Programmazione di sistemi multicore

Michele Martinelli
Michele.martinelli@uniroma1.it

Accesso diretto ai Registri

- Accedere direttamente ai registri del microcontrollore ha degli svantaggi
  - difficoltà di manutenzione del codice
  - perdita di portabilità
  - errori
- Ma anche dei vantaggi
  - facilità/velocità di accesso
  - Alcune volte serve di impostare diversi pin nello stesso momento
    PORTB = B00001100;
    (digitalRead() and digitalWrite() sono composte a molte righe di codice)

Maschere bitwise – accendere un bit

Accendere uno specifico bit di una stringa (o il pin di una porta)

Voglio ottenere xxxxxx11 (accendere gli ultimi due bit)

Maschere bitwise – accendere un bit

Accendere uno specifico bit di una stringa (o il pin di una porta)

Voglio ottenere xxxxxx11 (accendere gli ultimi due bit)

Esempi:

Stringa : 00000000

00000000 | 00000011 -> 00000011

Stringa: 11111111

11111111 | 00000011 -> 11111111

Stringa: 10101011

10101011 | 00000011 -> 10101011

Maschere bitwise – spegnere un bit

Accendere uno specifico bit di una stringa (o il pin di una porta)

Voglio ottenere xxxxxx00 (spegnere gli ultimi due bit)

Maschere bitwise – spegnere un bit

Accendere uno specifico bit di una stringa (o il pin di una porta)

Voglio ottenere xxxxxx00 (spegnere gli ultimi due bit)

Stringa : 11111111

11111111 & 11111100 -> 11111100

Stringa: 00000011

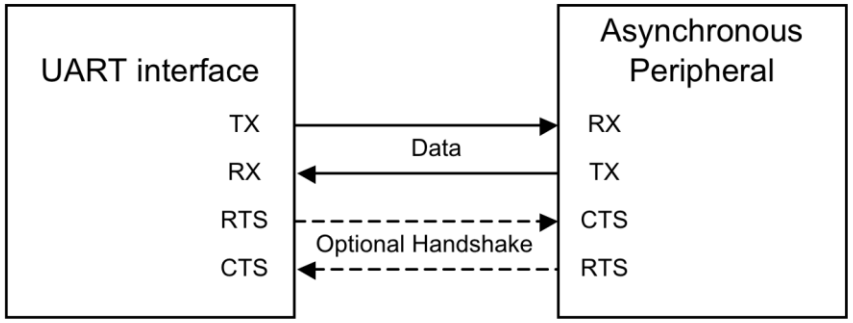
00000011 & 11111100 -> 00000000

Stringa: 01010111

01010111 & 11111100 -> 01010100

Comunicazione seriale
Asynchronous Serial Interface (UART)

Questo è quello che avete usato finora per «parlare» con Arduino

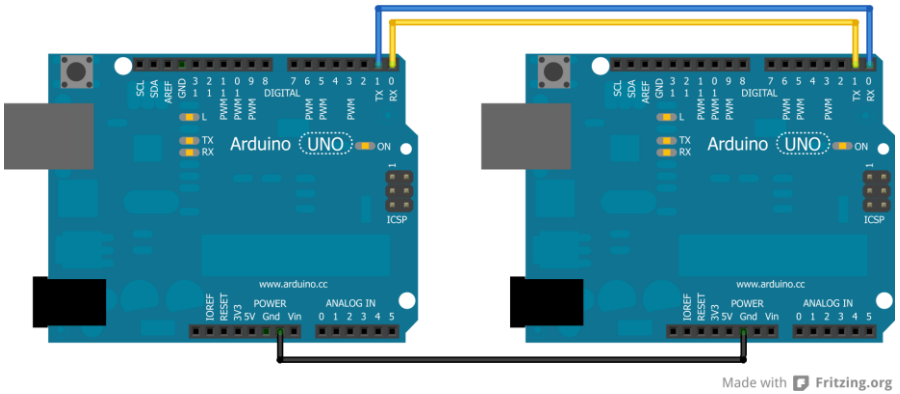


Caratteristiche:

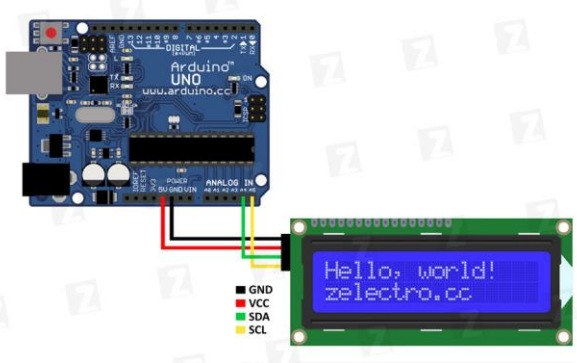
Molto lento
poco robusto

molto semplice da implementare e utilizzare

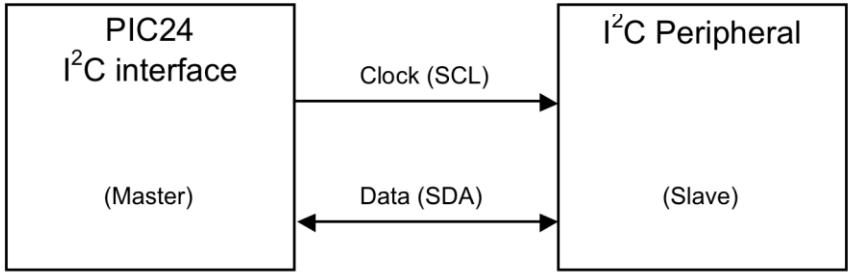
Prevede che ci sia una sola periferica connessa



I2C

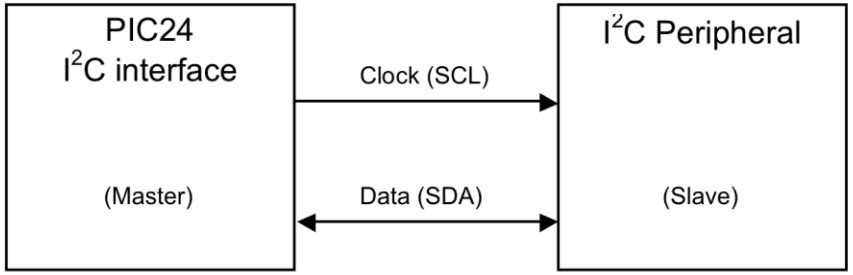


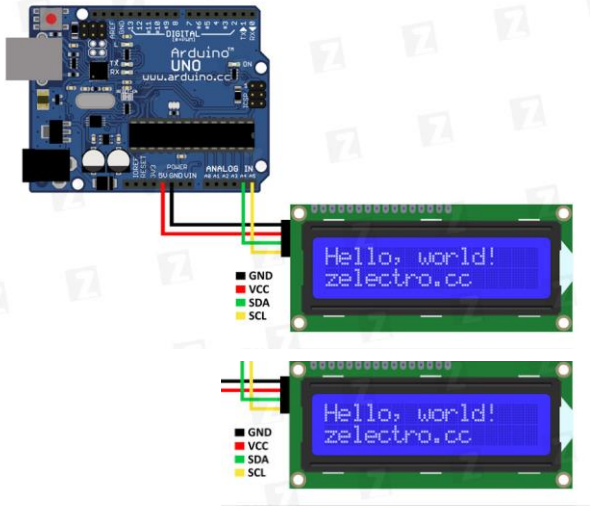
Indirizzo 123



Molto utilizzato per parlare con periferiche esterne (sensori…)
Relativamente veloce
Costituito da due fili
Possibilità di connessione di più device, MA SOLO UN MASTER

