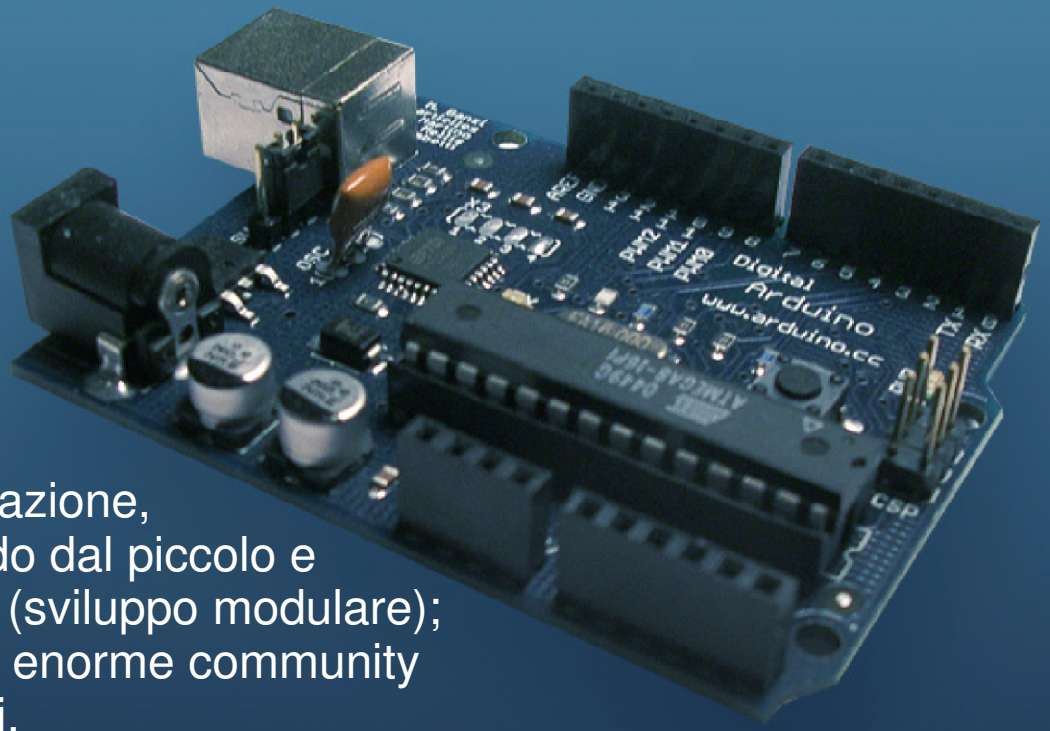
Arduino

Physical Computing I/O board



- Arduino è orientato all'applicazione,
- Permette di provare, partendo dal piccolo e crescendo pezzo per pezzo (sviluppo modulare);
- Lo sviluppo è aiutato da una enorme community online con esempi e consigli.
- Economico
- Connessione USB

Author: Ing. Sebastiano Giannitto (ITIS “M.BARTOLO” –PACHINO)

2^ parte

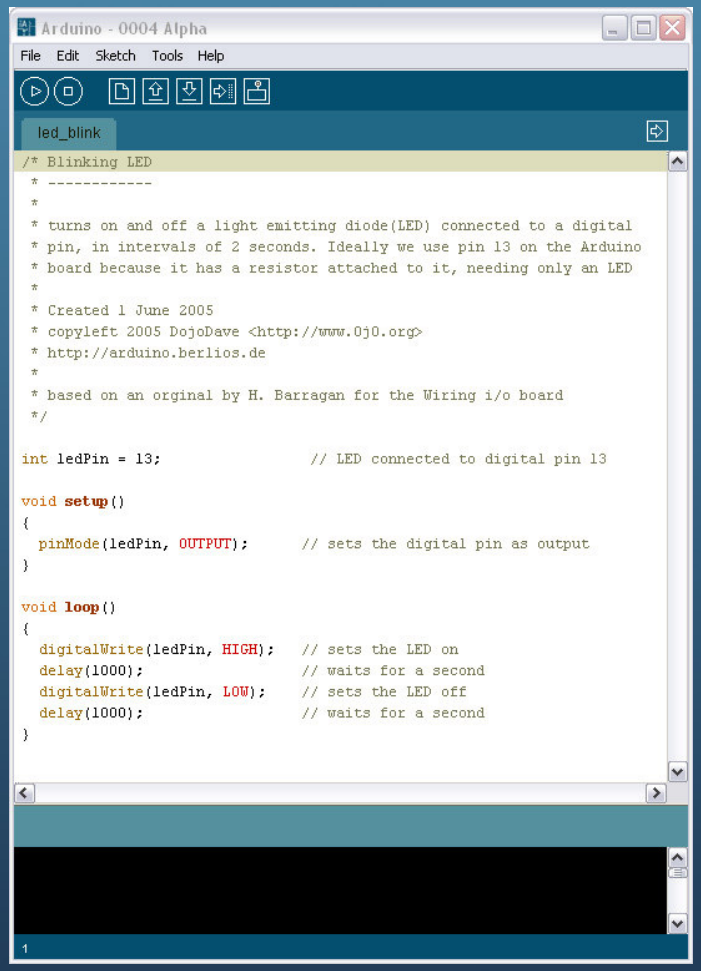
Arduino è una piattaforma

Un ambiente di sviluppo integrato ovvero IDE Integrated Development Environment for programming arduino-1.0.1 (scaricabile dal sito e installato sul tuo PC)

