

# Laboratorio di Segnali e Sistemi - Capitolo 9 -

## Introduzione ad Arduino



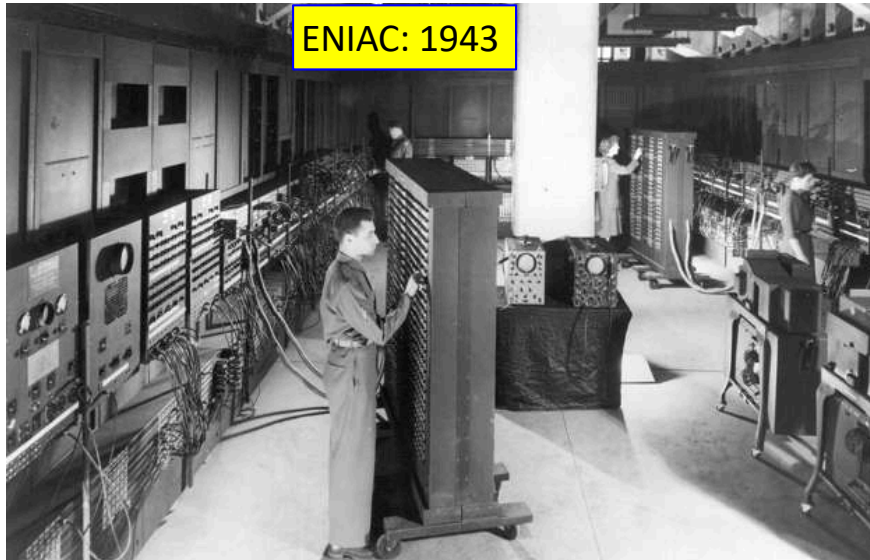
Claudio Luci  
SAPIENZA  
UNIVERSITÀ DI ROMA

*last update : 070117*

# Sommario del capitolo:

- Cenni all'architettura di un computer
- Cenni al microprocessore
- Dal microprocessore al microcontrollore
- il Progetto Arduino
- Comunicazione tra Arduino e il computer host
- Pulse Width Modulation (PWM)
- il DAC di Arduino Due
- ADC di Arduino

# Computer e tecnologia



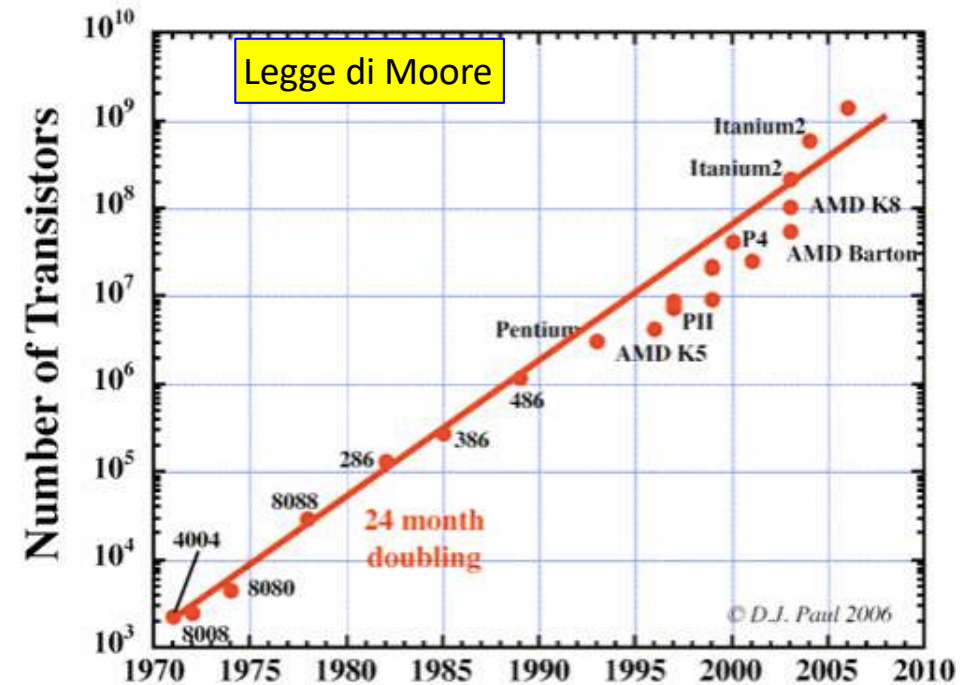
ENIAC: 1943

È il primo computer elettronico omni purpose della storia. Era fatto da 18000 valvole e 1500 rele'.

## DOWNSIZING AND UPGRADING

The inception of computing inspired a remarkable race for faster, smaller, lighter, cheaper hardware.

|               | ENIAC                       | Intel Core Duo chip  |
|---------------|-----------------------------|----------------------|
| Debut         | 1946                        | 2006                 |
| Performance   | 5,000 addition problems/sec | 21.6 billion ops/sec |
| Power use     | 170,000 watts               | 31 watts max         |
| Weight        | 28 tons                     | negligible           |
| Size          | 80' w x 8' h                | 90.3 sq. mm.         |
| What's inside | 17,840 vacuum tubes         | 151 B M transistors  |
| Cost          | \$487,000                   | \$637                |



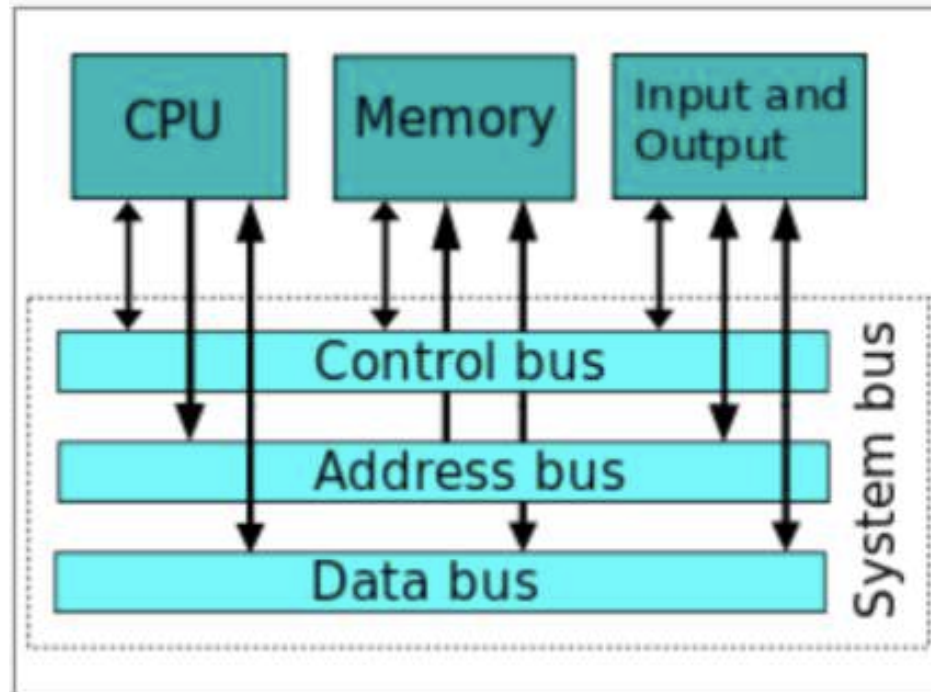
Prima legge di Moore (enunciata nel 1971): la complessità di un microcircuito, misurata ad esempio tramite il numero di transistor per chip, raddoppia ogni 18 mesi.

| Generation | Approximate Dates | Technology                         | Typical Speed (operations per second) |
|------------|-------------------|------------------------------------|---------------------------------------|
| 1          | 1946–1957         | Vacuum tube                        | 40,000                                |
| 2          | 1958–1964         | Transistor                         | 200,000                               |
| 3          | 1965–1971         | Small and medium scale integration | 1,000,000                             |
| 4          | 1972–1977         | Large scale integration            | 10,000,000                            |
| 5          | 1978–1991         | Very large scale integration       | 100,000,000                           |
| 6          | 1991-             | Ultra large scale integration      | 1,000,000,000                         |

# Architettura di un computer

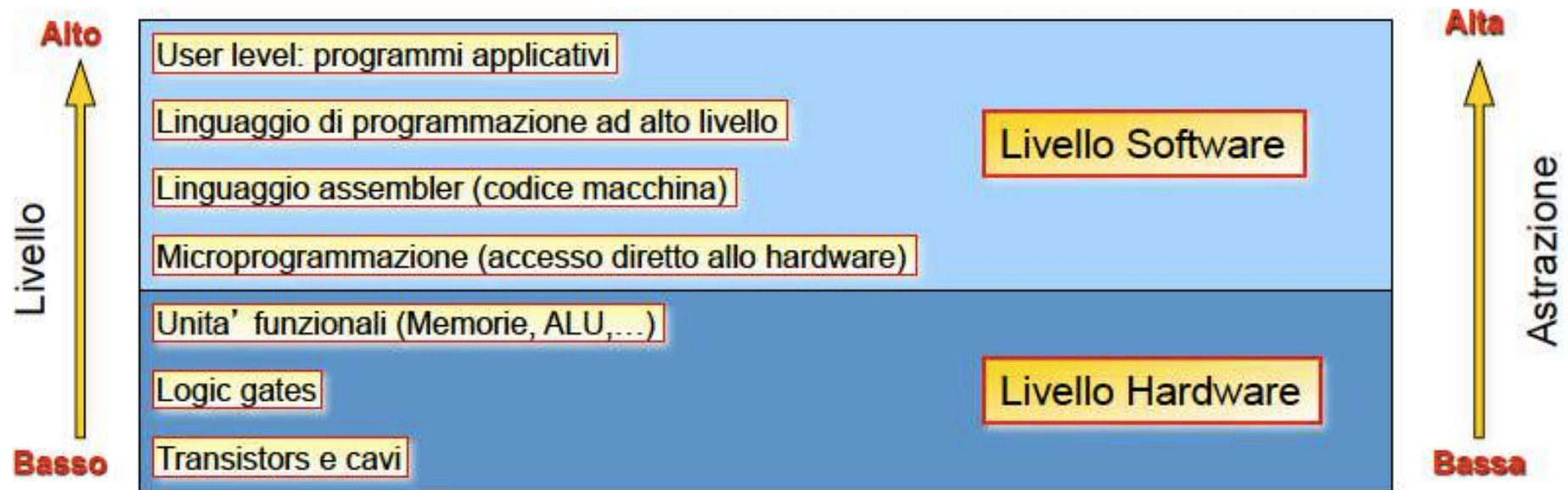
- ❑ Gran parte dei processori moderni sono basati sull'architettura di John von Neumann, che contribuì a svilupparla per l'EDVAC, che può essere considerato il successore dell'ENIAC.

