

# Sistemi di comunicazione

- Permettono di far dialogare la parte di elaborazione di un sistema embedded con altri sottosistemi digitali o con il mondo esterno
- Diverse tipologie con diverse caratteristiche:
  - Digitale vs analogico
  - Wired vs wireless
  - Seriale vs parallelo
  - Sincrono vs asincrono
  - Point-to-point vs bus vs stella vs...
  - Campionato o triggerato da eventi
  - Bit rate
  - Requisiti elettrici (voltaggio, corrente)
  - Controllo accesso, sicurezza, autenticazione
  - Connettori fisici (pin, porte...)



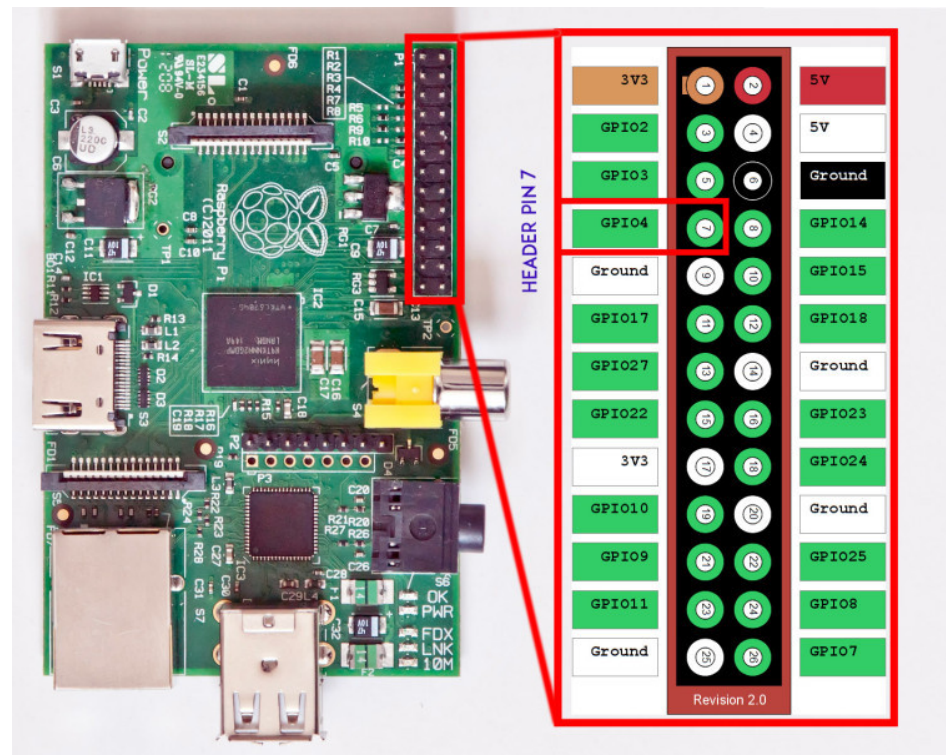
# Segnale digitale vs analogico

- Digitale = livelli fissi di corrente/tensione
  - Esempio: bit 0  $\rightarrow$  0 V, bit 1  $\rightarrow$  5 V
  - Utilizzati per input/output di tipo numerico, soprattutto da calcolatore a calcolatore
  - se la differenza tra le tensioni di 0 e di 1 è più grande, avrò più resistenza al rumore, ma meno velocità di trasmissione (banda passante analogica inferiore)
- Analogico = corrente/tensione variabile in un determinato range
  - Esempio: tutte le tensioni comprese tra 0 V e 5 V
  - Utilizzati per interfacciamento con parte fisica del sistema
  - Necessità di conversione A/D (input) o D/A (output)



# General-Purpose I/O (GPIO)

- Pin usati per input e/o output digitale
- Generano/leggono livelli di voltaggio binari, in genere 0 V e 5 V, anche 3.3V recentemente
  - In active high logic (logica diretta) 0 V → bit 0 e 5 V → bit 1
  - In active low logic (logica negata o inversa) 0 V → bit 1 e 5 V → bit 0
- Usato per interfacciamento con dispositivi esterni fisici

