# Free Software & Open Hardware

*Arduino è una piattaforma hardware / software rilasciata sotto licenza Creative Commons che permette la realizzazione oggetti fisici interattivi, in grado di utilizzare un ricco insieme di sensori e attuatori. Il seminario illustra come creare una piccola stazione di rilevamento di dati ambientali basata su Linux e accessibile via web.*

Stefano Sanna
http://www.gerdavax.it

# Sommario

- Free Software e Open Hardware
- Gli esempi più famosi di Open Hardware
- Arduino: un successo tutto italiano!
- Panoramica su Arduino
- Stazione di rilevamento Linux-based & web-based

# Speaker

- Co-fondatore del GULCh (correva l'anno 1996!)
- Senior Developer & Java ME Tech Lead
  - beeweeb technologies
- Autore del libro “Java Micro Edition”
  - Hoepli Informatica, Milano, 2007
- Tech & Food blogger
  - http://www.gerdavax.it

# Free Software

# Free Software

- Libertà di eseguire il programma, per qualsiasi scopo (libertà 0).
- Libertà di studiare come funziona il programma e adattarlo alle proprie necessità (libertà 1). L'accesso al codice sorgente ne è un prerequisito.
- Libertà di ridistribuire copie in modo da aiutare il prossimo (libertà 2).
- Libertà di migliorare il programma e distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio (libertà 3). L'accesso al codice sorgente ne è un prerequisito.

# Oltre la percezione comune...

- Per molti (troppi?) utenti della Rete “free software” e “open source” sono sinonimi di

**SALVA CON NOME**

- I valori di condivisione della conoscenza, di miglioramento collettivo di una risorsa (il software), di verifica incrociata (degli algoritmi) sono completamente ignorati

# Open Hardware

- L'“open source” del software diventa “open diagram” dell'hardware, ovvero:
  - elenco di tutti i componenti
  - schema elettrico con tutte le connessioni
  - schema del PCB, cui cui realizzare immediatamente il circuito stampato e provvedere alla realizzazione del prodotto
  - libertà di progettare e costruire varianti del circuito di partenza

# Attenzione: l'hardware costa!

- A differenza del software, l'hardware **non può** essere replicato a costi pressoché nulli
- La produzione di circuiti e l'uso di componenti di media complessità fa lievitare i costi di realizzazione e il processo potrebbe non essere disponibile a livello dilettantistico/amatoriale
- Open Hardware non significa “hardware gratuito”, ma godere delle medesime libertà del free software