Macchina di von Neumann



- ❑ Nei primi computer la CPU era costituita da un enorme volume di circuiti elettronici;
- ❑ Oggi i progressi fatti nel campo dell'integrazione hanno reso possibile racchiudere tutte le funzioni in un unico chip di silicio, in cui sono racchiuse migliaia o decine di migliaia di porte logiche elementari.
- ❑ **Questi integrati prendono il nome di microprocessori.**

# Livelli di astrazione di un architettura di calcolo



- Un architettura di calcolo puo' essere scomposta in diversi *livelli di astrazione*
- Questa descrizione definisce interfacce chiare tra funzionalita' diverse nascondendo i dettagli del singolo livello
  - Un cambiamento di un componente di un certo livello non comporta (non dovrebbe comportare...) cambiamenti negli altri livelli
- L'*astrazione* cresce dai livelli hardware fino al livello applicativo

# Introduzione ai microprocessori

- In base a quanto abbiamo visto finora studiando l'elettronica digitale, fondamentalmente abbiamo la corrispondenza:

**un progetto → un circuito (hardware)**


- La risoluzione di un problema con questa tecnica porta alla soluzione in **logica cablata** dello stesso; la soluzione si dice che è solo di tipo **hardware**.
- Questo modo di procedere aveva portato allo sviluppo di nuove applicazioni nei campi dell'elettronica e dell'automazione, ma alla fine degli anni 60 ci si era trovati davanti ad un vicolo cieco dato che ogni nuova applicazione richiedeva una complessa e costosa fase progettuale e realizzativa.
- Con l'avvento e la commercializzazione a basso costo del microprocessore lo sviluppo di un progetto viene fatto con la corrispondenza:

**un progetto → un circuito (hardware) + un programma specifico (software)**

- Fermo restando che si ha comunque una parte difficile da realizzare (l'hardware del microprocessore e la circuiteria di supporto), ciascun progetto (di una stessa tipologia) può essere realizzato ponendo in esecuzione una particolare procedura di funzionamento dell'hardware data dal **programma** specifico per quel progetto.
- Eseguendo un altro programma, il sistema funziona in un altro modo completamente differente pur utilizzando la stessa piattaforma hardware.
- L'avvento del microprocessore ha rivoluzionato con l'informatica il modo di vivere e di lavorare dell'uomo.

Il microprocessore è una CPU contenuta interamente all'interno di un unico chip

I computer (e i microprocessori) si contraddistinguono dalla dimensione del bus dei dati e da quella del bus degli indirizzi.

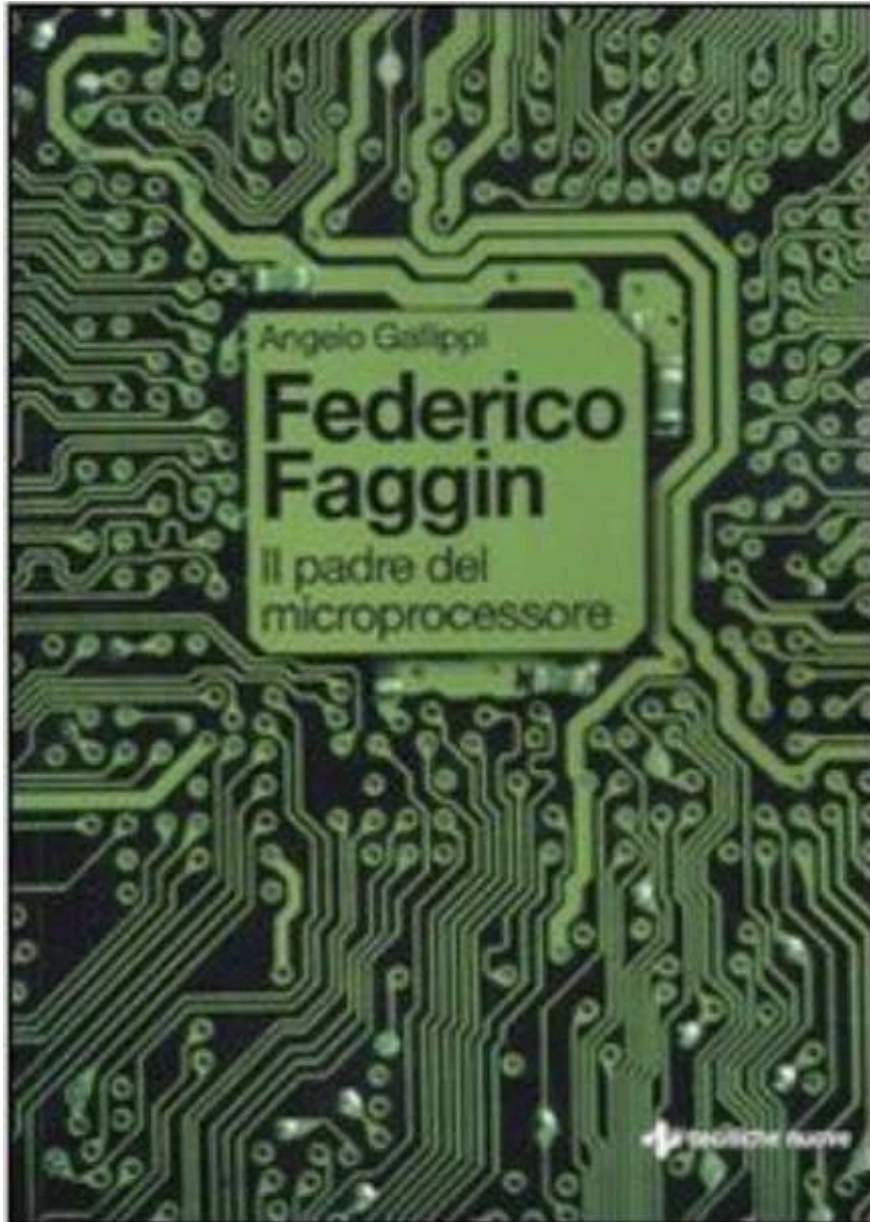


| Costruttore | Sigla | Dati<br>(bit) | Indirizzi<br>(bit) |
|-------------|-------|---------------|--------------------|
| Intel       | 8080  | 8             | 16                 |
| Motorola    | 6800  | 8             | 16                 |
| Zilog       | Z80   | 8             | 16                 |
| Intel       | 8086  | 16            | 16                 |
| Motorola    | 68000 | 16            | 16                 |
| Intel       | 80386 | 32            | 32                 |
| Motorola    | 68020 | 32            | 32                 |

Tabella 1: *Alcuni microprocessori "storici"*

I computer (e i microprocessori) sono macchine "sincrone": la sequenza delle operazioni è scandita da un clock.

# Federico Faggin



Federico Faggin  
il padre del microprocessore

Autore: Angelo Gallippi

Da wikipedia

**Federico Faggin** (Vicenza, 1° dicembre 1941) è un fisico, inventore e imprenditore italiano. Dal 1968 Faggin risiede negli Stati Uniti ed ha assunto anche la cittadinanza statunitense. Fu capo progetto dell'Intel 4004, il primo microprocessore al mondo, e di tutti i primi microprocessori dell'Intel (8008, 4040 e 8080) e creò anche l'architettura del 4040 e dell'8080, il primo microprocessore ad alta prestazione. Fu anche lo sviluppatore della tecnologia MOS con porta di silicio (MOS silicon gate technology), un contributo fondamentale che permise la fabbricazione dei primi microprocessori e delle memorie EPROM e RAM dinamiche e sensori CCD, gli elementi essenziali per la digitalizzazione dell'informazione. Nel 1974 fondò e diresse la ditta Zilog, la prima ditta dedicata esclusivamente ai microprocessori, presso cui dette vita al famoso microprocessore Z80 ancora in produzione fino ai nostri giorni. Nel 1986 Faggin fondò e diresse la Synaptics, ditta che sviluppò i primi Touchpad e Touch screen.

