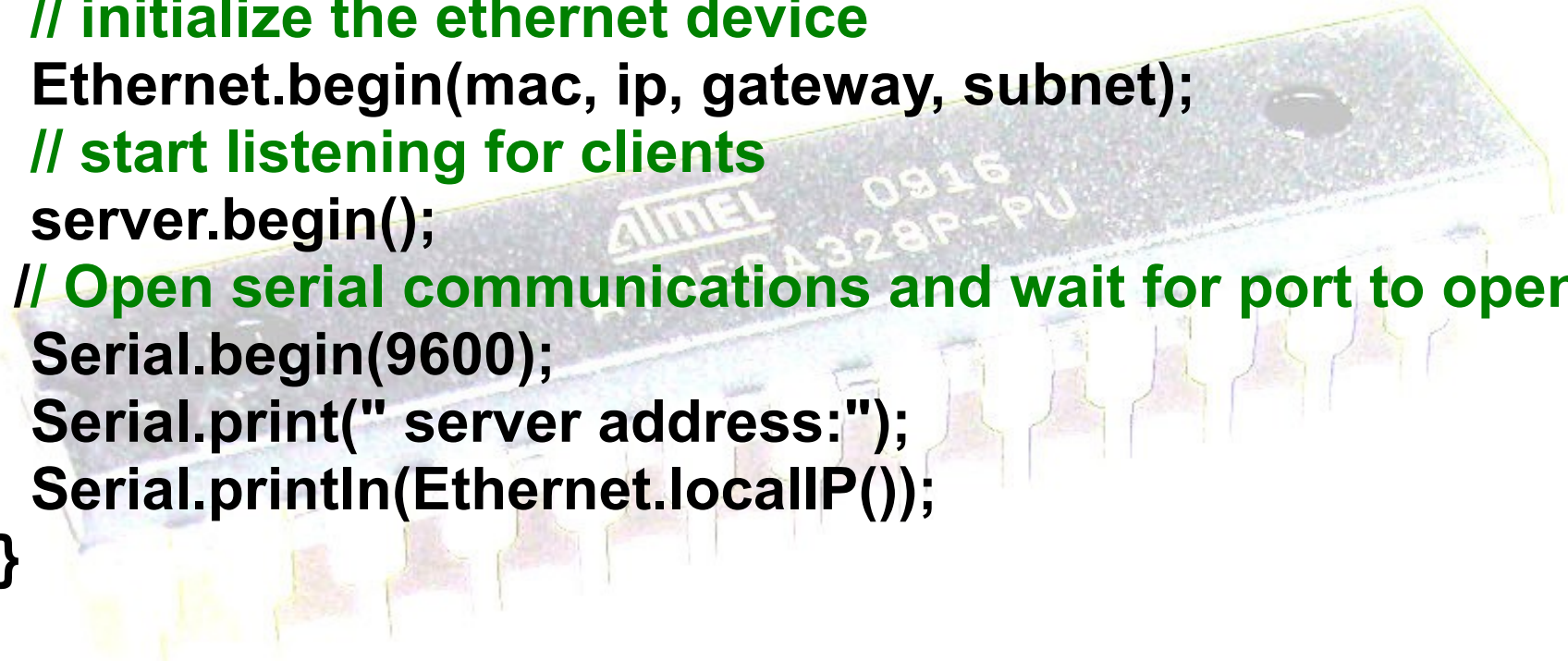
**int x1,x2,y1,y2;**

**int valore;**



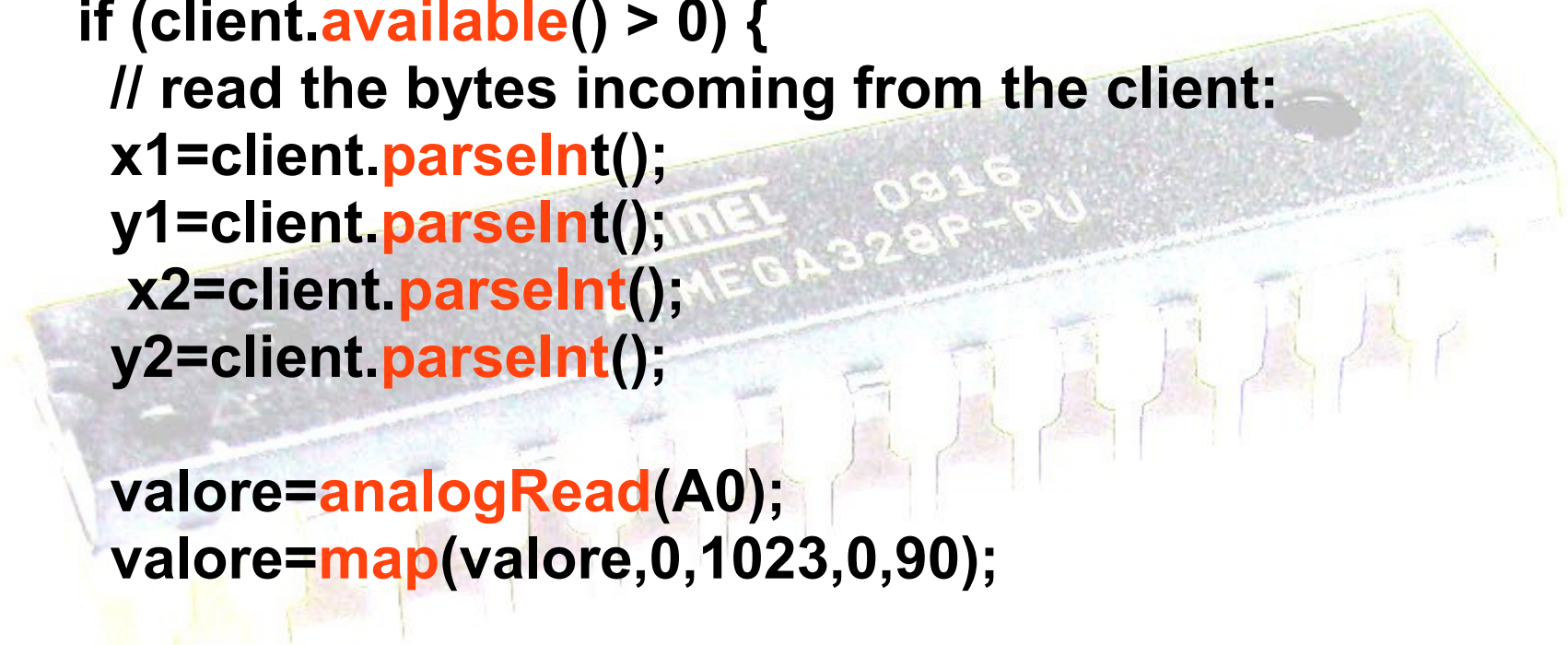
```
EthernetServer server(2300);
```

```
void setup() {  
  // initialize the ethernet device  
  Ethernet.begin(mac, ip, gateway, subnet);  
  // start listening for clients  
  server.begin();  
  // Open serial communications and wait for port to open:  
  Serial.begin(9600);  
  Serial.print(" server address:");  
  Serial.println(Ethernet.localIP());  
}
```