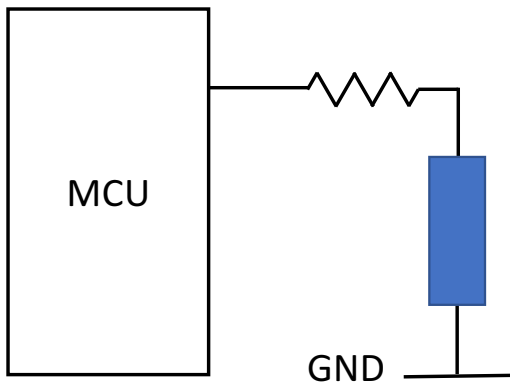


# Interfacciamento con GPIO (1)

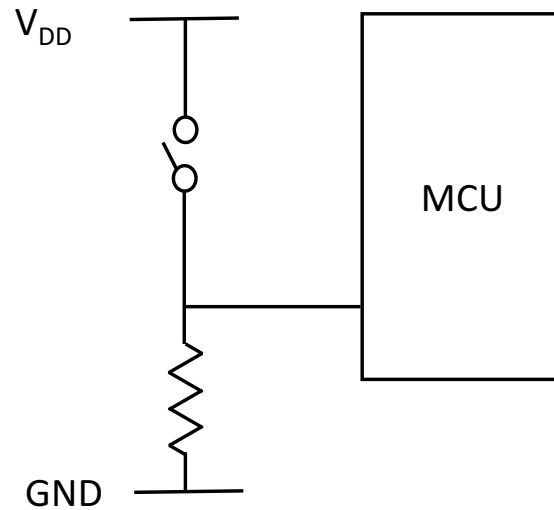
- Attenzione alla compatibilità elettrica!
  - Se un pin GPIO impone 5 V di tensione, la corrente nel dispositivo esterno è  $5/R$ , dove  $R$  è la resistenza del dispositivo
  - La corrente potrebbe superare la tolleranza del dispositivo (o del microcontrollore)!
- Se un pin GPIO è disconnesso (“flottante”) non è ben chiaro qual è l’input
- Se un pin GPIO è collegato ad un dispositivo esterno, questo potrebbe avere caratteristiche elettriche tali da introdurre rumore nel microcontrollore
- Infine, potrebbe essere necessario collegare più GPIO output allo stesso dispositivo (es., controllo con ridondanza)
  - Open collector / drain
  - Tri-state



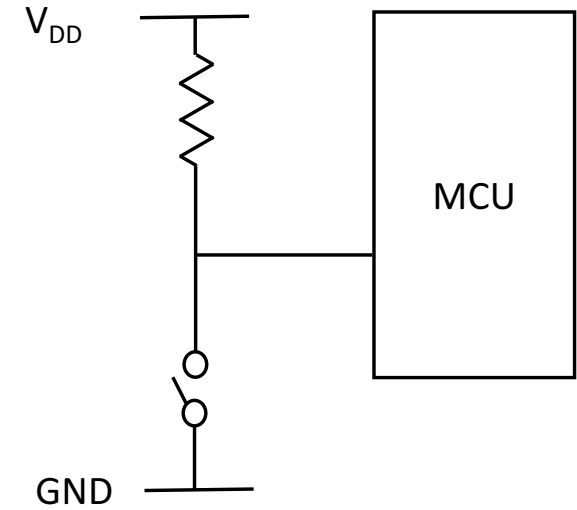
# Interfacciamento con GPIO (2)



Resistenza per limitazione corrente su pin di output



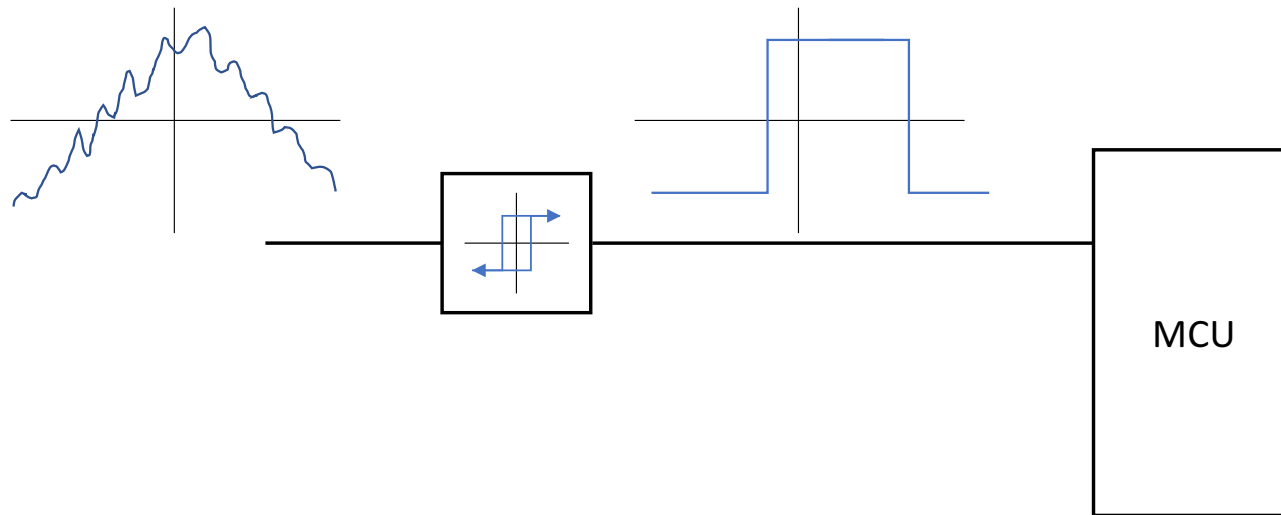
Resistenza pull-down per digital input ad alta impedenza