I2C

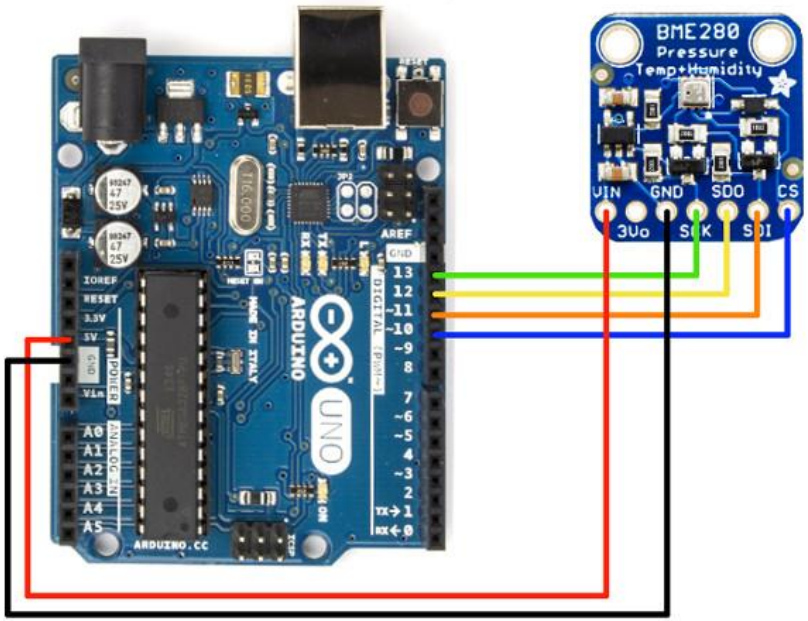


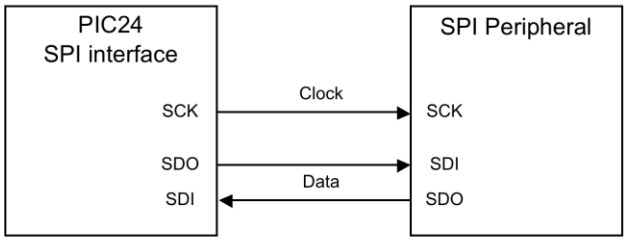


Molto utilizzato per parlare con periferiche esterne (sensori…)
Relativamente veloce
Costituito da due fili
Possibilità di connessione di più device, MA SOLO UN MASTER

SPI

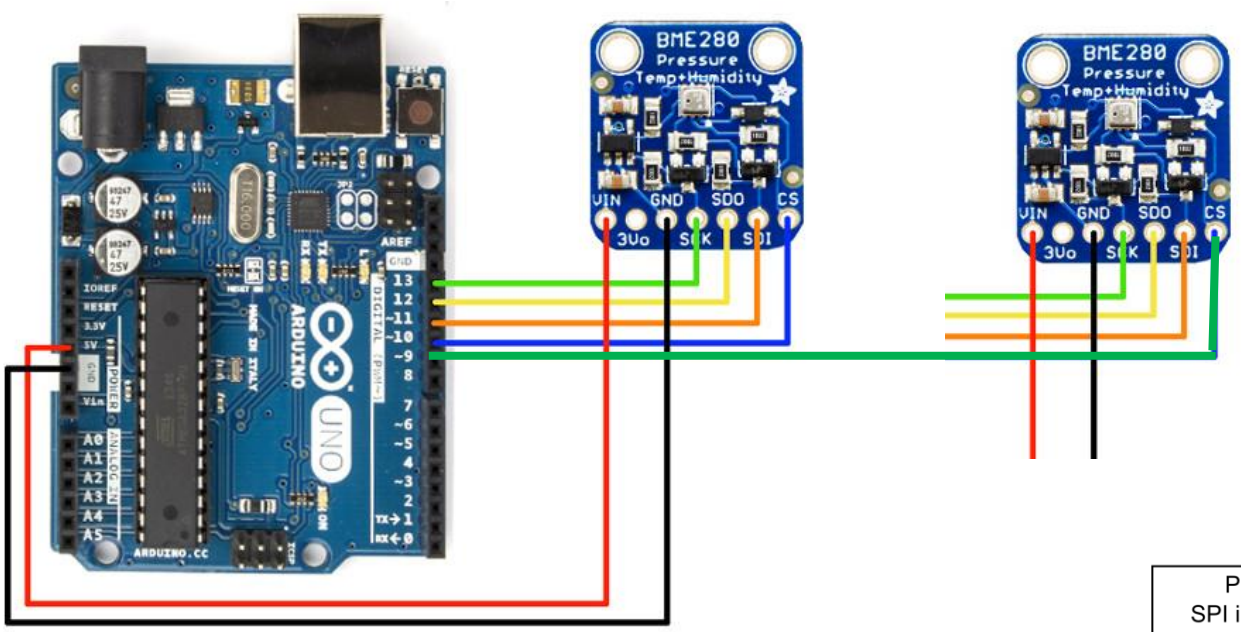
Molto utilizzato per parlare con periferiche esterne (sensori…)
Relativamente veloce
Costituito da 3 fili + 1 di indirizzamento per ogni slave
Possibilità di connessione di più device, MA SOLO UN MASTER

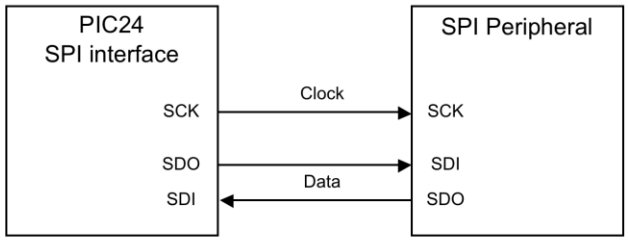




SPI

Molto utilizzato per parlare con periferiche esterne (sensori…)
Relativamente veloce
Costituito da 3 fili + 1 di indirizzamento
Possibilità di connessione di più device, MA SOLO UN MASTER



