

Introduzione

Questa guida permette ad un utente inesperto di iniziare a lavorare con la scheda ARDUINO. I prerequisiti necessari sono i seguenti:

- Programmazione con un linguaggio ad alto livello (linguaggio C o similare Java, Javascript, ...)
- Conoscenza del sistema binario e esadecimale
- Utilizzo delle porte logiche fondamentali
- Utilizzo dei principali componenti elettronici (resistenze, condensatori, diodi, led, transistor e integrati)

Le esercitazioni di difficoltà crescente sono inizialmente molto semplici e facili terminano con l'analisi di complessi problemi che richiedono sovente l'utilizzo di hardware progettato ad hoc magari montato su una basetta sperimentale (Breadboard).

Una volta installato, configurato e testato il corretto funzionamento della scheda Arduino dovete ricordarvi di lasciare configurato il Personal Computer per le lezioni future.

Materiale e software indispensabili

Occorre preparare tutto il necessario per questa lezione.

Avrete bisogno di alcuni componenti hardware e software! Assicurarsi di avere tutto il necessario o non si sarà in grado di completare questa lezione.

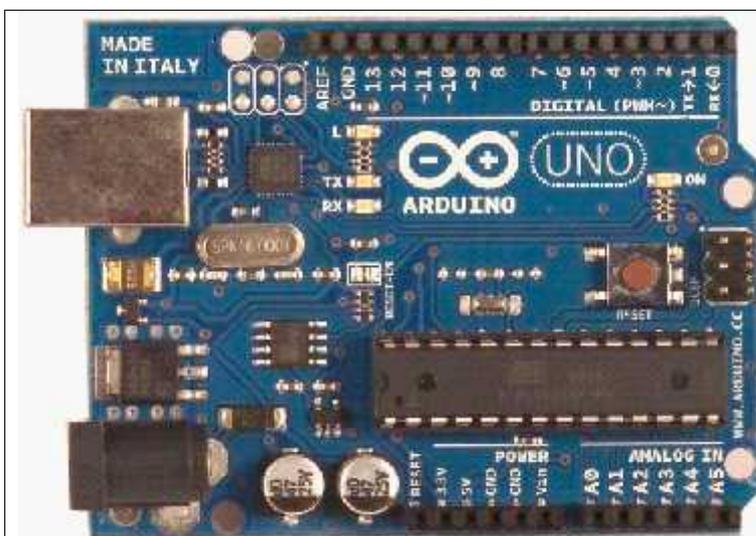


Figura 1 - Piastra Arduino Uno



Figura 2 - Cavo USB di collegamento al PC/Notebook



Figura 3 – Alimentatore 9Vcc (opzionale)

Estrarre la scheda Arduino dal suo sacchetto protettivo antistatico.

Posizionare la scheda su un pannello di supporto isolante (legno, masonite, vetro, cartone, etc.), oppure disporre 4 piedini isolanti come in figura 4.

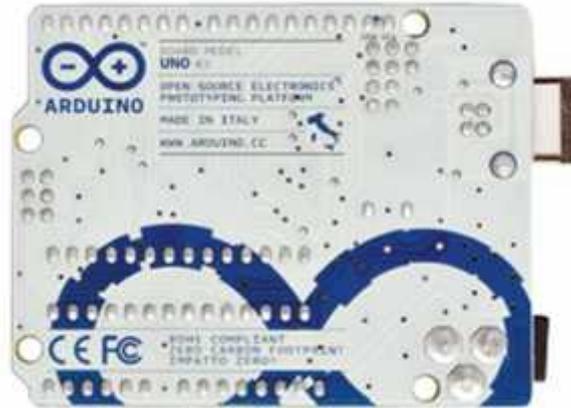


Figura 4 – Disposizione dei piedini in gomma

Caricare i driver della scheda Arduino UNO

La scheda Arduino Uno contiene al suo interno i seguenti componenti principali:

1. Integrato USB → FT232RL
2. Integrato della scheda micro → AT MEGA328P

Per procedere al corretto funzionamento della scheda occorre installare i driver.