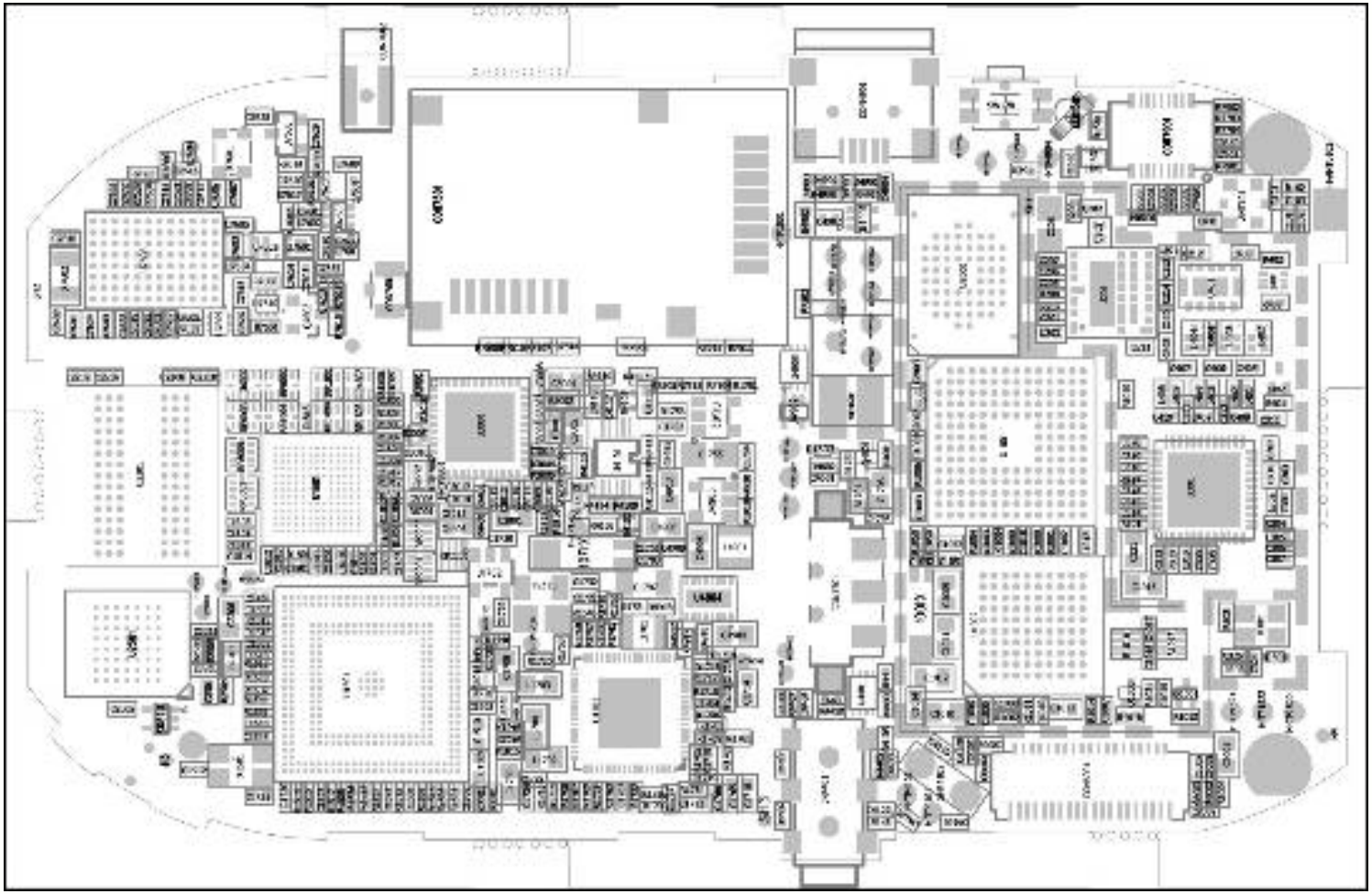
# Tre esempi importanti

- OpenMoko: free your phone!
  - Piattaforma hardware e software interamente open, basata su Linux
  - http://www.openmoko.org
- BeagleBoard
  - Piattaforma embedded hardware e software
  - http://www.beagleboard.org
- Arduino
  - Piattaforma di interaction design e sensor network

# OpenMoko FreeRunner

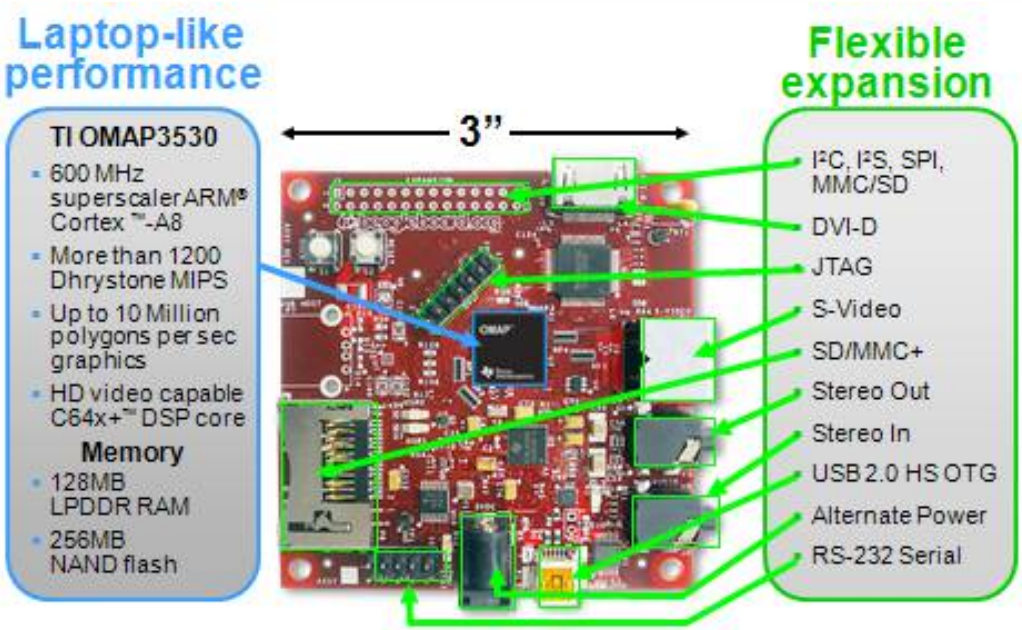
- Touch screen 2.84” (43mm x 58mm) 480x640 px
- 128MB SDRAM memory, 256 MB flash memory
- uSD slot supporting up to 8GB SDHC
- Internal GPS module
- Bluetooth e 802.11 b/g WiFi
- 400Mhz ARM processor
- Doppio accelrometro 3D
- Tri-band GSM and GPRS
- USB Host function with 500mA power

# FreeRunner “a nudo”



# BeagleBoard

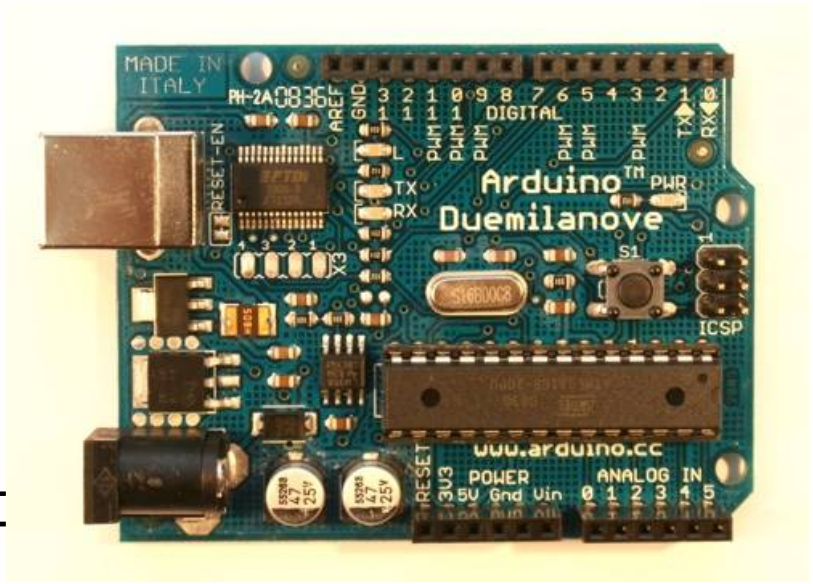
- Processore OMAP3530, oltre 1,200 Dhrystone MIPS
- OpenGL © ES 2.0 con acceleratore 2D/3D
- HD video con DSP TMS320C64x+ DSP (430MHz)
- Alimentazione via USB



