PROGETTO LAUREE SCIENTIFICHE
FISICA

a.s. 2015-2016

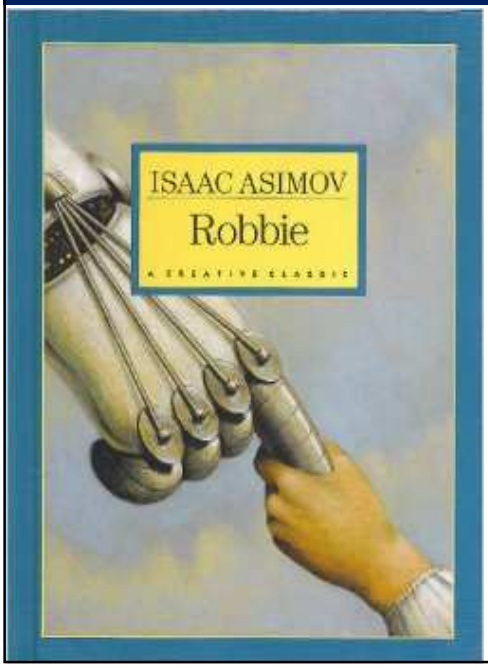
Liceo Scientifico “G. Galilei” – Macerata

PROGETTO ROBBIE

Prof. Angelo Angeletti

14 dicembre 2015

Il progetto ROBBIE



Robbie è un racconto di fantascienza scritto da Asimov nel 1940; la storia è ambientata nel 1998, un'epoca in cui la gente sta diventando sempre più diffidente nei confronti dei robot. Asimov intendeva reagire a tutte le precedenti storie sui "robot come minaccia", molto diffuse all'inizio del XX secolo, per conferire invece agli automi la figura di macchine utili e versatili che possono aiutare l'umanità durante il suo percorso nella storia.
Secondo Asimov le macchine non costituiscono un pericolo se sono ben progettate e correttamente utilizzate.
Per i suoi robot, inventa la ***TRE LEGGI DELLA ROBOTICA***:

1 – Un robot non può recar danno a un essere umano né può permettere che, a causa del proprio mancato intervento, un essere umano riceva danno.
2 – Un robot deve obbedire agli ordini impartiti dagli esseri umani, purché tali ordini non contravvengano alla Prima Legge.
3 – Un robot deve proteggere la propria esistenza, purché questa autodifesa non contrasti con la Prima o con la Seconda Legge.

Il progetto ROBBIE

Nelle Scuole e nelle Università italiane si sta sempre più diffondendo l'impiego di kit robotici basati su Arduino, come metodologia didattica in Informatica, Elettronica e Robotica. Fatte salve le naturali perplessità di alcuni docenti, occorre osservare che si tratta di uno *strumento didattico estremamente promettente*, che vanta una storia quasi cinquantennale (dal *Logo di Papert* ai *Progetti di Resnick* agli esperimenti al *Mit di Boston*, e presso le *Università Tuft* e la *Carnegie Mellon University*).

La Fisica non può esimersi dal giocare un ruolo fondamentale in questa piccola rivoluzione.

Osserviamo, infatti, che la ricerca in Fisica richiede, in modo sempre più stringente, competenze trasversali in elettronica, informatica, robotica ed un livello di progettualità, che in passato erano richieste solo agli ingegneri. Basti pensare al fatto che, come in tutti i grandi gruppi sperimentali e in tutti i laboratori, fisici, ingegneri elettronici, ingegneri meccanici, esperti di vuoto e informatici lavorino fattivamente fianco a fianco e che non esista oggi un solo grande esperimento in cui gli scienziati coinvolti non debbano avere competenze di questo tipo.

A questo occorre aggiungere che oramai tutti i corsi di Laurea Scientifici devono, volenti o nolenti, fronteggiare il problema del livello di occupazionale dei propri laureati e, nel far questo, hanno necessità di fornire competenze sempre più facilmente spendibili nel mondo del lavoro.

Il progetto ROBBIE

Questo progetto nasce da una nostra collaborazione ormai consolidata con il Dipartimento di Fisica dell'Università di Camerino e con il Liceo Scientifico di Recanati e vuol muoversi in questa direzione.

Per la prima volta, infatti, proponiamo un progetto di Fisica in cui non si presti attenzione solo alla misura di una o più quantità fisiche, come fine unico, ma si cerchi di partire dalle fondamenta dell'esperimento, dal capire cioè come costruire il set-up sperimentale, da quali sensori si possano utilizzare per misurare determinate quantità e come essi si interfaccino al PC, come si scrivano i programmi per l'acquisizione dati e, solo al termine di tutti questi passaggi, si proceda alla misura vera e propria e alla discussione dei risultati ottenuti.

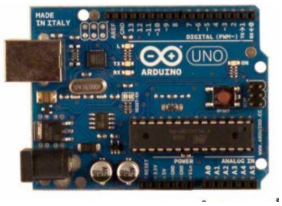
Crediti universitari

Agli studenti che parteciperanno al progetto e che poi si iscriveranno ad uno dei corsi di studio dell'Università di Camerino, saranno assegnati fino a quattro crediti formativi universitari, da utilizzare all'ambito di quelli riservati alle attività formative autonomamente scelte.

Le modalità di presentazione dei risultati e l'attribuzione dei crediti avverranno secondo la procedura prevista agli artt. 8 e 9 del "***Regolamento di Ateneo per la realizzazione di progetti formativi tra UNICAM e gli Istituti di Istruzione Secondaria Superiore ai fini dell'attribuzione di Crediti Formativi Universitari***".

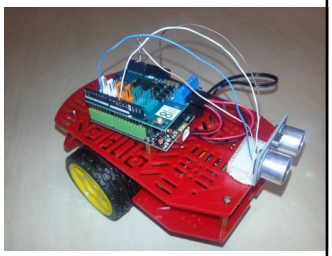
Programma di massima

Fase 1 - Introduzione
1) – Lunedì 14 dicembre 2015 – Presentazione del progetto. Teoria sulla scheda Arduino.
2) – Martedì 22 dicembre 2015 – Applicazioni pratiche sulla scheda Arduino



Fase 2 – Laboratorio
3) – Lunedì 11 gennaio 2016 – Laboratorio
4) – Mercoledì 20 gennaio 2016 – Laboratorio
5) – Giovedì 28 gennaio 2016 – Laboratorio
6) – Lunedì 1 febbraio 2016 – Laboratorio

Fase 3 - Robot
7 - Giovedì 11 febbraio 2016



Fase 4 – Preparazione materiali finali
8) – Mercoledì 17 febbraio 2016 – Preparazione presentazione
9) – Giovedì 25 febbraio 2016 – Preparazione presentazione

Fase finale – Università di Camerino
10) – Data da definire

Il progetto ROBBIE

LICEO SCIENTIFICO "G. GALILEI" - MACERATA

PROGETTO LAUREE SCIENTIFICHE - FISICA

Progetto ROBBIE - a.s. 2015-2016

Ultimo aggiornamento 8 dicembre 2015

Presentazione del progetto (pdf)

Le date degli incontri:

Fase 1 – Attività iniziali
1) – Lunedì 14 dicembre 2015 – Presentazione del progetto. Teoria sulla scheda Arduino.
2) – Martedì 22 dicembre 2015 – Applicazioni pratiche sulla scheda Arduino

Fase 2 – Attività di Laboratorio
3) – Lunedì 11 gennaio 2016
4) – Mercoledì 20 gennaio 2016
5) – Giovedì 28 gennaio 2016
6) – Lunedì 1 febbraio 2016

Fase 3 - Robot
7) - Giovedì 11 febbraio 2016 dalle 14.30 alle 17.30

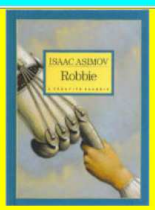
Fase 4 – Preparazione presentazione
8) – Mercoledì 17 febbraio 2016
9) – Giovedì 25 febbraio 2016

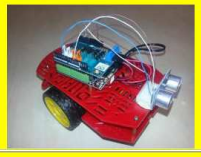
FASE FINALE - Presentazione dei lavori presso l'Università di Camerino
DATA DA DEFINIRE

Ad eccezione dell'ultimo, e se diversamente indicato, gli incontri si terranno nella sede centrale del Liceo Scientifico, in Via Manzoni, dalle ore 14.30 e avranno la durata di due ore circa.