Può essere programmato attraverso un linguaggio del tutto simile al C basato sul www.processing.org language.

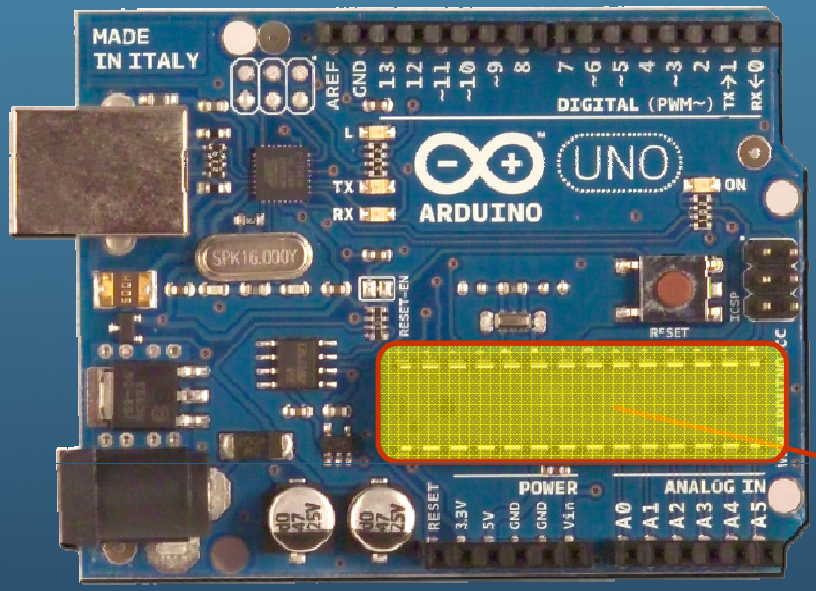
Una volta creato il codice lo si scarica sul µC della scheda che viene vista come una periferica dalla quale acquisire informazioni sulla porta seriale virtuale che il driver della scheda installa automaticamente

Scritto il programma Arduino potrà operare anche autonomamente eseguendo le istruzioni inserite al suo interno.



Interfaccia con cui sviluppare il software per arduino

Com'è fatto Arduino



Micro controllore:
Arduino è basato su un microcontrollore, della famiglia AVR di Atmel, l'**ATmega328** a 28 pin e a 8 bit

Il microcontrollore è il «cervello» di tutto il sistema, gestisce gli input, gli output, la comunicazione USB, la temporizzazione.

Necessita quindi di istruzioni → deve essere programmato !!!

Non necessita di un programmatore in qunato la scrittura del codice nella flash memory avviene tramite il collegamento USB al PC.

Microcontrollore: Approfondimento

Il microprocessore è il nucleo centrale di un calcolatore; esso è l'unità di elaborazione dei dati e di controllo del funzionamento del calcolatore stesso e viene spesso indicato con la sigla CPU (Central Processing Unit). Posto da solo, il uP non è utilizzabile, infatti sia i dati che i programmi su cui il processore opera sono immagazzinati in un'unità di memoria esterna a causa della grande quantità di memoria richiesta.

Per applicazioni particolari, tipiche del controllo industriale si fa invece uso dei così detti Microcontrollori (uC).
Un microcontrollore è un sistema a microprocessore completo, integrato in un solo *chip*, progettato per ottenere la massima autosufficienza funzionale ed ottimizzare il rapporto prezzo-prestazioni per una specifica applicazione. I uC comprendono, oltre all'unità di calcolo, anche la memoria (RAM e ROM) e ulteriori periferiche di input/output (convertitori analogico/digitali, timer, interfaccia USB) a seconda dell'applicazione specifica.

Microcontrollore: Approfondimento

I vantaggi dell'utilizzo dei micro:

- Sono richiesti meno dispositivi "discreti" per la realizzazione di un sistema
- Il sistema ha dimensioni ridotte
- Diminuiscono i costi
- Diminuisce il consumo di potenza
- Diminuisce la sensibilità all'ambiente (temperatura, EM, …)
- Utilizza meno componenti, quindi più affidabile.
- Riconvertibilità del progetto (riprogrammabile)
- Protezione contro la copia
- Interfacciamento semplice con altri dispositivi (PC, LCD, …)

Ma dove si usano???