Resistenza pull-up per digital input ad alta impedenza



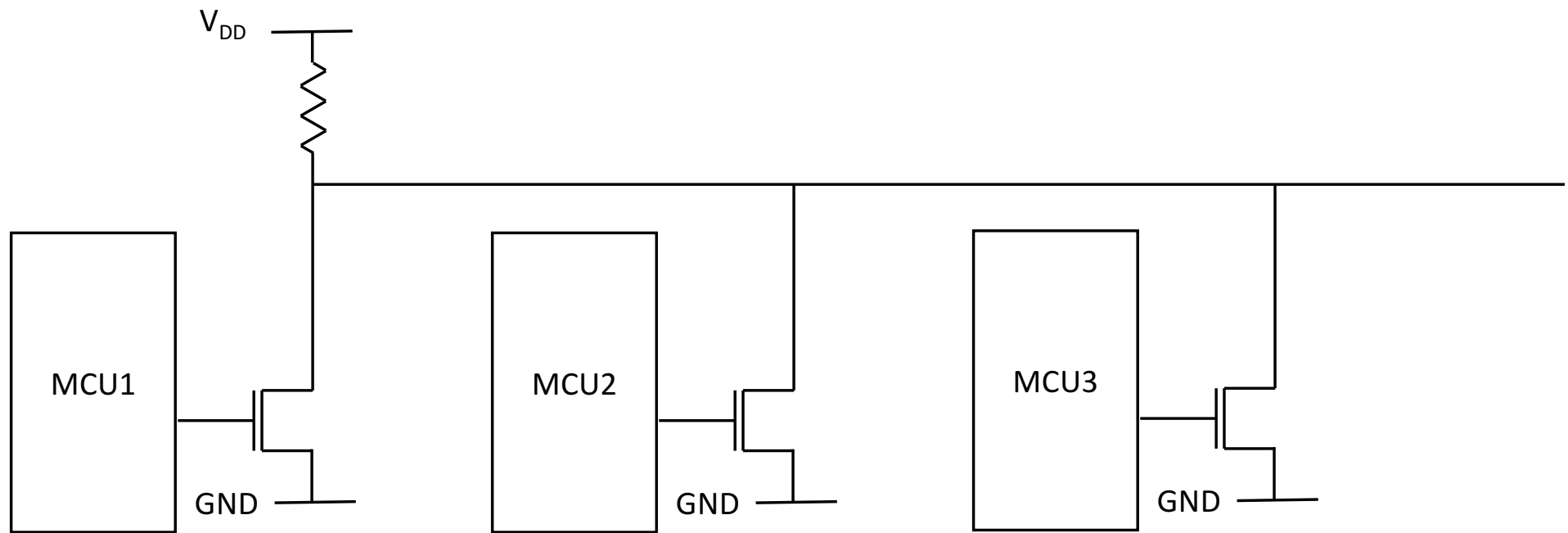
# Interfacciamento con GPIO (3)



Regolarizzazione digital input con trigger di Schmidt



# Interfacciamento con GPIO (4)




Circuiti open drain in configurazione wired NOR



# Pulse Width Modulation (1)

- La pulse width modulation (PWM) è un modo per fornire un segnale “pseudo-analogico” con dinamica lenta utilizzando esclusivamente circuiti digitali con commutazioni molto più frequenti
- Idea: commutare rapidamente il voltaggio su un pin di output digitale:  $0\text{ V} \rightarrow 5\text{ V} \rightarrow 0\text{ V} \rightarrow 5\text{ V} \rightarrow 0\text{ V} \dots$  se la velocità di questa commutazione è molto maggiore delle costanti di tempo del circuito che riceve il “segnale”, allora quel circuito non percepisce le commutazioni, ma solo il valore efficace (RMS)

$$x_{\text{RMS}} = \sqrt{\frac{1}{n} (x_1^2 + x_2^2 + \dots + x_n^2)} \quad f_{\text{RMS}} = \sqrt{\frac{1}{T_2 - T_1} \int_{T_1}^{T_2} [f(t)]^2 dt}$$