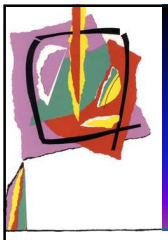
Materiali
Video della presentazione di Recanati
Arduino - Reference (inglese)
Elementi base del linguaggio di programmazione di Arduino (italiano)

Link
www.arduino.cc





HOMEPAGE



PROGETTO LAUREE SCIENTIFICHE
FISICA
a.s. 2015-2016

PROGETTO ROBBIE

UNICAM
Università di Camerino
1336

Liceo Scientifico "G. Galilei" – Macerata

Il laboratorio di Fisica con Arduino

Introduzione ad Arduino e alla sua programmazione

Prof. Angelo Angeletti

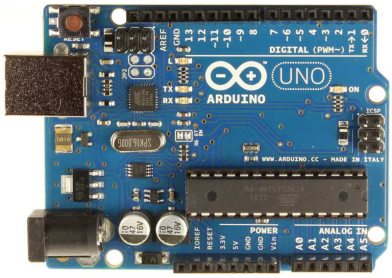
14 dicembre 2015

Premessa

Per queste slide devo ringraziare FABIO CAPODAGLIO, uno studente di Fisica della laurea magistrale di UNICAM, che insieme al dott. Alessandro Saltarelli ha fatto anche la maggior parte del lavoro di preparazione di questo progetto.

Che cosa è Arduino

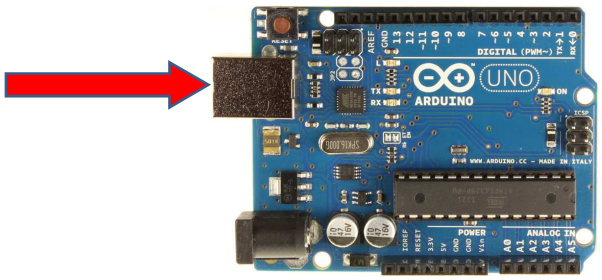
Arduino è una scheda elettronica di piccole dimensioni (comandata da un microprocessore) che presenta una serie di ingressi e uscite (pin) analogici o digitali che permettono di inviare e ricevere segnali.



La scheda nasce nel 2005 grazie al lavoro fatto di Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, e David Mellis.
Esistono diverse schede Arduino ma quella da noi usata è l'Arduino UNO REV 3 il cui processore è un ATmega328P.
Tale scheda ha una memoria di 32Kb.

Alimentazione della scheda

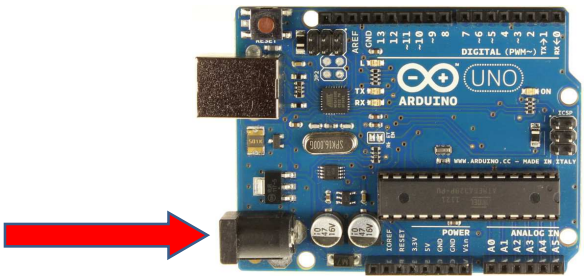
METODO 1: Collegando la scheda alla porta USB del computer



La scheda è alimentata con una tensione di 5 V e una corrente di 500 mA (valori ideali di funzionamento del processore).

Alimentazione della scheda

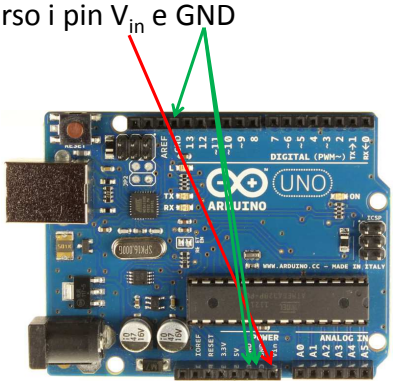
METODO 2: Collegando il connettore di alimentazione



Al connettore va collegata una tensione compresa tra i 7 V e i 12 V. Un regolatore di tensione montato sulla scheda porta tale tensione ai valori ideali; in questo caso la corrente può arrivare a 800 mA. Sul connettore è presente una protezione per eventuali inversioni della polarità.

Alimentazione della scheda

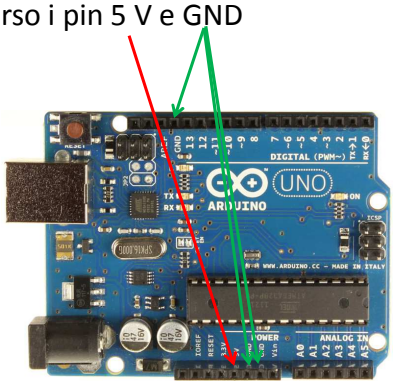
METODO 3: Attraverso i pin V_{in} e GND



In questo caso dobbiamo collegare una tensione compresa tra 6.5 V e 12 V (il polo positivo va collegato sul V_{in} e il polo negativo sul GND). Tale tensione passa attraverso il regolatore di tensione ma in questo caso non c'è protezione quindi bisogna fare attenzione a non invertire le polarità per non bruciare la scheda!!!

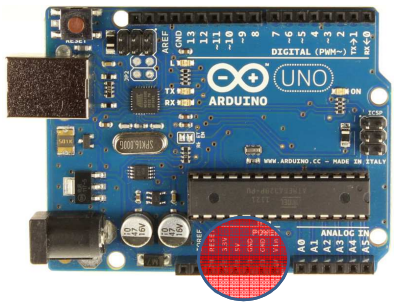
Alimentazione della scheda

METODO 4: Attraverso i pin 5 V e GND



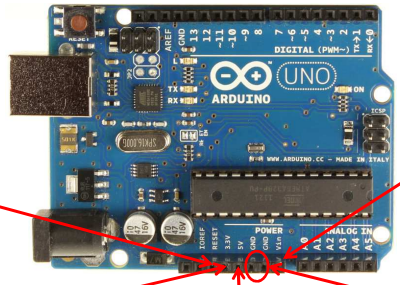
In questo caso dobbiamo collegare una tensione compresa tra 4.5 V e 5.5 V (il polo positivo va collegato sul 5 V e il polo negativo sul GND).
In questo caso non si ha né il regolatore di tensione né la protezione quindi se si superano i 5.5 V oppure si invertono le polarità si rompe la scheda.

Power pins



Power pins

RESET: è un pin che se collegato a un pin digitale di Arduino può riavviare la scheda (la scheda viene riavviata quando il pin digitale, che si trova normalmente allo stato HIGH, viene portato al valore LOW)



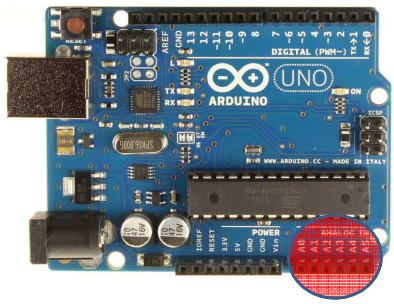
pin Vin: si può usare per alimentare la scheda

pin 3.3 V: è un'uscita che fornice una tensione costante di 3.3 V e una corrente massima di 150 mA

pin 5 V: è un'uscita che fornisce un potenziale costante di 5 V (in teoria non ci sono limiti di corrente)

pin GND: sono due pin di massa della scheda

Pin analogici



Pin analogici

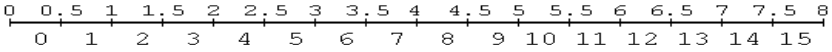
Arduino UNO R3 presenta 6 pin analogici (tali pin sono di input). Essi sono 6 convertitori analogico-digitale (ADC) con risoluzione di 10 bit. Ogni convertitore analogico digitale ha quindi un numero di canali pari a: $2^{10} = 1024$.

Gli ADC ci permettono di leggere un voltaggio nel range 0 – 5 V.

