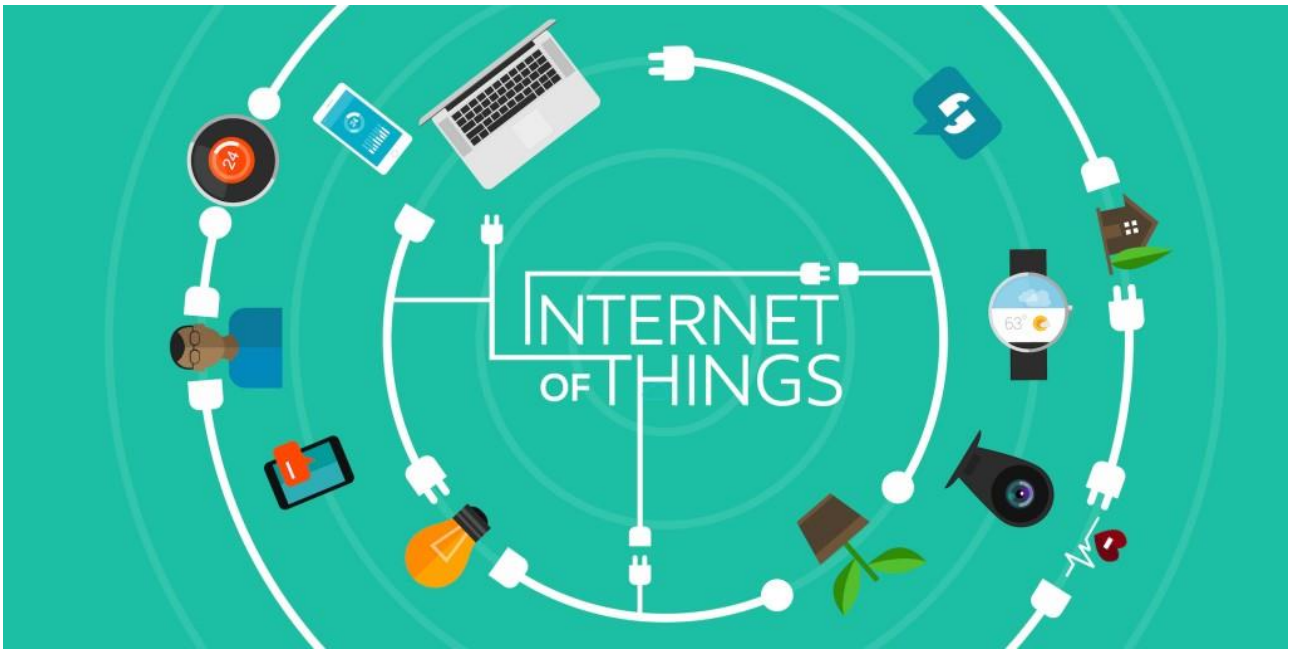
IIS *"Ettore Majorana"*
Via A. De Gasperi, 6 - 20811 Cesano Maderno (MB)

Internet of Things



Tesina di:

Butera Simone

Classe 5ªTL

Anno scolastico 2015/2016

Indice

Introduzione

Il tema che ho deciso di proporre è l' "Internet of Things" , in italiano Internet delle cose o Internet degli oggetti, poiché è chiaramente una delle invenzioni più importanti e potenti di tutta la storia dell'uomo.

Questa innovazione cambierà tutto, compresi noi stessi. Potrà sembrare un'affermazione azzardata, ma basta pensare all'impatto che Internet ha già avuto su istruzione, comunicazioni, economia, scienza, amministrazione pubblica e sull'umanità.

L'Internet delle cose rappresenta la prossima evoluzione di Internet, con un notevole miglioramento della capacità di raccogliere, analizzare e distribuire dati convertibili in informazioni, conoscenza e, in ultima istanza, saggezza. In questo scenario l'Internet delle cose diventa incredibilmente importante. Sono già in fase di realizzazione progetti di Internet delle cose volti a migliorare la distribuzione delle risorse mondiali a chi ne ha più bisogno e aiutarci a capire il nostro pianeta in modo da adottare un approccio maggiormente proattivo e meno reattivo. Ciononostante, esistono diverse problematiche che minacciano di rallentare lo sviluppo dell'Internet delle cose, tra cui la transizione a IPv6, la definizione di una serie di standard comuni e il reperimento di fonti di alimentazione per milioni (se non miliardi) di sensori di dimensioni minuscole. Tuttavia, grazie alla collaborazione tra aziende, governi, organismi di standardizzazione e università per far fronte a queste sfide, l'Internet delle cose continuerà a progredire. Scopo della tesina, quindi, è fornire informazioni chiare e semplici sull'Internet delle cose, sulla sua potenziale capacità di trasformare tutto ciò che costituisce la nostra realtà di oggi e di mostrare un progetto correlato ad esso.

Lo sviluppo dell'Internet delle cose e le sue problematiche

Attualmente l'Internet delle cose è costituito da una serie di reti eterogenee, non collegate tra loro e appositamente create per scopi specifici. Le automobili di oggi, ad esempio, dispongono di più reti per gestire il funzionamento del motore, le funzioni di sicurezza, i sistemi di comunicazione, ecc. Anche gli edifici commerciali e residenziali hanno vari sistemi di controllo per HVAC (Heating, Ventilation and Air Conditioning: riscaldamento, aerazione e condizionamento dell'aria), servizi di telefonia, sicurezza e illuminazione. Con l'evoluzione dell'Internet delle cose, queste reti (e molte altre) avranno accesso a ulteriori funzionalità di sicurezza, analisi e gestione. In questo modo l'Internet delle cose diventerà ancora più potente in termini di ciò che potrà aiutare a ottenere.
Lo sviluppo dell'Internet delle cose, tuttavia, potrebbe essere rallentato da alcune complicazioni. Le tre più importanti sono l'alimentazione dei sensori, l'accordo sugli standard e l'implementazione di IPv6.

Alimentazione dei sensori.
Affinché l'Internet delle cose sfrutti appieno il proprio potenziale, i sensori dovranno essere in grado di alimentarsi autonomamente. Immaginare di poter sostituire le batterie a miliardi di dispositivi installati in tutto il pianeta (e persino nello spazio) è pura follia. Occorre una soluzione che consenta ai sensori di generare elettricità da elementi dell'ambiente quali vibrazioni, luce e flusso dell'aria. In quella che può essere considerata una svolta epocale, in occasione della 241ª National Meeting & Exposition dell'American Chemical Society del marzo 2011, alcuni scienziati hanno annunciato il lancio di un nanogeneratore redditizio dal punto di vista commerciale, ovvero un chip flessibile che sfrutta i movimenti del corpo, ad esempio la pressione di pollice e indice, per generare elettricità.