Scaricare il file “**arduino-1.0.5-windows.exe**” relativo al software **versione 1.0.5** da installare dal sito “www.istitutoprimelevi.gov.it” selezionare menù “**Studenti**” ed in seguito selezionare “**Progetti e lavori**” oppure dal sito “<http://arduino.cc/en/Main/Software>”

Dopo aver scaricato eseguire un doppio clic sul file per installare il software

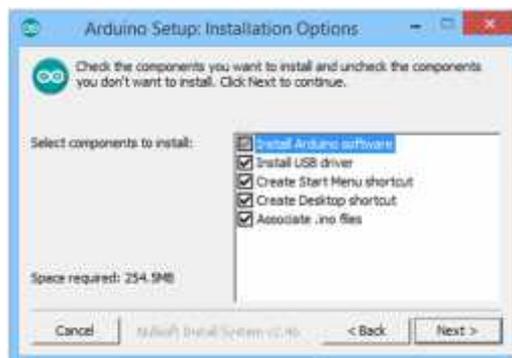


Figura 5 – Installazione delle opzioni

Effettuare un clic sul pulsante “**Next**” controllando di aver spuntato tutte le opzioni disponibili.

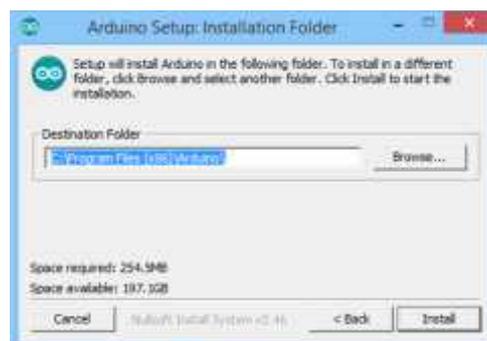


Figura 6 – Indicazione della cartella in cui verrà installato

Lasciare invariato il nome della cartella in cui verrà installato il software e procedere con un clic sul pulsante denominato **“Install”**.

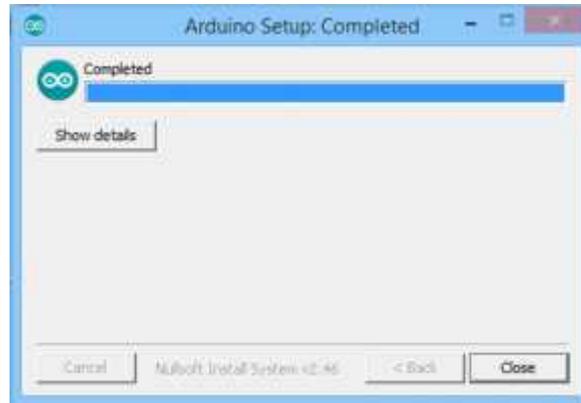


Figura 7 – Fase terminale dell'installazione

Dopo circa tre minuti di installazione occorre terminare l'installazione con un clic sul pulsante **“Close”**, si ottiene sul desktop la seguente icona



Figura 8 – Icona installata sul desktop per lanciare l'applicativo

Collegare il cavo USB alla scheda Arduino Uno e in seguito alla porta USB del Personal Computer oppure del Notebook.

ATTENZIONE! Dopo aver collegato il cavo USB al computer e alla scheda Arduino, la stessa viene alimentata direttamente tramite il cavo stesso con una tensione stabilizzata di +5V in corrente continua, quindi non è richiesto un alimentatore esterno.

Un altro metodo alternativo per alimentare la scheda Arduino è quello di inserire un alimentatore esterno alla presa DC della scheda Arduino. Verificare che si disponga di un alimentatore da 9V DC con una corrente 100÷500mA e che possieda un jack con il cavo POSITIVO collegato al terminale CENTRALE e quello NEGATIVO al terminale ESTERNO (vedi figura 9).



Figura 9 - Alimentatore a 9Vcc



Figura 10 – Cavo con connettori USB (sinistra lato PC e destra lato scheda Arduino)

Assicurarsi che il cavo USB sia collegato direttamente a una porta del computer. A volte il monitor o le tastiere hanno una porta USB a cui si può collegare. La maggior parte delle volte questo va bene, ma vi consiglio caldamente di collegarlo direttamente al computer in modo da eliminare gli eventuali problemi. Lo stesso vale per gli hub USB.

In seguito, una volta verificato il corretto funzionamento della scheda Arduino è possibile testare il collegamento con hub Usb o altre periferiche che gestiscono la porta Usb. Nel caso di dubbio è opportuno disconnettere qualsiasi periferica USB e lavorare momentaneamente solo con la porta USB della scheda Arduino per riconnettere le periferiche precedentemente scollegate (il consiglio è particolarmente valido perché risulta difficile procedere ad una installazione non andata a buon fine della periferica USB).



Figura 11 - Collegamento USB lato scheda Arduino

Si dovrebbe ottenere l'accensione di un led verde sul lato destro della scheda Arduino denominato "ON".