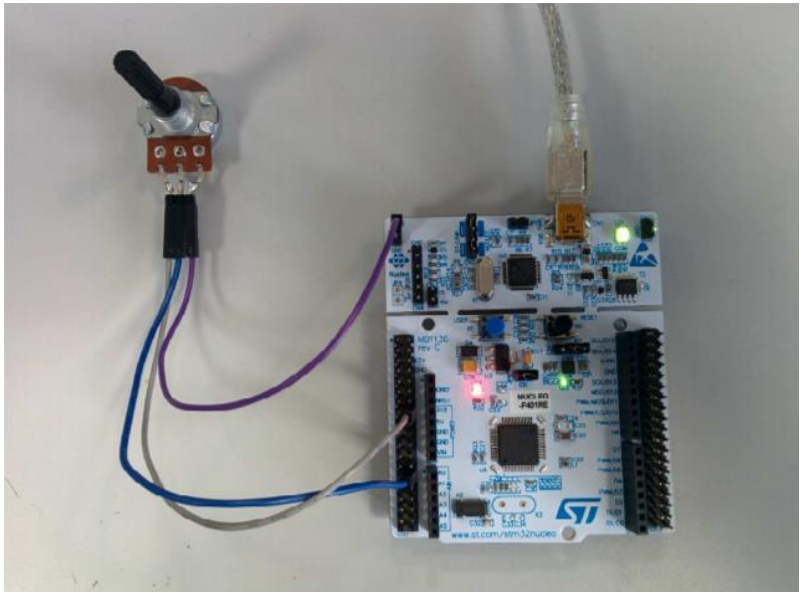


Nucleo F401RE – esempio

usiamo un potenziometro per regolare la luminosità di un led in PWM

PWM => per pilotare il LED
Timer 2 Channel 1 - PWM mode • 100Hz •

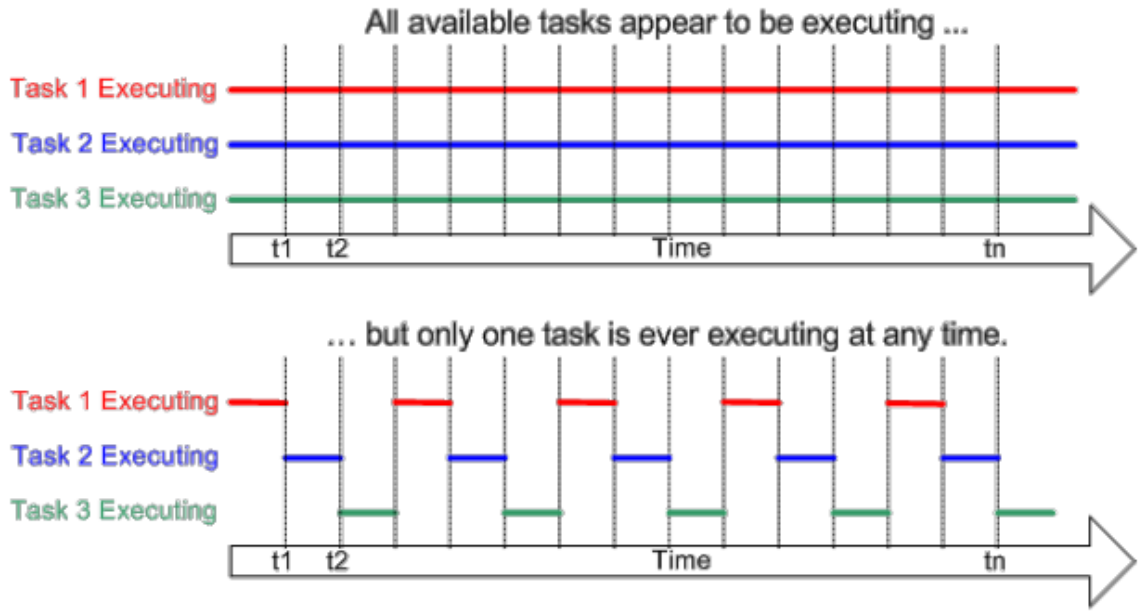
ADC => per variare la luminosità del LED variando proporzionalmente al valore analogico acquisito il Duty Cycle del PWM
• ADC1 IN0 – Single and regular conversion



AVVERTIMENTO:
Sarà PARECCHIO più complicato rispetto a scrivere «analogWrite»

RTOS

Un Real Time Operating System (RTOS) è un sistema operativo progettato per fornire un modello di esecuzione predicibile **(deterministico).**



Perché un RTOS

Un OS realtime è essenziale in tutti quegli scenari in cui il fattore tempo è cruciale (hard real time).

- **Hard real time:** è sempre necessario un tempo di esecuzione **massimo** garantito
- **Soft real time:** è necessario un tempo di esecuzione **medio**

Calcolo parallelo

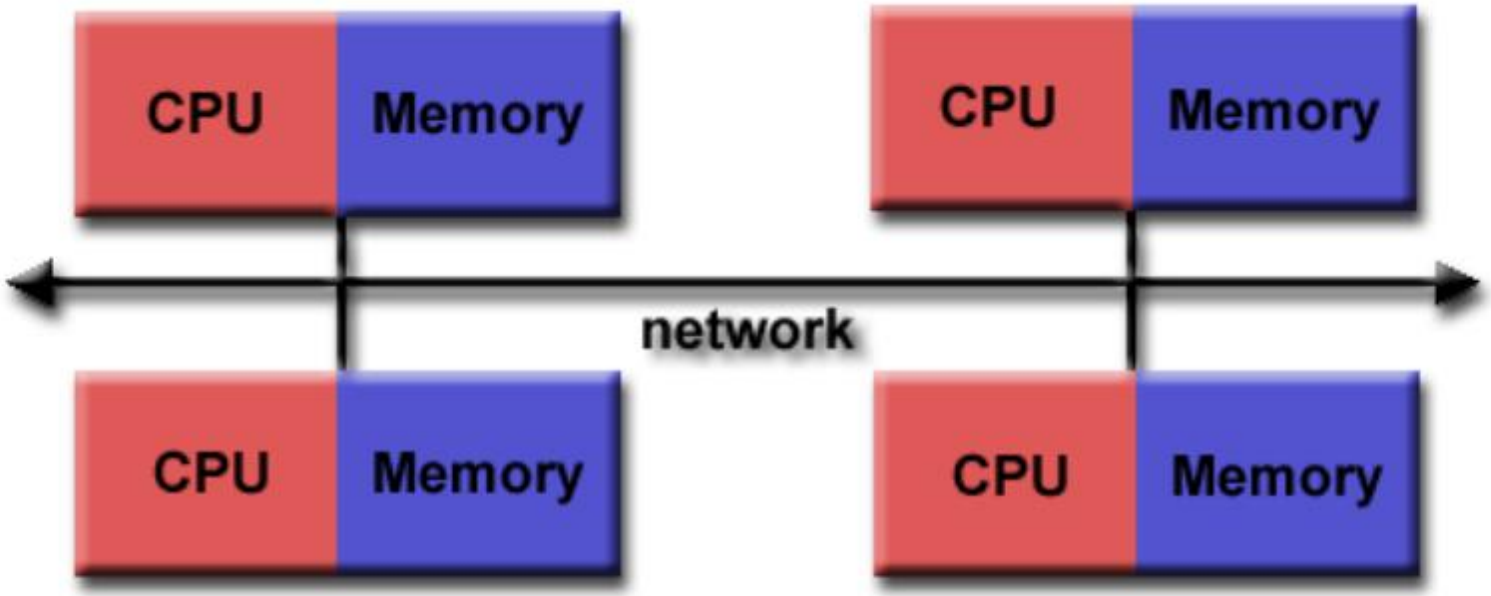
Il calcolo parallelo si basa sull'uso simultaneo di più processi per risolvere un problema in comune.