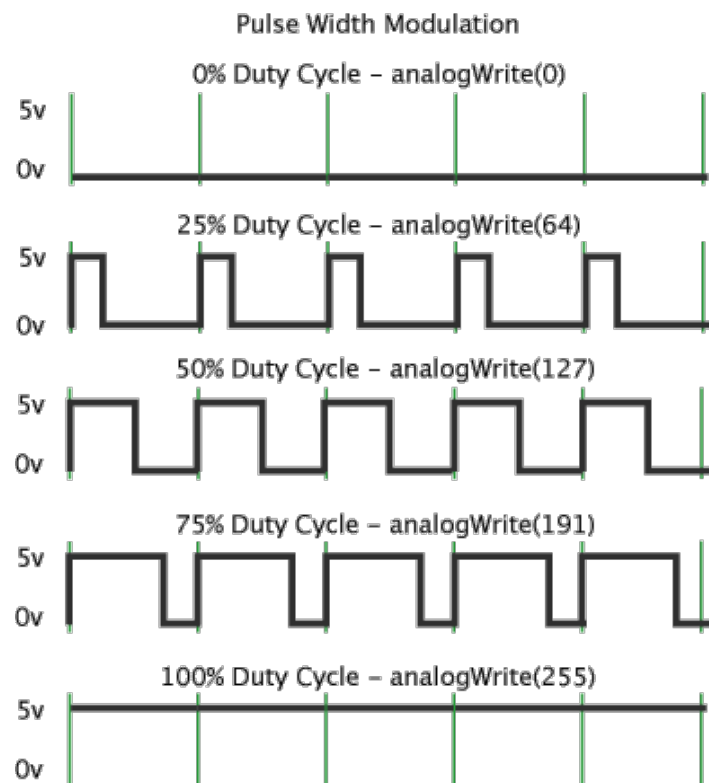


# Pulse Width Modulation (1b)

- Utile per comandare unità la cui risposta a cambiamenti di tensione/corrente è lenta rispetto alla frequenza a cui il sistema digitale può lavorare
  - Luci LED e ad incandescenza
  - Motori elettrici
  - Elementi termici (riscaldamento a resistenza)
- Ottimo metodo per controllare la potenza trasmessa ad un carico senza dissipare potenza aggiuntiva



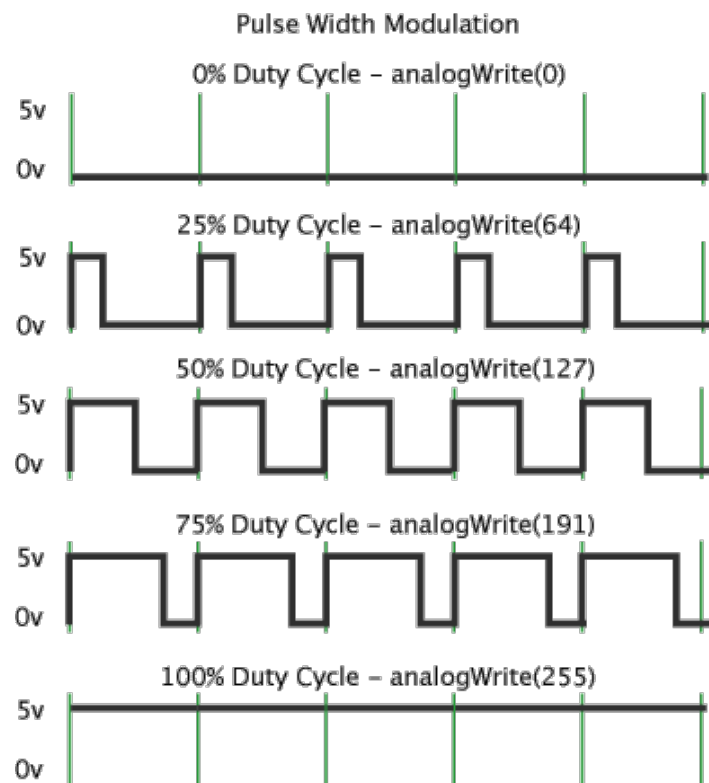
# Pulse Width Modulation (2)