Standard.
Nonostante nel campo degli standard siano stati fatti grandi passi avanti, ne servono ancora, soprattutto nei settori della sicurezza, della privacy, dell'architettura e delle comunicazioni. Lo IEEE è solo una delle organizzazioni che lavorano al superamento di queste sfide consentendo il routing dei pacchetti IPv6 in diversi tipi di rete. È tuttavia importante sottolineare che, nonostante ostacoli e sfide esistano, non sono insormontabili. Visti i vantaggi dell'Internet delle cose, si tratta di problemi che verranno risolti. È solo questione di tempo.

Implementazione di IPv6.
Gli ultimi indirizzi IPv4 disponibili sono stati assegnati nel febbraio del 2010. Anche se il grande pubblico non si è accorto praticamente di nulla, questa situazione ha rischiato di rallentare il progresso dell'Internet delle cose, dal momento che i potenziali miliardi di nuovi sensori richiederanno indirizzi IP univoci. Inoltre, IPv6 semplifica la gestione delle reti grazie alle funzionalità di autoconfigurazione e offre caratteristiche di sicurezza avanzate.

Il protocollo IPv6

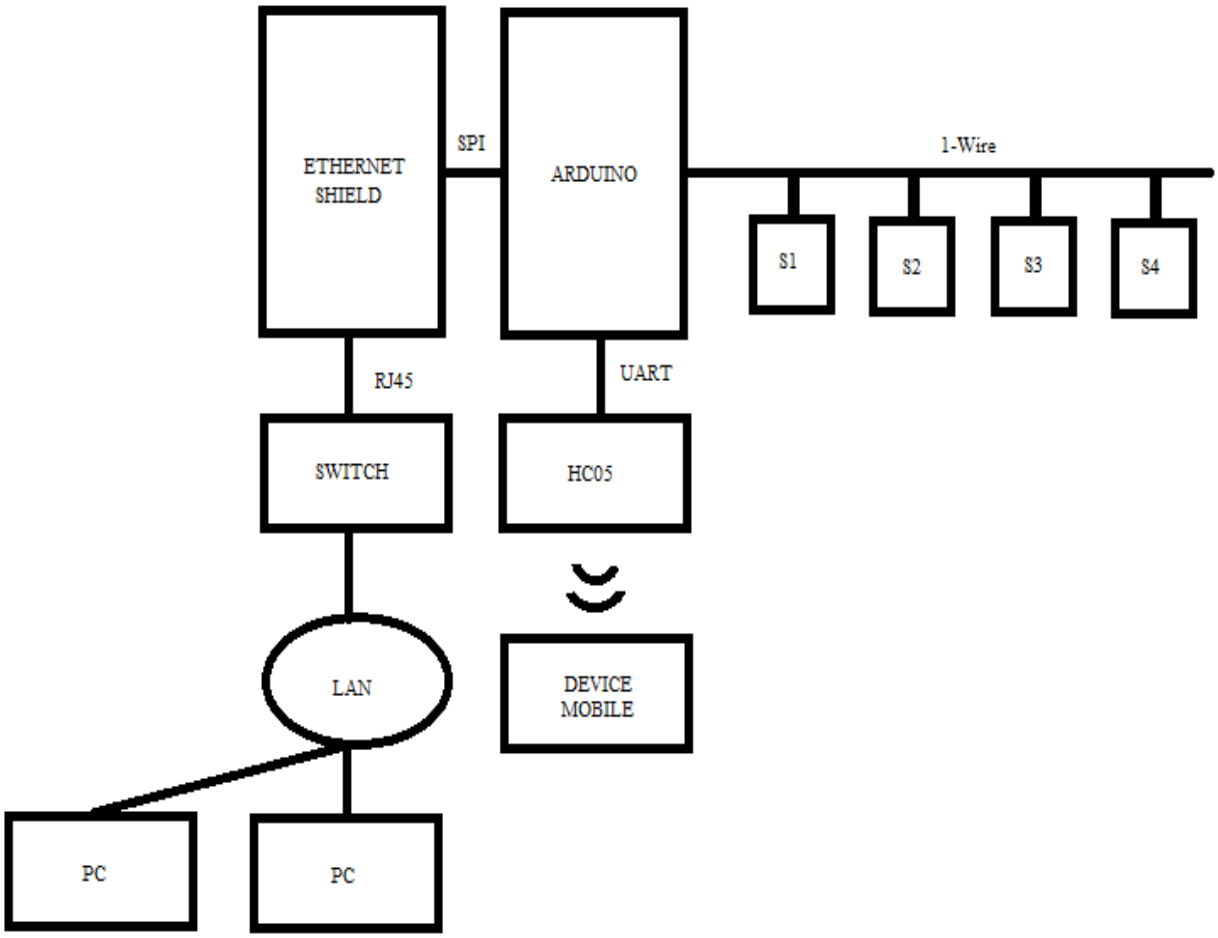
Il protocollo IPv6, noto anche come IPng (IP *next generation),* consente di supportare le esigenze attuali e future nell'ambito della comunicazione tra apparati che implementano la suite TCP/IP. Infatti, i principali limiti del protocollo IPv4 si possono così riassumere:

- lo spazio di indirizzi IPv4 a disposizione, pari a $2^{32} \approx 4{,}29 \cdot 10^{9}$ indirizzi IPv4, non è sufficiente a supportare lo sviluppo di reti IP multiservizio, in quanto il numero di apparati dotati di interfacce IP è cresciuto e crescerà ancora esponenzialmente.
- capacità di supporto della qualità di servizio (QoS) alquanto limitata.
- implementazione opzionale dell'architettura di sicurezza IPSec (*Internet Protocol Security architecture*), con crittografia dei singoli pacchetti IPv4, che limita la sicurezza.
- generalmente gli host non sono auto configuranti, la configurazione degli indirizzi IPv4 negli host deve essere fatta manualmente oppure deve essere presente un server DHCP;
- limitata capacità di un supporto di nuove funzionalità;
- va controllato il *checksum* dei pacchetti IPv4, anche se i nuovi sistemi di telecomunicazione, basati su fibre ottiche, hanno una bassissima probabilità d'errore.