1. Il **problema** deve essere **diviso** in più **parti indipendenti** che possono essere eseguite in parallelo.
2. Per raggiungere un vero parallelismo ogni parte dovrà essere **eseguita** realmente in **contemporanea** ad altre parti.

Quindi il multitask e il multithreading non sono veri e propri parallelismi.

Message passing

Un paradigma utilizzato nel calcolo parallelo è il **message passing**.



MPI

MPI viene definito come specifica e poi viene implementato da diverse librerie:

OPEN MPI, INTEL MPI, MPICH

È una libreria che permette l'implementazione del calcolo parallelo tramite message passing. Permette la:

- comunicazione tra processi/core,
- la gestione dei singoli
- e utilities per semplificare la vita al programmatore.

OPEN MPI è disponibile per C, Fortran e tramite wrapper per Python.

CUDA

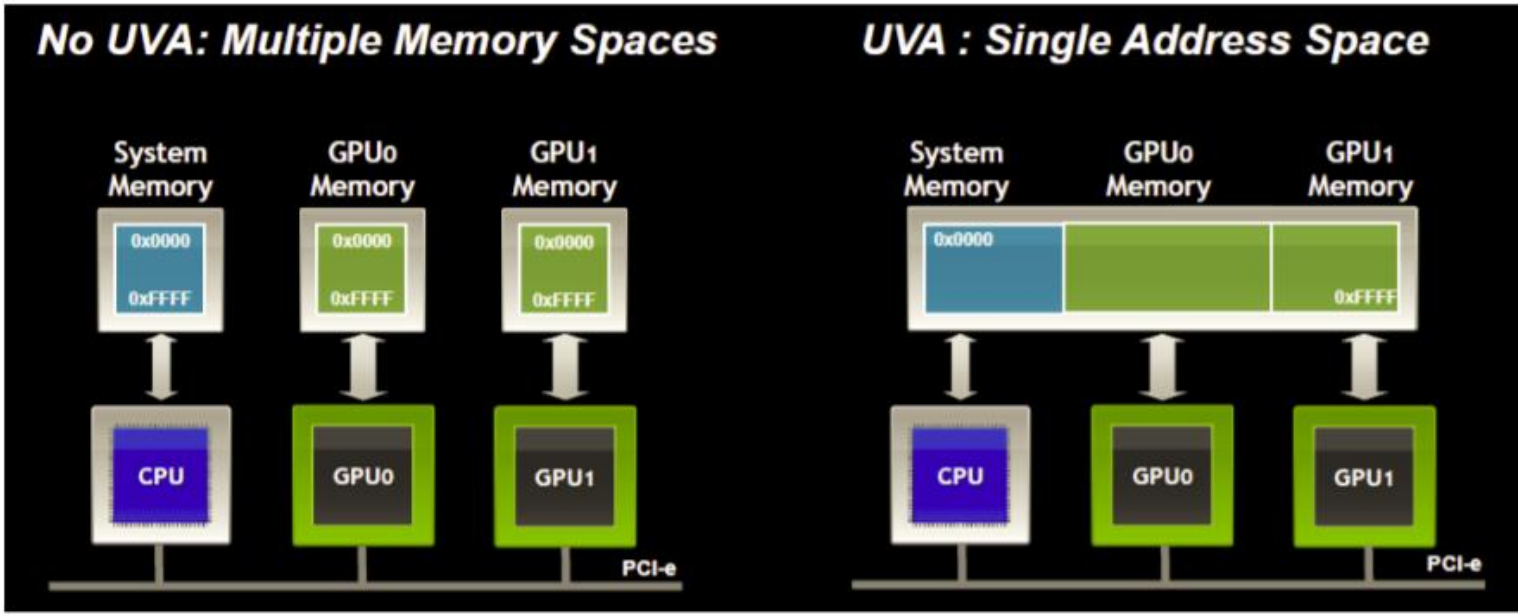
L'architettura **CUDA** è composta da due componenti:

1. L'**hardware** CUDA (le GPU NVIDIA)
2. Le **librerie** CUDA che estendono il **C** standard.

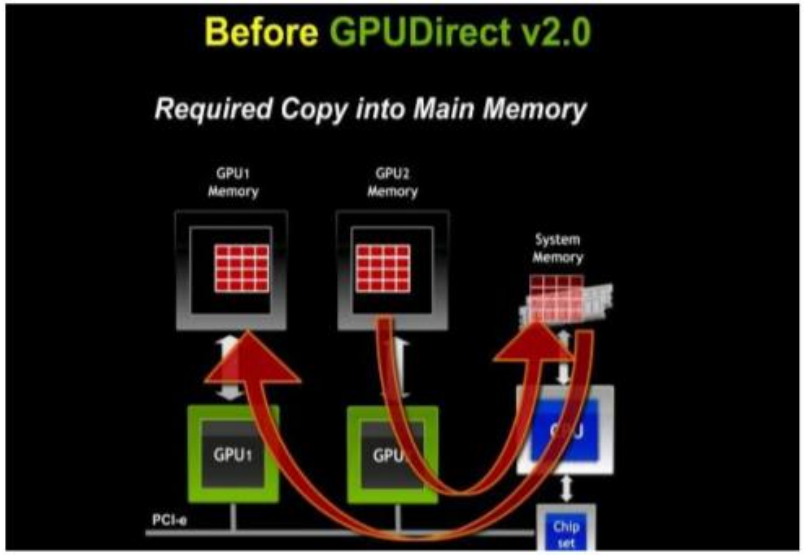
Ovviamente per eseguire il codice è necessario avere una scheda dedicata NVIDIA. In queste lezioni parleremo in modo molto generale del paradigma e del framework.

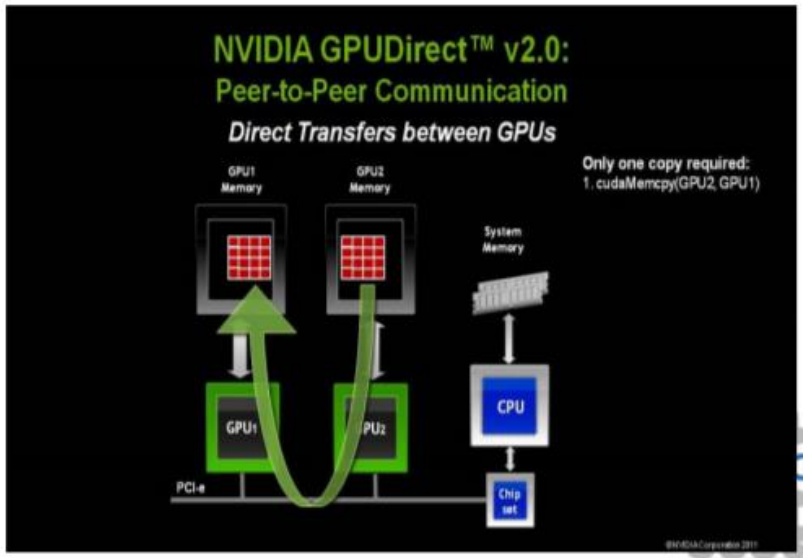
Indirizzamento virtuale unificato CPU – GPU