- Principali parametri:
  - Frequenza = numero di commutazioni al secondo
  - Duty cycle = % tempo in cui il voltaggio è alto
- Il duty cycle determina la tensione media (pseudo-analog voltage):
  - Luce LED: impostando il duty cycle posso comandare il “dimming” della luce
  - Motore elettrico a corrente continua: impostando il duty cycle posso comandare la velocità di rotazione del motore
  - Elemento termico: impostando il duty cycle controllo la temperatura dell'elemento termico



# Pulse Width Modulation (2)

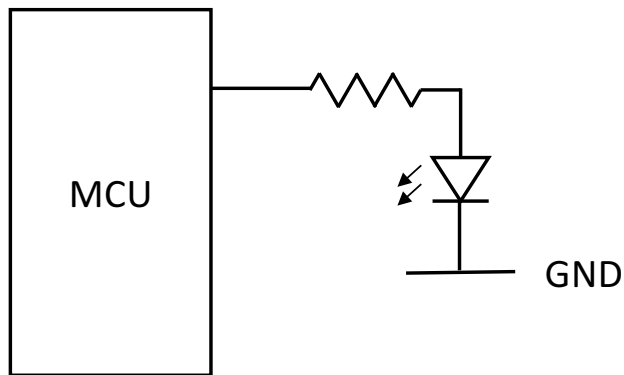


Si ipotizzi di fornire una tensione in uscita lavorando con PWM alla frequenza di 100KHz e di voler essere in grado di parzializzare la tensione RMS su 256 livelli. A quale frequenza minima dovrà lavorare il microcontrollore che accende e spegne l'uscita?

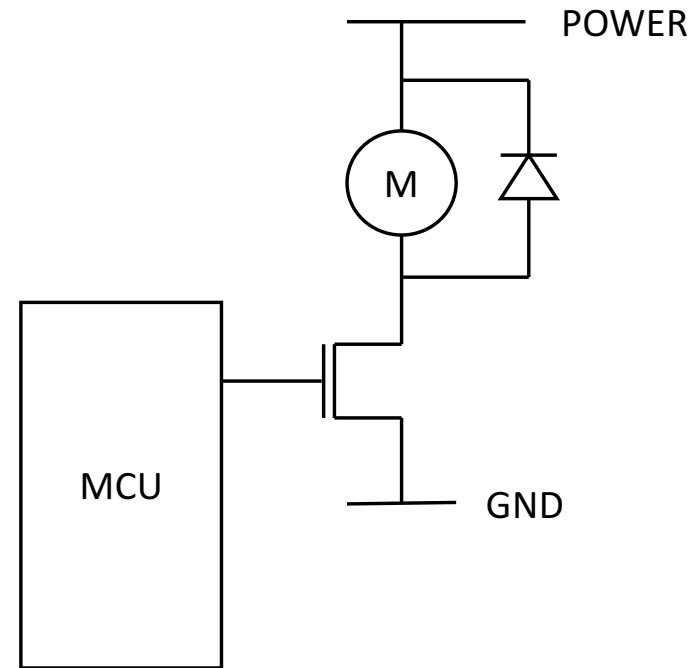
Si ipotizzi che il microcontrollore usato non abbia altro modo che gestire a controllo di programma questa periferica. Si facciano delle ipotesi sui cicli / istruzione e si determini quale frequenza minima deve avere il clock del processore.



# Pulse Width Modulation (3)



Collegamento a carico ridotto (LED)  
con resistenza per limitare la corrente



Collegamento a carico elevato (motore)  
in configurazione open drain con diodo flyback



# Connessioni digitali wired

- Parallelo = 1 linea per ogni bit, per una certa ampiezza di parola
  - (parallel) ATA, PCI, IEEE1284
- Seriale = 1 linea per direzione di flusso, singoli bit inviati in sequenza
  - RS232, USB, SATA, SPI, I2C
- Mixed = una o più “lanes” seriali
  - PCI express