- Componenti PC: mouse, tastiere, modem, carica batterie…
- Orologi, calcolatrici
- Serrature per porte, sistemi d'allarme
- Automotive: in una BMW X5 sono contenuti più di 70 microcontrollori.

Arduino è basato su un microcontrollore, l'ATmega328

La grande community online e il fatto di essere OpenSource lo rendono molto più semplice ed intuitivo rispetto al normale utilizzo di microcontollori (PIC, ARM…)

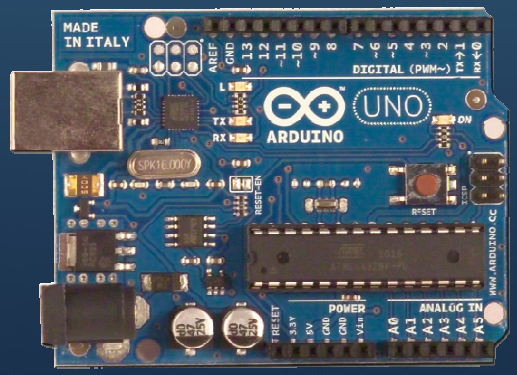
Arduino UNO

Dispone di 14 ingressi / uscite digitali (di cui 6 possono essere utilizzate come uscite PWM), 6 ingressi analogici, un oscillatore a 16 MHz, una connessione USB, un jack di alimentazione, un header ICSP, e un pulsante di reset. Esso contiene tutto il necessario per programmare il microcontrollore, per iniziare è sufficiente connettersi a un computer con un cavo USB e a un alimentatore AC-DC o batteria.

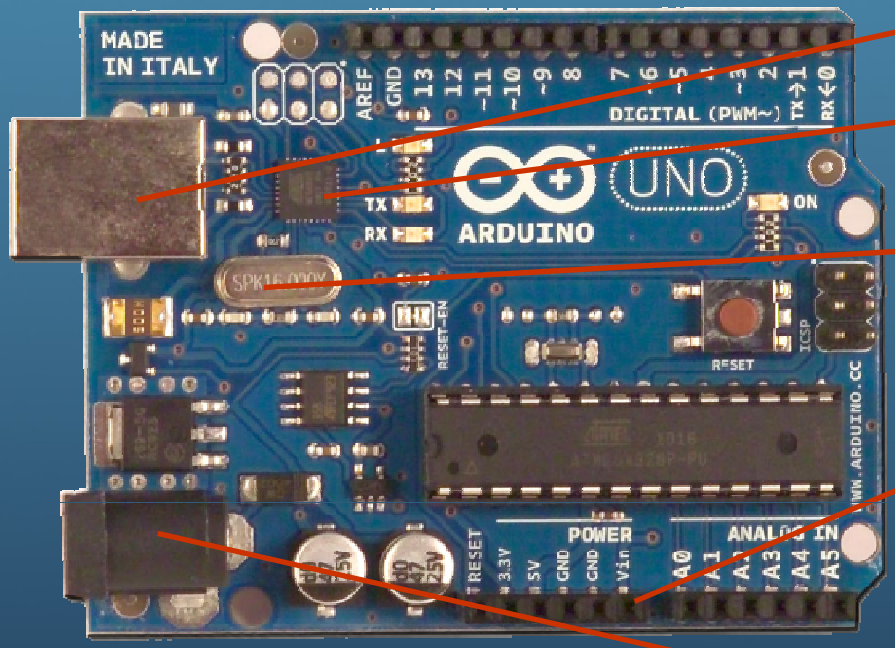
L'Arduino uno si differenzia da tutte le schede precedenti in quanto non utilizza il convertitore USB-seriale. La versione di partenza Arduino uno e la V1.0

Caratteristiche tecniche:

- Microcontrollore ATmega328
-Tensione di lavoro 5V
-Tensione di ingresso (consigliato) 7-12V
-Tensione limite 6-20V
-**14 pin digitali I / O** (di cui 6 forniscono uscite PWM)
-6 pin di ingresso analogico
-Corrente per I / O 40 mA
-Corrente per 3,3 V 50 mA
-32 KB di memoria Flash di cui 0,5 KB utilizzati dal bootloader
-SRAM 2 KB
-EEPROM 1 KB
-Velocità di clock 16 MHz



Com'è fatto Arduino

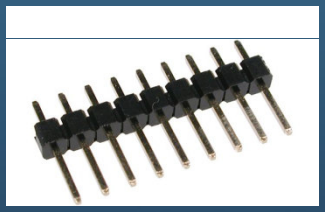


Alimentazione da USB da collegamento al pc

µC per convertire i segnali seriali in segnali USB

Quarzo per la generazione del clock 16 MHz

Alimentazione da Vin (6 - 20V)