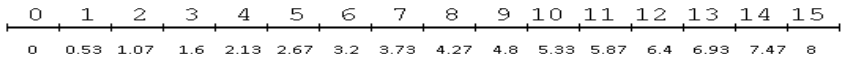
Tuttavia il comando di Arduino *analogRead* che permette di fare questa operazione ci dà il risultato della misura in canali (quindi un valore compreso fra 0 e 1023).
Per convertire il valore misurato in volt possiamo usare la seguente relazione:

$$\text{tensione(V)} = \text{tensione(Ch)} \times \frac{5}{1023}$$

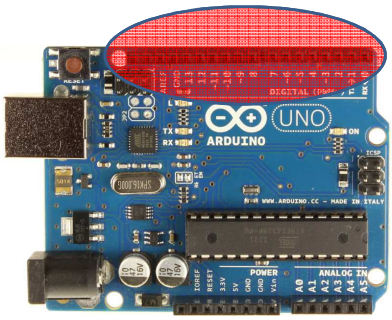
Convertitore analogico-digitale

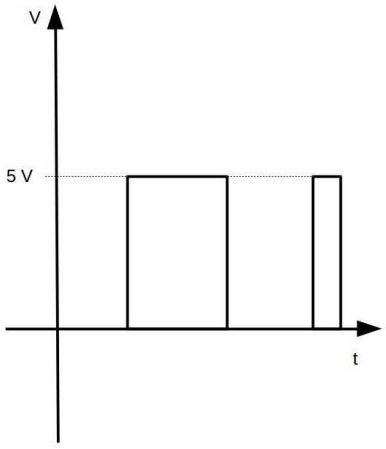


Supponiamo di avere un ADC con risoluzione di 4 bit.
Significa che possiamo avere 16 canali ($C = 2^4 = 16$) numerati da 0000 (= 0) fino a 1111 (=15), ossia C assume tutti i valori naturali da 0 a 15.
Supponiamo ora di dover trasformare una grandezza X che può assumere valori sa 0 fino a 8.
$C = 0 \Rightarrow$ X compreso tra 0 e 0.5, C = 1 significa X compreso tra 0.5 e 1 e così via fino al C = 15 che significa X compreso tra 7.5 e 8.
Ad esempio se la grandezza X = 3,25 avremo C = 6.
Se invece abbiamo C = 6 allora X = 6*(8/15) = 3,2.



Pin digitali





Pin digitali

I pin digitali sulla scheda Arduino UNO REV3 sono 13.

Tali pin possono essere utilizzati sia in input che in output (va specificato attraverso il comando pinMode).

Quando sono utilizzati come output essi possono essere immaginati come degli "interruttori" e possono avere due stati:
LOW: il pin digitale è "spento" e la tensione in uscita è 0 V
HIGH: il pin digitale è "acceso" e la tensione in uscita è 5 V.

La corrente massima in uscita è di 40 mA.

Lo stato dei pin digitali viene cambiato con il comando digitalWrite.

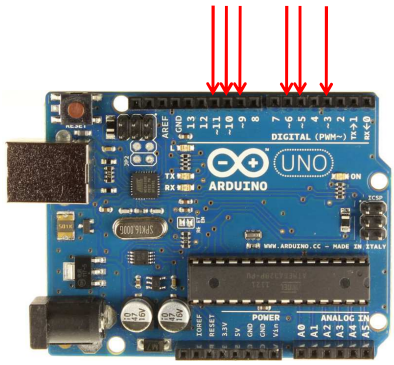
Pin digitali

Quando i pin digitali vengono impostati come input il loro stato può essere:

LOW: se riceve in ingresso un segnale di circa 0 V
HIGH: se riceve in ingresso un segnale di circa 5 V

Come vedremo il sensore di posizione a ultrasuoni hr04, che useremo negli esperimenti, utilizzerà due pin digitali, uno in input e uno in output.

Pin digitali speciali ⇒ Pulse Width Modulation (PWM)



I pin digitali PWM sono quelli contrassegnati dal simbolo tilde [~]
Nell'Arduino UNO REV3 sono i pin: 3, 5, 6, 9, 10, 11)

Pin digitali speciali ⇒ Pulse Width Modulation (PWM)

I pin PWM sono dei pin digitali che possono fornire in output una tensione variabile da 0 V a 5 V.

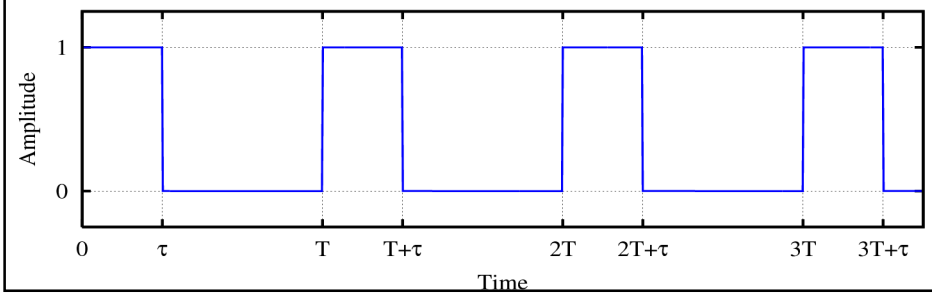
Analogamente al caso della lettura della tensione con le porte analogiche (per la lettura si usava **analogRead**) anche in questo caso la tensione va scritta in canali e non in volt. Per farlo si utilizza il comando **analogWrite**.

In questo caso i canali sono 256 quindi la relazione che lega la tensione in canali a quella in volt è la seguente:

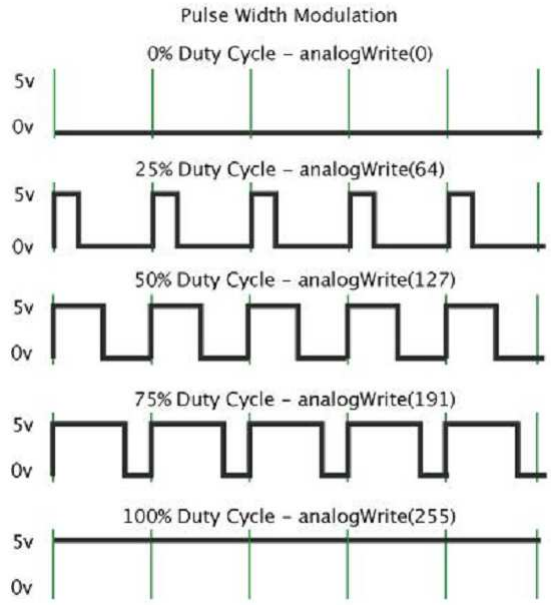
$$\text{tensione(Ch)} = \text{tensione(V)} \times \frac{255}{5}$$

Pin digitali speciali ⇒ Pulse Width Modulation (PWM)

La tensione viene modulata attraverso il metodo della Pulse Width Modulation
(modulazione di larghezza di impulso).
Esso consiste in una serie di impulsi ad una determinata frequenza (e quindi ad un determinato periodo T) che restano accesi per un tempo $\tau \leq T$.
In base al valore di τ abbiamo quindi un diverso valore della tensione in uscita.

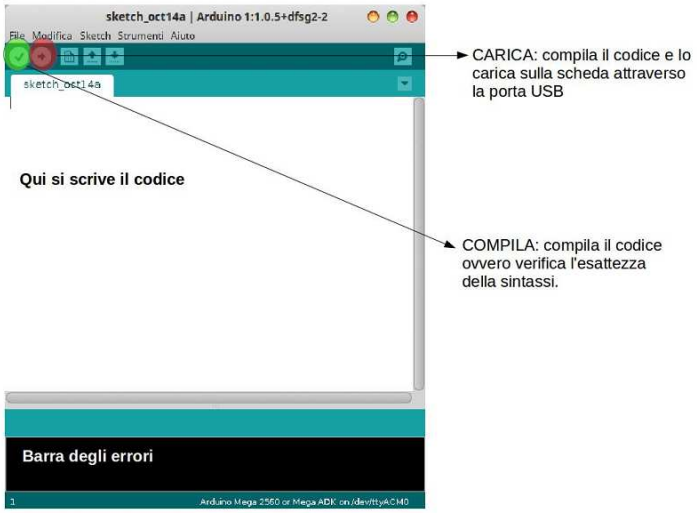


Pin digitali speciali ⇒ Pulse Width Modulation (PWM)



Come programmare Arduino

La scheda Arduino viene programmata in un linguaggio molto simile al C utilizzando l'Arduino ide.



Come programmare Arduino

Una volta aperto il programma si può iniziare a scrivere il codice nell'apposito spazio.
Quando il codice (o parte di esso) è stato scritto per verificarne la correttezza bisogna innanzitutto salvare il file.
Fatto questo si può cliccare sul pulsante compila.
Se la compilazione non va a buon fine significa che c'è qualche errore nella sintassi e tali errori sono riportati (in maniera più o meno chiara) nella barra degli errori.
Una volta che la compilazione va a buon fine e che l'intero codice è stato scritto si procede con il caricamento del programma (in gergo ***sketch***) sulla scheda cliccando sul pulsante carica.

Come programmare Arduino

Che cosa fare se la compilazione va a buon fine ma il caricamento sulla scheda fallisce?