# BeagleBoard

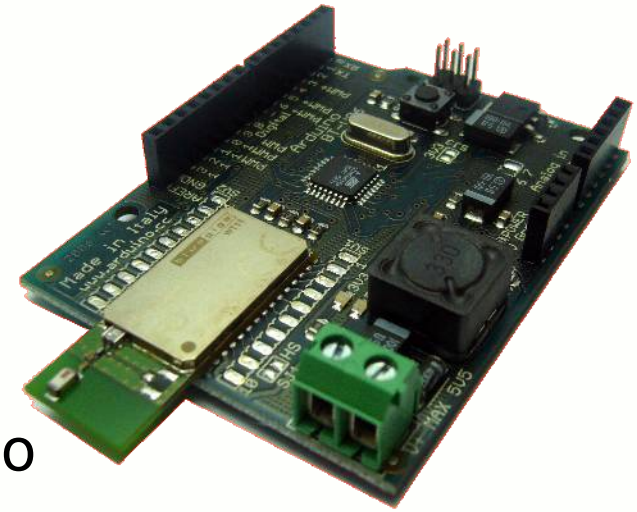

# Arduino

- E' un sistema economico, semplice, aperto, flessibile, programmabile per la realizzazione di sistemi in grado di acquisire dati da diversi sensori ed attivare diversi attuatori
- Il linguaggio di programmazione deriva dal C
- Ideato da Massimo Banzi
- Derivato da altri due progett open: Wiring e Processing
- **Arduino è orgogliosamente MADE IN ITALY!**



# Versioni di Arduino

- Prime versioni:
  - Arduino, Arduino USB, Arduino NG
- Version attuali:
  - Arduino Diecimila (“appena” sostituito da Arduino Duemilanove)
  - Arduino Mini
  - Arduino Nano
  - **Arduino Bluetooth**
  - Lilypad, Boarduino, Freeduino