Alimentazione da Jack (6 - 20V)

Si consiglia una tensione tra 7 e 12V. Che viene poi ridotta ai 5V e a 3,3 da un regolatore presente sulla scheda

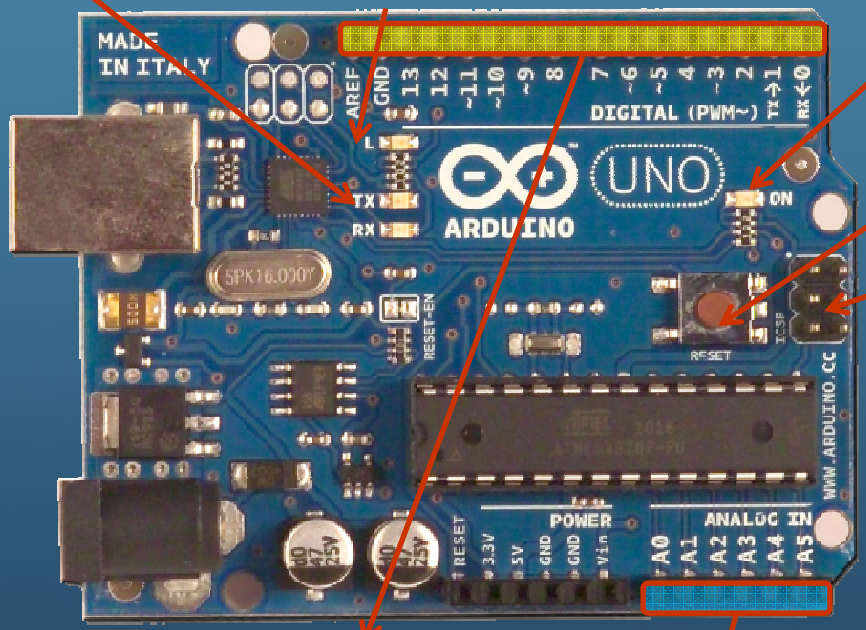
Se utilizziamo un alimentatore esterno, sul pin Vin si troverà la tensione di alimentazione vera e propria senza regolazione.

Arduino seleziona automaticamente la sorgente di alimentazione. Solo il vecchio Arduino diecimila necessita di un selettore.

Com’è fatto Arduino

Led di trasmissione e ricezione tra µC e PC

Led di lavoro



Led di accensione

Pulsante di reset

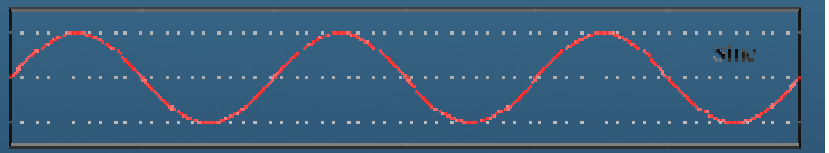
Connettore ICSP per collegare u programmatore

Connettore strip line a 14 Ingressi/Uscite digitali

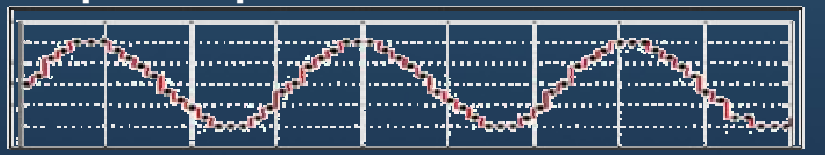
6 Ingressi analogici

Nota:

Segnale analogico: è un segnale a **tempo ed ampiezza continua**.



Segnale digitale o numerico: è un segnale a **tempo discreto e ad ampiezza quantizzata**.