RICORDA: una volta aperto il programma è opportuno selezionare subito il tipo di scheda utilizzata dal menù *strumenti* → *scheda* → *Arduino uno* e la porta seriale dal menù *strumenti* → *porta*.

ATTENZIONE: a volte nel menù delle porte seriali sono presenti più porte quindi l'unico modo che si ha per vedere qual è quella giusta è selezionarne una e provare a caricare il programma.

Se il caricamento va a buon fine la porta scelta è quella giusta.
Se il caricamento non va a buon fine si prosegue a tentativi!!!

Come fare in modo che Arduino inizi una misura quando vogliamo noi

Una volta che il programma è stato caricato sulla scheda Arduino, inizia immediatamente ad eseguirlo!
Per far si che la misura, o più in generale lo sketch, caricato sulla scheda venga eseguito a partire da un determinato istante di tempo abbiamo almeno tre modi:

- Spegnere e riaccendere Arduino togliendo e rimettendo l'alimentazione alla scheda.
- Premere il tasto reset.
- Utilizzare il software freeware (solo per Windows) Gobetwino

Noi utilizzeremo spesso l'ultimo modo in quanto ci permette anche di stampare i dati su file.

Come usare Gobetwino

Gobetwino è un software che non ha bisogno di istallazione e che si trova all'interno di una cartella chiamata Gobet (che si trova sul desktop).
Tale programma una volta lanciato esegue il reset della scheda e fa ripartire lo sketch dall'inizio.

Questo software permette inoltre di definire un comando per la stampa su file; nel nostro caso il comando LOGTEST.
Tale comando permette di stampare i dati presi con Arduino su un file di testo che abbiamo chiamato prova pls.txt e che si trova all'interno della cartella Gobet.

ATTENZIONE ATTENZIONE

Una volta terminata la misura la finestra di Gobetwino va assolutamente chiusa!!!

Se rimane aperta infatti avremo degli errori sia quando proveremo a caricare un nuovo sketch con l'Arduino ide sia quando lanceremo una nuova misura con Gobetwino stesso!!!

PER EVITARE DI PERDERE TEMPO PER CAPIRE PERCHÉ IL COMPUTER SEMBRA IMPAZZITO (QUANDO INVECE HA RAGIONE LUI) È BENE CHIUDERE LA FINESTRA DI GOBETWINO APPENA NON SERVE PIÙ, OSSIA APPENA FINITA LA MISURA

Metodo alternativo per lanciare una misura e stampare i dati!

Un metodo alternativo all'utilizzo di Gobetwino è quello di utilizzare il ***monitor seriale*** di Arduino (menù *strumenti* → *monitor seriale*).
Infatti una volta aperto il monitor seriale Arduino inizia da capo l'esecuzione dello sketch e inoltre si ha anche la possibilità di stampare i dati su tale finestra.

Tuttavia si hanno due grossi svantaggi:

1) Non si ha la possibilità di esportare i dati su un file. L'unico metodo è il copia e incolla che non è ideale quando si devono selezionare grandi quantità di dati.
2) Se i dati sono davvero molti non si ha la possibilità di vederli tutti.

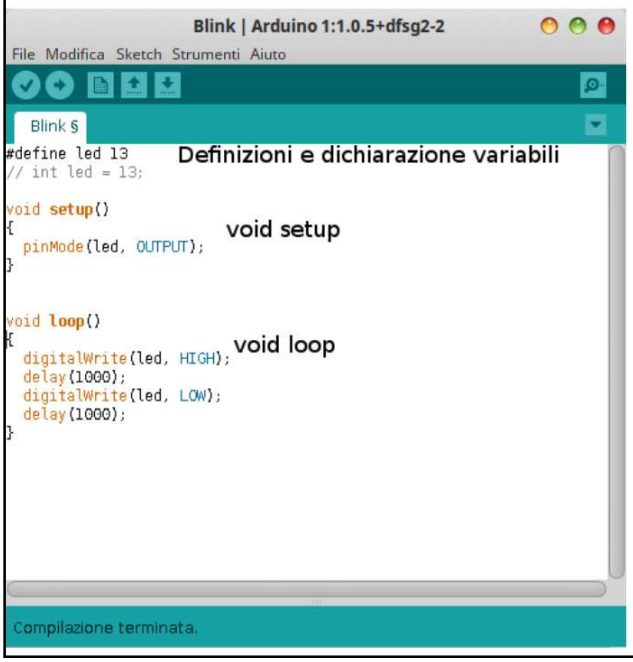
Per questo motivo utilizzeremo spesso Gobetwino.

Struttura di uno *sketch* per Arduino

Uno sketch per Arduino si divide in tre parti:

Definizione e dichiarazione variabili
In questa parte del programma si possono assegnare dei nomi ai pin analogici o digitali usati e si possono inoltre dichiarare le variabili (in realtà le variabili possono essere dichiarate in qualsiasi parte del programma purché sia prima del loro utilizzo).

Struttura di uno *sketch* per Arduino



Uno sketch per Arduino si divide in tre parti:

void setup

È un ciclo che Arduino esegue una sola volta e in cui si definisce se i pin digitali usati sono di input o di output e il loro stato iniziale (LOW o HIGH).
In questo ciclo di definiscono inoltre gli ADC utilizzati e si inizializza anche la comunicazione con la porta seriale.

Struttura di uno *sketch* per Arduino

Uno sketch per Arduino si divide in tre parti:

void loop

È un ciclo che viene eseguito dalla scheda infinite volte e che contiene tutte le istruzioni che la scheda deve eseguire.

Importazione delle librerie

A volte, soprattutto quando si utilizzano dei sensori e degli shield (cioè dei moduli che si applicano sulla scheda Arduino) potrebbe essere necessario importare delle librerie esterne.
Il modo più semplice per fare ciò è avere il file .zip di tale libreria ed importarla dal menù *sketch → aggiungi libreria*.
Una volta aperto il menù bisogna selezionare la libreria che si vuole importare nella cartella dove è contenuta.
Una volta importata la libreria (tale operazione va fatta una sola volta in quanto una volta fatta la libreria rimane nella cartella libraries di Arduino) bisogna richiamarla all'inizio dello sketch utilizzando il seguente comando:
#include <nomelibreria.h> (ad esempio **#include <OneWire.h>**

Note generali sulla programmazione

Tutti i comandi devono terminare con ;

Le parentesi graffe delimitano un blocco di comandi {}
Ad ogni parentesi aperta deve sempre corrispondere una parentesi chiusa

Nella stesura di programmi è buona norma inserire commenti.

Tutto quello compreso tra i simboli /* e */ viene considerato come commento.

Tutto quello che va dal simbolo // fino alla fine della riga è un commento.

Per i comandi vedi *Arduino – reference* (in inglese) o *Elementi base del linguaggio di programmazione di Arduino*.
Collegamenti in angeloangeletti.it

Definizione e dichiarazione variabili

Definizioni: si ha la possibilità di assegnare un nome ai pin di Arduino utilizzati utilizzando il seguente comando:
#define *nomepin numeropin*

Il *nomepin* è arbitrario e può contenere sia lettere che numeri mentre il *numeropin* è quello riportato sulla relativa porta di Arduino.
Ad esempio si può avere: **#define chargePin 3**

Nelle definizioni non è necessario distinguere tra porte analogiche e digitali in quanto tale distinzione sarà automatica nel momento in cui i nomi dati alle porte verranno inseriti nei vari comandi (infatti i comandi per le porte analogiche sono diversi da quelli per quelle digitali).

Definizione e dichiarazione variabili

Dichiarazione delle variabili: le variabili usate nel codice possono essere di diverso tipo e vanno dichiarate, per esempio:

int → intero a 16 bit, va da –32768 a + 32767;
unsigned int → intero a 16 bit, va da 0 a + 65535;

long → intero a 32 bit, va da –2 147 483 684 a + 2 147 483 647;
unsigned long → intero a 32 bit, va da 0 a + 4 294 967 295;

float → numero in virgola mobile a 32 bit, va da $-3.4028235 \cdot 10^{+38}$ e $3.4028235 \cdot 10^{+38}$;
double → numero in virgola mobile a 64 bit (solo in alcune schede).

Esistono anche altri tipi di variabili come ad esempio interi senza segno, costanti, ecc.