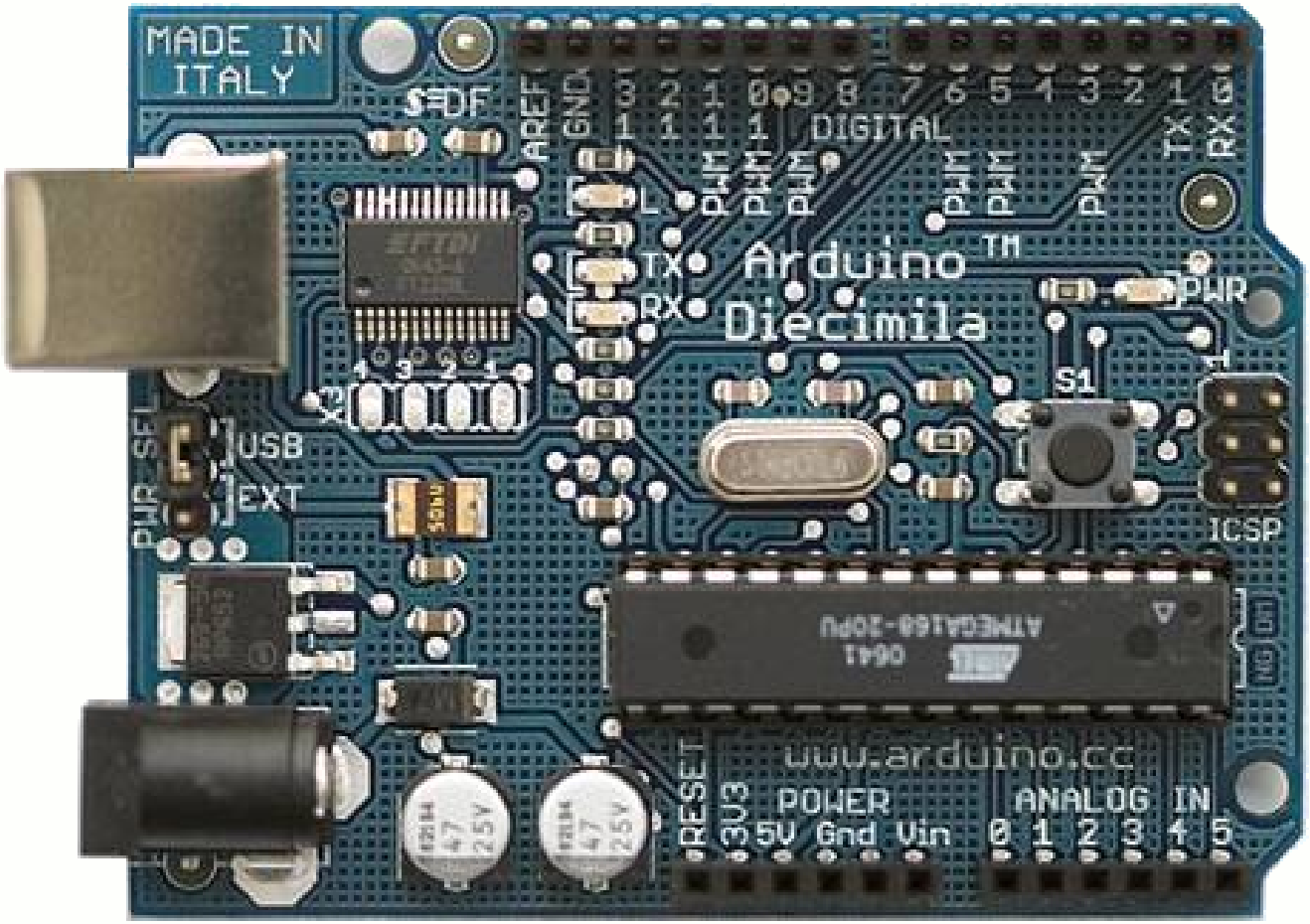
# Arduino FAQ

- Da dove si inizia?
  - Da qui: http://www.arduino.cc
- Dove si acquista?
  - http://www.smartprj.com
- Quando costa?
  - Il modello più economico (Arduino Duemilanove) costa meno di 30 euro
- Funziona solo con Linux?
  - No, funziona perfettamente anche con Windows e Mac OS X