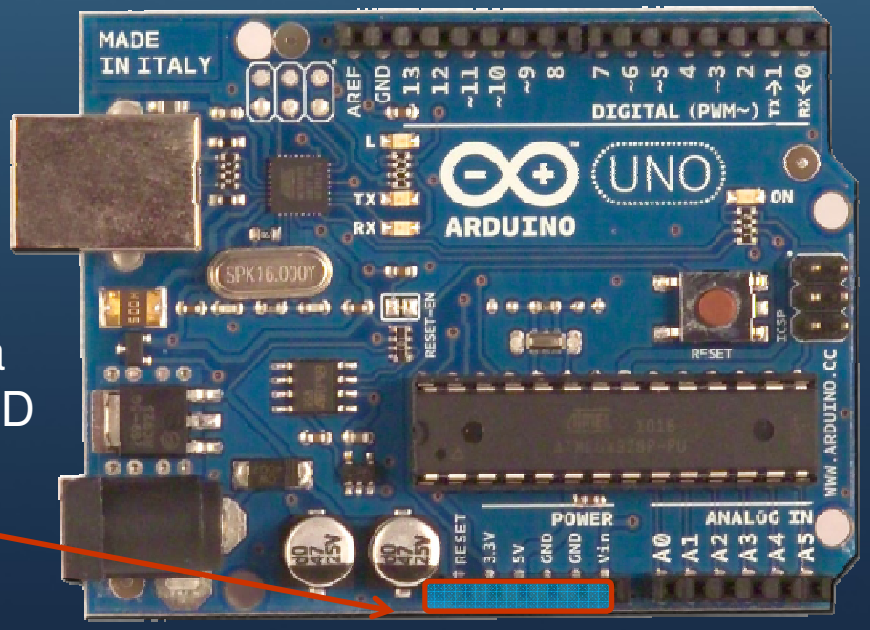


Principali blocchi della scheda ARDUINO

- **Microcontrollore Atmega328**, presenta:
  - 8 bit con frequenza di clock a 16 MHz
  - 20 pin di I/O multifunzionali suddivisi in 3 Port: B,C e D
  - Memoria Flash da 32KB (di cui 0,5 KB occupati dal bootloader per il caricamento dei programmi)
  - una EEPROM da 1KB
  - una memoria volatile SRAM da 2KB
  - Dispone di periferiche per la comunicazione seriale di tipo I2C e SPI
  - Ingressi analogici e digitali e uscite PWM

- **Convertitore seriale/USB** svolta dal controlllore **ATmega8U2**

- **Alimentazione** fornita da pc tramite cavo USB o da adattatore AC/DC fra 7 e 12 V da applicare tramite jack o tra i piedini 6 e GND del connettore POWER ai quali si uò applicare in alternativa una batteria. Nel connettore abbiamo anche la possibilità di prelevare 5V e 3,3 V da appositi piedini.



Alimentazione a batteria

Dimensionamento della batteria

Il dimensionamento della batteria è un aspetto fondamentale di un progetto.

Arduino One, Mega e 2009, grazie al connettore esterno, permettono di alimentare il sistema per mezzo di batterie… ma quali usare?

La batteria da utilizzare dipende da:

1. Tempo d'uso
2. Corrente richiesta dal sistema
3. Tensione richiesta dal sistema
4. Dimensione massima (in cm) della batteria

Se per esempio abbiamo un circuito che necessità 3,3V, consuma 150mA e vogliamo che la batteria duri almeno 8 ore, dovremo scegliere un batteria con tensione maggiore di 3,3V (il circuito avrà un regolatore, come Arduino) e una 'capacità' di almeno 150mA*8*1,2 = 1440mAh

1,2 è un fattore di correzione per assicurarsi la durata voluta.

Pin Digitali: Generici ed specifici

Ognuno dei **14 pin Digital I/O** dell'Arduino può essere utilizzato sia come input che come output.

Pertanto Arduino è in grado di acquisire informazioni da sensori e gadget elettronici e al contempo pilotare motori, emettere suoni o accendere luci.

I pin Digital I/O operano ad una tensione di **5V** e possono fornire fino a **40mA** di corrente

Alcuni di questi pin hanno funzioni specifiche:

Serial: pin 0 (RX) e pin 1 (TX). Sono rispettivamente il pin di trasmissione e ricezione per la comunicazione seriale. Lavorano a 5V e sono connessi con l'USB. Possono essere utilizzati per connettere un modulo **bluetooth**. In questo caso il modulo deve essere scollegato per permettere la scrittura del firmware attraverso l'USB.

External Interrupts: pin 2 e 3. Questi pin possono essere configurati per la **generazione di interrupt**. Possono cioè essere configurati in modo che se il valore del pin cambia, l'esecuzione del codice viene interrotta momentaneamente per eseguire un'altra operazione, associata al cambiamento del pin