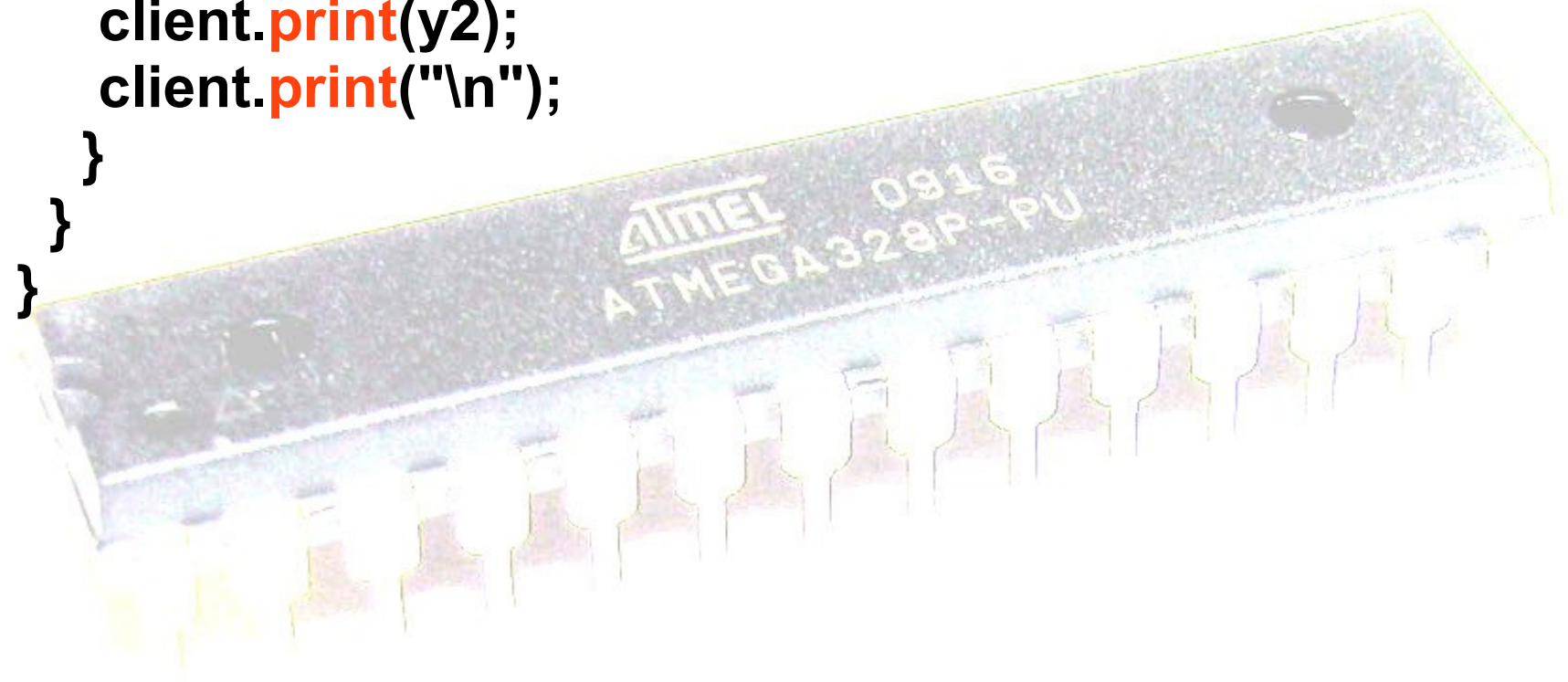


```
void loop() {  
  // wait for a new client:  
  EthernetClient client = server.available();  
  if (client) {  
    if (client.available() > 0) {  
      // read the bytes incoming from the client:  
      x1=client.parseInt();  
      y1=client.parseInt();  
      x2=client.parseInt();  
      y2=client.parseInt();  
  
      valore=analogRead(A0);  
      valore=map(valore,0,1023,0,90);  
  
      y1=y1-(valore-45);  
      y2=y2+(valore-45);  
      client.print(x1);  
      client.print(" ");  
      client.print(y1);  
      client.print(" ");  
    }  
  }  
}
```



```
client.print(x2);  
client.print(" ");  
client.print(y2);  
client.print("\n");
```

```
}  
}  
}
```