Il protocollo IPv6 supera tali limiti in quanto è caratterizzato da:

- indirizzi IPv6 a128 bit, che significa uno spazio di indirizzamento enorme, pari a $2^{128} \approx 3{,}4 \cdot 10^{38}$ indirizzi IPv6, ciò consente di realizzare una gerarchia di livelli di indirizzamento che rende più efficiente il routing, di avere host autoconfiguranti, di introdurre nuovi tipi di indirizzi IPv4, come l'*anycast*, di supportare, meglio il traffico di tipo multi cast ecc.;
- header IPv6 ridefinito e semplificato, di lunghezza fissa (40 ottetti), senza checksum e senza campi opzionali;
- introduzione nell'header IPv6 di un campo di 20 bit denominato *flow label*, che consente l'identificazione e la differenziazione dei flussi informativi;
- header opzionali denominati *extension header*, che è possibile porre dopo l'header IPv6, la cui lunghezza è limitata solo dalle dimensioni di un pacchetto;
- protezione delle informazioni di utente e della privacy grazie al supporto all'architettura IPsec.

Esempio di Internet delle cose: "progetto per la rilevazione di temperature da un bus di sensori."

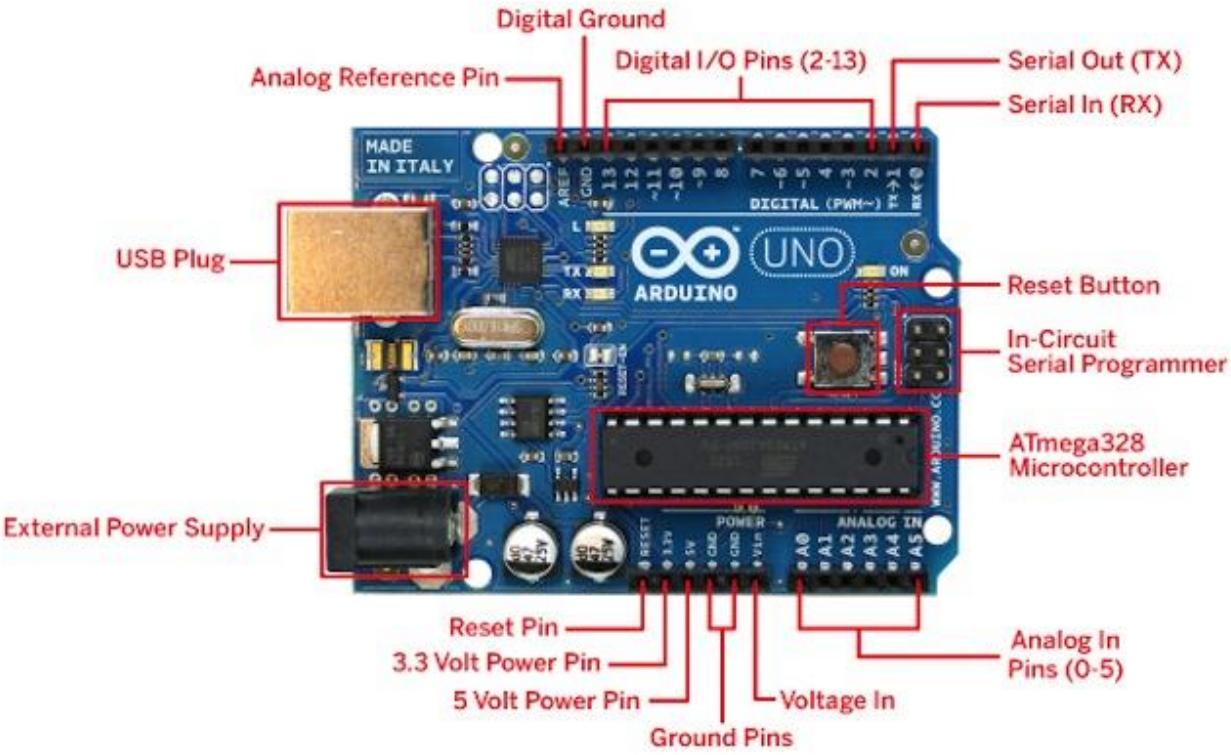


Il progetto nasce dall'idea di voler dare una dimostrazione pratica dell'Internet of Things.
Un esempio di Internet delle cose potrebbe essere l'acquisizione di temperature da un bus di sensori, situato in un appartamento, che inviano i dati rilevati a uno *smartphone* via Bluetooth e una pagina web via Ethernet.
A questo si è aggiunta la volontà di voler far azionare una ventola nel caso in cui la stanza principale dell'appartamento raggiunga una temperatura superiore ai 30°C, simulando così un impianto di raffreddamento.

Per realizzare questo progetto si è utilizzato Arduino, l'Ethernet Shield, un bus formato da quattro sensori Dallas, un resistore da 4,7 kΩ, una ventola, un cavo RJ45 e un modulo Bluetooth HC05.

Arduino

Arduino è una piattaforma hardware low-cost programmabile, con cui è possibile creare circuiti "quasi" di ogni tipo per molte applicazioni, soprattutto in ambito di robotica ed automazione. Si basa su un Microcontrollore della ATMEL, l'ATMega168/328: per esempio l'Arduino Uno monta un ATMega328.



Spiegazione dei Pin:

•AREF- Questo pin viene utilizzato per gli input analogici, voltaggio di riferimento. •GND- massa

•PWM- I pin a disposizione con questa funzionalità sono 6. Il PWM, o pulse width modulation permette di creare onde quadre con duty cycle regolabile.

•TX - RX- porta seriale

•RESET- Questo è un PIN digitale. Se la lettura di questo PIN=HIGH il controller si resetta

•PIN uscita corrente a 3.3V

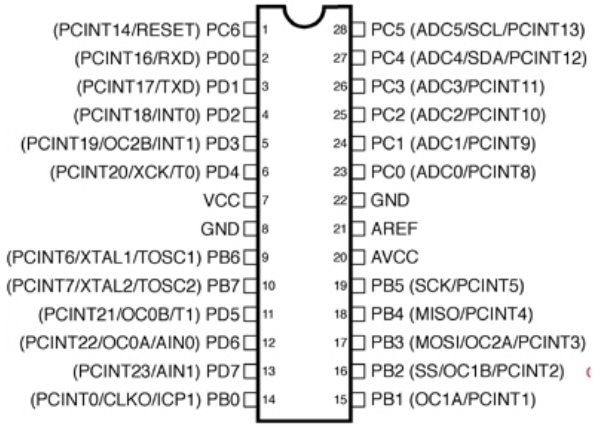
•PIN uscita corrente a 5V

•Vin- PIN input corrente per alimentazione controller

•Analog in- PIN input analogici. Possono percepire molto precisamente una corrente DC tra 0 e 5V, resistuendo un valore da 0 a 1023.