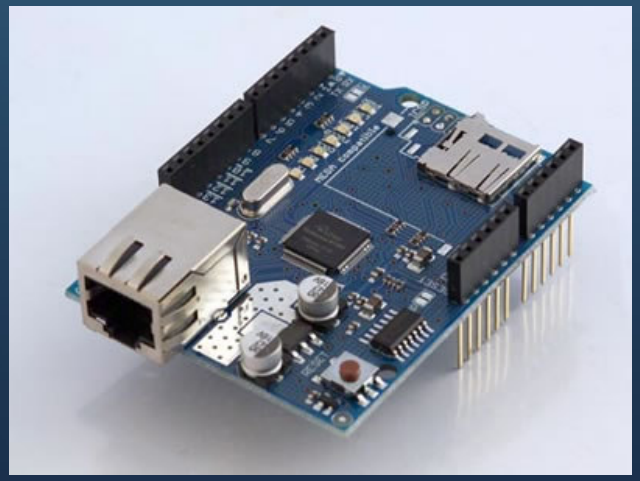
Pin Digitali:Generici ed specifici

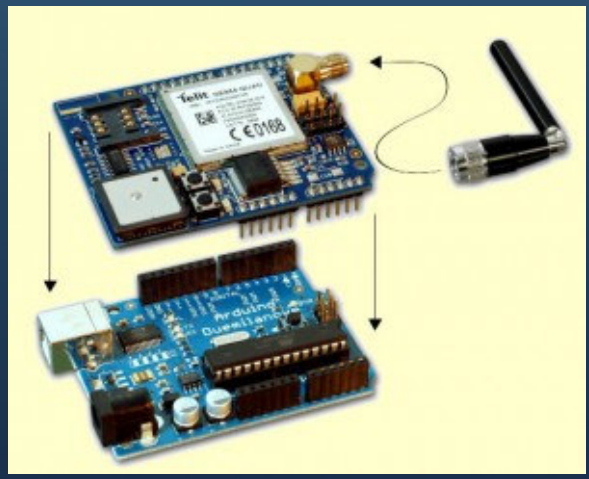
PWM: 3, 5, 6, 9, 10, e 11. Possono essere utilizzati come uscite PWM ossia come onde quadre con duty cicle regolabile

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Questi pin possono essere utilizzati per la **comunicazione** SPI con alcuni dispositivi (SDCARD, Ethernet shield, GSM).

LED: 13. Il pin 13 è collegato ad un led SMD presente sulla scheda.

I²C: 4 (SDA) e 5 (SCL). Configurabile come I²C (TWI) (simile a SPI).





Pin Digitali:
Pulse Width Modultation (PWM)

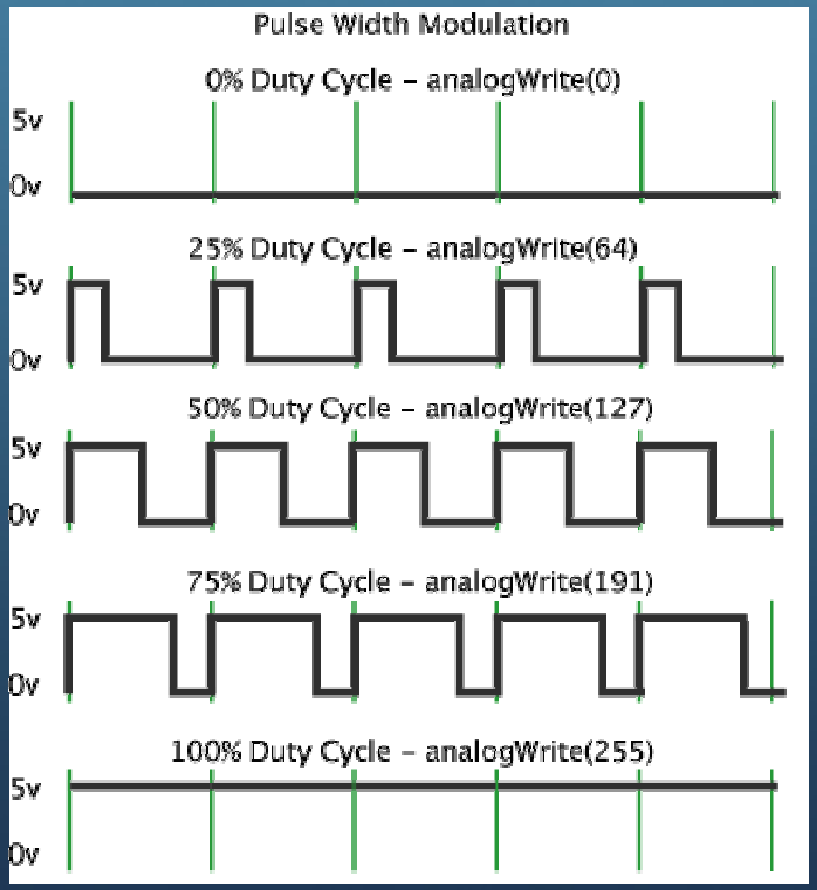
La PWM – Pulse Width Modulation è una tecnica utilizzata per la generazione di un segnale analogico utilizzando un uscita digitale.

Variando la lunghezza dell'impulso posso generare dei valori analogici da 0 a Vcc (5V per Arduino).

Il duty-cycle è il rapporto tra il periodo dell'impulso al valore logico alto sul periodo in percentuale.

La frequenza di lavoro del PWM di Arduino è circa 470Hz.

Pilotando un led con questa tecnica posso far assumere diverse gradazioni di luminosità, l'occhio non percepisce il continuo on/off ma un livello differente di luminosità.