Dopo qualche secondo compare una nuova finestra in cui occorre scegliere di non visualizzare più questo messaggio.



Compare una finestra come quella in figura 12

Figura 12 - Schermata iniziale Arduino



Procedere al lancio del software tramite la seguente icona  presente sul desktop ed in seguito si otterrà l'ambiente di lavoro vuoto che permette di procedere con la prima esercitazione.

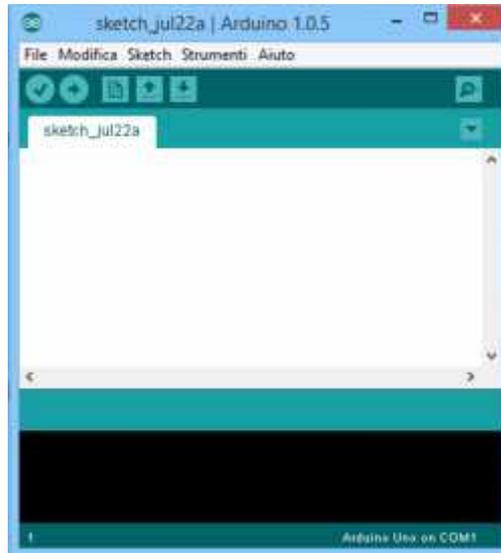


Figura 13 - Finestra software Arduino

Selezionare **“Strumenti”** → **“Tipo di Arduino”** → **“Arduino Uno”** (occorre selezionare il tipo di scheda con cui lavorare)

Seleziona **“Strumenti”** → **“Porta seriale”** → **“COM3”**. Prestare attenzione ad effettuare un clic sulla scritta **“COMxx”**, dove xx in questo esempio è il numero 3 **“COM3”** (vedere figura 13 occorre controllare che sia il numero della porta seriale corretto! Conviene eseguire un controllo da: **“Pannello di Controllo”** → **“Gestione dispositivi”** → **“Porte (COM e LPT)”**)

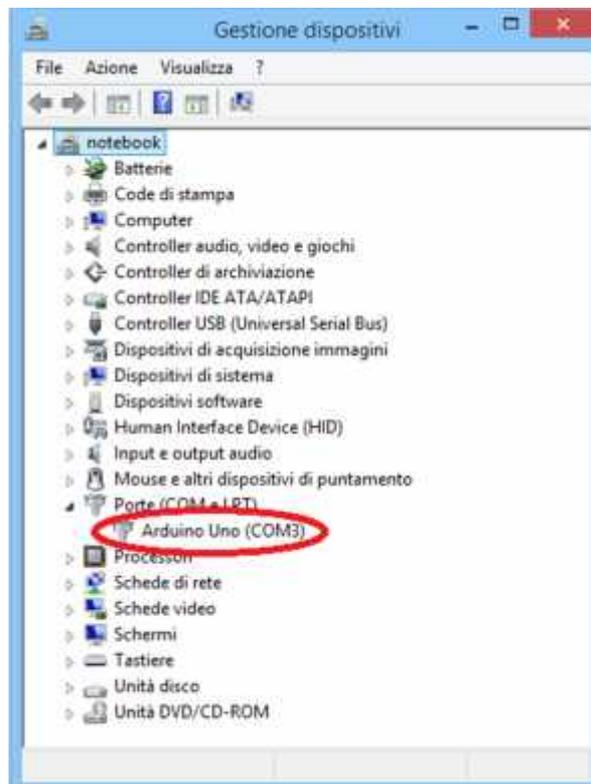


Figura 14 - Esempio di USB rilevata come COM3

Introduzione alla scheda Arduino

Dalla figura 15 è possibile capire come collegare la scheda con i circuiti esterni montati su una basetta sperimentale (Breadboard).

In particolare guardando in senso orario a partire dal centro in alto:

Denominazione	Descrizione
AREF	Tensione Analogica di riferimento
GND	Massa digitale (GND=GROUND)
Digital Pins dal 2 al 13	sono i pin del microcontrollore con la possibilità di essere configurati come Input o Output digitali
Digital Pins dal 0 al 1	sono i pin del microcontrollore con la possibilità di essere configurati come Input o Output digitali e vengono denominati “Serial In / Out” oppure “TX / RX”. Questi pin non possono essere utilizzati per Input / Output digitale (tramite le istruzioni “digitalRead” e “digitalWrite”) se si utilizza anche la comunicazione seriale (per esempio con l’istruzione “Serial.begin”).
S1	Pulsante Reset
ICSP	In-circuit Serial Programmer
Analog Input dal 0 al 5	sono i pin del microcontrollore con la possibilità di essere configurati come Input analogici
Vin POWER	Tensione di alimentazione esterna compresa tra +9 e +12Vcc
GND	Massa digitale (GND=GROUND).
5V	Tensione di alimentazione +5V stabilizzata
3V3	Tensione di alimentazione +3,3V stabilizzata
RESET	Ingresso digitale di RESET del microcontrollore
X1	Connettore a jack per l’alimentazione tramite alimentatore esterno compreso tra +9 e +12V con il positivo sul terminale centrale del jack. La commutazione tra l’alimentazione con USB e alimentatore esterno avviene in automatico.
USB	Interfaccia di comunicazione con il Personal Computer che utilizza un apposito integrato il quale simula la comunicazione tramite la porta seriale virtuale (ad esempio COM5)

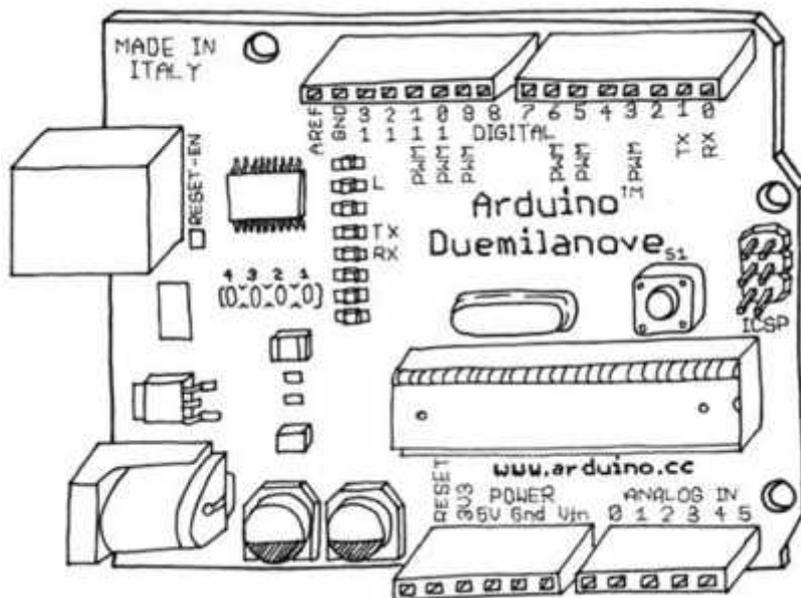


Figura 5 - Scheda Arduino Duemilanove / Uno Lato Componenti (collegamenti nuova versione)

Caratteristiche del Microcontrollore

Molti di noi sanno cosa è un computer. Esso di solito ha una tastiera, un monitor, una CPU (Central Processing Unit), una stampante e un mouse. Questi tipi di computer come i Mac o i PC, sono essenzialmente progettati per comunicare con l'uomo.

Database management, analisi finanziaria, o ogni “word-processing” sono tutti eseguiti all'interno della “grande scatola” che contiene la CPU, la memoria, l'hard-disk, etc...

L'attuale “computing”, comunque, ha luogo nella CPU. Se ci pensate, lo scopo di un monitor, della tastiera, del mouse e la stampante è “collegare” la CPU col mondo esterno.

Altri computer sono nell'auto, sullo “Space Shuttle”, nei giochi del fratello più piccolo e potrebbero perfino essere nell'asciugacapelli.

Questi dispositivi vengono denominati “microcontrollori”. Micro perché sono piccoli e controller perché “controllano” le macchine, i gadgets e ogni elettrodomestico. La principale caratteristica è il loro basso costo in rapporto alla CPU di un Personal Computer.

Il microcontrollore è poi per definizione, progettato per collegare la macchina con la gente. Essi sono duttili perché si può costruire una macchina o un dispositivo, scrivere programmi personalizzati automaticamente.

Ci sono un'infinità di applicazioni per i microcontroller. L'unico limite è la fantasia del progettista! Sono disponibili sul mercato italiano centinaia (se non migliaia) di differenti modelli di microcontroller. Alcuni sono programmati e prodotti per specifiche applicazioni così come il controllo del forno a microonde o della lavastoviglie. Altri sono riprogrammabili, questo significa che possono essere utilizzati per molte applicazioni differenti scaricando il software necessario alla singola necessità.