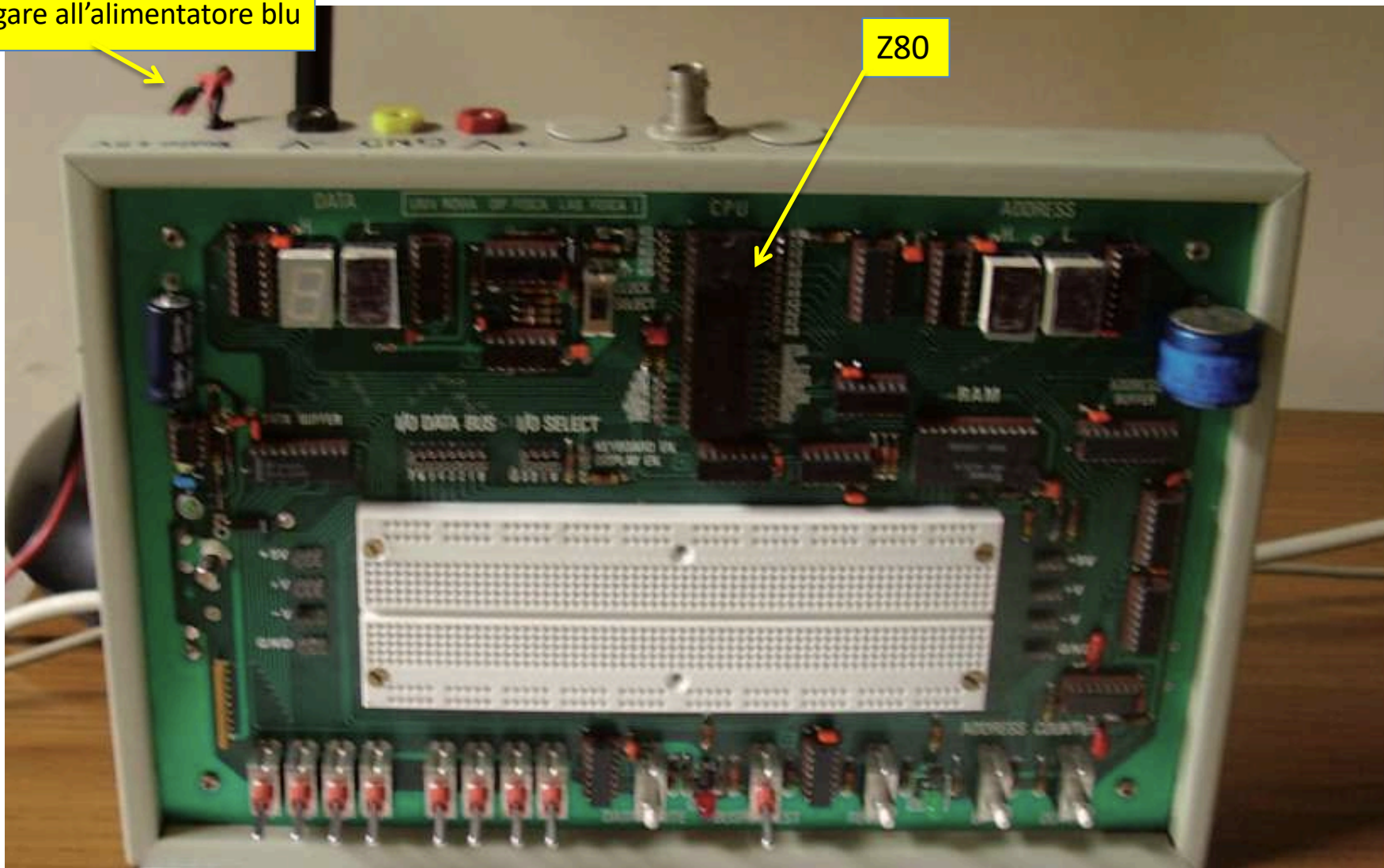


# La scheda didattica Z80

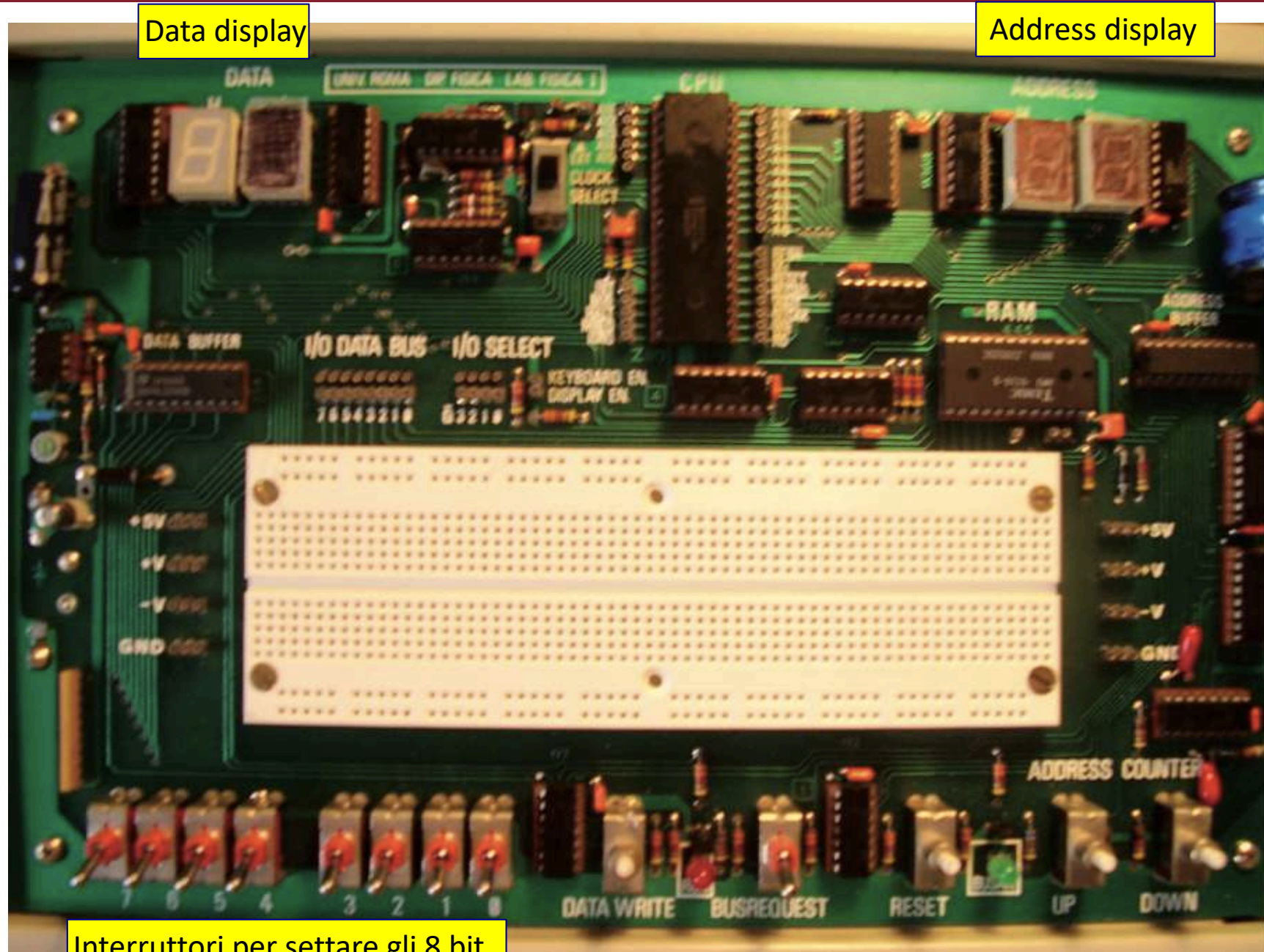
- Questa scheda fu realizzata dallo studente I.Vannucci come progetto nell'ambito del corso di Laboratorio di Fisica 2 (un esame del quarto d'anno di Fisica) nel 1986

Alimentazione: 5 V

Da collegare all'alimentatore blu

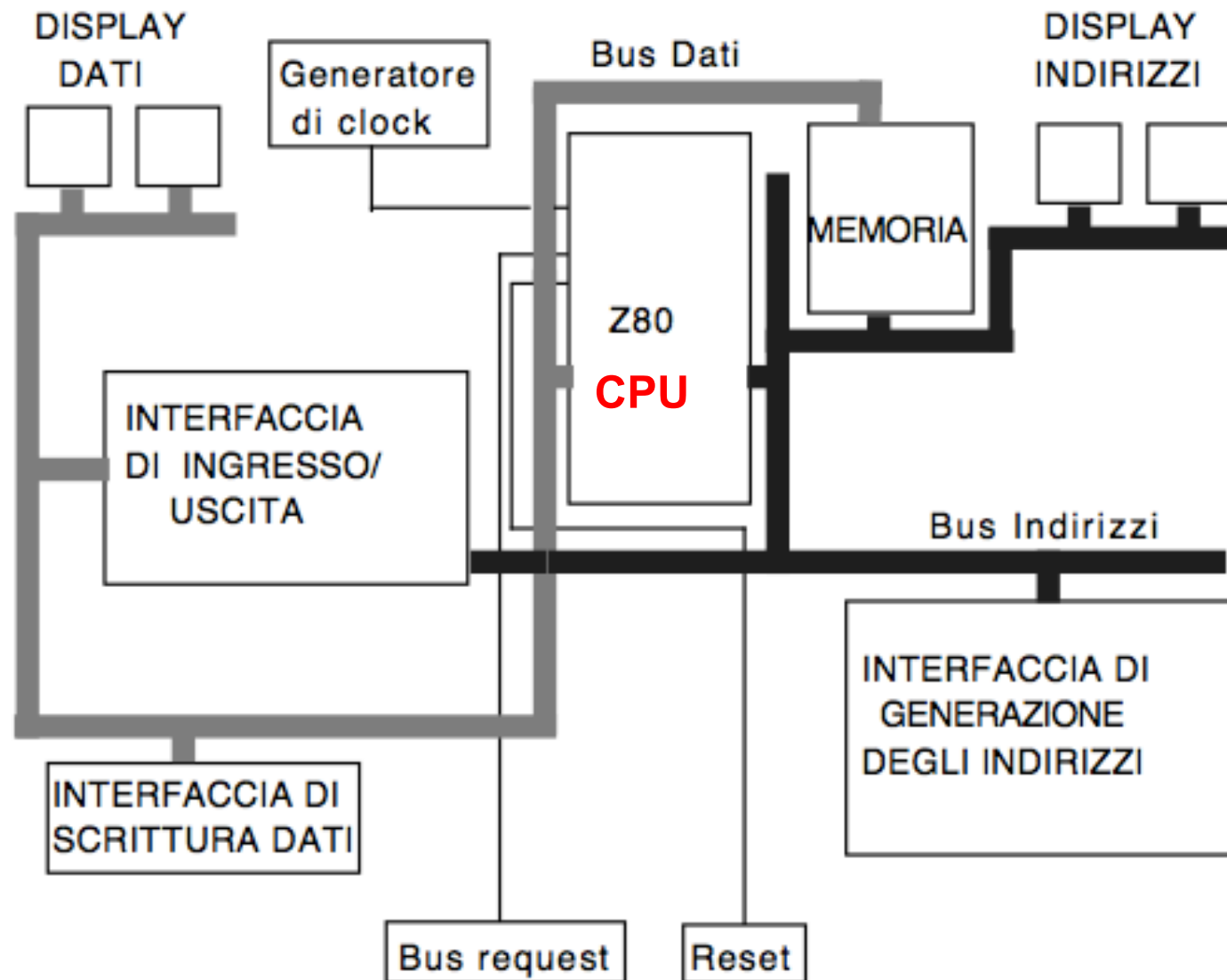


# La scheda didattica Z80



Interruttori per settare gli 8 bit

# Scheda didattica Z80: schema a blocchi



# Il microcontrollore

## Microprocessore

Un microprocessore integra sul chip la logica di elaborazione ma richiede sempre delle unità esterne ( memorie, gestori di segnali e dispositivi periferici) per poter scambiare informazioni e interagire con l'esterno.

## Microcontrollore

Un microcontrollore è invece un sistema completo, che integra in uno stesso chip il processore, la memoria permanente, la memoria volatile e i canali (pin) di I/O, oltre ad eventuali altri blocchi specializzati.

È progettato per interagire direttamente con il mondo esterno tramite un programma residente nella propria memoria interna e mediante l'uso di pin specializzati o configurabili dal programmatore. Sono disponibili in 3 fasce di capacità elaborativa (ampiezza del bus dati): 8 bit, 16 bit e 32 bit.

- ❑ Il primo microcontrollore fu il TMS 1000, creato da Gary Boone and Michael Cochran nel 1971 e commercializzato nel 1974.
- ❑ Esso combinava read-only memory (ROM), read/write memory (RAM), processor (CPU) e clock in un singolo chip.



A TMS1000 "computer on a chip".

## Componenti di un microcontrollore:

- CPU
- Memoria di programma: ROM, EPROM, FLASH
- Memoria dati: RAM, EPROM
- Oscillatore
- Porte di I/O
- Gestione di Interrupt
- Timer, contatori
- Moduli di comunicazione (SPI, I2C, USB....)
- ADC, DAC, PWM

## Sistema di sviluppo:

L'insieme di strumenti (hardware e software) necessari per generare il codice che deve essere eseguito dal processore, metterlo a punto e collaudarlo.

Arduino utilizza i microcontrollori della ATMEL corporation

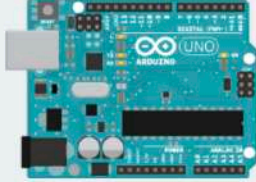
PROFESSIONAL EDUCATION STORE  SIGN IN


ARDUINO


HARDWARE SOFTWARE DOCUMENTATION COMMUNITY BLOG ABOUT


Learn about Arduino Response to the COVID-19 outbreak

WHAT IS ARDUINO?



BUY AN ARDUINO 

LEARN ARDUINO 

DONATE 

LATEST

## ARDUINO EDUVISION

Watch the whole Season 2

10 episodes with amazing guests and topics: learn about IoT, space missions, engineering skills, coding at an early age and much more!

Available on YouTube

[Watch it now!](#)





### New Portenta Vision Shield

Low power embedded computer vision made easy

[Check it out now!](#)

The Arduino® Student Kit: bring the buzz home



NEW BOARDS AND SHIELDS AND KITS, OH MY! CHECK THEM OUT!