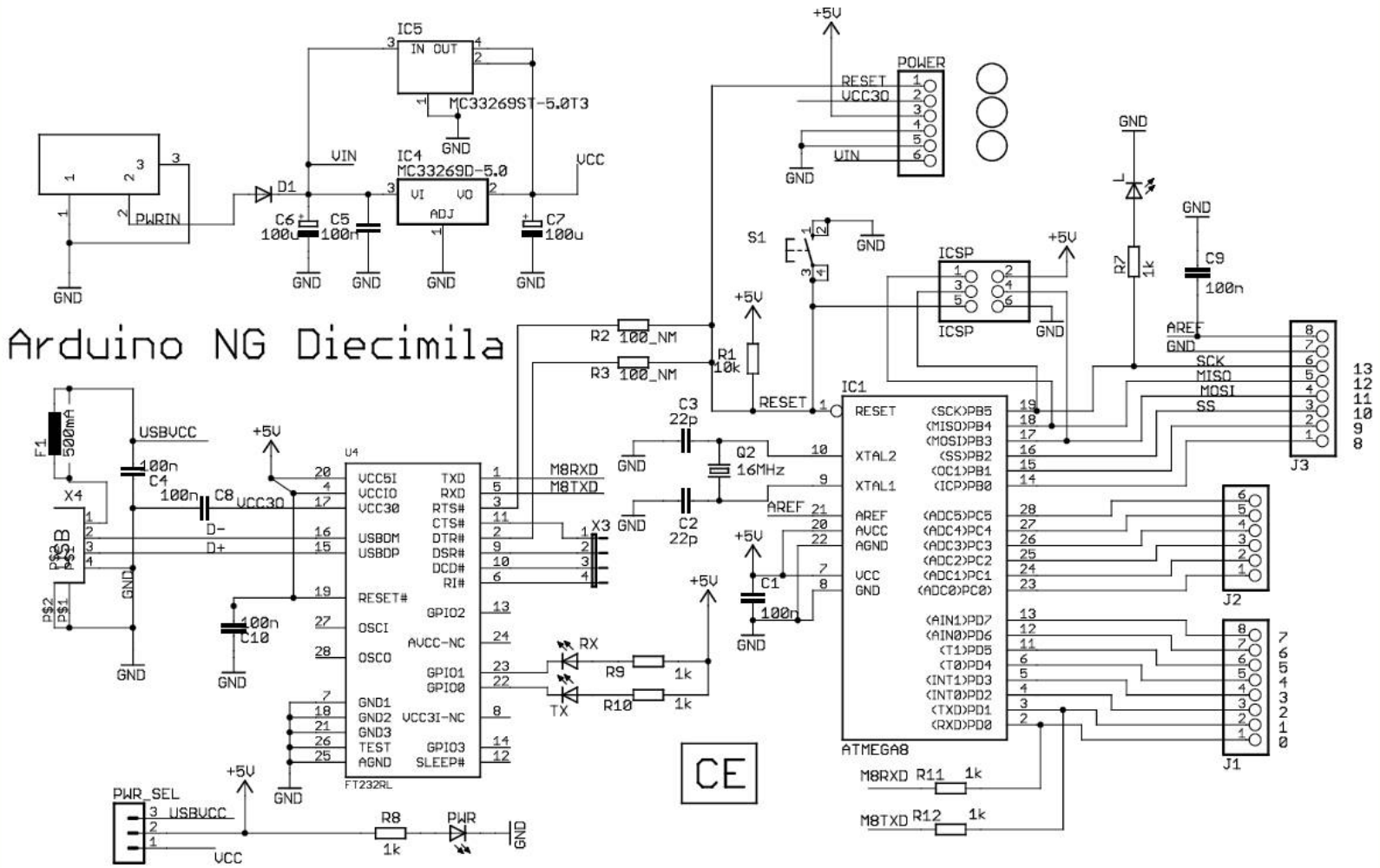
# Punti di forza di Arduino

- Completamente aperto: codice, diagrammi, librerie sono rilasciate sotto licenza Creative-Commons
- Molto molto molto economico!
- Facile da programmare e da interfacciare virtualmente con qualsiasi tipo di hardware
- E' supportato da una vastissima community di sviluppatori, appassionati, designer e progettisti
- E' un'ottima scuola per creare sensor network e sistemi di interaction design, prima di passare a piattaforme più complesse (e costose)