•Digital- PIN digitali programmabili per essere input o output, percepiscono se è presente o no corrente restituendo LOW se non c'è corrente e HIGH se c'è corrente, oppure possono essere programmati per generare corrente in output di massimo 40mA.

Spiegazione dell'Atmega328:
Atmega328 è la sigla di questo microcontrollore. All'interno di questo componente viene salvato il programma scritto dall'utente e tutta la configurazione di base che permette ad Arduino un funzionamento corretto.



Ethernet Shield

Arduino Ethernet Shield consente alla scheda Arduino di connettersi a Internet in modo molto semplice collegando il modulo prima all' Arduino poi tramite il cavo RJ45 alla rete
L'installazione, dal punto di vista hardware è davvero semplice, in quanto occorre posizionare la scheda sopra Arduino Uno, inserendo i PIN in modo corretto. Sarà possibile comunque usare i PIN per i nostri progetti, dal momento in cui esiste un contatto tra la scheda Ethernet e Arduino Uno. Dispone di uno slot per schede micro-SD accessibile attraverso la Libreria SD , che può essere utilizzata per memorizzare i file da mettere in rete.

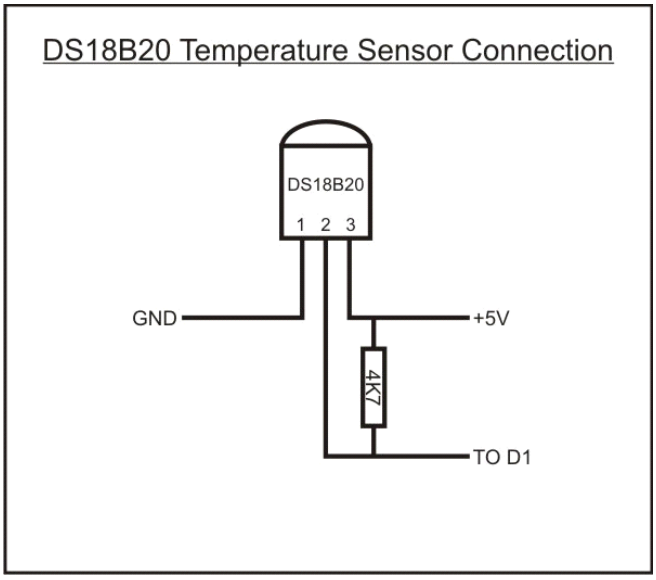
Arduino comunica sia con il W5100 sia con la scheda SD utilizzando il bus SPI .Il pin 10 viene utilizzato per comunicazione con il W5100 e il pin 4 con la scheda SD.

Caratteristiche principali:

- 5V Tensione di esercizio (fornita dalla scheda Arduino)
- Ethernet Controller: W5100 con 16K buffer interno
- Velocità di connessione: 10 / 100 Mbit/s
- Collegamento con Arduino sulla porta SPI

Trasduttore di temperatura DS18B20

Trasduttore intelligente digitale di temperatura con una uscita serale (1- Wire). Identificabile grazie all' indirizzo fisico da 64 bit contenuto nella ROM. Molto preciso (0,5°C di accuratezza) sfrutta una risoluzione a 9 bit (8 di misura e 1 di segno). È dotato di tre piedini: GND,segnale e alimentazione.



Per poter funzionare necessita di due librerie: OneWire e DallasTemperature le quali devono essere scaricate sul desktop del proprio computer, e poi caricate da Sketch -> Importa libreria -> Add library. Le nuove librerie verranno installate nella cartella Arduino presente nella directory Home. Ogni sensore è dotato di un indirizzo univoco.

Bus 1-WIRE

Quando parliamo di Bus 1-Wire parliamo di un protocollo di comunicazione, introdotto dalla ditta americana Dallas (ora Maxim/Dallas) semiconductor, che permette di interfacciare dispositivi logici quali memorie, sonde, e molti altri ad un microcontrollore mediante un solo filo (più la massa). Proprio questa semplicità di architettura/cablaggio ha reso questo bus molto diffuso in campo hobbystico e non laddove la velocità nella acquisizione dati non rappresenta un requisito fondamentale. E' possibile collegare al bus molti dispositivi contemporaneamente in quanto ognuno di essi dispone di un proprio indirizzo univoco che permette di distinguerlo dagli altri. Il bus 1-Wire è pensato per essere un bus del tipo single-master, multi-slave.

Single-master in quanto è prevista la presenza di un solo master che ha il compito di "gestore" del bus. A lui infatti spetta il compito di dettare le temporizzazioni, di verificare quanti e quali dispositivi sono collegati, di interrogare i vari dispositivi e riceverne le risposte. Generalmente il compito di master è assunto dal microcontrollore.

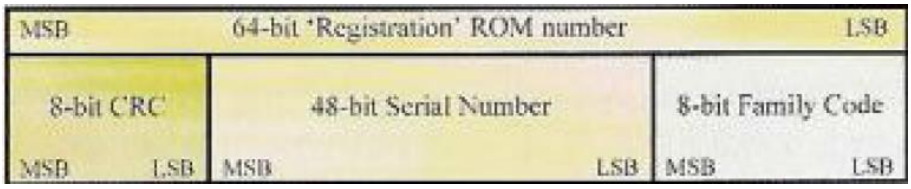
Multi-slave in quanto più dispositivi possono essere connessi contemporaneamente. Ogni slave ha l'obbligo di lasciare sempre libero il bus e di occuparlo solamente quando viene interrogato dal master.

Nel Bus 1-Wire la trasmissione e' bi-direzionale e di tipo halfduplex (ricezione e trasmissione non possono avvenire contemporaneamente). Nella comunicazione si trasmettono i singoli bit iniziando sempre dal bit LSB. Non e' richiesto un clock di sistema in quanto la sincronizzazione avviene con il fronte di discesa del bus pilotato dal master. Tipicamente si possono avere due velocità di comunicazione:

1) Standard: circa16kbps (Kilo bits per secondo).

2) Overdrive: circa 140kbps, questa modalità è attivabile solo in determinate condizioni (pochi dispositivi collegati e linea non eccessivamente lunga – non superiore ai 10/15 metri).