BLOG

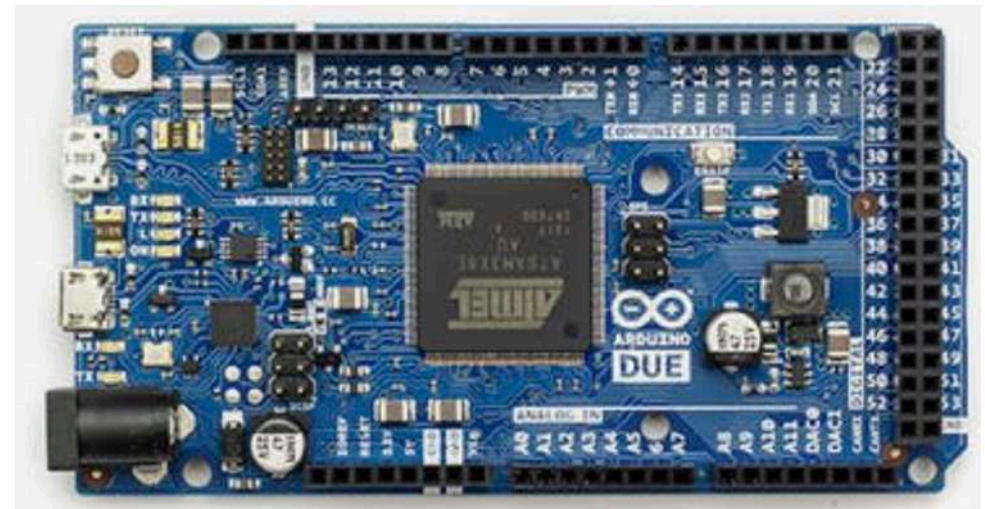
BLOG

TRANSGENDER AWARENESS WEEK: SPREADING GENDER AWARENESS IN THE ARDUINO COMMUNITY

BLOG

# Che cos'è Arduino?

- ❑ Il progetto Arduino è nato a Ivrea nel 2005 per permettere agli studenti dell'Istituto di Design Interattivo di realizzare in modo rapido e semplice (e a basso costo) dei prototipi senza avere una conoscenza approfondita di elettronica o programmazione.
- ❑ L'intero progetto, hardware e software, è "open source", ovvero chiunque può accedere agli schemi della scheda Arduino e riprodurla. Questo ha permesso una rapida diffusione su scala globale in tutti gli ambiti, sia lavorativi che didattici.
- ❑ Come corollario ad Arduino ci sono tutta una serie di sensori e software interfacciabili ad Arduino che ne permette svariati utilizzi in tutti i campi; non c'è limite alla fantasia.
- ❑ Noi abbiamo utilizzato la scheda **Arduino Uno** basata sul microcontrollore Atmel AT328, ma nel 2019 siamo passati ad **Arduino Due** basato sul microcontrollore a 32 bit Atmel SAM3X8E ARM (non esistono solo Arduino Uno e Arduino Due, ma ce ne sono diversi altri, lo scoprirete visitando il sito di Arduino).



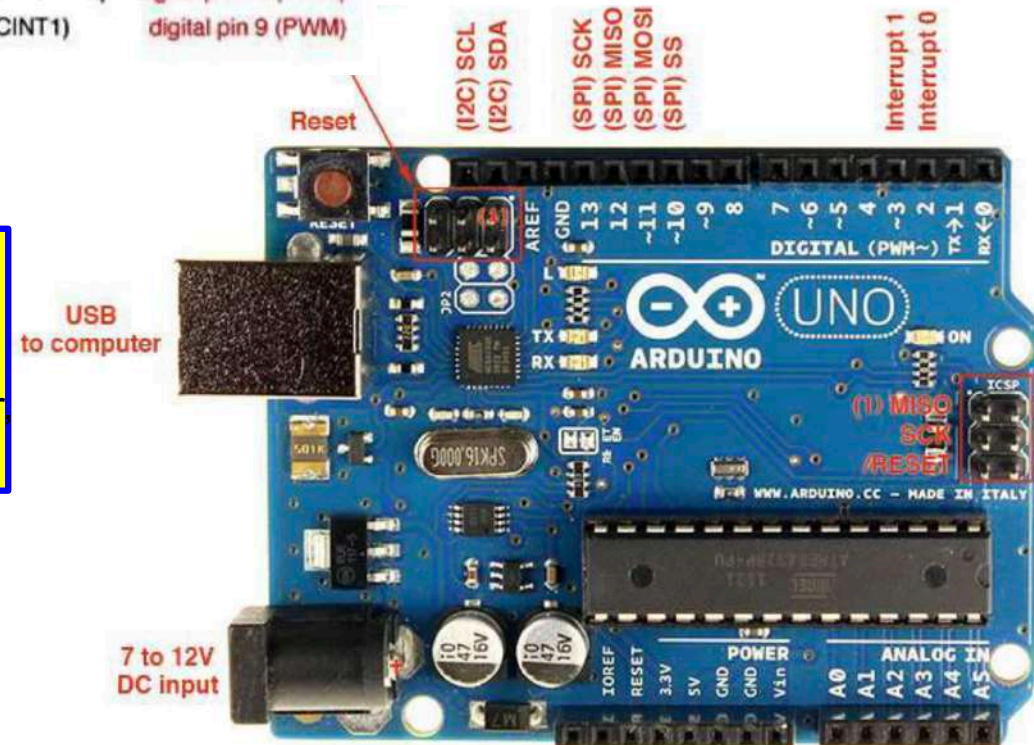
La differenza principale è la presenza del DAC

# Arduino Uno: pin mapping dell'AT328

| Arduino function    | Microcontroller pin      | Microcontroller pin | Arduino function     |
|---------------------|--------------------------|---------------------|----------------------|
| reset               | (PCINT14/RESET) PC6      | 1                   | analog input 5       |
| digital pin 0 (RX)  | (PCINT16/RXD) PD0        | 2                   | analog input 4       |
| digital pin 1 (TX)  | (PCINT17/TXD) PD1        | 3                   | analog input 3       |
| digital pin 2       | (PCINT18/INT0) PD2       | 4                   | analog input 2       |
| digital pin 3 (PWM) | (PCINT19/OC2B/INT1) PD3  | 5                   | analog input 1       |
| digital pin 4       | (PCINT20/XCK/T0) PD4     | 6                   | analog input 0       |
| VCC                 | VCC                      | 7                   | GND                  |
| GND                 | GND                      | 8                   | analog reference     |
| crystal             | (PCINT6/XTAL1/TOSC1) PB6 | 9                   | VCC                  |
| crystal             | (PCINT7/XTAL2/TOSC2) PB7 | 10                  | digital pin 13       |
| digital pin 5 (PWM) | (PCINT21/OC0B/T1) PD5    | 11                  | digital pin 12       |
| digital pin 6 (PWM) | (PCINT22/OC0A/AIN0) PD6  | 12                  | digital pin 11(PWM)  |
| digital pin 7       | (PCINT23/AIN1) PD7       | 13                  | digital pin 10 (PWM) |
| digital pin 8       | (PCINT0/CLKO/ICP1) PB0   | 14                  | digital pin 9 (PWM)  |
|                     |                          | 15                  |                      |
|                     |                          | 16                  |                      |
|                     |                          | 17                  |                      |
|                     |                          | 18                  |                      |
|                     |                          | 19                  |                      |
|                     |                          | 20                  |                      |
|                     |                          | 21                  |                      |
|                     |                          | 22                  |                      |
|                     |                          | 23                  |                      |
|                     |                          | 24                  |                      |
|                     |                          | 25                  |                      |
|                     |                          | 26                  |                      |
|                     |                          | 27                  |                      |
|                     |                          | 28                  |                      |


La scheda Arduino facilita l'accesso ai pin del microcontrollore, mette qualche protezione, dei diodi di visualizzazione dello stato, un'interfaccia USB con il computer, ma niente altro (ma tutto questo non e' poco!)

Vedremo più avanti le funzionalità che useremo in questo laboratorio.






# Arduino Uno versus Arduino Due

|                             |                                                       |                                                                                   |
|-----------------------------|-------------------------------------------------------|-----------------------------------------------------------------------------------|
| Microcontroller             | ATmega328P                                            |  |
| Operating Voltage           | 5V                                                    |                                                                                   |
| Input Voltage (recommended) | 7-12V                                                 |                                                                                   |
| Input Voltage (limit)       | 6-20V                                                 |                                                                                   |
| Digital I/O Pins            | 14 (of which 6 provide PWM output)                    |                                                                                   |
| PWM Digital I/O Pins        | 6                                                     |                                                                                   |
| Analog Input Pins           | 6 (ADC)                                               |                                                                                   |
| DC Current per I/O Pin      | 20 mA                                                 |                                                                                   |
| DC Current for 3.3V Pin     | 50 mA                                                 |                                                                                   |
| Flash Memory                | 32 KB (ATmega328P) of which 0.5 KB used by bootloader |                                                                                   |
| SRAM                        | 2 KB (ATmega328P)                                     |                                                                                   |
| EEPROM                      | 1 KB (ATmega328P)                                     |                                                                                   |
| Clock Speed                 | 16 MHz                                                |                                                                                   |
| LED_BUILTIN                 | 13                                                    |                                                                                   |
| Length                      | 68.6 mm                                               |                                                                                   |
| Width                       | 53.4 mm                                               |                                                                                   |
| Weight                      | 25 g                                                  |                                                                                   |



|                                          |                                                |                                                                                     |
|------------------------------------------|------------------------------------------------|-------------------------------------------------------------------------------------|
| Microcontroller                          | AT91SAM3X8E                                    |  |
| Operating Voltage                        | 3.3V                                           |                                                                                     |
| Input Voltage (recommended)              | 7-12V                                          |                                                                                     |
| Input Voltage (limits)                   | 6-16V                                          |                                                                                     |
| Digital I/O Pins                         | 54 (of which 12 provide PWM output)            |                                                                                     |
| Analog Input Pins                        | 12 (ADC)                                       |                                                                                     |
| Analog Output Pins                       | 2 (DAC) (DAC)                                  |                                                                                     |
| Total DC Output Current on all I/O lines | 130 mA                                         |                                                                                     |
| DC Current for 3.3V Pin                  | 800 mA                                         |                                                                                     |
| DC Current for 5V Pin                    | 800 mA                                         |                                                                                     |
| Flash Memory                             | 512 KB all available for the user applications |                                                                                     |
| SRAM                                     | 96 KB (two banks: 64KB and 32KB)               |                                                                                     |
| Clock Speed                              | 84 MHz                                         |                                                                                     |
| Length                                   | 101.52 mm                                      |                                                                                     |
| Width                                    | 53.3 mm                                        |                                                                                     |
| Weight                                   | 36 g                                           |                                                                                     |

# ”comunicare” con Arduino

Potete consultare la guida di Arduino: <https://www.arduino.cc/en/Guide>

- ❑ Arduino comunica con un computer host (Win, Mac, Linux) tramite una porta USB
- ❑ Il cavo di collegamento, per Arduino Due, è uguale al cavo di alimentazione dei (vecchi) smartphone
- ❑ È necessario installare sull’host il software **Arduino IDE** (Integrated Development Environment) che:
  - Permette di scrivere il codice del programma (chiamato **sketch**) in (pseudo) C;
  - Carica lo sketch nella board di Arduino e lo fa partire.
- ❑ Se si ha una connessione di rete affidabile, si può usare **Online Arduino IDE** (*Arduino Web Editor*), invece del **Desktop Arduino IDE** (che utilizzeremo in laboratorio).
  - In questo modo gli sketches verranno salvati sul cloud di Arduino.
  - Anche il tipo di board di Arduino dovrebbe essere riconosciuta automaticamente.  
*Remember that boards work out-of-the-box on the Web Editor, no need to install anything.*
- ❑ La programmazione è in C, anche se ci sono delle piccole variazioni (per facilitare la scrittura)

Se collegate Arduino al computer tramite il cavo USB non è necessario alimentarlo separatamente, perché l’alimentazione é presa via porta USB.  
In laboratorio usatelo sempre in questo modo.