# Usare il protocollo UDP

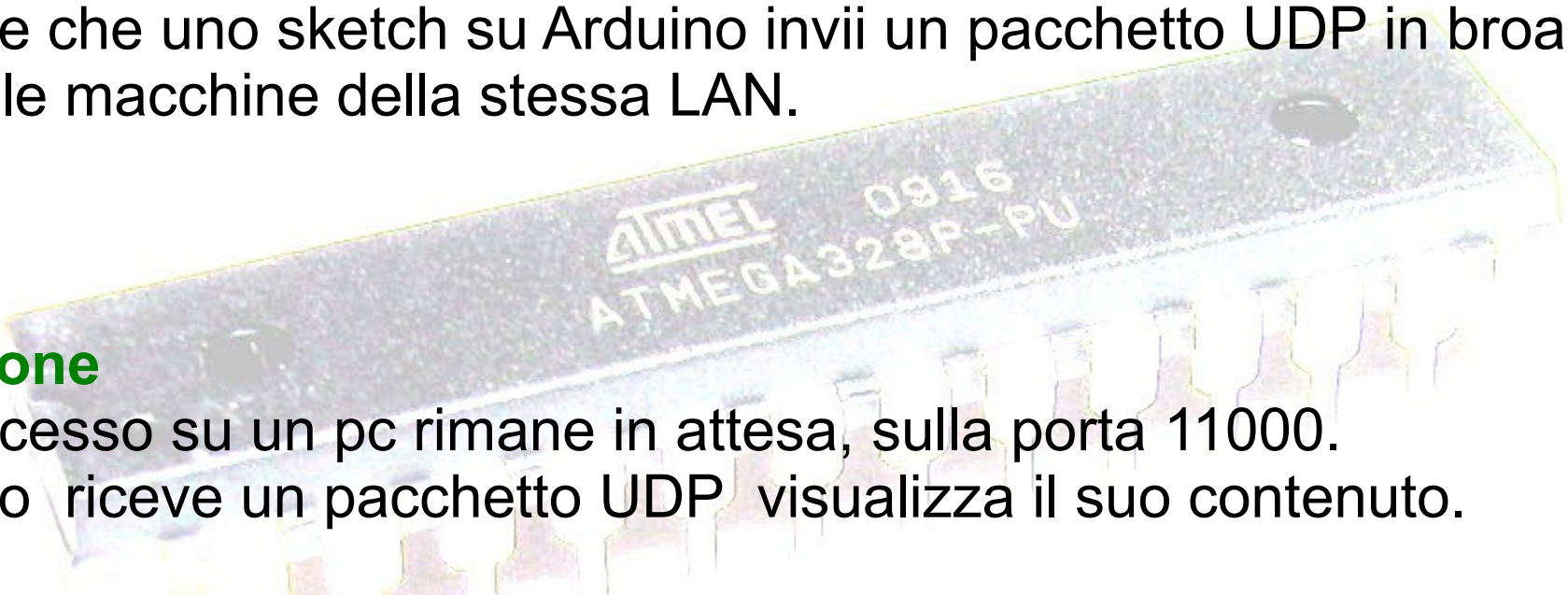
## Problema

Si vuole che uno sketch su Arduino invii un pacchetto UDP in broadcast a tutte le macchine della stessa LAN.

## Soluzione

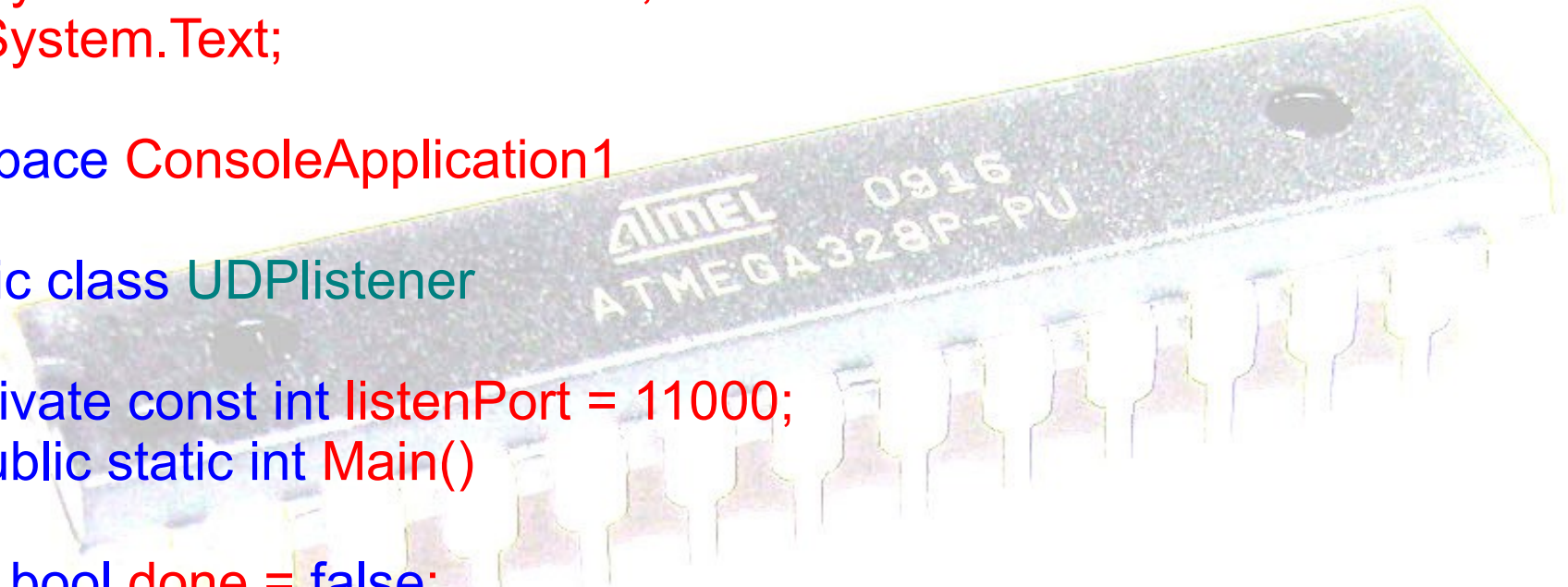
Un processo su un pc rimane in attesa, sulla porta 11000. Quando riceve un pacchetto UDP visualizza il suo contenuto.