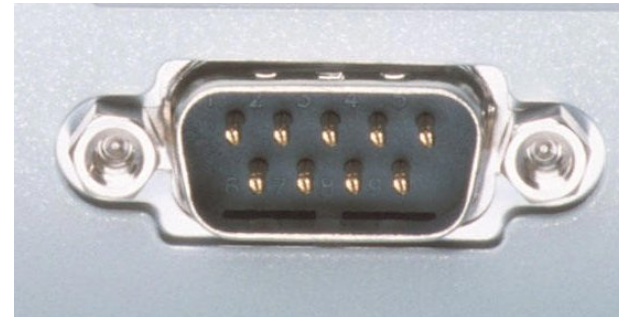
# Seriale vs parallelo

- Un'interfaccia parallela richiede un elevato numero di pin
  - Poco adatto per un sistema embedded
  - Di solito un'interfaccia parallela è half duplex per dimezzare il numero di linee
- Inoltre le linee dei diversi bit:
  - Devono essere sincronizzate tra di loro (problematico all'aumentare della distanza percorsa)
  - Fanno interferenza reciproca (inter-symbol interference) e sono più sensibili al rumore
  - Devono essere "pilotate" più lentamente a causa della mutua induttanza/capacità
- Oggi tutte le interfacce più recenti (e veloci) sono seriali
- Le interfacce parallele sono spesso emulate per retrocompatibilità usando i pin GPIO



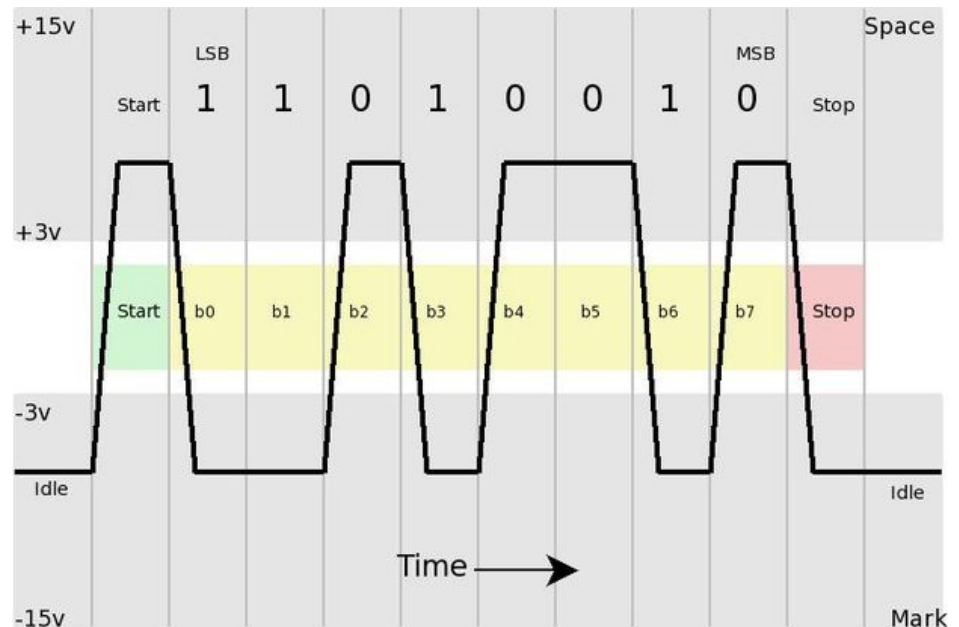
# RS-232 (1)

- Un “antico” protocollo ancora in uso (la vecchia “porta seriale” dei PC)
- Progettato per comunicazione terminale-modem
- Protocollo point-to-point asincrono (quello a 9 pin)
- Velocità non maggiore di 20 Kbit/sec a max 300 m di distanza



# RS-232 (2)

- Sincronizzazione con bit di start (1) e di stop (1 o 2)
- Trasmissione dei bit con livelli di tensione tra -15 V e -3 V (bit 1) e tra 3 V e 15 V (bit 0)
- Durata bit a seconda della velocità di trasmissione negoziata