# Arduino reference

La programmazione di Arduino usa alcune librerie. L'elenco lo trovate qui: <https://www.arduino.cc/reference/en/>

## Ad esempio:

### Digital I/O

`digitalRead()`  
`digitalWrite()`  
`pinMode()`

### Analog I/O

`analogRead()`  
`analogReference()`  
`analogWrite()`

### Zero, Due & MKR Family

⇒ `analogReadResolution()`  
 ⇒ `analogWriteResolution()`

### Time

`delay()`  
`delayMicroseconds()`  
`micros()`  
`millis()`

### Math

`abs()`  
`constrain()`  
`map()`  
`max()`  
`min()`  
`pow()`  
`sq()`  
`sqrt()`

### Trigonometry

`cos()`  
`sin()`  
`tan()`

### Conversion

`(unsigned int)`  
`(unsigned long)`  
`byte()`  
`char()`  
`float()`  
`int()`  
`long()`  
`word()`

### Random Numbers

`random()`  
`randomSeed()`

### Bits and Bytes

`bit()`  
`bitClear()`  
`bitRead()`  
`bitSet()`  
`bitWrite()`  
`highByte()`  
`lowByte()`

### Arithmetic Operators

`% (remainder)`  
`* (multiplication)`  
`+ (addition)`  
`- (subtraction)`  
`/ (division)`  
`= (assignment operator)`

### Data Types

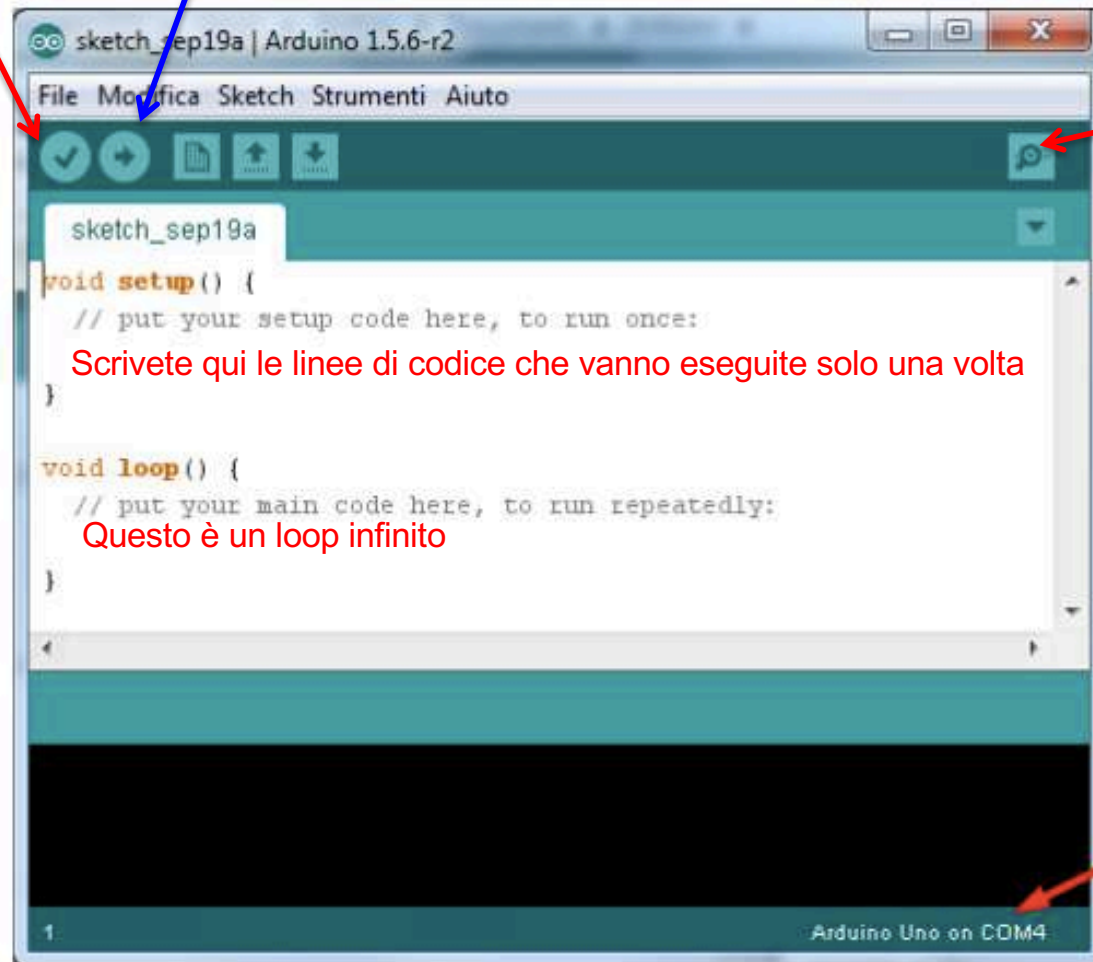
`array`  
`bool`  
`boolean`  
`byte`  
`char`  
`double`  
`float`  
`int`  
`long`  
`short`  
`size_t`  
`string`  
`String()`  
`unsigned char`  
`unsigned int`  
`unsigned long`  
`void`  
`word`

# Come iniziare

- ❑ Collegate Arduino Due ad una delle porte USB del PC
- ❑ Avviate l'applicazione Arduino IDE (già installata sui PC del Laboratorio)
- ❑ Si apre una finestra che offre un **framework** per la scrittura dello **sketch** (cioè, il programma)

compila

Compila, carica e run



Scrivete qui le linee di codice che vanno eseguite solo una volta

Questo è un loop infinito

Apri il serial monitor per eseguire operazioni di input/output

Dopo aver scritto il programma esso può essere compilato utilizzando il pulsante apposito (il primo in alto a sinistra): il risultato della compilazione viene riportato nella piccola finestra nera, dove vengono anche segnalati gli eventuali errori presenti.

Porta USB utilizzata

# Un esempio: programma blinking led

The screenshot shows the Arduino IDE interface with a sketch named "blinkingLED". The code is as follows:

```
// 2 LED LAMPEGGIANO ALTERNATIVAMENTE
const int LED1 = 13; // Assegna il pin 13
const int LED2 = 8; // Assegna il pin 8
void setup() {

pinMode(LED1, OUTPUT); // Il pin e' definito come output
pinMode(LED2, OUTPUT); // Il pin e' definito come output
}

void loop() {

digitalWrite(LED1, HIGH); // Mette il pin LED1 a 5 V
digitalWrite(LED2, LOW); // Mette il pin LED2 a 0 V
delay(500); // Aspetta 500 ms
digitalWrite(LED1, LOW); // Mette il pin LED1 a 0 V
digitalWrite(LED2, HIGH); // Mette il pin LED2 a 5 V
delay(500); // Aspetta 500 ms
}
```

Annotations in red text:

- La variabile (costante) LED1 = 13 e la variabile LED2 è uguale a 8
- La funzione **pinMode** definisce se un pin digitale è di ingresso oppure di uscita. In questo caso entrambi i pin 8 e 13 sono di uscita
- La funzione **digitalWrite** scrive su un dato pin il valore HIGH oppure Low
- La funzione **delay** mette in pausa il programma per un tempo prefissato

Annotation in blue text:

- Questo è un loop infinito. Si ripete fino a quando non si carica su Arduino un altro programma

Caricamento completato

Notare l'occupazione di memoria del programma e dei dati

Lo sketch usa 1.132 byte (3%) dello spazio disponibile per i programmi. Il massimo è 32.256 byte.  
Le variabili globali usano 9 byte (0%) di memoria dinamica, lasciando altri 2.039 byte liberi per le variabili locali. Il massimo è 2.048 byte.

17 Arduino Uno on /dev/cu.usbmodem411

# Programma blinking led: versione in C

Prima di compilare Arduino trasforma lo sketch creando il programma in C completo:

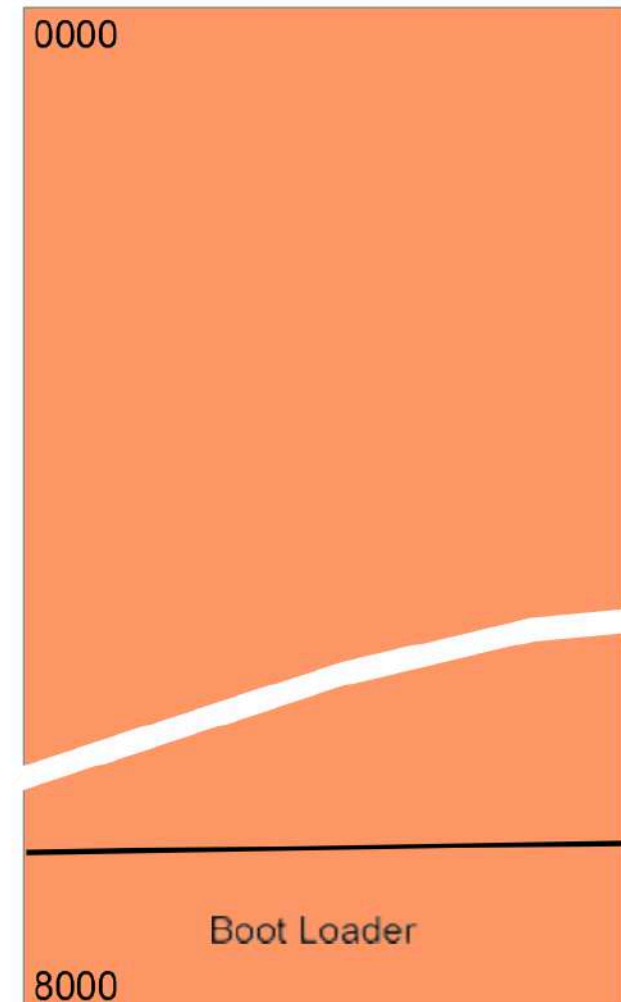
```
#include "Wprogram.h";
void setup ();
void loop ();
void setup () {
pinMode ( LED1 , OUTPUT); // Il pin e' definito come output
pinMode ( LED2 , OUTPUT); // Il pin e' definito come output
}
void loop () {
digitalWrite ( LED1 , HIGH ); // Mette il pin LED1 a 5 V
digitalWrite ( LED2 , LOW ); // Mette il pin LED2 a 0 V
delay ( 500 ); // Aspetta 500 ms
digitalWrite ( LED1 , LOW ); // Mette il pin LED1 a 0 V
digitalWrite ( LED2 , HIGH ); // Mette il pin LED2 a 5 V
delay ( 500 ); // Aspetta 500 ms
}
int main () {
setup ();
for (;;) { loop (); }
return 0;
}
```

Ad esempio nello sketch di Arduino non c'è il main, ma questo è necessario in un programma

Il compilatore (residente sul PC) crea il codice eseguibile. Il codice viene poi trasferito sulla memoria flash del  $\mu C$  (attraverso la connessione USB) ed eseguito.