I microcontroller sono incredibilmente versatili, lo stesso dispositivo può controllare un modello di missile, un tostapane, il sistema di antibloccaggio dei freni dell'auto.

La scheda Arduino è una sofisticata piastra elettronica contenente un insieme di circuiti, tutti assemblati in un piccolo circuito stampato (PCB).

I programmi scritti per la scheda Arduino sono realizzati con una speciale versione del linguaggio C/C++. Molti altri microcontroller richiedono altre forme di programmazione che possono essere molto difficili da imparare ad esempio se utilizzano il linguaggio assembly.

Con il linguaggio C si possono creare programmi in pochi minuti e/o in parecchie ore. È ovvio che il grado di difficoltà risulta dipendente da diversi fattori:

- esperienza nel campo dei microcontrollori per la programmazione con un linguaggio ad alto livello del tecnico programmatore
- necessità di utilizzo con più ingressi da acquisire contemporaneamente
- necessità di gestire contemporaneamente più uscite.
- disponibilità di strumenti di misura idonei al progetto (esempio: tester, oscilloscopi, oscillatori, etc...)

Comunque, non si deve dimenticare che essendo la scheda Arduino parte integrante di un progetto Open Source si hanno a disposizione una serie completa di “librerie” ovvero di software già pronto per l'utilizzo che permettono la gestione semplificata anche di protocolli di comunicazione particolarmente difficili da implementare.

Quando creiamo dispositivi che utilizzano un microcontroller, questo agisce come un cervello, in molti casi tentiamo di imitare le operazioni del corpo umano.

Il cervello degli esseri umani si basa su informazioni ordinate per prendere le decisioni. Tali informazioni si acquisiscono attraverso vari sensi come la vista, l'udito, il tatto etc... Questi sensi percepiscono ciò che succede nel “mondo reale”, e mandano le informazioni al cervello attraverso “processi”. Comunque, quando il cervello prende una decisione, manda segnali esterni ad alcune parti specifiche del corpo per fare qualcosa nel “mondo reale”. Utilizzando gli “impulsi” ricevuti dai organi di senso, e generando le azioni per controllare le gambe, le braccia, le mani etc..., il cervello è una “unità di elaborazione” che interagisce con il mondo eterno.

Guidando per strada, ad esempio gli occhi percepiscono un gatto che sta correndo di fronte all'auto. Il cervello analizza gli impulsi, prende una decisione, e dopo da istruzioni alle braccia e mani, facendo girare il volante e le ruote per evitare l'impatto con l'animale. Questi impulsi/decisioni/azioni possono essere integrati nel microcontroller.

Noi li chiamiamo "Input/Output" o "I/O" per abbreviare. La prima lezione ci introdurrà sulle funzioni di "output" della scheda Arduino mentre le lezioni seguenti introdurranno nuove idee ed esperimenti per le prove. Ovviamente si possono utilizzare le idee fornite in queste lezioni per inventare una nuova applicazione o progetto.

Tutti i sistemi a microcontroller (o computer) consistono in due componenti primari:

- "Hardware"
- "Software"

L'Hardware è l'effettivo componente fisico del sistema.

Il Software è una lista di istruzioni che risiedono dentro l'hardware e ne permettono il corretto funzionamento. L'obiettivo che ci prefiggiamo non è quello di costruire l'hardware perché le schede sono già assemblate e collaudate, ma di scrivere un programma software in linguaggio C per "controllare" l'hardware.

Per ordinare al microcontroller come agire con il mondo reale, abbiamo però bisogno di collegare pochi componenti hardware alla piastra del microcontrollore tramite una basetta sperimentale (Breadboard) che non richiede nessuna saldatura. Questa Breadboard viene utilizzata per semplici connessioni di interfacciamento alla scheda Arduino.

Questi esperimenti/esercizi proposti ci introdurranno nell'affascinante mondo dei microcontroller utilizzando una scheda molto popolare chiamata ARDUINO.