Pin Digitali: Come settarli

La modalità di uscita o di ingresso dei pin digitali da 0 a 13 viene stabilita dalle seguenti istruzioni, dove alla variabile X va sostituito il numero che contraddistingue il pin sulla scheda:

pinMode (X,OUTPUT) ; ***es. pinMode(3,OUTPUT) pin3 settato come uscita***

pinMode (X,INPUT); ***es. pinMode(8,INPUT) pin8 settato come ingresso***

Se il pin X è impostato come uscita per settarlo a '1' o a '0' si utilizzano le seguenti istruzioni

digitalWrite(x,HIGH);

digitalWrite(x,LOW);

Per leggere il livello di un ingresso si utilizza la seguente istruzione che trasferisca alla variabile val i valori HIGH o LOW

Val=digitalRead(x);

Uscite PWM: Come settarle

I pin 3,5,6,9,10,11, sono in grado di fornire uscite PWM, ossia onde quadre con duty cicle regolabile, tramite l'istruzione:

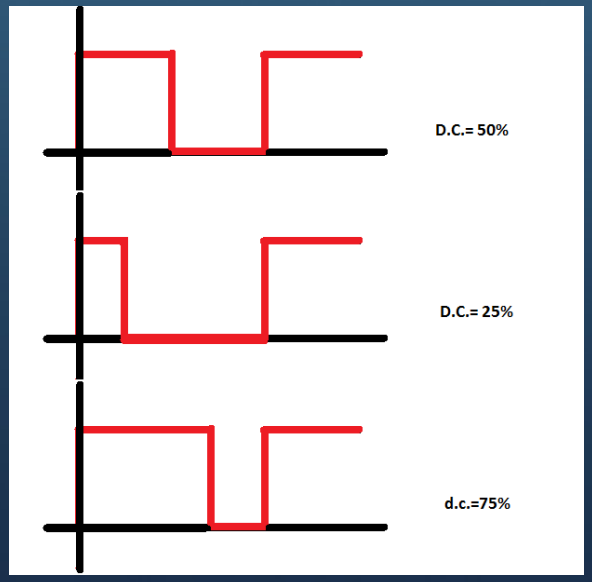
analogWrite(X,valore) ;

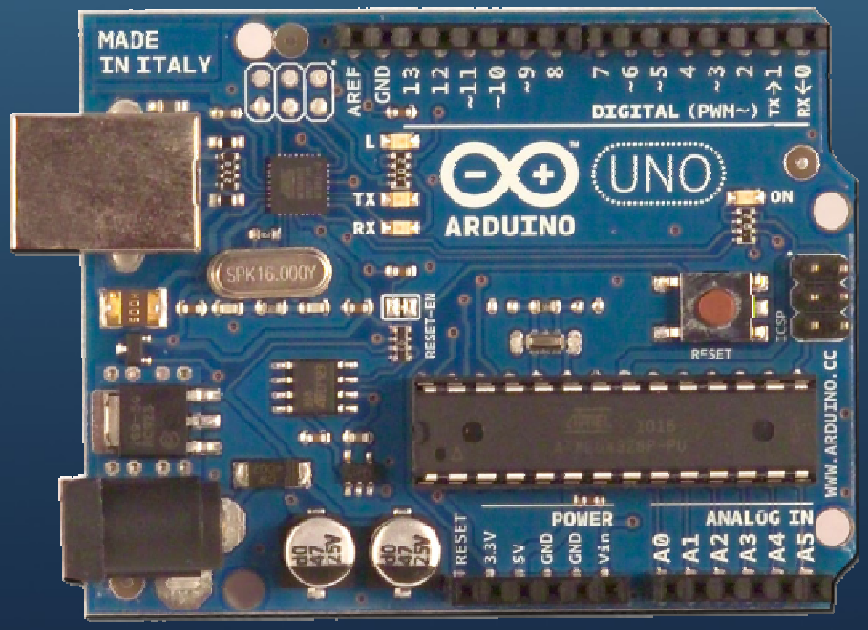
Dove valore è un numero compreso tra 0 e 255 cui corrisponde un duty cicle da 0 al 100%

analogWrite(3,127) ; genera sul pin 3 un'onda quadra con D.C. del 50%

analogWrite(5,63); genera sul pin 5 un'onda quadra con D.C. del 25%

analogWrite(3,191); genera sul pin 3 un'onda quadra con D.C. del 75%





Ingressi Analogici:
Convertitore Analogico-Digitale (ADC)

Arduino ha inoltre **6 ingressi analogici** (A0…A5) ognuno dei quali ha una risoluzione a 10bit (cioè riconosce 2^10 = 1024 intervalli di tensione differenti).

Il **convertittore analogico-digitale (ADC)** interno di Aduino è settato di default per acquisire valori tra **0 e 5V.**

Questo vuol dire che l'intervallo di 5V sarà diviso in 1024 intervalli.