Si può scrivere il programma con un qualsiasi linguaggio che supporti UDP. L'esempio seguente è in C#

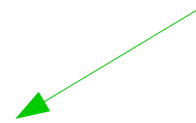


```
using System;
using System.Net;
using System.Net.Sockets;
using System.Collections.Generic;
using System.Text;

namespace ConsoleApplication1
{
    public class UDPListener
    {
        private const int listenPort = 11000;
        public static int Main()
        {
            bool done = false;
            UdpClient listener = new UdpClient(listenPort);
            IPEndPoint groupEP = new IPEndPoint(IPAddress.Any, listenPort);
            string received_data;
            byte[] receive_byte_array; // i dati di un pacchetto
```

An image of an ATMEL ATMEGA328P-PU microcontroller chip. The chip is a small, rectangular integrated circuit with a dark grey surface and a gold-colored lead frame. The top surface of the chip is printed with the ATMEL logo, the part number 'ATMEGA328P-PU', and the date code '0916'. The chip is shown from a slightly elevated perspective, highlighting its 28-pin DIP package.

Si blocca fino alla ricezione di un datagram



```
try
{
    while (!done)
    {
        receive_byte_array = listener.Receive(ref groupEP);
        Console.WriteLine("Received a broadcast from {0}", groupEP.ToString());
        received_data = Encoding.ASCII.GetString(receive_byte_array, 0,
receive_byte_array.Length);
        Console.WriteLine("data follow \n{0}", received_data);
    }
}
catch (Exception e)
{
    Console.WriteLine(e.ToString());
}
listener.Close();
return 0;
}
}
```