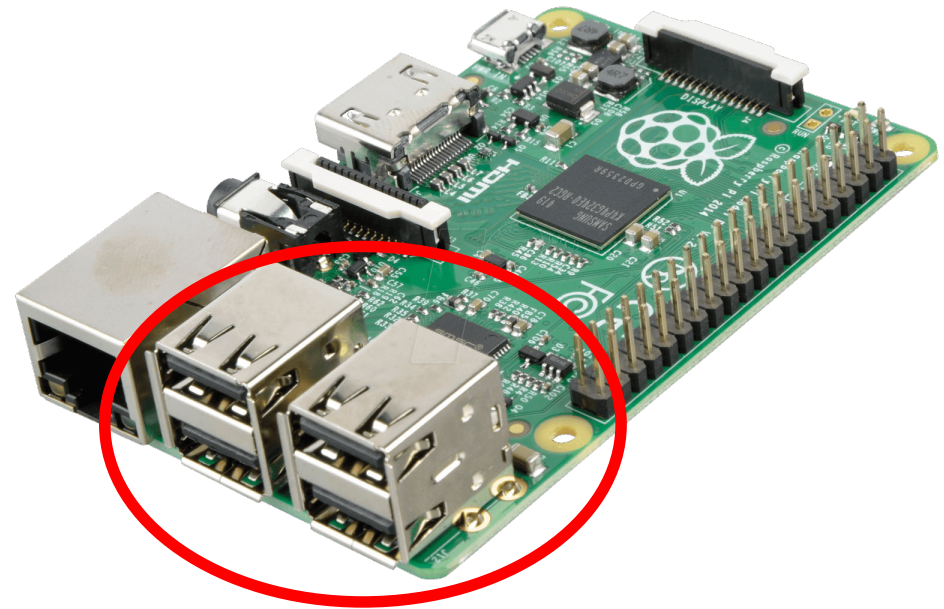
# UART e USART

- UART = Universal Asynchronous Receiver/Transmitter
- USART = Universal Synchronous/Asynchronous Receiver/Transmitter
- Sono componenti di un computer embedded (e di un PC) che trasformano una o più parole di memoria in una sequenza di trasmissione su un link seriale, e viceversa
- Usati per diversi tipi di protocolli seriali (RS-232, RS-422, RS-485)



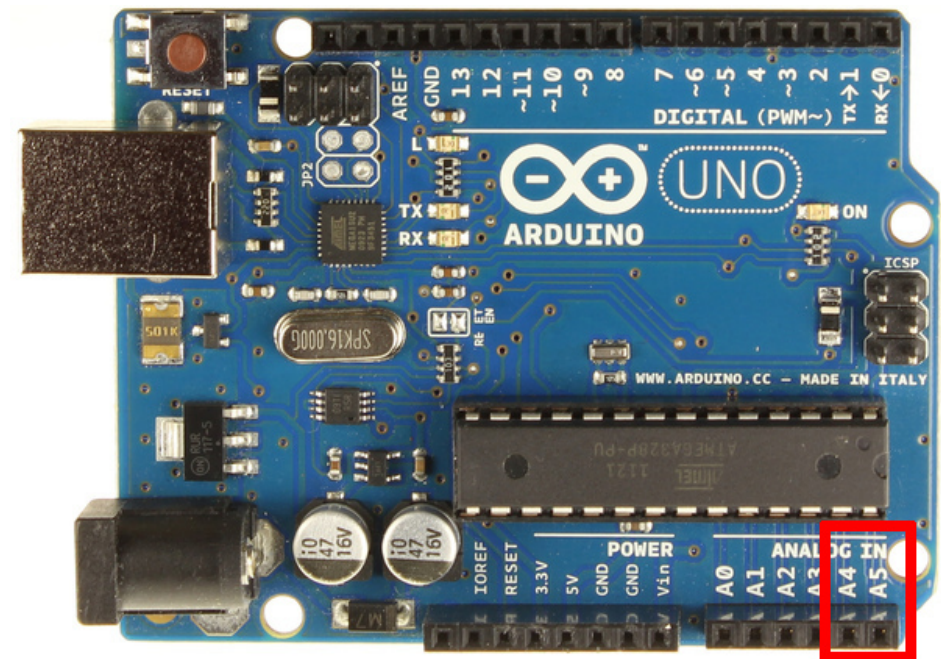
# Universal Serial Bus (USB)

- Protocollo seriale, topologia ad albero con un controllore (master), gli altri dispositivi sono controllati (slave), ma "appare" come un bus
- Max 25 m distanza, max 10 Gbit/sec
- Elettricamente più semplice e robusto di RS-232, richiede una logica di controllo sofisticata
- Consente di collegare una grandissima varietà di dispositivi



# Inter-Integrated Circuits (I2C) e System Management Bus (SMBus)

- Bus di comunicazione seriale tra circuiti integrati, con multi-master e arbitrato
- Max 3.4 Mbit/sec, corta distanza
- Solo 2 fili!
- Interfacciamento open drain (richiede resistenze di pull-up)
- Principalmente per comunicazione tra componenti a bassa velocità su stessa scheda, o anche su schede diverse
- Le differenze tra I2C e SMBus sono minime (compatibilità!)