Ogni dispositivo-slave 1-Wire è individuabile mediante un numero seriale unico di 64 bit, assegnato dal costruttore e contenuto in opportuna memoria ROM del dispositivo. Questo ID è suddiviso in 8 bytes raggruppati in 3 opportune sezioni. Il primo byte (LSB) identifica il tipo di dispositivo (sensore di temperatura, memoria eeprom, ADC programmabili, etc…). I successivi 6 bytes (48 bit) rappresentano l'indirizzo univoco del dispositivo. L'ultimo byte (MSB) è il codice di controllo (CRC – Cyclic Redundancy Checksum) relativo ai 56 bit precedenti. La tecnica di ridondanza di codice (CRC), molto applicata nei processi di trasmissione dati, permette al dispositivo MASTER di individuare se la lettura dei precedenti 56 bit è avvenuta correttamente, cioè non sono presenti "conflitti" nel processo di trasmissione.



Il sistema 1-wire è un modello di comunicazione "prettamente software" e per la gestione di ogni dispositivo si richiede il software opportuno.

Dal momento in cui ogni sensore è dotato di un indirizzo univoco, è necessario ricavarlo poiché andrà inserito nel programma per l'acquisizione delle temperature.

Lettura degli indirizzi MAC dei Dallas

```
#include <OneWire.h>

#define SENSOR_PIN 5

OneWire  ourBus(SENSOR_PIN);  // Create a 1-wire object

void setup()

{

  Serial.begin(9600);


  discoverOneWireDevices();

}

void loop()

{

}
```

```
void discoverOneWireDevices(void) {
  byte i;
  byte present = 0;
  byte data[12];
  byte addr[8];
  Serial.print("Looking for 1-Wire devices...\n\r");// "\n\r" is NewLine
  while(ourBus.search(addr)) {
    Serial.print("\n\r\n\rFound \'1-Wire\' device with address:\n\r");
    for( i = 0; i < 8; i++) {
      Serial.print("0x");
      if (addr[i] < 16) {
        Serial.print('0');
      }
      Serial.print(addr[i], HEX);
      if (i < 7) {
        Serial.print(", ");
      }
    }
    if ( OneWire::crc8( addr, 7) != addr[7]) {
      Serial.print("CRC is not valid!\n\r");
      return;
    }
  }
  Serial.println();
  Serial.print("Done");
  ourBus.reset_search();
  return;
}
```

Rivoluzione Wireless: Bluetooth

Nell'analizzare il mercato hi-tech presente e futuro, tutti gli analisti concordano nell'identificare una vera e propria "rivoluzione wireless".
Le prospettive di crescita per la commercializzazione di prodotti e servizi che utilizzano la comunicazione senza fili, infatti, sono impressionanti: nel momento in cui le performance velocistiche dei computer hanno raggiunto livelli tali da soddisfare le esigenze dell'utenza consumer, i nuovi parametri di valutazione per la qualità di un prodotto sono la sua portabilità e la sua capacità di comunicare.

Come per tutte le nuove ondate di innovazione tecnologica in campo informatico, l'unico ostacolo ad un pieno successo è la creazione di standard comuni che permettano una diffusione massiccia delle tecnologie. Nel caso del mercato wireless sembra che il problema sia stato risolto brillantemente: esistono principalmente tre standard, con funzionalità e caratteristiche ben definite e assolutamente non concorrenziali: Bluetooth , Wi-Fi e ZigBee

Soffermiamoci sul primo:

Il Bluetooth è una forma "leggera" di comunicazione wireless: ha un raggio limitato e fondamentalmente si pone come un valido sostituto ai cavi di collegamento tra periferiche. E' facile, quindi, trovare cellulari dotati di tecnologia Bluetooth per potersi sincronizzare con computer, palmari, ecc. Oppure di recente sono stati messi in commercio kit composti di mouse e tastiera wireless basati sulla tecnologia Bluetooth: un ottimo modo di sgombrare le scrivanie da cavi inutili. Indubbiamente il Bluetooth dispone di notevoli potenzialità in questo ambito, ma sicuramente non è adatto alla realizzazione di reti senza fili, proprio in virtù della sua portata limitata.

Bluetooth è uno Standard Aperto per telecomunicazioni wireless a corto raggio. [Standard Aperto (Open Standard) vuol dire utilizzabile da chiunque liberamente. Le specifiche sono pubbliche] Bluetooth è stata sviluppato nel 1994 (il nome deriva da Harald Blatant "Bluetooth", Re di Danimarca, 940-981 A.D.) La trasformazione di bluetooth in standard avviene a opera del Bluetooth Special Interest Group (SIG), fondato nel 1998 e costituito da Ericcson, IBM, Nokia, Intel, Toshiba. Ora comprende più di 1900 società. Lo standard Bluetooth serve a realizzare le cosiddette Personal Area Network (PAN), cioè piccole reti dell'estensione tipica di qualche metro. Per esempio, reti che collegano computer con varie periferiche (in sostituzione del cavo), reti dati e/o audio in singole abitazioni (home networking) [collegamento tra elettrodomestici intelligenti ("smart appliances"), dispositivi di riscaldamento, condizionamento, intrattenimento, e sistemidi allarme], Bluetooth funziona in banda ISM 2.45 GHz (2400-2483.5 MHz), con una modulazione frequency hopping – spread spectrum (FHSS) per ridurre l'effetto di interferenze e fading (cammini multipli). Sono disponibili 79 canali FHSS. In ogni canale, per minimizzare la complessità del radiotrasmettitore si usa una modulazione binaria FM a 1Msimbolo/s = 1Mbit/s (lordo). Da standard, la distanza massima di comunicazione è 10 m (100 m con un trasmettitore potenziato). Un ricetrasmettitore bluetooth è anche detto "dispositivo" bluetooth. Tipicamente, un canale radio fisico è occupato da più dispositivi sincronizzati su una stessa sequenza di frequency hopping. Tale insieme di dispositivi si chiama "piconet". Un dispositivo, detto "master", è responsabile della sincronizzazione. Gli altri sono "slave".