Dal lato del  $\mu C$  il trasferimento e la scrittura in memoria sono eseguiti da un programma residente (BootLoader) che occupa gli ultimi 512 bytes della memoria flash. (In Arduino Uno)

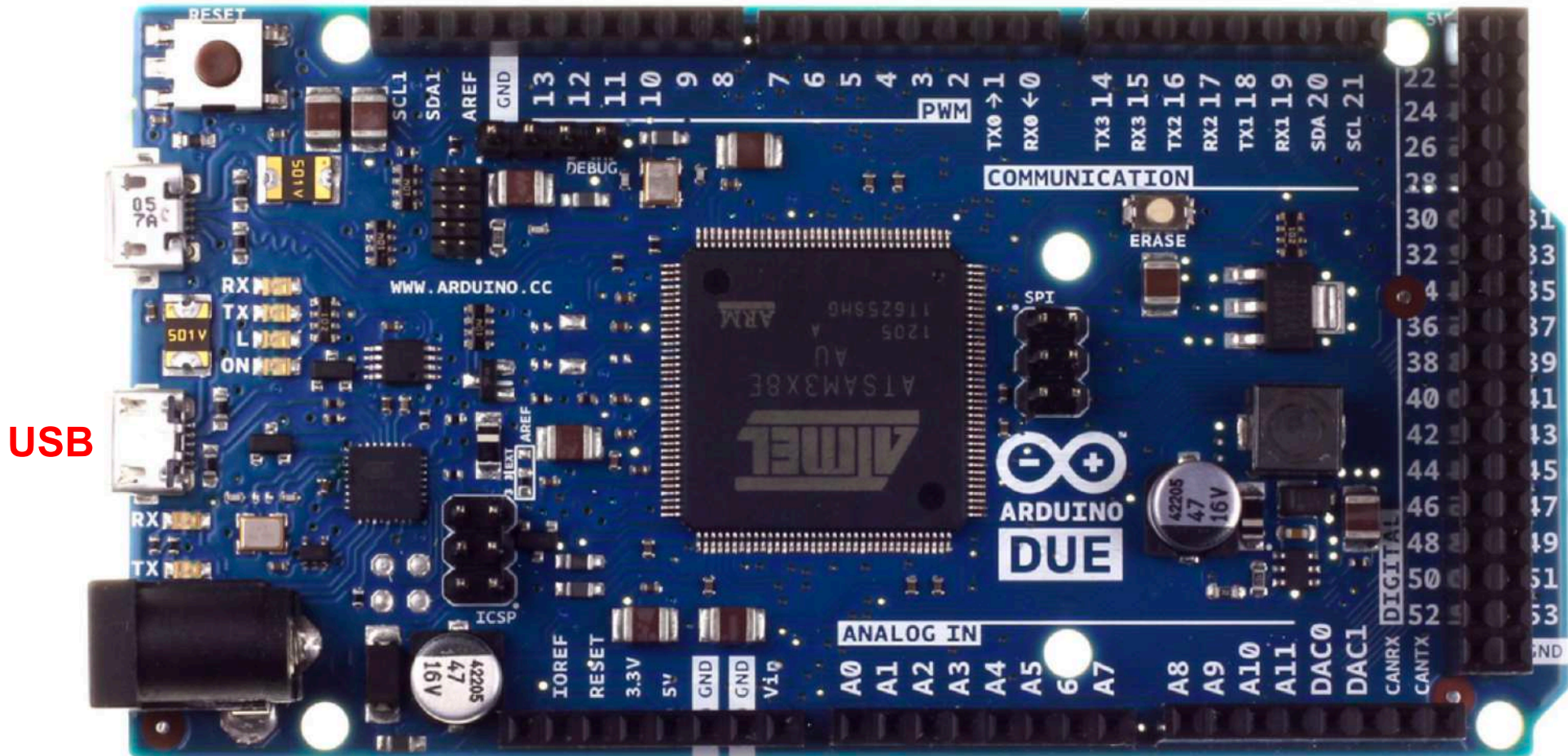
Il programma è memorizzato permanentemente e viene eseguito ogni volta che si accende la scheda (anche senza aprire Arduino sul PC), finché non è sostituito da un altro programma.



# La scheda Arduino Due

Pin digitali da 0 a 53.

I pin da 2 a 13 hanno anche la PWM (Pulse Width Modulation)



USB

Pin Analogici da A0 a A11.

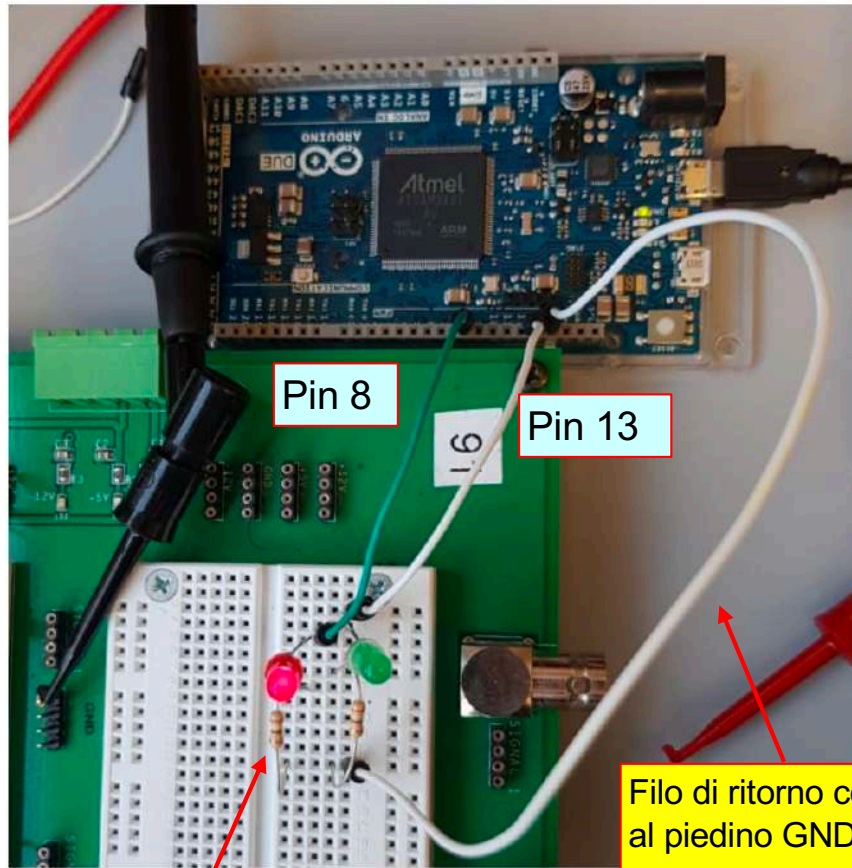
Le uscite del DAC sono DAC0 e DAC1

N.B. il GND digitale è diverso dal GND analogico

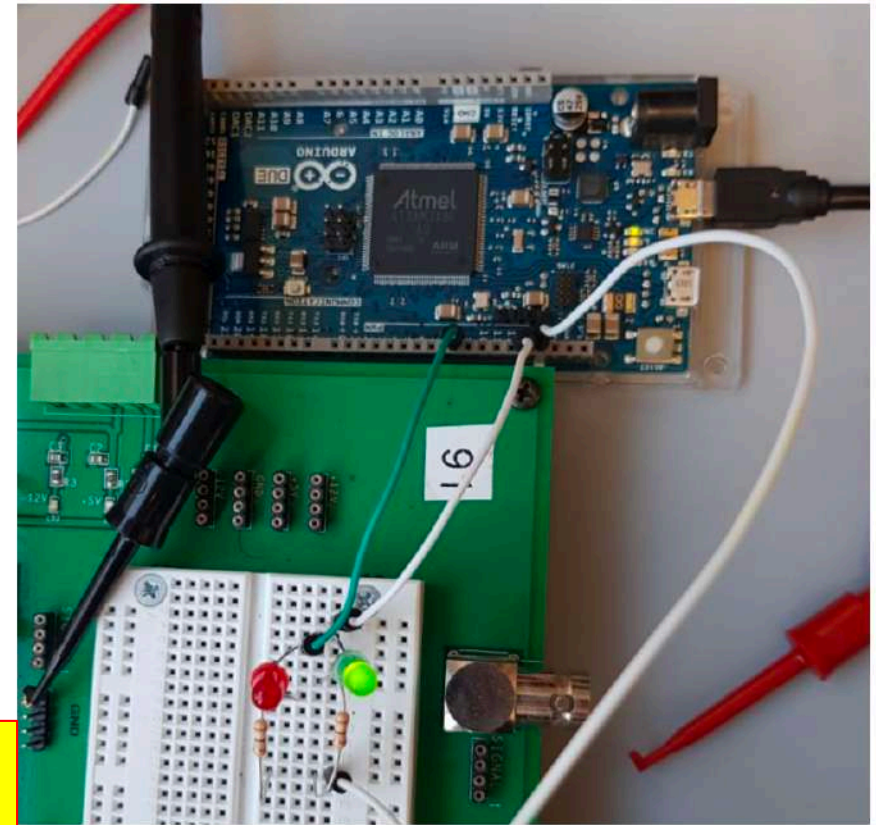


# Collegamento dei due Led su Arduino

La scheda didattica è usata come supporto per il collegamento dei componenti



((a)) Blinking LED: rosso acceso, verde spento



((b)) Blinking LED: verde acceso, rosso spento.