Definizione e dichiarazione variabili

Le variabili oltre ad essere dichiarate possono anche essere inizializzate ad un valore iniziale. Ad esempio:
int pippo;
int pippo1 = 0;
double misura;

In pratica la struttura è la seguente:

tipo di variabile nome variabile = valore iniziale ;

ATTENZIONE al punto e virgola: come nel linguaggio C il punto e virgola va messo alla fine di ogni istruzione (sono escluse le importazioni delle librerie e le definizioni).

Definizione di vettori

Oltre alle variabili scalari si possono definire anche dei vettori (o meglio delle liste di numeri → ***array***).
I numeri contenuti in tali liste sono tutti dello stesso tipo (o tutti interi o tutti in virgola mobile).

Per definire un vettore si usa la seguente sintassi:

tipo nomevariabile[numero elementi vettore]
int vettore[100];

Possiamo allocare il vettore (specificarne in numero di elementi) anche dinamicamente ovvero:
int Nmis = 100;
double vettore[Nmis];
Tuttavia in questo caso il vettore va definito dentro il void loop.

Che cosa mettere nel *void setup*

I comandi da mettere necessariamente nel ***void setup*** sono:

Serial.begin(9600)
se si ha intenzione di stampare dei dati su file o sul monitor seriale. Tale comando inizializza la porta seriale.

pinMode(nomeporta / numeroporta , INPUT / OUTPUT)
se si usano delle porte digitali. Tale comando imposta le porte digitali usate o in INPUT o in OUTPUT.

bitClear(ADCSRA,ADPS0)
se si usano delle porte analogiche per la lettura di una tensione. Tale comando esegue il reset dell'ADC e ne stabilisce la frequenza di campionamento.

Che cosa mettere nel *void setup*

Nel caso in cui si utilizzino delle porte digitali è opportuno specificare anche lo stato iniziale della porta (ovvero lo stato in cui si troverà la porta una volta usciti dal ***void loop***); il comando è:

digitalWrite(nomeporta / numeroporta, HIGH / LOW)
Tale comando serve a scrivere HIGH o LOW sulla porta digitale specificata.

Che cosa mettere nel *void loop*

Il ***void loop*** è il ciclo in cui diciamo ad Arduino cosa fare.

Le istruzioni che possiamo inserire sono tutte quelle del ***linguaggio C*** (operazioni matematiche, cicli, ecc) insieme ad alcuni comandi specifici di Arduino che servono a leggere e scrivere su porte digitali o analogiche.

RICORDA: vanno messi nel ***void loop*** anche le dichiarazioni dei vettori allocati dinamicamente ovvero quei vettori il cui numero di elementi non è specificato direttamente da un intero ma da una variabile intera.

Nelle prossime slides andremo a vedere sia la sintassi dei comandi di Arduino per il ***void loop*** sia quella dei cicli.

Comandi di Arduino per il *void loop*: porte digitali

digitalWrite(nomeporta / numeroporta, HIGH / LOW)
Comando per scrivere su una porta digitale

digitalRead(nomeporta / numeroporta)
Comando per leggere lo stato di una porta digitale

pulseIn(nomeporta / numeroporta, HIGH/LOW)
Tale comando ci da il tempo (in s) durante il quale una porta digitale è rimasta rispettivamente nello stato HIGH o LOW (es: pulseIn(4,HIGH) ci da in s il tempo che la porta digitale 4 resta nello stato HIGH). Questa funzione è molto utile con il sensore a ultrasuoni che useremo negli esperimenti.

Comandi di Arduino per il *void loop*: porte digitali

analogWrite(nomeporta / numeroporta, 0-255)
Comando per scrivere su una porta digitale PMW
Il valore 0 corrisponde a 0 V mentre il valore 255 corrisponde a 5 V

analogRead(nomeporta / numeroporta)
Comando per leggere il valore di tensione in ingresso a una porta analogica .
Il valore in uscita varia tra 0 (0 V) e 1023 (5V)

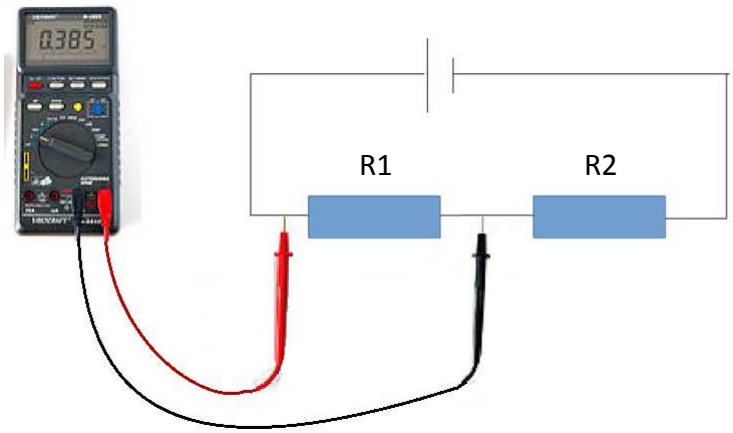
Come fare una misura di tensione con Arduino

RICORDA
Quando si vuole fare una misura di tensione ai capi di un dispositivo (ad esempio una resistenza) si deve immaginare di usare Arduino come un multimetro; quindi i cavi da collegare sono due (il puntale rosso e quello nero del multimetro).

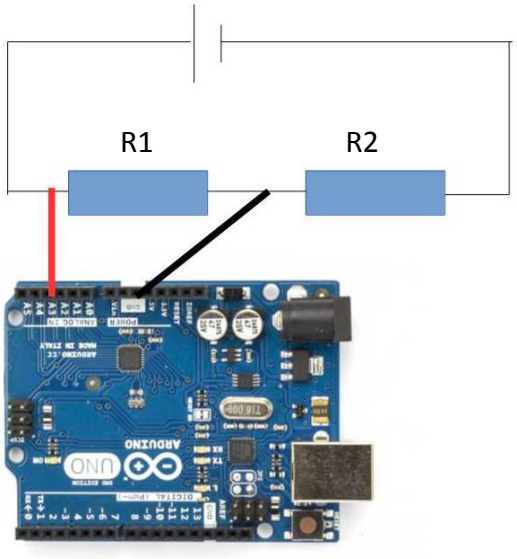
Il puntale rosso è la nostra porta analogica mentre il puntale nero è una delle porte di ground (GND) di Arduino.

Attenzione a non invertire perché Arduino legge tensioni tra 0 e 5 V quindi non è in grado di leggere tensioni negative e se invertiamo rosso e nero (più e meno) in realtà abbiamo una tensione negativa ma in pratica Arduino legge 0 V.

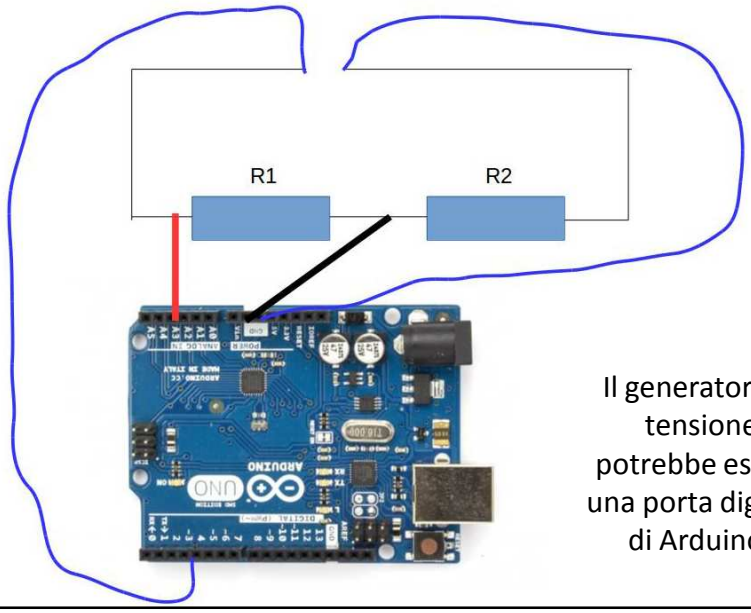
Come fare una misura di tensione con Arduino



Come fare una misura di tensione con Arduino

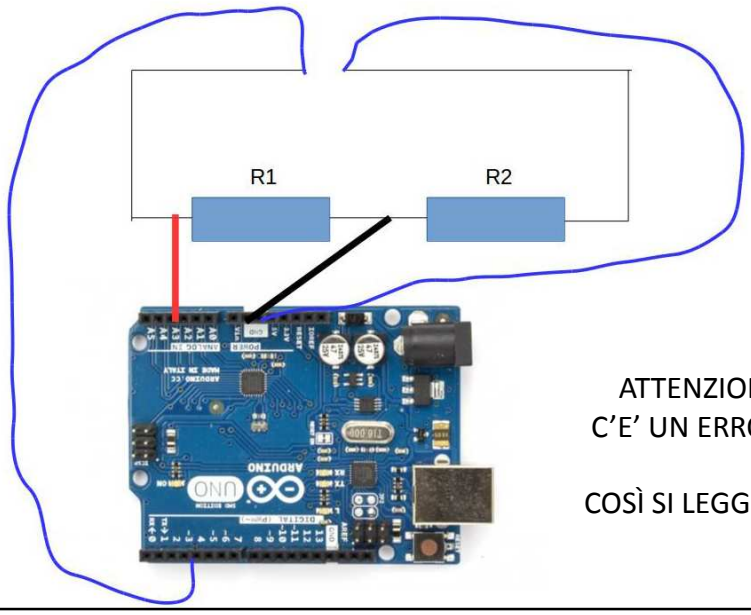


Come fare una misura di tensione con Arduino



Il generatore di tensione potrebbe essere una porta digitale di Arduino!