# Arduino Diecimila



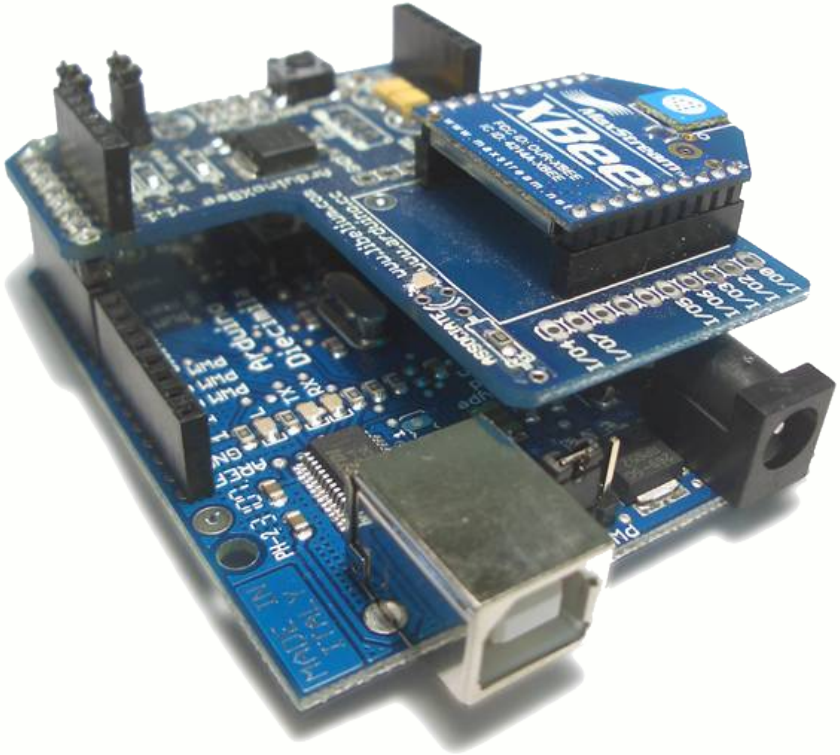
# Arduino: schema elettrico



# Caratteristiche hardware

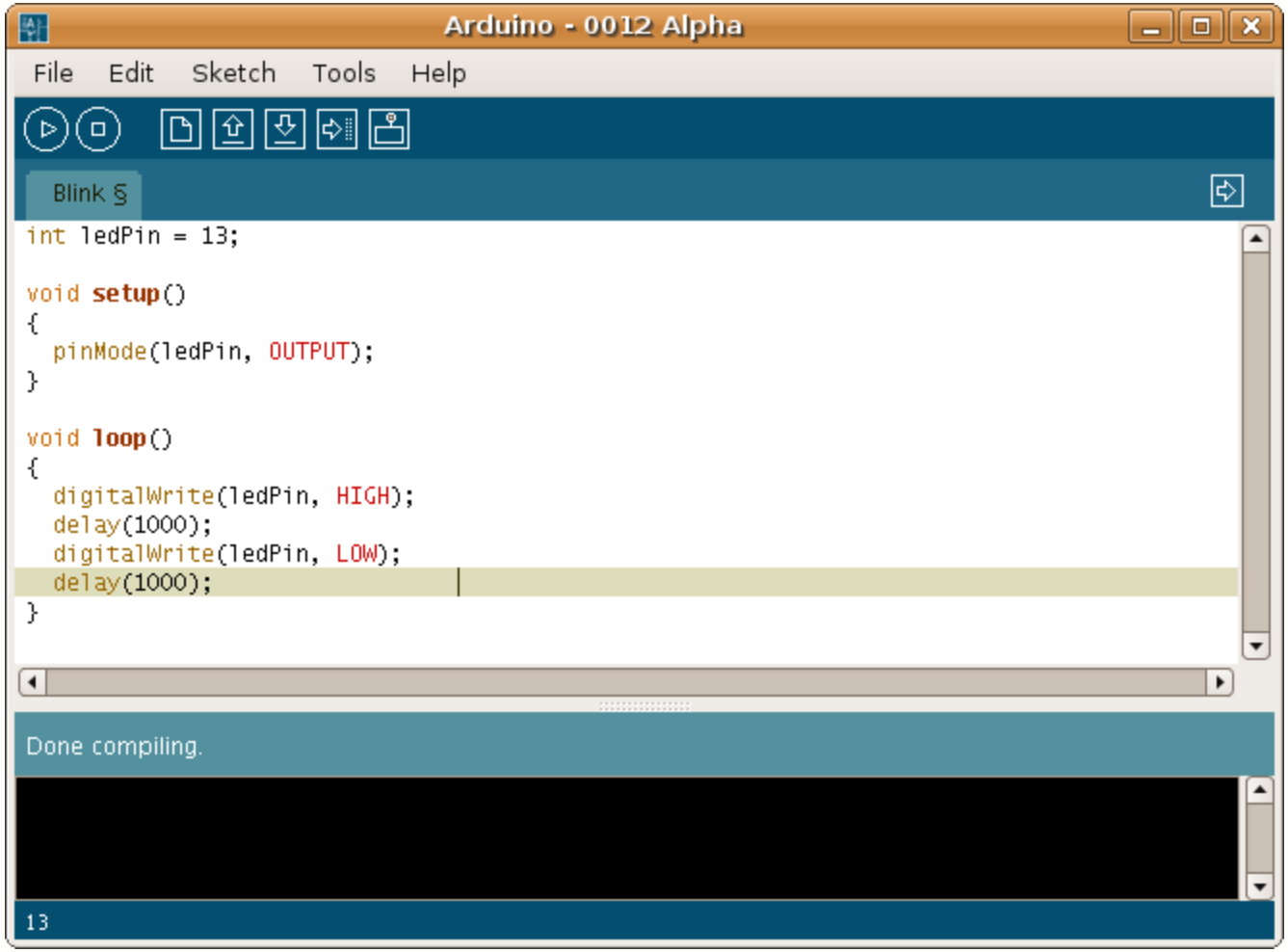
- Microcontroller Atmel Atmega168
- Alimentazione 6V-20V
- 14 ingressi/uscite digitali (6 uscite PWM)
- 6 ingressi analogici
- 1 porta seriale TTL
- 16KB di memoria flash (2KB bootloader)
- 1KB RAM
- EEPROM 512B
- Clock 16MHz