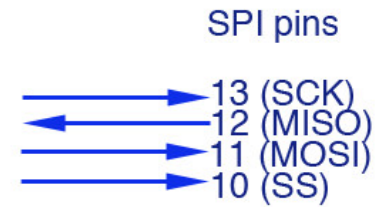
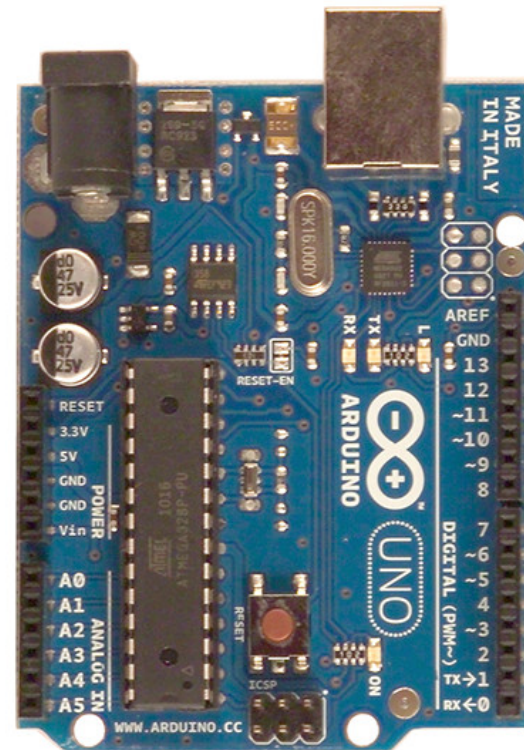


I2C Pins



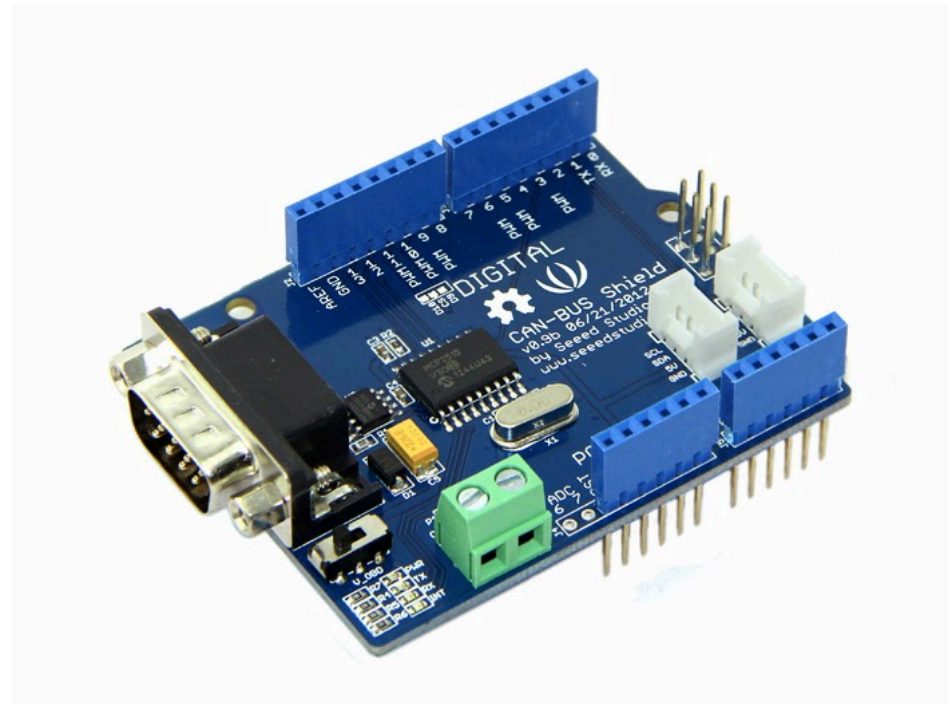
# Serial Peripheral Interface (SPI)

- Un altro tipo di bus seriale
- Differenze con I2C:
  - Non richiede resistenze di pull-up (minori consumi)
  - Il numero di linee è più elevato
  - Il throughput è molto più alto
  - Supporta un singolo master
- SPI è più adatto per trasferire data streams, I2C è più adatto per trasferire parole singole



# Controller Area Network (CAN) bus

- Bus seriale sviluppato da Bosch per connessione dispositivi in ambito industriale (elevato rumore elettromagnetico)
- Fino 1 Mbit/s sotto i 40 m, max 500 m a 125 Kbit/s
- Asincrono, multi-master basato su priorità



# JTAG

- Protocollo hardware per il testing di circuiti integrati e stampati
- Originariamente creato per il boundary testing (ossia, test dei pin dei circuiti integrati presenti sulla scheda)
- Nelle implementazioni più recenti JTAG viene usata anche come porta per il debugging