Come fare una misura di tensione con Arduino

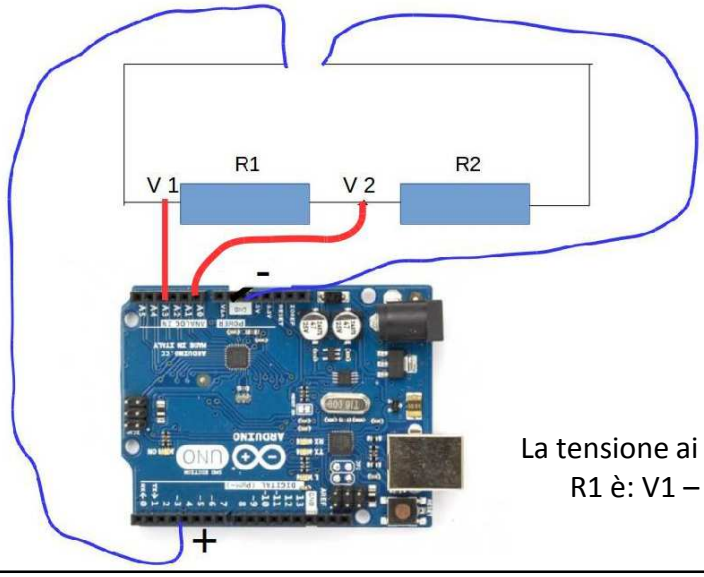


ATTENZIONE
C'E' UN ERRORE!

COSÌ SI LEGGE 5V!

Come fare una misura di tensione con Arduino

ECCO IL METODO CORRETTO!



La tensione ai capi di R1 è: V1 – V2

Altre funzioni nel *void loop*

Un altro tipo di funzioni che incontreremo spesso sono quelle che permettono di inserire dei ritardi di tempo (o meglio dei tempi di attesa) tra un'istruzione e l'altra e quelle che permettono di contare il tempo trascorso da quando è stato lanciato lo sketch.

delay(variabile/numero)
aspetta un numero di millisecondi pari al valore della variabile o del numero messo fra parentesi tonde (deve essere un intero)

delayMicroseconds(variabile/numero)
aspetta un numero di microsecondi pari al valore della variabile o del numero messo fra parentesi tonde (deve essere un intero)

Altre funzioni nel *void loop*

Un altro tipo di funzioni che incontreremo spesso sono quelle che permettono di inserire dei ritardi di tempo (o meglio dei tempi di attesa) tra un'istruzione e l'altra e quelle che permettono di contare il tempo trascorso da quando è stato lanciato lo sketch.

millis()
restituisce in millisecondi il tempo trascorso da quando è stato lanciato lo scketch

micros()
restituisce in microsecondi il tempo trascorso da quando è stato lanciato lo scketch

Strutture principali

I principali costrutti che utilizzeremo nei nostri sketch sono i seguenti:

Ciclo **for**: è un ciclo che esegue tutte le istruzioni contenute al suo interno un numero fissato di volte.

Ciclo **while**: è un ciclo che esegue tutte le istruzioni che sono al suo interno fino a quando è vera una determinata condizione logica.

Struttura **if**: è una struttura che esegue tutte le istruzioni contenute al suo interno una sola volta e solo se si verifica una data condizione logica.

RICORDA
Queste strutture vanno inserite sempre all'interno del void loop

Come scrivere un ciclo *for*

Abbiamo bisogno di:

- Variabile intera → indice del ciclo (nell'esempio sotto i)
- Variabile intera o numero che indica il numero di iterazioni del ciclo da compiere (nell'esempio sotto 100)
- step del ciclo (nell'esempio sotto lo step è pari a uno in quanto abbiamo inserito i++)

```
int i;
for(i=0;i<100;i++)
{
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13,LOW);
delay(1000);
}
```

Tale ciclo fa lampeggiare 100 volte il led di Arduino collegato alla porta 13 a intervalli di 1s

Come scrivere un ciclo *for*

Se vogliamo incrementare lo step del ciclo di un numero diverso da 1 al posto di scrivere i++ possiamo scrivere: i = i + 2

Esempio:

```
int N=100;
int i;
for(i=0;i<N;i=i+2)
{
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13,LOW);
delay(1000);
}
```

Proviamo i due esempi precedenti

Per provare i due esempi precedenti dobbiamo scrivere il void setup dove dichiariamo il pin 13 come output e lo poniamo allo stato di LOW e poi inseriamo gli sketch precedenti all'interno del void loop.

Se facciamo questo vediamo che il led non lampeggia un numero fissato di volte (N) ma lampeggia all'infinito.

Questo perché non appena il ciclo for termina, siccome siamo nel void loop, si ricomincia da capo.

RICORDA
Tutte le istruzioni vanno messe nel void loop.
Quindi la soluzione non è togliere il void loop oppure mettere il ciclo for al di fuori di esso!!

Come scrivere un ciclo *while*

```
while(condizione logica)
{
istruzioni
}
```

Come scrivere un ciclo *while*

Esempio

```
int N=100;
int i=0;
while(i<N)
{
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13,LOW);
delay(1000);
i++;
}
```

Tale ciclo fa lampeggiare 100 volte il led di Arduino collegato alla porta 13 a intervalli di 1s

Come fermare il *void loop*

Per fermare Arduino possiamo usare un “trucco” molto semplice ... possiamo metterlo a non fare niente!

Possiamo quindi inserire (alla fine di tutte le istruzioni del void loop) un ciclo while da cui non esce mai e non mettere nessuna istruzione in questo ciclo.

In pratica l’abbiamo messo a dormire!

```
int b=1;
while(b<2){}
```

Struttura *if*

if(condizione logica)
{
istruzioni
}
else if(condizione logica)
{
istruzioni
}
else
{
istruzioni
}

Se la condizione logica dell'if è soddisfatta vengono eseguite solo le istruzioni relative all'if. Se viene soddisfatta la condizione logica dell'else if vengono eseguite solo le relative istruzioni. Se nessuna delle due condizioni viene soddisfatta si eseguono le istruzioni nell'else.

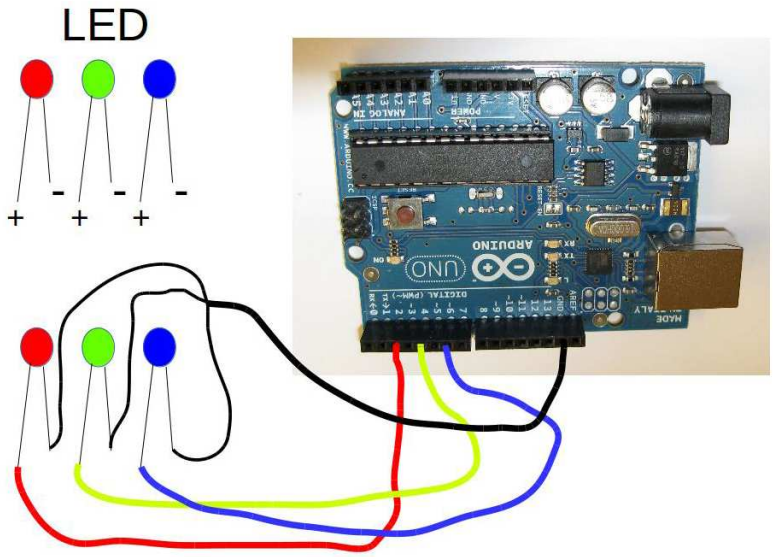
Esempio di struttura *if*

Supponiamo di avere tre led (rosso, verde e blu) e di collegarli rispettivamente alle porte digitali 2, 4, 6.