Modulo Bluetooth HC-05

Il modulo bluetooth HC è un modulo che permette di trasformare una porta UART\USART più comunemente conosciuta come seriale (come il protocollo RS-232 unica differenza i livelli dei segnali per cui necessiterebbe di un max232 che è un circuito integrato per un collegamento diretto esempio ad un PC) in una porta Bluetooth, generalmente con profilo SPP(Serial Port Profile), diventando cosi una seriale over Bluetooth.
Normalmente questo dispositivo viene usato quando si vuole far comunicare un microprocessore(MCU) che può essere un PIC della microchip od un processore ATmel di norma montato su schede Arduino con il mondo esterno, dove il mondo esterno può essere un Personal Computer, un Telefonino SmartPhone, un Tablet o quant'altro la tecnologia possa fornire con una connessione Bluetooth.
Le classi per i dispositivi Bluetooth sono 3:
Classe 1 : potenza in mW 100, potenza dBm 20, distanza metri circa 100.
Classe 2 : potenza in mW 2,5, potenza dBm 4, distanza metri circa 10.
Classe 3 :potenza in mW 1, potenza dBm 0, distanza metri circa 1.
Il dispositivo HC05 è di classe 2.
Per tutti gli esperti di protocollo Bluetooth non c'è bisogno di specificare altro, per tutti coloro che sono meno esperti possiamo dire che fondamentalmente il protocollo Bluetooth funziona in modalità Master/Slave, un master ed n°(pochi) slave.
Quindi affinché due moduli HC comunichino tra di loro c'è bisogno che uno entri in modalità Master ed uno entri in modalità Slave, per far questo si utilizzano due HC-05 e li si configura uno in modalità Master ed uno in modalità Slave.
Se vogliamo far comunicare un modulo Bluetooth con uno Smartphone è necessario configurare il modulo in modo che possa trasmettere dati al device.
In particolare è necessario configurare uno di questi come SLAVE e per farlo possiamo utilizzare i comandi AT.

Programmazione modulo Bluetooth HC-05

```
#include <SoftwareSerial.h>
int rxPin = 10;
int txPin = 11;
SoftwareSerial bluetooth(rxPin, txPin);
void setup()
{
  Serial.begin(9600);
  bluetooth.begin(38400);
  Serial.println("Comunicazione inizializzata, Lista comandi per modulo HC-05:");
  Serial.println("AT              Se la comunicazione funziona il modulo risponde OK");
  Serial.println("AT+VERSION      Restituisce la versione del firmware");
  Serial.println("AT+BAUDx        Imposta il Baudrate, al posto di x mettere 1 per 1200 bps, 2=2400, 3=4800, 4=9600, 5=19200, 6=38400, 7=57600, 8=115200, 9=230400, A=460800, B=921600, C=1382400");
  Serial.println("AT+NAMEstring   Al posto di string mettere il nome che vuoi dare al modulo (massimo 20 caratteri)");
  Serial.println("AT+PINxxxx      Imposta il pincode del modulo bluetooth (es.1234)");
}
void loop()
{
  if (bluetooth.available())
  {
    Serial.write(bluetooth.read());
  }
  if (Serial.available())
  {
    bluetooth.write(Serial.read());-
  }
}
```

Questo sketch permette di eseguire dei comandi AT sul modulo Bluetooth e leggere le risposte. Una volta caricato lo sketch su Arduino eseguiamo dei comandi AT per impostare il modulo Bluetooth come SLAVE:

- Apriamo il monitor seriale del programma Arduino
- Assicuriamoci che sia impostato il valore "Both NL & CR" e la velocità di connessione a 9600
- Quando resettiamo Arduino apparirà sul monitor seriale. "Enter AT commands: "
- Connettiamo il PIN Vcc del modulo Bluetooth al PIN 5V di Arduino, dovremmo vedere il led del modulo Bluetooth lampeggiare lentamente (circa ogni due secondi)
- Scriviamo nel monitor seriale il comando "AT+ROLE=0";
  questo configurerà il modulo Bluetooth come SLAVE. Per verificare il comando appena eseguito, digitiamo "AT+ROLE?" e il monitor seriale ci restituirà "+ROLE:0"
- Ora eseguiamo il comando "AT+BIND=" e premiamo INVIO.
  In questo modo rimuoviamo qualunque binding esistente. Per verificare eseguiamo il comando "AT+BIND?" e dovremmo ottenere "+BIND:0:0:0"
- Eseguiamo quindi il comando "AT+ADDR?" per ottenere l'indirizzo MAC del nostro modulo.

Ethernet

Il termine Ethernet si riferisce a una famiglia di tecnologie per le LAN conformi agli standard IEEE 802.3xx, che implementano gli strati OSI 1 e 2 di una rete a commutazione di pacchetto e adottano un protocollo MAC che fa uso del metodo di accesso multiplo CSMA/CD.
La tecnologia Ethernet costituisce lo standard di fatto cpm cui si realizzano praticamente tutte le LAN attuali. Ciò è dovuto alla sua efficienza, ai bassi costi e alle prestazioni che sono ormai elevatissime (hanno superato i 1000 Mbit/s).
Nelle LAN Ethernet lo strato fisico delle stazioni ha il compito di effettuare la ricetrasmissione sul cavo dei bit che costituiscono ciascun frame passato allo strato fisico dello strato MAC.
Terminata la trasmissione di un frame, però, una stazione deve attendere un tempo denominato *Inter-Frame Gap* (IFG) o *Inter-Packet Gap* (IPG) prima di iniziare a trasmettere il frame successivo.
Lo standard IEEE 802 prescrive che l'IFG deve avere una durata equivalente a 96 tempi di bit, corrispondenti a 12 byte.