Resistenze di protezione di 470 Ohm per limitare la corrente

## ATMega328P

Il microcontrollore supporta vari meccanismi di comunicazione con dispositivi esterni:

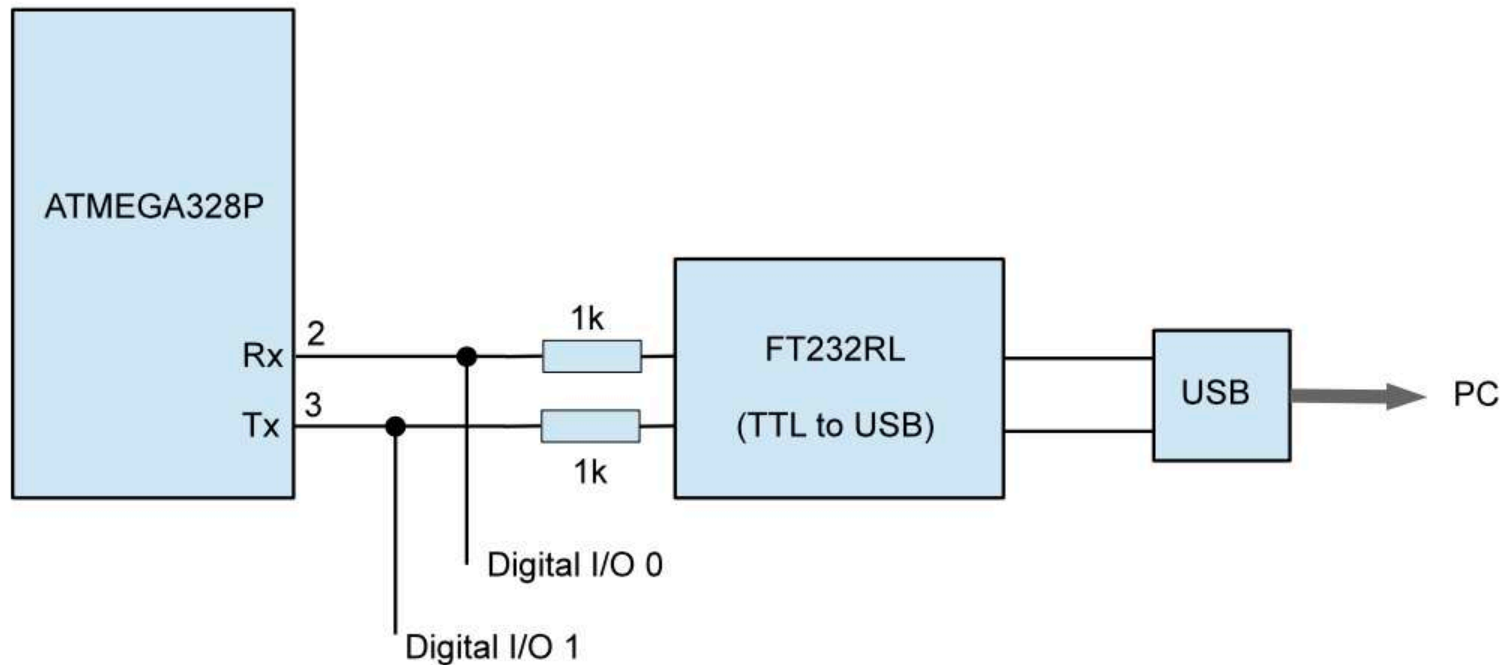
- USART (Universal Synchronous Asynchronous Receive Transmit)  
Digital I/O 0 e 1
- SPI (Serial Peripheral Interface)  
Digital I/O 10, 11, 12, 13
- I2C/TWI (Inter Integrated Circuit / Two Wire Interface)  
Analog pin A4, A5

Arduino fornisce librerie che facilitano l'uso di questi protocolli.

Su Arduino Due ci sono, oltre a questi, anche altri Pin dedicati a questi canali di comunicazione

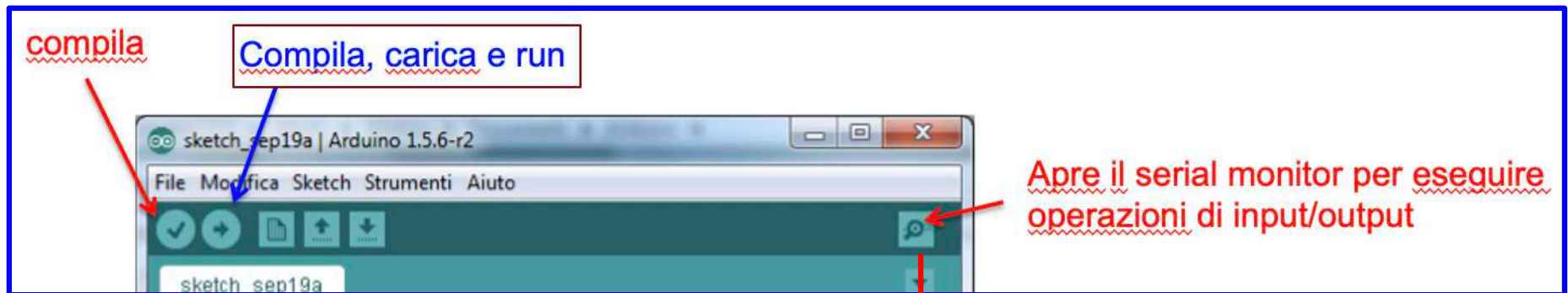
# Comunicazione seriale USART

E' utilizzata da Arduino per connettere il  $\mu C$  al PC, tramite la porta USB: da qui viene fatto il download del programma.



Puo' liberamente essere poi utilizzata dall'utente.

# Comunicazione seriale: finestra di monitor



L'Arduino IDE si occupa di gestire le comunicazioni tra il microcontrollore e il computer. Non dobbiamo preoccuparci di nulla.

Seleziona la velocità di trasmissione dati

# Esempio comunicazione seriale

```

long int data=0;
long unsigned t0,t1;
int i;

void setup() {
  //open serial communication and wait for port to open
  Serial.begin(9600); Velocità porta seriale
  Serial.println("Ci siamo!");
  Serial.println(" ");
}

void loop() {
  if(Serial.available()>0){ Controlla se c'è qualcosa in input
    data=Serial.parseInt(); legge l' input
    t0=micros();
    Serial.print("Hai scritto il numero: ");
    Serial.println(data);
    Serial.print("E questo moltiplicato per 10 fa:");
    Serial.println(data*10); println va a "capo"
    t1=micros();
    Serial.print("durata: t0=");
    Serial.print(t0);
    Serial.print(" t1=");
    Serial.println(t1);
    Serial.print(t1-t0);
    Serial.print(" ");
    Serial.println("microsecondi");
    delay(100);
  }
}

```

Scrive una linea vuota

Monitor seriale

```

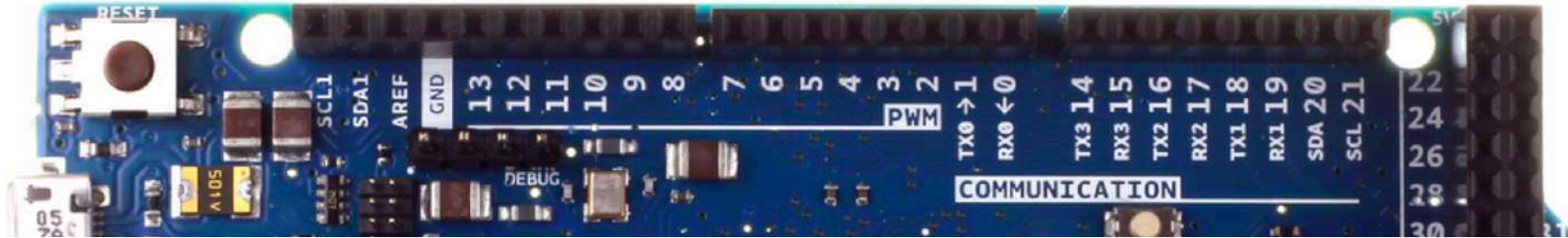
Ci siamo!

Hai scritto il numero: 1
E questo moltiplicato per 10 fa:10
durata: t0=3099002 t1=3099099
97 microsecondi
Hai scritto il numero: 10
E questo moltiplicato per 10 fa:100
durata: t0=7040002 t1=7040103
101 microsecondi
Hai scritto il numero: 50
E questo moltiplicato per 10 fa:500
durata: t0=11594002 t1=11594103
101 microsecondi
Hai scritto il numero: 100
E questo moltiplicato per 10 fa:1000
durata: t0=16439002 t1=16439106
104 microsecondi
Hai scritto il numero: 200
E questo moltiplicato per 10 fa:2000
durata: t0=21128002 t1=21128106
104 microsecondi
Hai scritto il numero: 300
E questo moltiplicato per 10 fa:3000
durata: t0=23747003 t1=23747106
103 microsecondi

```

# Pulse Width Modulation (PWM)

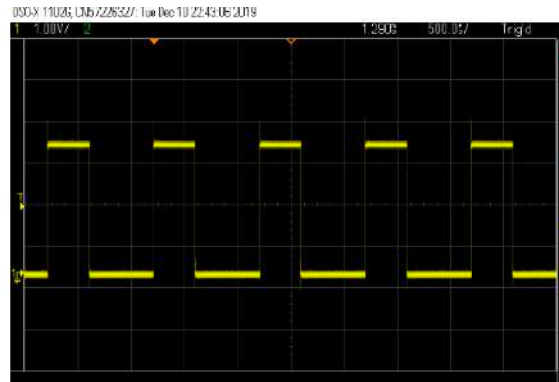
I pin da 2 a 13 hanno anche la PWM (Pulse Width Modulation)



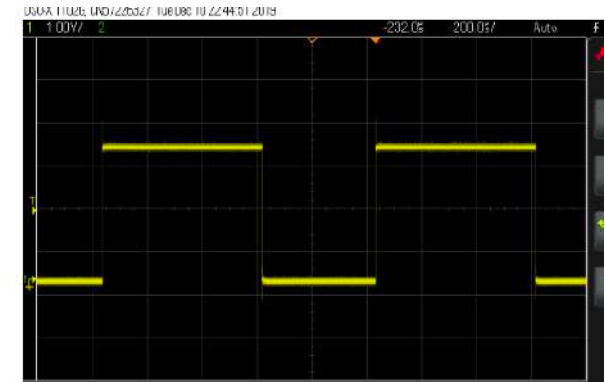
- Le uscite PWM sono uscite digitali, ovvero 0 oppure 1, però si può variare il duty cycle, cioè la durata dello "stato" 1 rispetto allo "stato" 0



64/255



100/255



150/255

```
analogWrite(pin, value)
```

$f = 1 \text{ kHz}$

pin: the Arduino pin to write to. Allowed data types: int.

value: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: int.

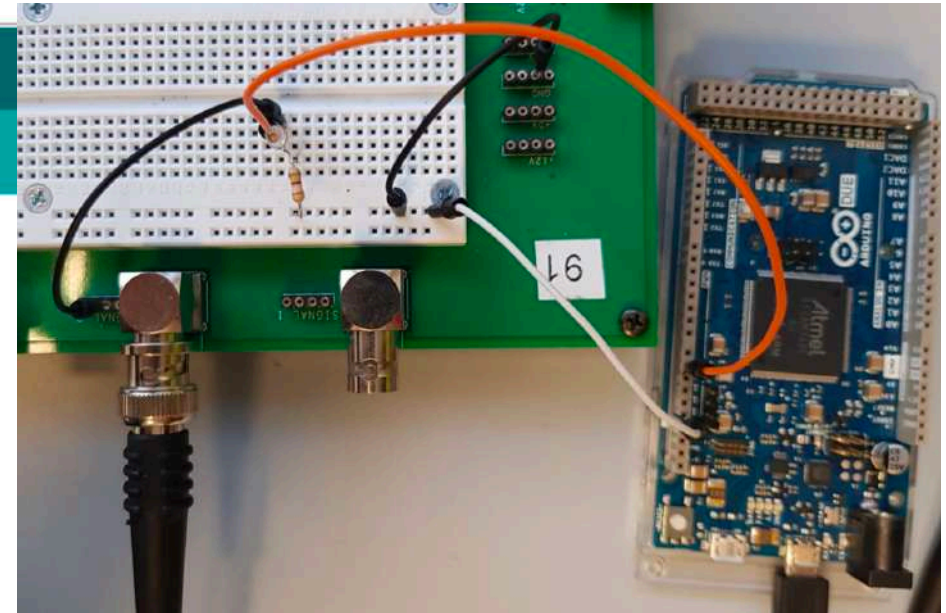
# Programma Analog\_write

```

analog_write $
int ledPin = 9;  Pin di scrittura
int val = 128;
void setup() {
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
  Serial.println("Accendi il fuoco...");
  Serial.println(" ");
}

void loop() {
  // put your main code here, to run repeatedly:
  if (Serial.available() > 1) {
    val=Serial.parseInt(); |
    // check value in
    if (val>=0 && val <=255) {
      analogWrite(ledPin, val);
      Serial.print("Scritto valore :");
      Serial.println(val);
    }
    else {
      Serial.println("Valore fuori range (0:255)!!!");
    }
  }
}

```



La luminosita' del Led dipende dal numero scritto sulla riga di input

Questo e' il valore scritto con la tastiera del computer sulla porta seriale di input di arduino

Attenzione: c'e' un carattere di controllo a fine linea che viene interpretato come uno zero!