La scheda Arduino Uno utilizza il microcontrollore ATMEGA328P-PU che possiede le seguenti principali caratteristiche:

- Prestazioni elevate, basso consumo per i microcontroller a 8-Bit
- Architettura Avanzata di tipo RISC
 - 131 potenti istruzioni - La maggior parte richiedono solo un singolo ciclo di clock per l'esecuzione dell'istruzione
 - 32 x 8 Registri di uso generale per il lavoro
 - Funzionamento completamente statico
 - Fino a 20 MIPS con un quarzo alla frequenza di clock di 20 MHz
 - 32K Byte di In-sistema self-programmabile Flash Program Memory (ATmega328P)
 - 1K Byte di memoria EEPROM (ATmega328P)
 - 2K Byte di memoria SRAM interna (ATmega328P)
 - Cicli di cancellazione/scrittura: 10.000 su Flash/100, 000 su EEPROM
 - Conservazione dei dati: 20 anni a 85 °C oppure 100 anni a 25 °C
 - Optional Boot Code, sezione con lock bit indipendenti
- Caratteristiche periferiche
 - Due 8-bit del timer / contatori separati con Prescaler e modalità di confronto
 - Un 16-bit Timer / Counter con separato Prescaler, modalità confronto e Modalità Capture
 - 6 canali di controllo con PWM (Pulse With Modulation)
- Misurazione della tensione analogica
 - 6 canali con risoluzione di 10-bit del convertitore ADC nel contenitore DIP
- Misura della temperatura
 - USART seriale programmabile
 - Interfaccia Seriale Master / Slave SPI
 - Byte-oriented a 2 fili interfaccia seriale (Philips I2C compatibile)
 - Watchdog Timer programmabile con separato oscillatore on-chip
 - Comparatore Analogico On-chip
 - Interrupt e Wake-up su Pin Change
- Funzioni speciali

- Power-on Detection Brown-out reset e programmabile
- Oscillatore calibrato internamente
- Sorgenti di interrupt sia esterne che interne
- 6 modalità di risparmio potenza (Sleep): Idle, ADC Noise Reduction, Power-save, Power-down, in standby, ed Extended Standby
- I / O e contenitori
 - 23 pin di I/O programmabili indipendentemente come Input o come Output
 - Contenitore disponibile a 28-pin PDIP
- Tensione di funzionamento:
 - da 1,8V a 5,5V per ATmega328P
- Temperatura:
 - da -40 °C a 85 °C
- Velocità di elaborazione:
 - da 0 a 20 MHz con una tensione di alimentazione compresa tra 1,8V e 5,5V
- Basso consumo energetico a 1 MHz con 1,8 V e una temperatura di 25 °C per ATmega328P si ottiene un consumo di corrente di:
 - in modalità attiva (Active Mode): 0,2 mA
 - in modalità di risparmio energetico (Power-down Mode): 0,1 μ A
 - in modalità di risparmio energetico massimo: 0,75 μ A (compresi 32 kHz RTC)