# Estensioni hardware





# Arduino IDE


Arduino - 0012 Alpha
File Edit Sketch Tools Help
Blink §
int ledPin = 13;
void setup()
{
pinMode(ledPin, OUTPUT);
}
void loop()
{
digitalWrite(ledPin, HIGH);
delay(1000);
digitalWrite(ledPin, LOW);
delay(1000);
}
Done compiling.
13

# Installazione

- Scaricare e decomprimere l'IDE di Arduino (verificare su http://www.arduino.cc l'ultima versione disponibile)

- Installare il compilatore e librerie per il microcontrollore AVR. Su Ubuntu:
  - sudo aptitude install **gcc-avr**
  - sudo aptitude install **avr-libc**

# Hello Arduino

```
int ledPin = 13;

void setup()
{
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  digitalWrite(ledPin, HIGH);
  delay(1000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

# Programmazione

- Linguaggio e librerie standard ereditano molte delle funzioni del C

- E possibile scrivere librerie native utilizzando il C o l'assembler standard del microcontroller

# Programmazione

- Variabili: byte, int, unsigned int, long, unsigned long, float, double, boolean, char, array
- Costanti specifiche:
  - Livelli logici: HIGH, LOW
  - Stati booleani: true, false
  - Configurazione dei PIN: INPUT, OUTPUT
- Operatori, funzioni, controllo di flusso, include di librerie...

# Libreria standard

- I/O digitale:
  - pinMode(PIN, INPUT | OUTPUT)
  - digitalWrite(PIN, HIGH | LOW)
  - int digitalRead(PIN)
- I/O analogico
  - int analogRead(PIN)
  - analogWrite(PIN, valore)
- Invio e lettura treni di bit, temporizzazioni, comunicazione seriale...

# Librerie estese “ufficiali”

- Accesso EEPROM
- Display LCD
- Motori: trazione diretta, stepper, servo
- Software Serial (no buffer)
- Interfaccia 1-Wire e I2C
- LED Matrix
- LED Sprite

# Librerie estese “community”

- Gestione date e orari
- Libreria di comunicazione per PC
- Vari driver per display LCD
- LED Driver
- Manipolazione stringhe
- Controller X10
- Controller PWM
- Interfacce per cellulari

# Sensori

- Sono disponibili moltissimi sensori analogici e digitali, in grado di rilevare pressoché qualsiasi grandezza fisica:
  - temperatura, luminosità
  - pressione, umidità, CO2
  - accelerazione, direzione goniometrica, posizione
  - sensori di contatto, prossimità, distanza
  - lettori RFID

# Attuatori

- Attraverso opportuni driver è possibile controllare virtualmente qualsiasi tipo di attuatore:
  - motori a trazione diretta, stepper, servo
  - display LED e LCD
  - buzzer
  - memory card

# Stazione di Rilevamento

- Arduino consente di realizzare una piccola interfaccia USB universale per qualsiasi tipo di sensore:
  - attraverso gli ingressi analogici e digitali è possibile leggere la grandezza fisica di interesse
  - il programma caricato sul microcontrollore si occupa di effettuare conversioni, verifiche sui dati acquisiti, eventuali interpolazioni o confronti da più sensori
  - il dato “ben formato” può essere inviato al PC attraverso la porta USB sotto forma di stringa di testo

# Sensore di temperatura LM35

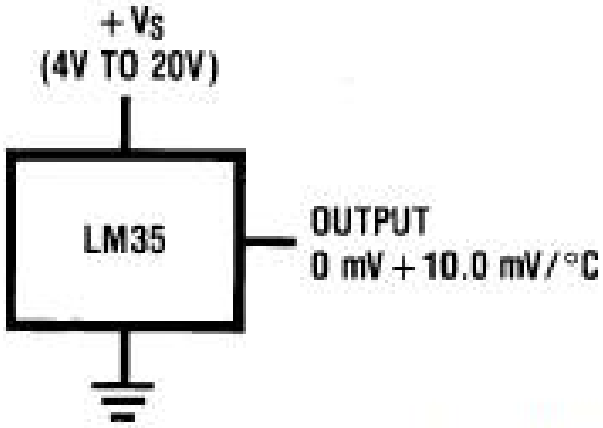
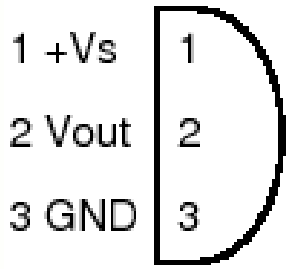



**FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)**

LM35
TO92



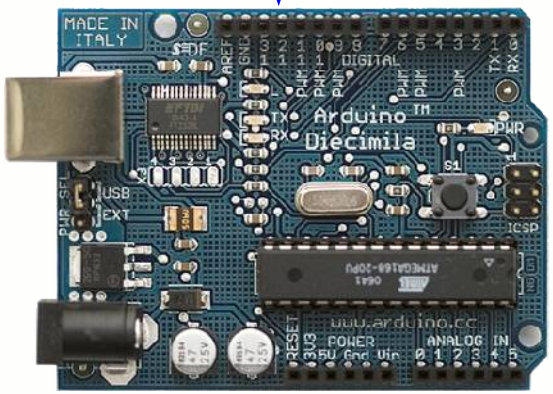

Fonte: http://www.national.com/ds/LM/LM35.pdf

# Chi fa cosa

**Jetty**

**Linux Day “Sensor Station”**

**JavaComm**



# Arduino: input analogico

- Attraverso i 6 ingressi analogici, Arduino è in grado di leggere tensioni comprese tra 0V e 5V, effettuando un campionamento a 10 bit

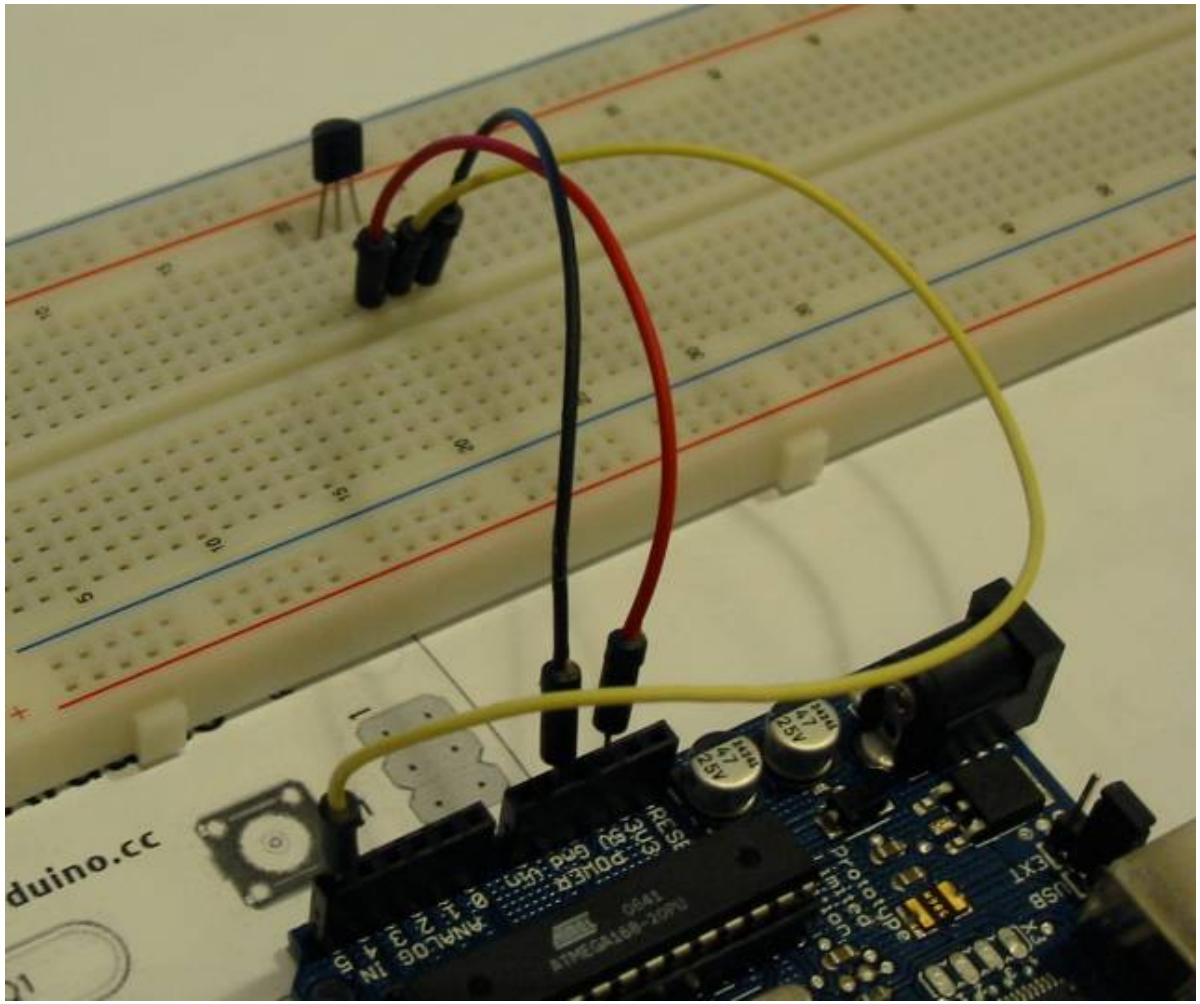
- La funzione analogRead() riceve come parametro il numero del PIN su cui effettuare la lettura e restituisce un valore compreso tra 0 e 1023

- Ogni “step” di analogRead() corrisponde a circa 4.9mV

# Arduino: comunicazione seriale

- I PIN digitali 0 e 1 sono connessi alla seriale TTL fornita dal microcontroller
- Due LED sulla board (pilotati dal controller USB FTDI) notificano il transito di dati in ingresso e in uscita
- La classe (!) Serial permette di gestire la porta:
  - Serial.begin(SPEED) imposta la velocità
  - int Serial.read() legge un byte
  - Serial.print(VALORE) invia un byte sulla seriale

# Cablaggio su breadboard



# Sketch su Arduino