Un led ha due contatti ... quello positivo (quello più lungo) va collegato alla porta digitale mentre quello negativo (quello più corto) va collegato a massa (quindi alla porta GND di Arduino).

Supponiamo di avere inoltre un sensore i distanza (ad esempio il sensore a ultrasuoni che vedremo in seguito) che legge la distanza e la registra nella variabile r.

Esempio di struttura *if*



Esempio di struttura *if*

Vogliamo accendere:

led rosso se r < 30 cm

led verde se r > 50 cm

led blu se 30 cm < r < 50 cm

Esempio di struttura *if*

Questo è un tipico caso in cui è utile la struttura if.

```
void loop()
{
digitalWrite(2,LOW);
digitalWrite(4,LOW);
digitalWrite(6,LOW);
r = vedremo in seguito come misurarla
if(r<30) {
        digitalWrite(2, HIGH);
}
else if(r>50)  {
         digitalWrite(4, HIGH);
}
else  {
        digitalWrite(6, HIGH);
}
}
```

Operatori di confronto

Spesso nelle strutture *if* e *while* si ha la necessità di fare dei confronti tra variabili e costanti. Per fare ciò si utilizzano gli operatori di confronto.

==	UGUALE
!=	DIVERSO
>	MAGGIORE
>=	MAGGIORE UGUALE
<	MINORE
<=	MINORE UGUALE

Operatori logici

Nelle strutture if a volte si mettono non una ma due o più condizioni e ad esempio si richiede di eseguire l'if se una delle due, tutte e due o nessuna delle due è soddisfatta. In questo caso abbiamo bisogno degli operatori logici.

&& AND
|| OR
! ! NOT

Esempi con operatori di confronto e logici

Supponiamo di avere due variabili x, y che sono lette in qualche modo con la scheda (ad esempio con qualche sensore o con gli ADC).

```
if(x <= 1 && y == 3)
{
istruzioni (eseguite solo se x minore uguale di 1 e y uguale 3)
}
if(x > 1 || y != 3)
{
istruzioni(eseguite solo se x maggiore di 1 o y diverso 3)
}
```

Esempi con operatori di confronto e logici

Supponiamo di avere due variabili x, y che sono lette in qualche modo con la scheda (ad esempio con qualche sensore o con gli ADC).

```
if(x <= 1 && y == 3)
{
istruzioni (eseguite solo se x minore uguale di 1 e y uguale 3)
}
if(x > 1 || y != 3)
{
istruzioni(eseguite solo se x maggiore di 1 o y diverso 3)
}
if(! x <= 1)
{
istruzioni(eseguite solo se x non è minore uguale di 1)
}
```

Esempi con operatori di confronto e logici

```
if(! x <= 1)
{
istruzioni(eseguite solo se x non è minore uguale di 1)
}
if(x > 1 || y != 3 || z < 5)
{
istruzioni(eseguite solo se o x è maggiore di 1 o y è diverso
da 3 o z è minore di 5)
}
```

Come stampare i dati

Supponiamo di avere due variabili x e y su cui registriamo dei dati (supponiamo i valori di tensione letti nelle porte analogiche 1 e 2). Se vogliamo stampare queste due variabili su monitor seriale possiamo usare il seguente codice (da mettere sempre all'interno del void loop).

```
void loop()
{
x = analogRead(A1);
y = analogRead(A2);
Serial.print("Tensione 1: ");
Serial.print(x);
Serial.print(" ");
Serial.print("Tensione 2: ");
Serial.println(y);
}
```

Come stampare i dati

L'output dello sketch precedente sul monitor seriale è il seguente.

Tensione 1: 500 Tensione 2: 400
Tensione 1: 100 Tensione 2: 700
Tensione 1: 400 Tensione 2: 10
.....

I valori 500, 400, ecc. sono ovviamente solo indicativi.

RICORDA
i dati che compaiono sul monitor seriale non vengono memorizzati, se si ha bisogno di conservarli è necessario, con copia e incolla, incollarli in un file di testo.

Come stampare i dati usando *Gobetwino*

Riscriviamo il codice precedente con una sintassi che permette la stampa dei dati tramite ***Gobetwino*** (lanciato con il file go.bat come detto precedentemente) su un file di testo.

```
void loop()
{
x = analogRead(1);
y = analogRead(2);
Serial.print("#S|LOGTEST|[");
Serial.print("Tensione 1: ");
Serial.print(x);
Serial.print(" ");
Serial.print("Tensione 2: ");
Serial.print(y);
Serial.println("]#");
}
```

Come stampare i dati usando *Gobetwino*

L'output dello sketch precedente sul file di testo **prova pls.txt** (contenuto nella cartella **Gobet**) sarà lo stesso di prima ovvero:

Tensione 1: 500 Tensione 2: 400
Tensione 1: 100 Tensione 2: 700
Tensione 1: 400 Tensione 2: 10
.....

I valori 500, 400, ecc. sono ovviamente solo indicativi.

RICORDA

Ogni volta che vengono stampati i dati su file con Gobetwino il file di salvataggio è sempre **prova pls.txt**, per non perdere i dati salvati in precedenza il vecchio file **prova pls.txt** viene rinominato automaticamente **prova pls-mm-anno_mi.ss.txt** dove mm sta per mese, anno sta per anno mi sta per minuti e ss per secondi, per esempio **prova pls-09-2015_10.18.txt**

Come stampare i vettori con *Gobetwino*

Supponiamo di avere definito due vettori t[100] e v[100] (vettori con 100 componenti indicizzate però da 0 a 99) il primo contiene dei tempi e le velocità in quell'istante.
Supponiamo inoltre di voler stampare le componenti dei vettori su un file di testo contenente due colonne (la prima con le componenti di t e la seconda con le componenti di v) e come separatore vogliamo mettere il punto e virgola.
Supponiamo inoltre di voler mettere un'intestazione in cui si indichi che cosa rappresentano le due colonne.

OUTPUT:
tempi;velocità
t1; v1
t2; v2
t3; v3
t4; v4
.....

Come stampare i vettori con *Gobetwino*

Per fare ciò possiamo utilizzare il seguente codice:

```
void loop() {
int i;
Serial.print("#S|LOGTEST|[");
Serial.print("tempi;velocità");
Serial.println("]#");
for(i=0;i<100;i++)
{
Serial.print("#S|LOGTEST|[");
Serial.print(t[i]);
Serial.print(";"); (se si vuole cambiare il separatore basta sostituire il ;)
Serial.print(v[i]);
Serial.println("]#");
}
}
```

I sensori

I sensori che possono essere collegati alla scheda Arduino sono tantissimi e variano a seconda dello scopo dell'esperimento.
Sono riportate solo le tipologie che utilizzeremo

Sensori di posizione
misurano la distanza tra il sensore e un ostacolo.

Sensori di temperatura
misurano la temperatura di un corpo.

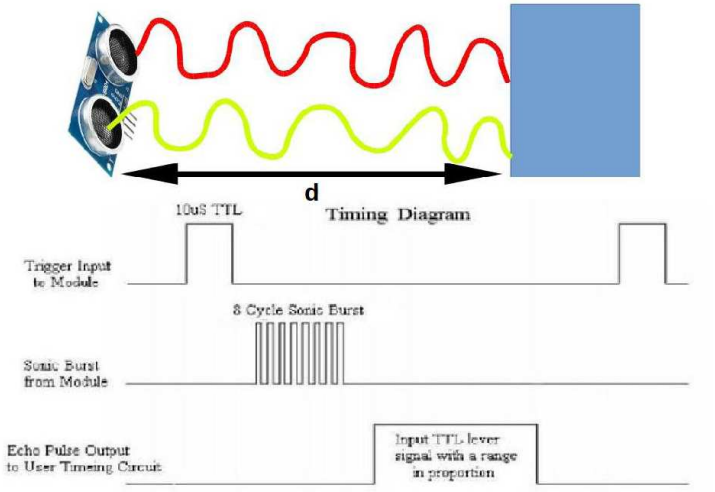
Oltre a questi ne esisto moltissimi altri come: sensori di corrente, di pressione, di umidità, sensori in grado di rilevare la presenza di un essere vivente nei paraggi ecc ecc