Programmazione Arduino per la rilevazione delle temperature dal bus di sensori e l'invio dei dati via Ethernet e via Bluetooth

```
#include <DallasTemperature.h>
#include <OneWire.h>
#include <Ethernet2.h>
#include <SPI.h>
#include <SoftwareSerial.h>
  float temp1;
  float temp2;
  float temp3;
  float temp4;
  int ventola=2;
  SoftwareSerial BT(6,7); // RX|TX
int num;  //inizializzazione variabili
OneWire bus(5); //istanza ow da classe onewire
DallasTemperature miobus(&bus); //istanza bus da classe
DeviceAddress sens1={0x28,0x80,0x22,0x09,0x07,0x00,0x00,0x66}; //indirizzo
DeviceAddress sens2={0x28,0xCA,0x2B,0x74,0x05,0x00,0x00,0x55}; //indirizzo
DeviceAddress sens3={0x28,0xFB,0x37,0x74,0x05,0x00,0x00,0xD2}; //indirizzo
DeviceAddress sens4={0x28,0xB0,0x75,0x09,0x07,0x00,0x00,0x4F}; //indirizzo

byte mac[]={0x90, 0xA2,0xDA, 0x10, 0x20, 0x48};
byte ip[]={10,0,0,112};
char Data_RX;
EthernetServer ArduinoServer(80);
void setup()
{
 miobus.begin();
 num=miobus.getDeviceCount();  //rilevazione numero sensori effettivo
 Serial.print("numero sens");
 Serial.print(num);
 Serial.begin(9600);
```

```
 BT.begin(9600);
 Ethernet.begin(mac,ip);
 ArduinoServer.begin();
 pinMode(ventola,OUTPUT); //programmo  il pin 8 digitale di arduino in uscita

}
void loop()
{
 miobus.requestTemperatures(); // avvio conversione
 Serial.print("\nRILEVAZIONE TEMPERATURE\n"); //lettura e stampa temperature
  temp1=miobus.getTempC(sens1);
  if (temp1>30)
    {
      digitalWrite(ventola,HIGH);
      }
      else
      {
        digitalWrite(ventola,LOW);
        }
  temp2=miobus.getTempC(sens2);
  temp3=miobus.getTempC(sens3);
  temp4=miobus.getTempC(sens4);

  temp1=miobus.getTempC(sens1);
  Serial.print("Sensore Soggiorno:");
  BT.print("Sensore Soggiorno: ");
  Serial.print(temp1);
  BT.print(temp1);
  Serial.println("Gradi C ");
  BT.println("Gradi C ");
```

```
 temp2=miobus.getTempC(sens2);
 Serial.print("Sensore Camera da letto:");
 BT.print("Sensore Camera da letto: ");
 Serial.print(temp2);
 BT.print(temp2);
 Serial.println("Gradi C ");
 BT.println("Gradi C ");


 temp3=miobus.getTempC(sens3);
 Serial.print("Sensore Cameretta 1:");
 BT.print("Sensore Cameretta 1: ");
 Serial.print(temp3);
 BT.print(temp3);
 Serial.println("Gradi C ");
 BT.println("Gradi C ");


 temp4=miobus.getTempC(sens4);
 Serial.print("Sensore Cameretta 2:");
 BT.print("Sensore Cameretta 2: ");
 Serial.print(temp4);
 BT.print(temp4);
 Serial.println("Gradi C ");
 BT.println("Gradi C");

 BT.println("\n");
/*
ascolto le richieste  dei client, controllo se ci sono dati da leggere
e creo un oggetto relativo al client che sta interrogando l'ethernet shield
 */
EthernetClient pc_client = ArduinoServer.available();
```

```
//controllo se pc_client è "true"
//if(pc_client != false)
if(pc_client)
{
 //se pc_client è true continua ad utilizzarlo
 //controllo se il client è connesso
 while(pc_client.connected())
 {
  //eseguo questo codice finché il client è connesso...
  //controllo se ci sono byte disponibili per la lettura
  if(pc_client.available())
  {
   //leggo i byte disponibili
   //provenienti dal client
   Data_RX = pc_client.read();
   //invio i dati letti al serial monitor
   /*attendo che tutti i byte siano letti
     quando Data_RX contiene il carattere
     di nuova line capisco tutti i byte sono stati letti
    */
    if(Data_RX == '\n')
    {
     /*invio la risposta al client
       invio lo status code
      */
      pc_client.println("HTTP/1.1 200 OK");
      //imposto il data type
      pc_client.println("Content-Type: text/html");
      pc_client.println();
      //invio codice html
       pc_client.print("<html>");
       pc_client.print ("<p align='center'>"); //allinea testo
```

```
pc_client.print ("<p align='center'><font size='12' face='Comic Sans MS'><p align='center'> Rilevazione temperature dell'appartamento <br></head>");
pc_client.print ("<br>"); //nuova riga
pc_client.print ("<p align= 'center'>");
pc_client.print ("<body bgcolor='gray'>");
pc_client.print ("<table width='25%' border='3'>"); //comando per creare tabella
pc_client.print ("<tr bgcolor='yellow'>"); //tr e' il tag per aggiungere righe alla tabella td per le colonne
pc_client.print ("<td> <h2 align='center'> SENSORI </h2> </td>");
pc_client.print ("<td> <h2 align='center'> TEMPERATURA &#176 C </h2> </td>");
pc_client.print ("</tr>");
pc_client.print ("<tr>");
if(temp1>30)
{
pc_client.print ("<tr bgcolor='red'>");
pc_client.print ("<td> <h2 align='center'> Sensore Soggiorno </h2> </td>");
pc_client.print ("<td> <h2 align='center'>");
pc_client.print (temp1);
pc_client.print ("</h2> </td>");
pc_client.print ("</tr>");
}
else
{
pc_client.print ("<tr bgcolor='green'>");
pc_client.print ("<td> <h2 align='center'> Sensore Soggiorno </h2> </td>");
pc_client.print ("<td> <h2 align='center'>");
pc_client.print (temp1);
pc_client.print ("</h2> </td>");
pc_client.print ("</tr>");
}
if(temp2>30)
{
pc_client.print (" <tr bgcolor='red'>");
```

```
pc_client.print ("<td><h2 align='center'> Sensore Camera </h2></td>");
pc_client.print ("<td><h2 align='center'>");
pc_client.print (temp2);
pc_client.print ("</h2></td>");
pc_client.print ("</tr>");
}
else
{
pc_client.print (" <tr bgcolor='green'>");
pc_client.print ("<td><h2 align='center'> Sensore Camera </h2></td>");
pc_client.print ("<td><h2 align='center'>");
pc_client.print (temp2);
pc_client.print ("</h2></td>");
pc_client.print ("</tr>");
}
if (temp3>30)
{
pc_client.print ("<tr bgcolor='red'>");
pc_client.print ("<td><h2 align='center'> Sensore Cameretta 1 </h2></td>");
pc_client.print ("<td><h2 align='center'>");
pc_client.print (temp3);
pc_client.print ("</h2></td>");
pc_client.print ("</tr>");
}
else
{
pc_client.print (" <tr bgcolor='green'>");
pc_client.print ("<td><h2 align='center'> Sensore Cameretta 1 </h2></td>");
pc_client.print ("<td><h2 align='center'>");
pc_client.print (temp3);
pc_client.print ("</h2></td>");
pc_client.print ("</tr>");
```

```
        }
        if (temp4>30)
        {
        pc_client.print ("<tr bgcolor='red'>");
        pc_client.print ("<td><h2 align='center'> Sensore Cameretta 2 </h2></td>");
        pc_client.print ("<td><h2 align='center'>");
        pc_client.print (temp4);
        pc_client.print ("</h2></td>");
        pc_client.print ("</tr>");
        }
        else
        {
        pc_client.print ("<tr bgcolor='green'>");
        pc_client.print ("<td><h2 align='center'> Sensore Cameretta 2 </h2></td>");
        pc_client.print ("<td><h2 align='center'>");
        pc_client.print (temp4);
        pc_client.print ("</h2></td>");
        pc_client.print ("</tr>");
        }
        pc_client.print ("</p>");
        pc_client.print ("</table>");
        pc_client.print ("</body>");
        pc_client.print ("</html>");
       //aspetto 10 ms affinchè la risposta giunga al browser del client
       delay(10);
       //esco dal ciclo while una volta completato l'invio della risposta
       break;
     }
   }
}
//chiudo la connessione
pc_client.stop();
```

```
 }
 delay(3000);
}
```