- La memoria lato host deve essere pinned
- Allocazione lato host con cudaHostAlloc()
- Allocazione lato device con cudaMalloc()
- Copia con cudaMemcpy()e parametro cudaMemcpyDefault
- E’ possibile accedere alla memoria host direttamente da device tramite cudaHostAlloc() e **cudaHostGetDevicePointer** (CUDA zero copy)



Comunicazioni Peer-to-Peer





Paradigma RDMA

- È un meccanismo che permette ad un host su una rete di avere accesso diretto ai dati nella memoria di un altro host.
- Server principalmente ad eliminazione la necessità della copia intermedia dei dati tra il buffer di ricezione e l'applicazione finale nell'host ricevente.
- Con RDMA i dati in arrivo vengono riconosciuti come dati per una particolare applicazione e vengono quindi inviati in memoria.

- In questo caso è importante che la memoria sia lockata oltre che pinnata, altrimenti la scheda di rete potrebbe sovrascrivere aree di memoria utilizzate da altri processi

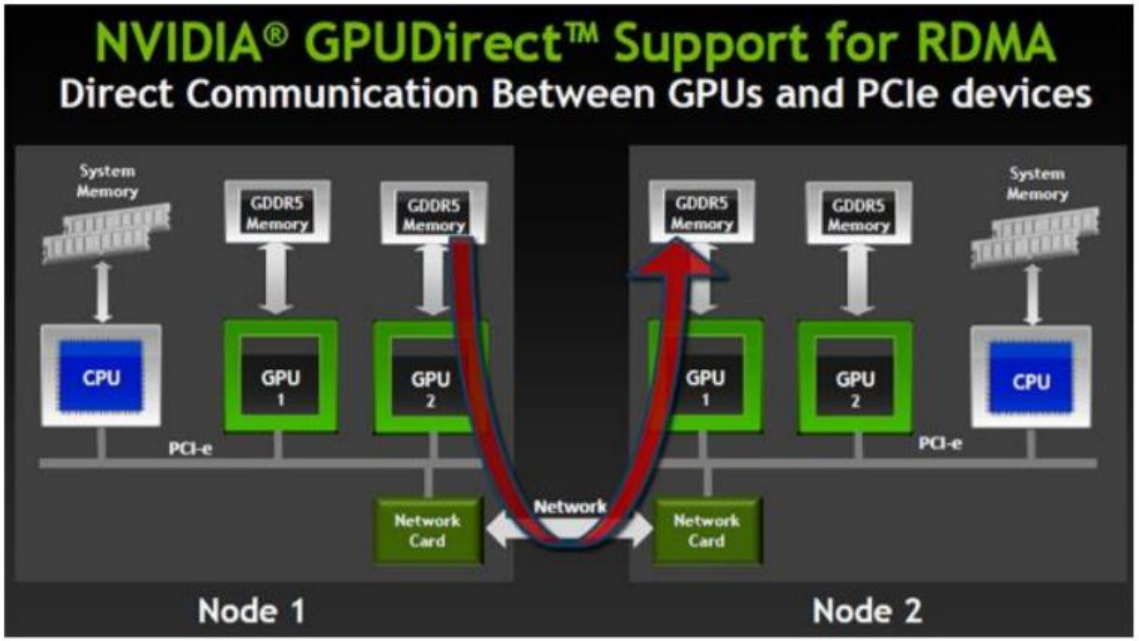
- RDMA è un'evoluzione del DMA!

Comunicazioni GPUDirect RDMA

A partire da CUDA 5.0 La tecnologia GPUDirect permette la comunicazione diretta tra GPU e altri dispositivi PCI-E presenti nel sistema;

supporta l'accesso diretto alla memoria tra la GPU e le schede di rete.

La comunicazione tra le GPU avviene tramite chiamate MPI dove al buffer vengono passati gli indirizzi di memoria della GPU.



Block(chain) idea

1. Everyone **tries to solve** a puzzle

2. The **first one** to solve the puzzle **gets 1 coin**

3. The solution of **puzzle *i*** **defines puzzle *i+1***

3			2	4			6	
	4						5	3
1	8	9	6	3	5	4		
				8		2		
		7	4	9	6	8		1
8	9	3	1	5		6		4
		1	9	2		5		
2			3			7	4	
9	6		5			3		2

It's EASY to VERIFY the solution, not to find it!

Block(chain) concetti chiave

1. Ogni “peer” della rete mantiene la sua copia della blockchain (TUTTA)

2. La soluzione al problema è “difficile” da trovare per evitare che si possa facilmente riscrivere le storia (trovare la soluzione è il “mining”)

3. La soluzione è facile da verificare perché tutti possano verificare che un blocco è valido

4. Esempio: un’idea è utilizzare un funzione per calcolare l’hash del blocco che contiene le transazioni da “bloccare” + un riferimento all’hash del blocco precedente. NB in questo modo è sempre possible risalire alla provenienza dei fondi a partire dale transazioni che li hanno riguardati

SHA256 hash functions

Here how you can try it in Python:

```
>>> import hashlib


>>> hashlib.sha256(“hello world”).hexdigest()
‘B94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9’

>>> hashlib.sha256("helli world").hexdigest()
'bd6952606ca18ccc9ff86bb8874ce0c61d7aa4fb72363e323f9eb2d3e783a487'
```

The puzzle

I want to find a particular hash…
For example, I want two ZEROs at the end of the hash (“00”)…
What string I have to use?

```
>>> import hashlib


>>> hashlib.sha256(“hello world”).hexdigest()
‘B94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9’

>>> hashlib.sha256("hello world 1").hexdigest()
'063dbf1d36387944a5f0ace625b4d3ee36b2daefd8bdaee5ede723637efb1cf4’

…

>>> hashlib.sha256("hello world 238").hexdigest()
‘d622375f29c4b3586747946104O4bdd1f4e060a7244568b5549eed9754dfa400’
```

The puzzle

…and 3 ZEROs at the end of the hash (“000”)…?

```
>>> import hashlib


>>> hashlib.sha256(“hello world 3198”).hexdigest()
‘838ee925a5eff45a3337363e9a2e993e02eddcc2edd90cdd4fec14b0c6e59000’


…and 6?
>>> hashlib.sha256(“hello world 28114982”).hexdigest()
3d28877e8af972f6732de4591245a87545b0221c26e446f09e98665a6d000000
```

It’s getting harder to find a solution, but easy to verify it!