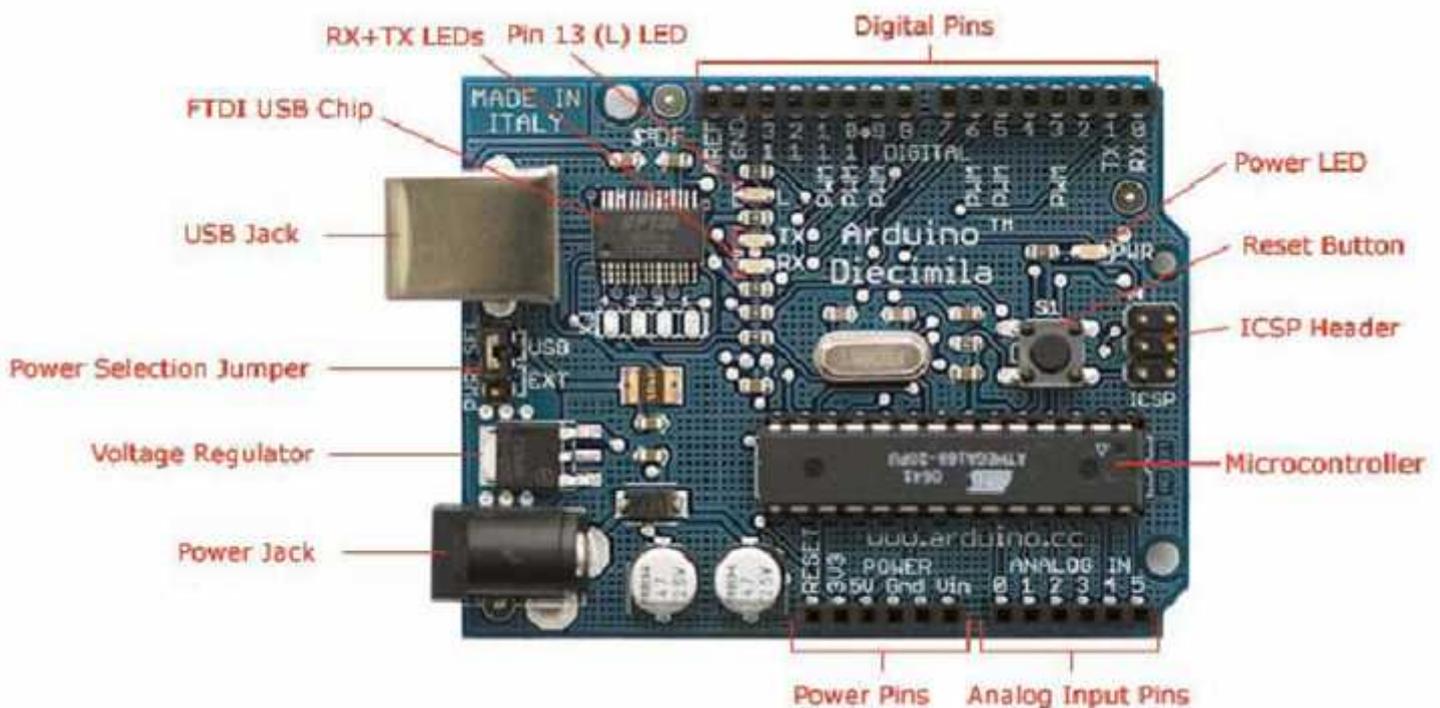


Figura 6 - Riferimenti per i collegamenti alla scheda Arduino

Utilizzo delle basette sperimentali senza saldature (Breadboard)

Questa sezione permette di utilizzare correttamente le basette sperimentali per il cablaggio di piccoli schemi da collegare alla piastra Arduino.

1) Esecuzione dei montaggi

Ogni gruppo di alunni preleva dall'armadio / magazzino i componenti necessari al montaggio.

Al termine dell'esercitazione il circuito deve essere smontato, e i componenti devono essere rimessi esattamente nel posto da cui sono stati prelevati.

Eventuali componenti guasti, danneggiati, o con i terminali non adatti al montaggio (troppo corti, sporchi, attorcigliati, ...) vanno tenuti da parte e consegnati al personale di laboratorio.

Nel laboratorio sono disponibili:

- resistenze a strato di carbone da $\frac{1}{4}W$, 5%, secondo la serie E12;
- condensatori con valori compresi tra 10pF e $1\mu F$ di vario tipo (ceramici, ceramici multistrato, film plastico), con tolleranze del 10% o 20%, secondo la serie E6 (qualche valore secondo E12);
- condensatori elettrolitici da 1 a 2200 μF , con tolleranza del 20% o del 40%;
- i componenti attivi richiesti per ciascuna esercitazione;
- potenziometri di vari valori;
- interruttori, pulsanti e deviatori e LED (per forzare e rilevare stati logici).

Usare solo componenti e cavi con terminali di diametro $0,5\div 0,7$ mm (resistenze e condensatori piccoli); terminali più grossi danneggiano i contatti della basetta breadboard.

Valori normalizzati secondo la **serie E6** (6 valori per decade):

1 - 1.5 - 2.2 - 3.3 - 4.7 - 6.8 e suoi multipli

Valori normalizzati secondo la **serie E12** (12 valori per decade, **è quella più utilizzata**):

1 - 1.2 - 1.5 - 1.8 - 2.2 - 2.7 - 3.3 - 3.9 - 4.7 - 5.6 - 6.8 - 8.2 e suoi multipli

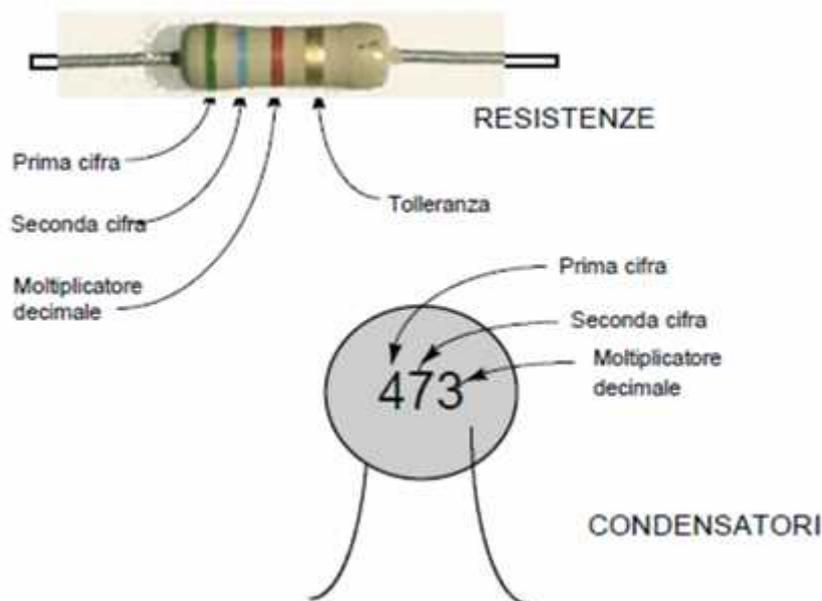


Figura 17 - Resistori e Condensatori

Il valore dei componenti passivi (resistenze, condensatori) di uso corrente è indicato con tre cifre.