Le librerie “SPI.h” e “Ethernet2” permettono di interfacciarsi allo shield ethernet dopodichè sono stati creati due vettori che andranno a contenere l'indirizzo MAC e l'indirizzo IP della scheda di rete. Successivamente è stato creato un oggetto Server che permette di rimanere in ascolto sulla porta 80 che è quella utilizzata dal protocollo http.
Per memorizzare i byte provenienti dal client è stata impiegata la variabile Data_RX
Nel setup() vi sono due importanti funzioni:
(Ethernet.begin(mac, ip);): per inizializzare il chip WIZnet con l’indirizzo mac e l’indirizzo IP,
(ArduinoServer.begin();): avvia il server e lo mette in ascolto sulla porta 80 per le avvenutali richieste dei client.

HTML

Per riportare i valori in una pagina web è stato necessaria la conoscenza di alcune semplici nozioni di HTML;

L'**HyperText Markup Language** (**HTML**), in informatica è il linguaggio di markup solitamente usato per la formattazione e impaginazione di documenti ipertestuali disponibili nel World Wide Web sotto forma di pagine web, nato assieme al web 1.0.

È un linguaggio di pubblico dominio, la cui sintassi è stabilita dal World Wide Web Consortium (W3C).

Di seguito è riportata la parte di codice del software dedicata alla scrittura della pagina web in HTML. Nella pagina web saranno riportati i valori di temperatura dei locali dell' appartamento. Nel caso in cui la temperatura di uno dei quattro locali superi la soglia di 30°, la cella riservata a quel locale viene contrassegnata in rosso.

```
pc_client.print("<html>");
pc_client.print ("<p align='center'>"); //allinea testo
pc_client.print ("<p align='center'><font size='12' face='Comic Sans MS'><p align='center'> Rilevazione temperature dell'appartamento <br></head>");
pc_client.print ("<br>"); //nuova riga
pc_client.print ("<p align= 'center'>");
pc_client.print ("<body bgcolor='gray'>");
pc_client.print ("<table width='25%' border='3'>"); //comando per creare tabella
pc_client.print ("<tr bgcolor='yellow'>"); //tr e' il tag per aggiungere righe alla tabella td per le colonne
pc_client.print ("<td> <h2 align='center'> SENSORI </h2> </td>");
pc_client.print ("<td> <h2 align='center'> TEMPERATURA &#176 C </h2> </td>");
pc_client.print ("</tr>");
pc_client.print ("<tr>");
if(temp1>30)
{
pc_client.print ("<tr bgcolor='red'>");
pc_client.print ("<td> <h2 align='center'> Sensore Soggiorno </h2> </td>");
pc_client.print ("<td> <h2 align='center'>");
pc_client.print (temp1);
pc_client.print ("</h2> </td>");
```

```
pc_client.print ("</tr>");
}
else
{
pc_client.print ("<tr bgcolor='green'>");
pc_client.print ("<td> <h2 align='center'> Sensore Soggiorno </h2> </td>");
pc_client.print ("<td> <h2 align='center'>");
pc_client.print (temp1);
pc_client.print ("</h2> </td>");
pc_client.print ("</tr>");
}
if(temp2>30)
{
pc_client.print (" <tr bgcolor='red'>");
pc_client.print ("<td><h2 align='center'> Sensore Camera </h2></td>");
pc_client.print ("<td><h2 align='center'>");
pc_client.print (temp2);
pc_client.print ("</h2></td>");
pc_client.print ("</tr>");
}
else
{
pc_client.print (" <tr bgcolor='green'>");
pc_client.print ("<td><h2 align='center'> Sensore Camera </h2></td>");
pc_client.print ("<td><h2 align='center'>");
pc_client.print (temp2);
pc_client.print ("</h2></td>");
pc_client.print ("</tr>");
}
if (temp3>30)
{
pc_client.print ("<tr bgcolor='red'>");
```

```
pc_client.print ("<td><h2 align='center'> Sensore Cameretta 1 </h2></td>");
pc_client.print ("<td><h2 align='center'>");
pc_client.print (temp3);
pc_client.print ("</h2></td>");
pc_client.print ("</tr>");
}
else
{
pc_client.print (" <tr bgcolor='green'>");
pc_client.print ("<td><h2 align='center'> Sensore Cameretta 1 </h2></td>");
pc_client.print ("<td><h2 align='center'>");
pc_client.print (temp3);
pc_client.print ("</h2></td>");
pc_client.print ("</tr>");
}
if (temp4>30)
{
pc_client.print ("<tr bgcolor='red'>");
pc_client.print ("<td><h2 align='center'> Sensore Cameretta 2 </h2></td>");
pc_client.print ("<td><h2 align='center'>");
pc_client.print (temp4);
pc_client.print ("</h2></td>");
pc_client.print ("</tr>");
}
else
{
pc_client.print ("<tr bgcolor='green'>");
pc_client.print ("<td><h2 align='center'> Sensore Cameretta 2 </h2></td>");
pc_client.print ("<td><h2 align='center'>");
pc_client.print (temp4);
pc_client.print ("</h2></td>");
pc_client.print ("</tr>");
```

```
}
pc_client.print ("</p>");
pc_client.print ("</table>");
pc_client.print ("</body>");
pc_client.print ("</html>");
```

The man who invented the web

The World Wide Web is basically the work of a single man. Tim Barners-Lee, a computer scientist born in London in 1955. He was 25 when he started working at CERN. He wrote a simple program called “Enquire” to help him remember the connections amoung the people, computers, and projects at the lab. Enquire developed into a kind of “hypertext” notebook. Words in one document could be “linked” to other files on his PC. Barners-Lee invented four tools to create web:

- Browser: a program to allow users to view and move around the web on computer screens;
- HTML: a language for creating document to be placed on the web;
- URL: a system to establish a precise location for each page on the web
- HTTP: a set of rules to enable documents to be linked togheter on computer across the Internet

Sitografia

www.wikipedia.it
www.cisco.com
www.polinienrico.altervista.org
www.pic-ap-board.eu

Bibliografia

“Corso di telecomunicazioni” di Onelio Bertazioli
“English for New Technology” di Kiaran O’Malley