The puzzle – python code

```
import hashlib

i=0
while True:
    res = hashlib.sha256("hello world "+str(i)).hexdigest()
    if res.endswith("000000"):
        print i
        print res
        break
    i += 1
```

The puzzle – create a chain

How can the puzzle "depends" on the previous one?

```
>>> hashlib.sha256("hello world 3198").hexdigest()
'838ee925a5eff45a3337363e9a2e993e02eddcc2edd90cdd4fec14b0c6e59000'
```

The puzzle – create a chain

How can the puzzle "depends" on the previous one?

```
>>> hashlib.sha256("hello world 3198").hexdigest()
'838ee925a5eff45a3337363e9a2e993e02eddcc2edd90cdd4fec14b0c6e59000'
```

A new string!

```
>>> hashlib.sha256("
SECOND hello world + 838ee925a5eff45a3337363e9a2e993e02eddcc2edd90cdd4fec14b0c6e59000
    ").hexdigest()
'9b582a1b5fd41f36cc0a396d4747507148f3bdde58a4bff18891406a9158a72e'
```

But this hash doesn't end with "000"…

The puzzle – create a chain

How can the puzzle "depends" on the previous one?

```
>>> hashlib.sha256("hello world 3198").hexdigest()
'838ee925a5eff45a3337363e9a2e993e02eddcc2edd90cdd4fec14b0c6e59000'
```

A new string!

```
>>> hashlib.sha256("
SECOND hello world + 838ee925a5eff45a3337363e9a2e993e02eddcc2edd90cdd4fec14b0c6e59000
    ").hexdigest()
'9b582a1b5fd41f36cc0a396d4747507148f3bdde58a4bff18891406a9158a72e'
```

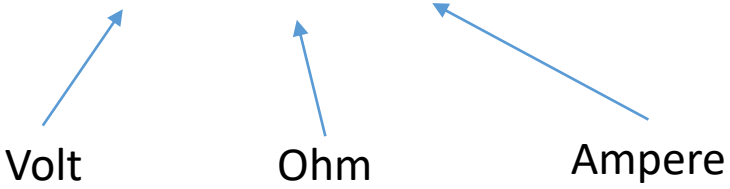
But this hash doesn't end with "000"… let's do the puzzle again:

```
SECOND hello world +
838ee925a5eff45a3337363e9a2e993e02eddcc2edd90cdd4fec14b0c6e59000 + 1659
'46eb8a72e26d849f9b755c39f0caa08013b61b4592f02ab446c4fa1b498d4000'
```

Per l’esame completo

Legge di Ohm

V=R*I



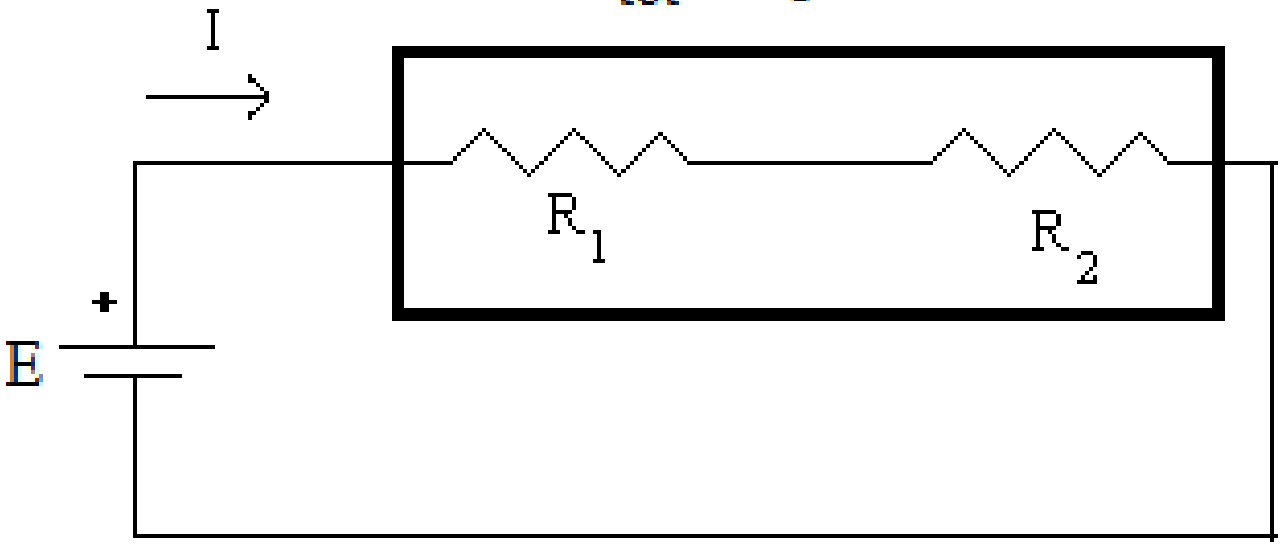
1)Se l'intensità di corrente che percorre un filo è 0,03 A, quando la tensione ai suoi estremi è 6 V , quanto vale la sua resistenza? [soluzione: 200 Ω]

2)Calcola la tensione ai capi di un conduttore , sapendo che l'intensità di corrente è 50mA e la reistenza conplessiva 50 Ω. [soluzione:2,5V]

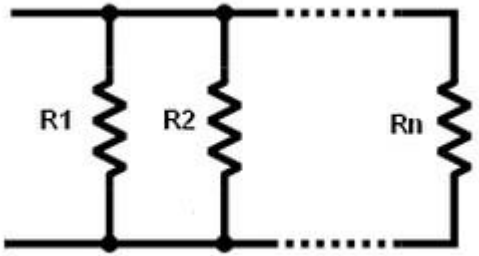
3)Calcola l'intensità di corrente in un circuito , avente tensione 4V e resistenza 40Ω. [soluzione: 100mA]