Sensore di posizione a ultrasuoni hc-sr04

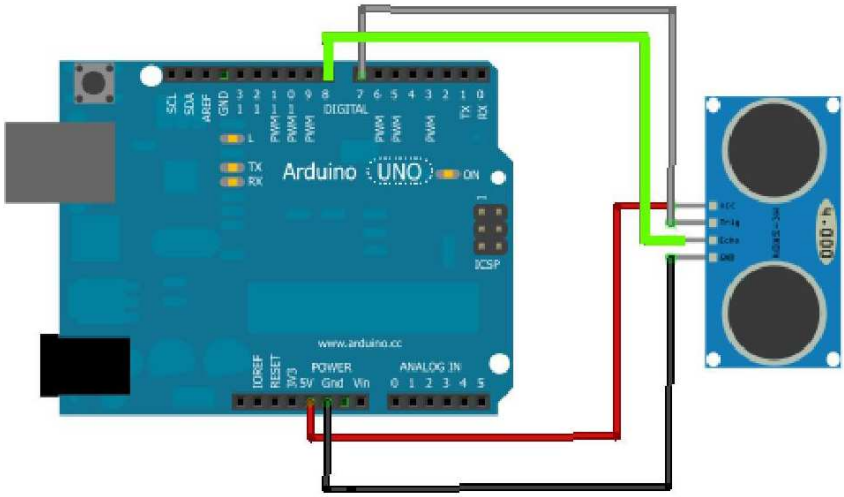
Sensore di posizione a ultrasuoni hc-sr04

Il principio di funzionamento del sensore di posizione a ultrasuoni è basato sulla riflessione del segnale a ultrasuoni, generato dal sensore stesso, sul primo ostacolo che tale segnale incontra lungo il suo cammino.
Si può misurare il tempo impiegato dal segnale per compiere il percorso di andata (sensore-ostacolo) e ritorno (ostacolo sensore) e considerando il fatto che la velocità media degli ultrasuoni in aria a temperatura ambiente è 340 m/s si può calcolare la distanza dell'ostacolo dal sensore

Sensore di posizione a ultrasuoni hc-sr04



hc-sr04 : interfacciamento con Arduino



hc-sr04 : funzionamento

Il funzionamento di questo sensore a ultrasuoni si basa sui seguenti step:

- Si invia sulla porta digitale di trigger un impulso della durata di 10µs (tale impulso è lo start per l'invio del segnale);
- Il sensore appena finito l'impulso invia un treno di 8 impulsi a ultrasuoni a una frequenza di 40 kHz e porta lo stato della porta di echo nel valore HIGH;
- Nel momento in cui l'echo (il segnale riflesso dall'ostacolo) viene rilevato dal sensore la porta di echo viene portata di nuovo allo stato LOW.

hc-sr04 : funzionamento

Con la funzione pulseIn possiamo misurare l'intervallo di tempo t in cui la porta di echo è stata nello stato HIGH (t corrisponde al tempo impiegato dal segnale per colpire l'ostacolo e tornare indietro). Si ha quindi che:

$$d = v_{suono} \cdot \frac{\Delta t}{2} = 340\left[\frac{m}{s}\right]\frac{\Delta t}{2}[s]$$

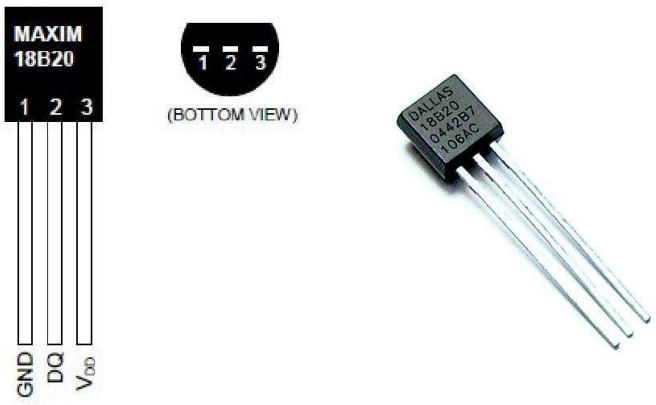
RICORDA: la funzione pulseIn ci dà l'intervallo di tempo t in microsecondi quindi è necessario convertirlo in secondi.

Un esempio di codice che permette la lettura della distanza utilizzando tale sensore è riportato nella cartella sketch all'interno della cartella PLS2016.

hc-sr04 : caratteristiche

- Costo: 4,39 € (Campustore)
- Max range: 4 m (massimo range effettivo: 2.5 m)
- Min range: 2 cm
- Errore: ±1 cm
- intervallo di tempo minimo tra una misura e l'altra: 25ms
- intervallo di tempo minimo consigliato tra una misura e l'altra: 100 ms

Sensore di temperatura digitale: DS18B20



Sensore di temperatura digitale DS18B20 caratteristiche

Come tutti i termometri di questo tipo il principio di funzionamento è la variazione della resistenza del materiale di cui è fatto, al variare della temperatura.

- Range – 55°C +125°C
- Errore: ± 0.5°C
- Costo: 4.37 €

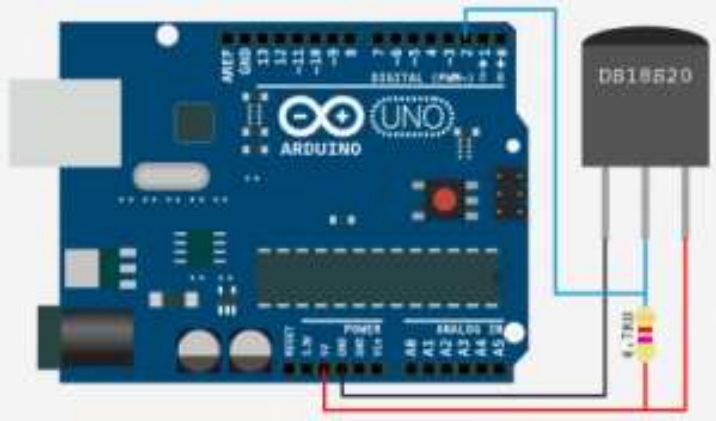
Si tratta di un termometro digitale che può essere usato con le seguenti librerie:

- DallasTemperature.h
- OneWire.h

Tali librerie sono già state caricate.

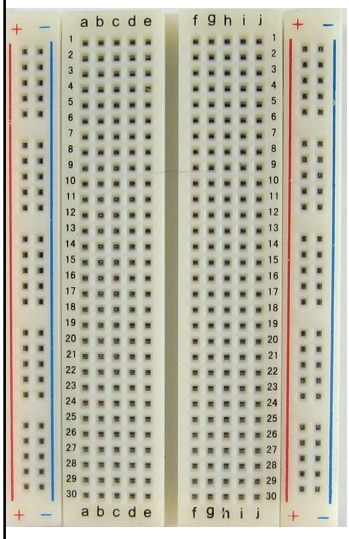
Comunque per importarle basta andare sull'Arduino ide menù sketch → importa libreria → importa libreria zip e poi selezionare la libreria da importare.

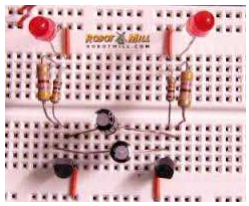
Sensore di temperatura digitale DS18B20 Interfacciamento con Arduino



Un esempio di codice per leggere la temperatura con questo tipo di sensore é contenuto nella cartella sketch all'interno della cartella PLS2016.

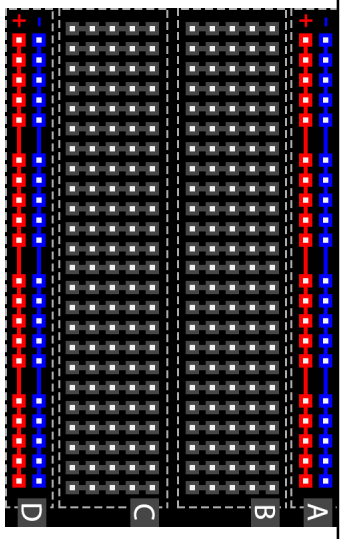
La basetta *breadboard*





È il luogo fisico dove costruiremo i nostri circuiti.

Non serve fare saldature.



I fori delle due file laterali sono collegati in verticale, gli altri sono collegati, 5 a 5, in orizzontale.

FINE

GRAZIE PER L'ATTENZIONE!