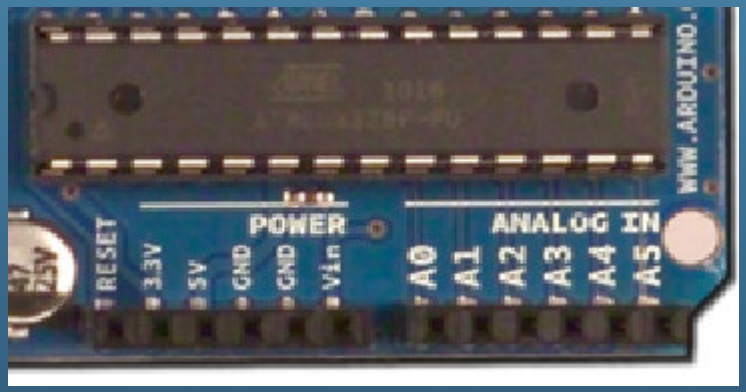
E se volessimo acquisire un segnale tra

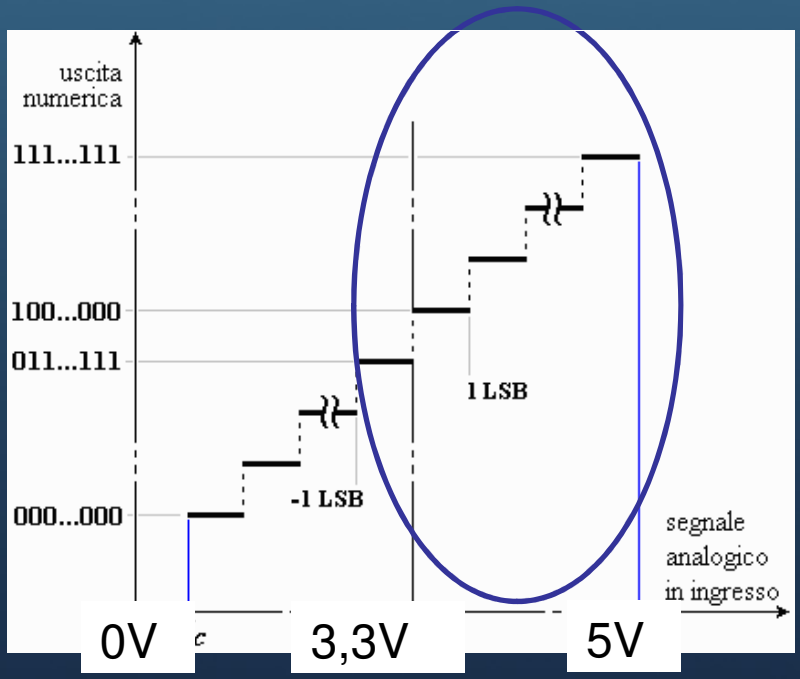
0 e 3,3V?

Parte dei livelli di quantizzazione sarebbero

Inutili.

Per tale motivo è presente il pin 21 detto **AREF,** col quale per mezzo di una apposita funzione che vedremo in seguito, si può fissare il valore di riferimento (il valore massimo) per l'ADC.





Ingressi Analogici : Convertitore Analogico-Digitale (ADC)

L'istruzione che compie la lettura di un ingresso analogico (tra 0 e 5V) e di convertirlo in un numero compreso tra 0 e 1023 assegnadolo ad una variabile val, è la seguente:

Val=analogRead(x); con x compreso tra 0 e 5

Per cui se Vin=5V sarà val=1023
se Vin=2,5V sarà val=511
se Vin=1V sarà val=205

Un altro pin molto utile è il **RESET**.

Questo pin, se posto a 0, permette di resettare lo stato dell'arduino.

È possible resettare Arduino sia per mezzo del pulsante presente sulla board, sia attraverso questo pin.
Durante la programmazione questo pin è posto basso per resettare Arduino e permetterne la programmazione.

Linee seriali

I pin 0 (RX) e 1 (TX) del connettore DIGITAL della scheda consentono la comunicazione seriale fra il microcontrollore e il PC attraverso il convertitore seriale-USB.
Grazie a queste linee il programma scritto sul pc viene inviato al µC e dati dal µC possono essere inviati al pc.

Per la comunicazione dal **µC** ⇨ **PC** sono necessarie queste istruzioni:

- ***Serial.begin(speed) ;*** impostando in speed la velocità di trasmissione (9600bps)
- ***Serial.println(data);*** per inviare I dati da visualizzare sul monitor nell'apposita finestra Serial Monitor dell'interfaccia grafica.

La trasmissione dal **PC** ⇨ **µC** sono gestite dalle istruzioni:

Val=Serial.available(); fornisce Il numero di byte presenti nel buffer della porta seriale in attesa di essere letti. Se il buffer è vuoto val=0;

Val=Serial.read(); legge il primo bute disponibile nel buffer e lo assegna a val