```
int TEMP_PIN= 5;
int temp;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
  temp = analogRead(TEMP_PIN) * 0.5 + 2;
  Serial.println(temp, DEC);
  delay(1000);
}
```

Il valore letto da analogRead() è diviso per dieci, così da avere uno incremento di una unità per ogni grado centigrado

# Server Java

```
public class SensorStation
{
    private static String temp;
    private static String time;
    private Timer timer;
    private SerialWatcher serialWatcher;
    private StationHandler handler;
    private SerialPort serialPort;
    private BufferedReader reader;
```

# Server Java

```
public static void main(String[] args) {
  new SensorStation();
}

public SensorStation() {
  try {
    timer = new Timer();
    handler = new StationHandler();
    Server server = new Server(8080);
    server.setHandler(handler);
    server.start();
    serialWatcher = new SerialWatcher();
    timer.scheduleAtFixedRate(serialWatcher, 5000, 1000);
    initSerial();
  } catch (Exception ex) { }
}
```

# Inizializzazione porta seriale

```
private void initSerial() {
  try {
    CommPortIdentifier id =
CommPortIdentifier.getPortIdentifier("/dev/ttyUSB0");

    serialPort = (SerialPort) id.open("GULCh", 1000);

    serialPort.setSerialPortParams(9600,
SerialPort.DATABITS_8, SerialPort.STOPBITS_1,
SerialPort.PARITY_NONE);

    reader = new BufferedReader(new
InputStreamReader(serialPort.getInputStream()));

  } catch (Exception e) {  System.exit(1); }
}
```

# Lettura dei dati

```
private class SerialWatcher extends TimerTask {
  public void run() {
    String read = null;

    if (reader != null) {
      try {
        while ((read = reader.readLine()) != null) {

          time = (new
Date(System.currentTimeMillis())).toString();

          temp = read;
        }
      } catch (Exception e)  { }
    }
  }
}
```

# Modulo web (1)

```
private class StationHandler extends AbstractHandler {
  public void handle(String target, HttpServletRequest
request, HttpServletResponse response, int dispatch)
                throws IOException, ServletException {

    response.setContentType("text/html");
    response.setStatus(HttpServletResponse.SC_OK);
    PrintWriter writer = response.getWriter();
    writer.println("<html>");
    writer.println("<head>");
    writer.println("<meta http-equiv=\"refresh\"
content=\"10\">");
    writer.println("</head>");
    writer.println("<body>");
    writer.println("<h1>Linux Sensor Station</h1>");
    writer.println("<hr/>");
```

# Modulo web (2)

```
    writer.println("<p>");
    writer.println("Last temperature read: " + temp + " @
" + time);
    writer.println("</p>");
    writer.println("</body>");
    writer.println("</html>");

    ((Request) request).setHandled(true);
  }
}
```

# Rullo di tamburi...

# DEMO

# Conclusioni

- Lo spirito del Free Software muove i primi passi nel mondo dell'hardware e i primi progetti sono validi ed estremamente interessanti

- Arduino è una delle piattaforme di hardware aperto di maggior successo: le sue caratteristiche lo rendono estremamente flessibile e adattabile a numerosi contesti di utilizzo
- Utilizzando Arduino, Linux e un po' di codice Java è possibile realizzare una piccola stazione di rilevamento accessibile via web