Resistori in serie

$$R_{tot} = R_1 + R_2$$

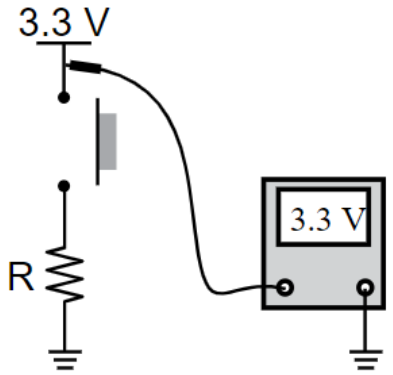


Resistori in parallelo

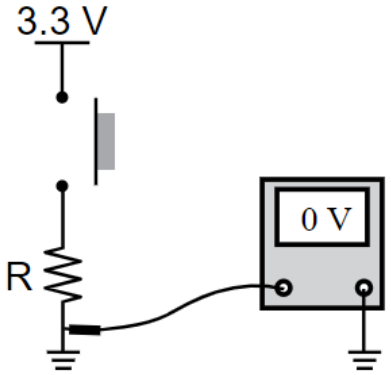


$$R_P = \frac{R_1 \cdot R_2}{R_1 + R_2}$$

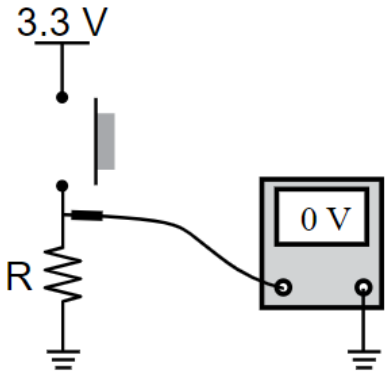
Pull-Down



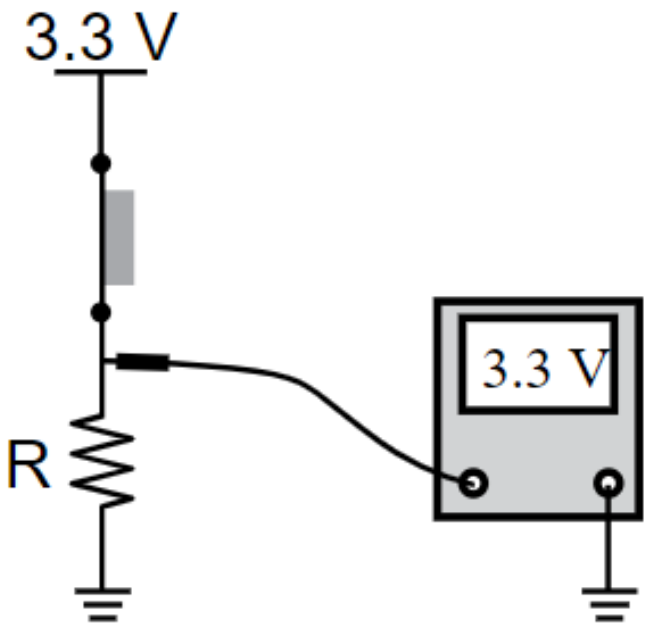
(a).



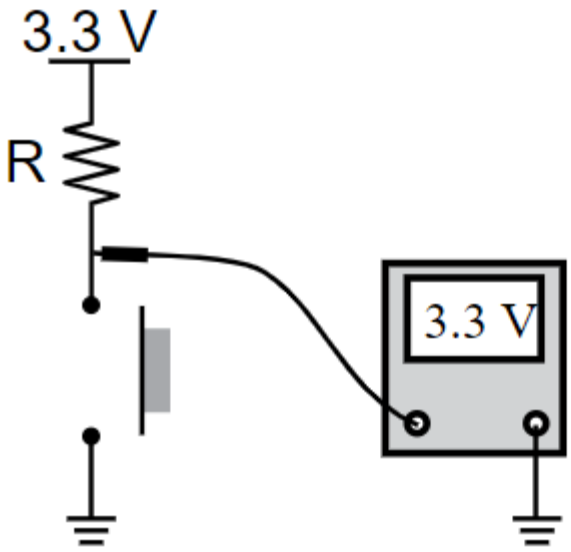
(b).

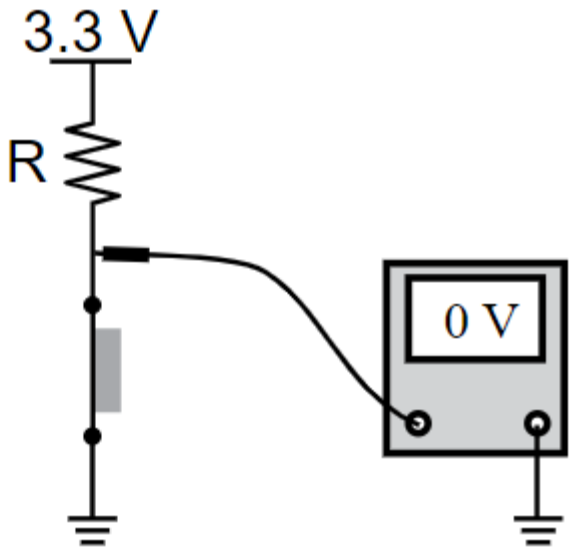


Pull-Down



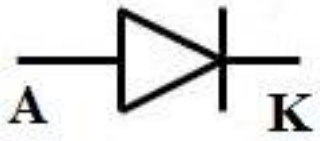
Pull-up





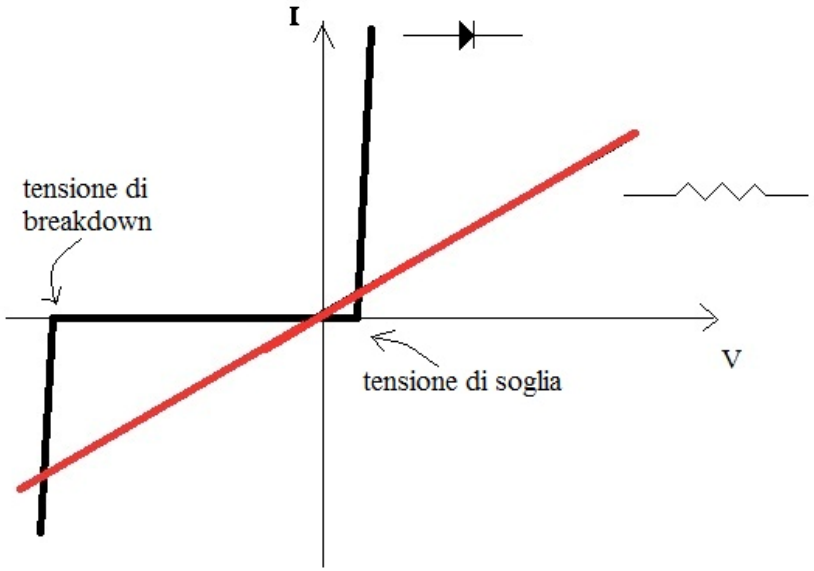
DIODO

- Il diodo permette il passaggio di corrente elettrica in un verso e la blocca nell'altro
  - il triangolo indica la freccia di direzione in cui il flusso di corrente è possibile
  - I due terminali del diodo vengono detti anodo (A) e catodo (K)



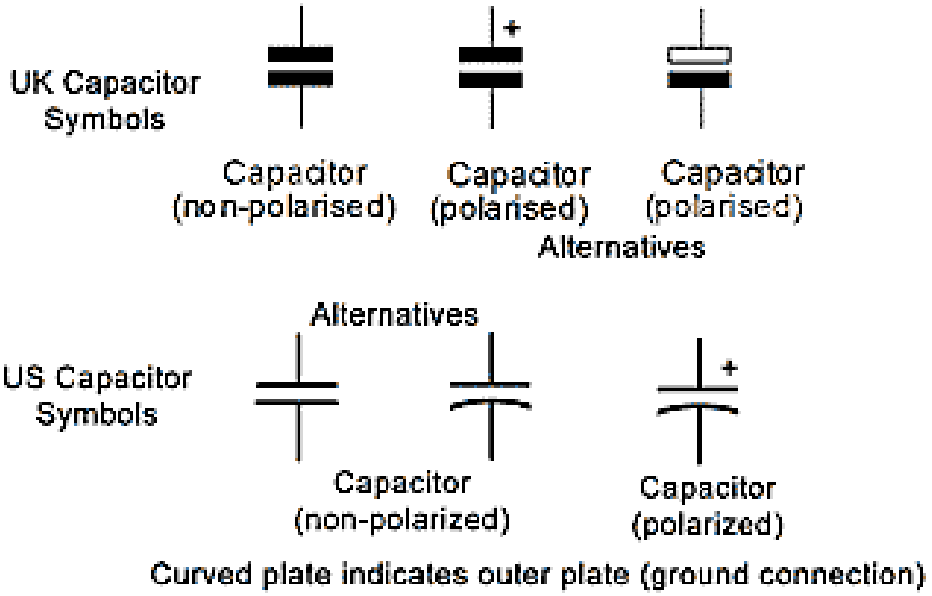
Non linearità del diodo

- La curva caratteristica del resistore è una retta (è un componente lineare)
- Il diodo invece è un componente non lineare, come si può facilmente osservare dalla forma della sua curva caratteristica, tutt'altro che rettilinea. Tensione e corrente non sono [illegible] proporzionalità diretta