# Arduino Due: DAC

- ❑ **Arduino Due ha due uscite DAC: DAC0 e DAC1**
- ❑ Si può scegliere il numero di bit del DAC, da 8 a 12, con la funzione `AnalogWriteResolution(nbit)`

`analogWriteResolution()` sets the resolution of the `analogWrite()` function. It defaults to 8 bits (values between 0-255) for backward compatibility with AVR based boards.

$$V = \frac{V_{max}}{2^n - 1} \sum_0^{n-1} 2^i a_i \quad \xrightarrow{\text{12 bit}} \quad V = \frac{V_{REF}}{4095} \cdot N \quad (0 \leq N \leq 4095)$$

$V_{REF}$  è la tensione di riferimento di Arduino Due (3.3 V)

## ATTENZIONE

Arduino Due non ha un'uscita analogica compresa tra 0 V e  $V_{REF}$ , ma tra 1/6 e 5/6 della tensione  $V_{REF}$ , cioè tra **0.55 V** e **2.75 V** con  $V_{REF} = 3.3$  V.

L'intervallo della tensione di uscita del DAC è di **2.75-0.55 = 2.2 V**,  
con una risoluzione di **2.2/4095 = 0.5372 mV**

- ❑ Corrispondenza tra numero **N** e tensione di uscita **V** del DAC:

$$V = \frac{1}{6} \cdot V_{REF} + \frac{4}{6} \cdot \frac{V_{REF}}{4095} \cdot N \quad (0 \leq N \leq 4095)$$



# Esempio di uso del DAC

```
Prog_DAC
// Reads from keyboard a value and set the DAC
// val needs to be in the appropriate range 0-4096 if 12 bits
int val;
int i;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("DAC testing");
}

void loop() {
  analogWriteResolution(12); //sets the DAC resolution to 12 bits.
  Serial.println("inserire un valore");
  val = Serial.read(); ← Valore scritto con la tastiera sull'input della porta seriale
  Serial.println(val, DEC);
  analogWrite(DAC0, val); ← Verificare con il multimetro la tensione presente sul pin DAC0
  while(!Serial.available()){
  }
}
```

\*\* In addition to PWM capabilities on the pins noted above, the Due has true analog output when using `analogWrite()` on pins `DAC0` and `DAC1`.

# ADC di Arduino UNO (AT328)

Il microcontroller contiene un ADC a 10 bit (sample and hold, successive approximation). E' collegato tramite un multiplexer a 8 ingressi (6 effettivamente utilizzabili su Arduino).

$$\text{Output} = \frac{2^{10}}{V_{ref}} V_{in}$$

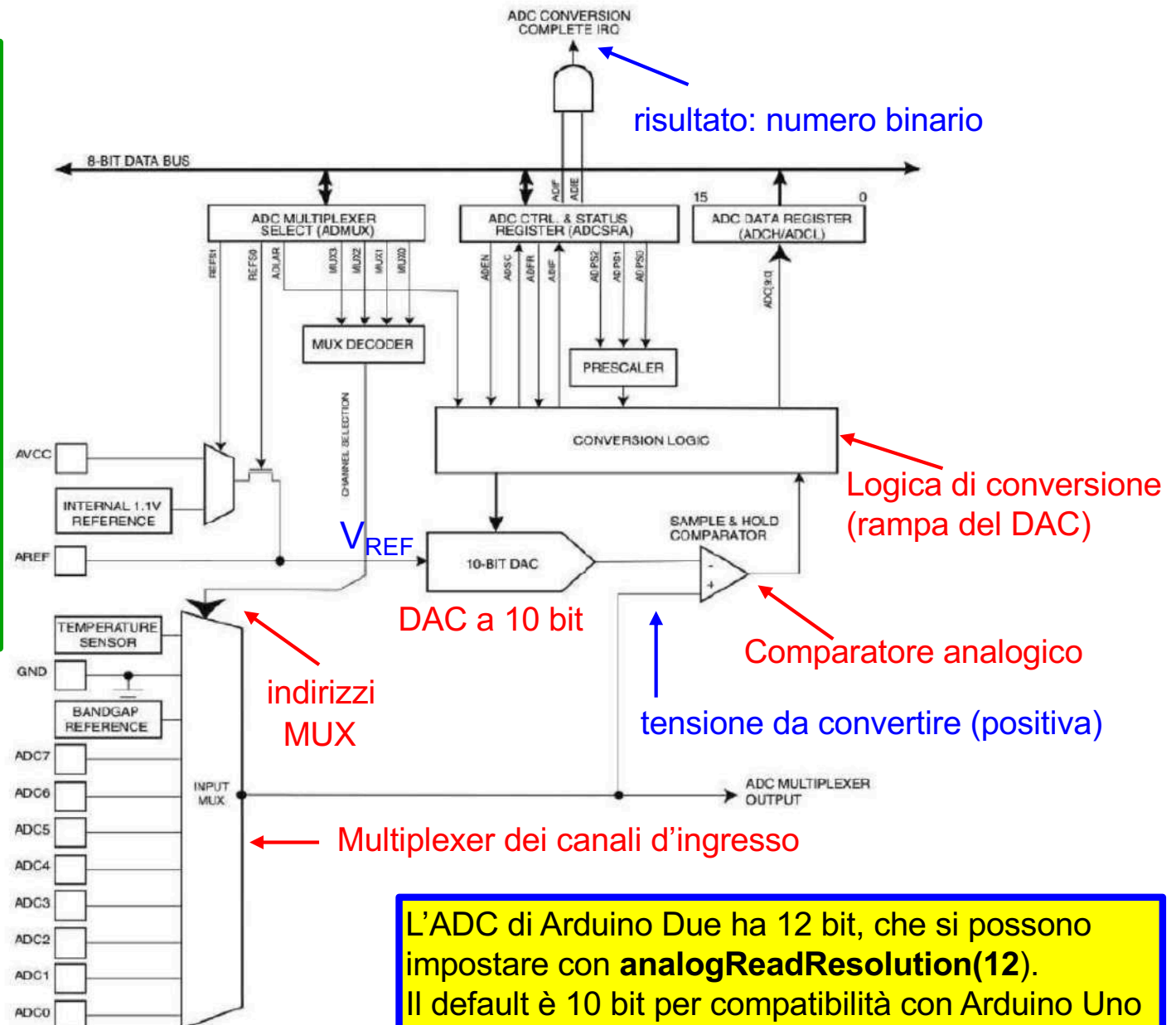
3 possibili  $V_{ref}$ :

- 5 V (default)
- 1.1 V
- $A_{ref}$  (esterna)

Nota bene:

$$0 \leq A_{ref} \leq +5 V$$

*conversion time*  $\approx 100 \mu s$



# Esempio dell'uso dell'ADC

analog\_read

```
int analogPin = A3; // analog pin A3
int RdVal = 0;
float Vx = 0.0;

void setup() {
  analogReference(AR_DEFAULT); // Arduino Due 3.3 V
  // analogReadResolution(12); // set the ADC to 12 bit
  Serial.begin(9600);
  Serial.println(" Arduino Voltage meter ");
  Serial.println(" ");
}
void loop() {
  RdVal = analogRead(analogPin); // ADC value
  Vx = float(3.3)*(float(RdVal)/float(1023)); // 10 bit ADC
  // Vx = float(3.3)*(float(RdVal)/float(4095)); // 12 bit ADC
  delay(1000);
  Serial.print(" Numero letto sul pin analogico: ");
  Serial.print(RdVal);
  Serial.print(" Tensione letta: ");
  Serial.println(Vx);
}
```

Sul pin A3 va mandata la tensione da convertire

$$\text{Output} = \frac{2^{10} - 1}{V_{ref}} V_{in}$$

(se usate 12 bit il 10 va sostituito con 12, ovviamente)

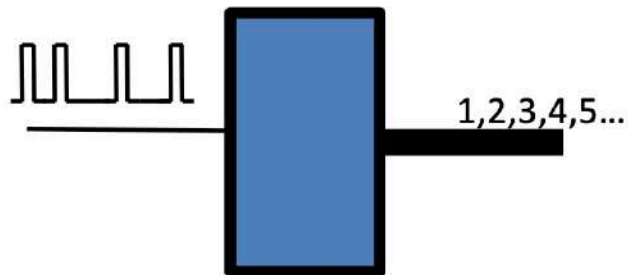
il valore di default dell'ADC è di 10 bit

Confrontate la tensione letta con il multimetro con il valore Vx



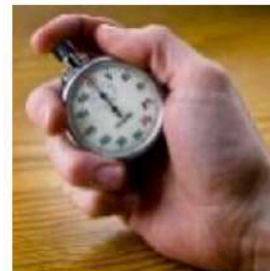
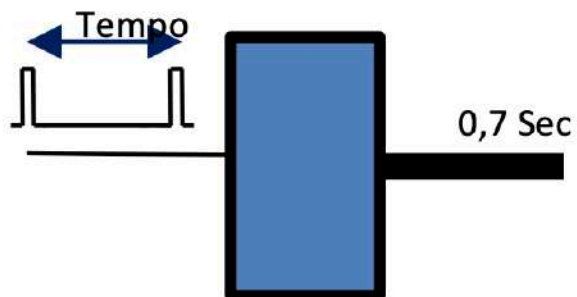
# Acquisizione dati con Arduino

Si possono utilizzare gli ingressi digitali per contare il numero di eventi



## Contatori Digitali

Consentono di contare il numero di eventi.



## Misuratori di tempo digitali (TDC)

Consentono di misurare il tempo tra due eventi.

Utilizzando i contatori e le routine di timing, si possono fare delle misure di intervalli di tempo (TDC)

**Nota finale:** Arduino può fare molte altre cose e ha molte altre funzionalità che non abbiamo descritto, tipo ad esempio la gestione degli interrupt. In queste slides ho accennato solo alle cose che utilizzeremo nelle due esercitazioni di laboratorio.



SAPIENZA  
UNIVERSITÀ DI ROMA

Fine del capitolo 9