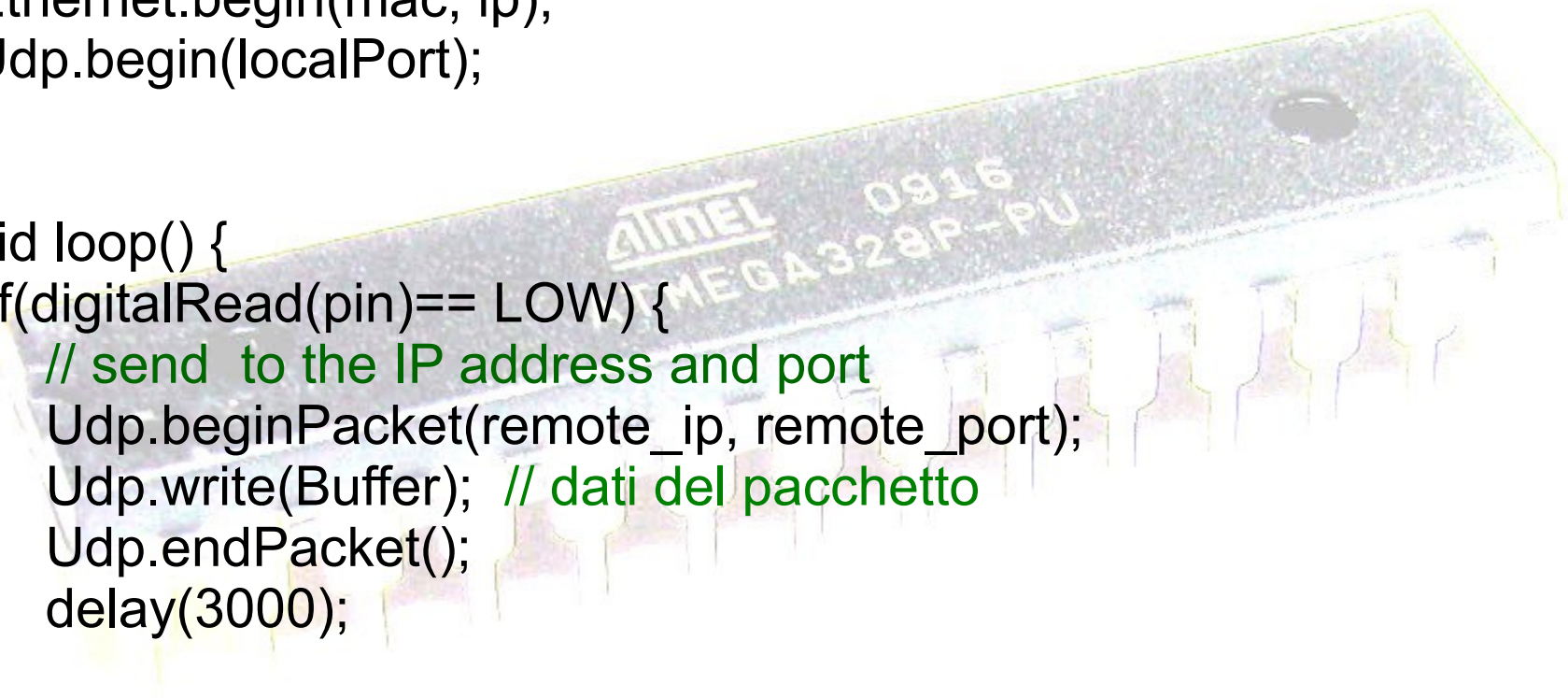
```
#include <SPI.h>          // needed for Arduino versions later than 0018
#include <Ethernet.h>
#include <EthernetUdp.h>  // UDP library

// Enter a MAC address and IP address for your controller below.
// The IP address will be dependent on your local network:
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192, 168, 1, 177);
IPAddress remote_ip(192,168,1,255); // broadcast su 192.168.1.0
unsigned int localPort = 8888;     // local port to listen
unsigned int remote_port= 11000;
const int pin=4;
char Buffer[] = "attenzione bottone pigiato"; // a string to send

// An EthernetUDP instance to let us send and receive packets
EthernetUDP Udp;
```

```
void setup() {  
  pinMode(pin, INPUT_PULLUP);  
  // start the Ethernet and UDP:  
  Ethernet.begin(mac, ip);  
  Udp.begin(localPort);  
}
```

```
void loop() {  
  if(digitalRead(pin)== LOW) {  
    // send to the IP address and port  
    Udp.beginPacket(remote_ip, remote_port);  
    Udp.write(Buffer); // dati del pacchetto  
    Udp.endPacket();  
    delay(3000);  
  }  
}
```



```
file:///C:/Users/Sergio/Documents/Visual Studio 2005/Projects/UDPricevitore/UDPricevitore/bin/De...
Received a broadcast from 192.168.1.177:8888
data follow
attenzione bottone pigiato
Received a broadcast from 192.168.1.177:8888
data follow
attenzione bottone pigiato
Received a broadcast from 192.168.1.177:8888
data follow
attenzione bottone pigiato
-
```