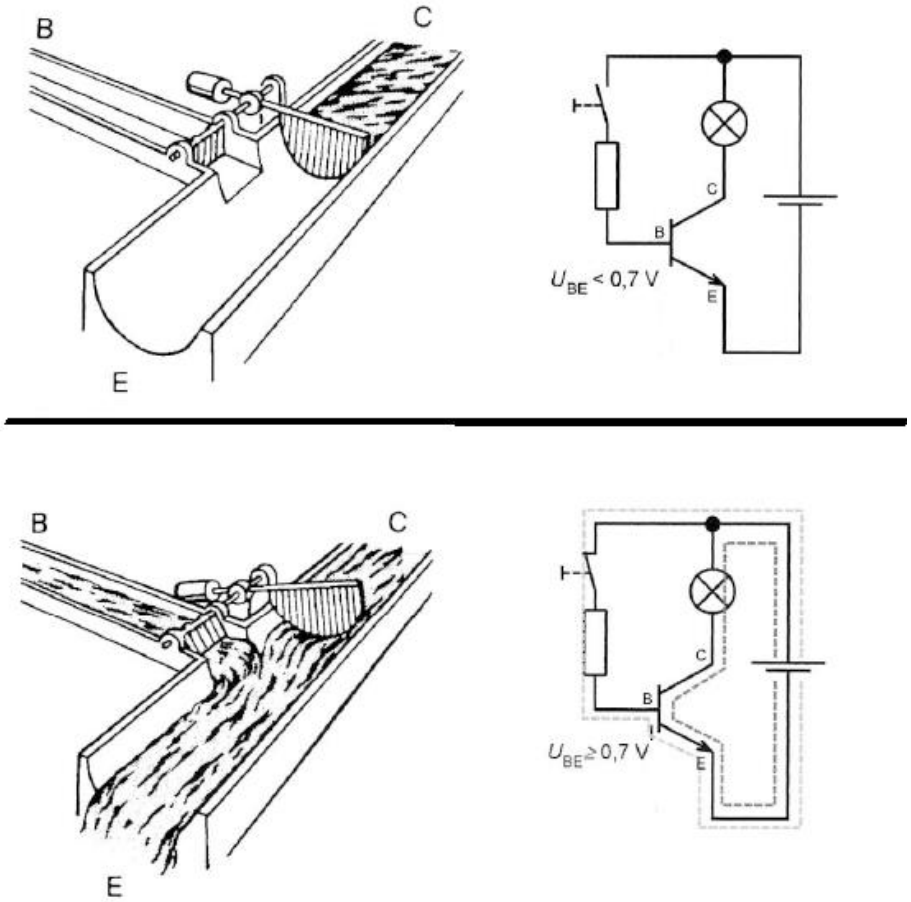
Condensatori

- Il condensatore (*capacitor*) è un bipolo in grado di immagazzinare al suo interno una certa quantità di carica elettrica
- La capacità C (Farad, F) è (intuitivamente) una misura della carica che il condensatore è in grado di accumulare

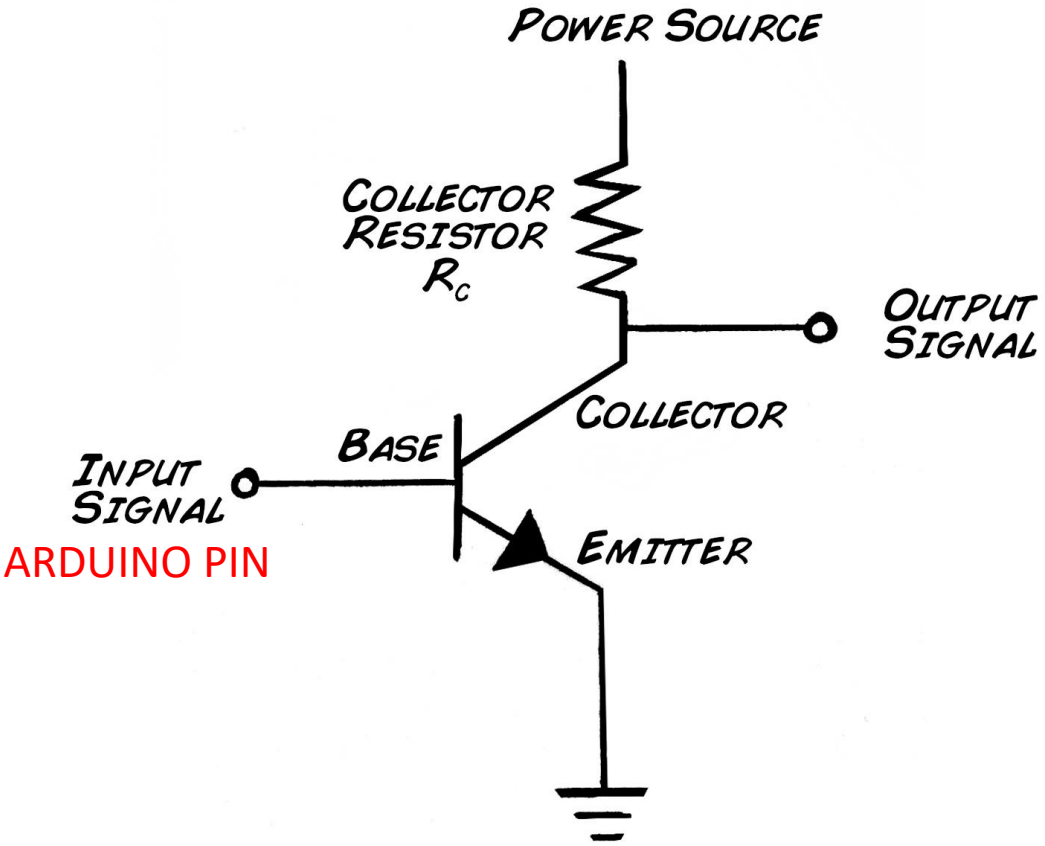


<u>Contrariamente alle resistenze, bisogna stare attenti alla polarità</u>

Transistor – principio di funzionamento



Transistor – schema tipico di collegamento



I transistor vengono utilizzati come switch digitali ma principalmente per controllare carichi che hanno bisogno di più corrente di quella erogabile da un microcontrollore