Lette da sinistra, le cifre indicano:

- prima cifra significativa
- seconda cifra significativa
- moltiplicatore decimale

Nel caso delle resistenze (e per alcuni tipi di condensatori) le tre cifre sono rappresentate con fasce colorate secondo il "**codice dei colori**". In altri casi (ad esempio per la maggior parte dei condensatori), le tre cifre sono direttamente stampate sul componente.

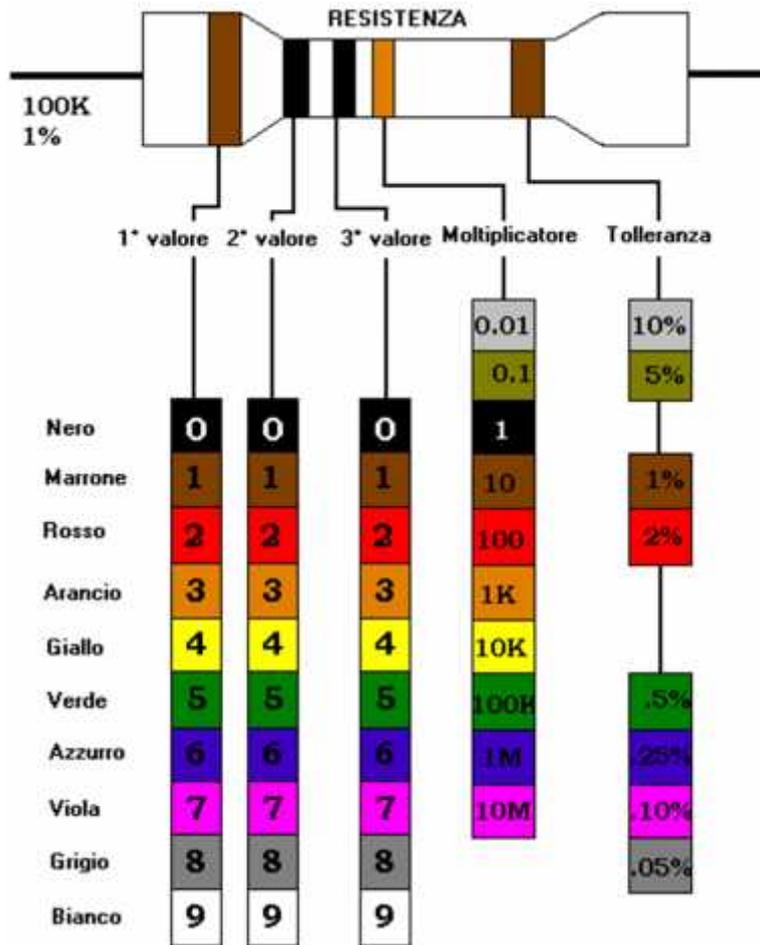


Figura 18 - Codice colore dei resistori

Codice dei colori:

nero	0
marrone	1
rosso	2
arancio	3
giallo	4
verde	5
blu	6
viola	7
grigio	8
bianco	9

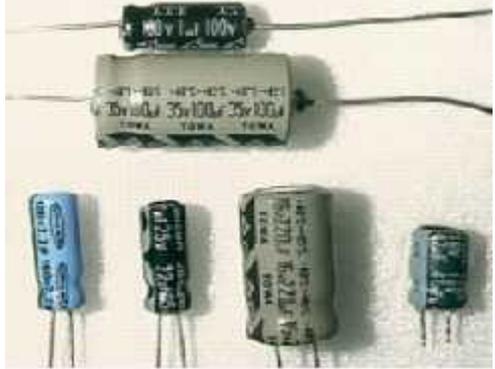
Per le resistenze il valore è espresso in Ohm, per i condensatori ceramici e a film plastico in pF. Altre indicazioni (fasce colorate o lettere) indicano la tolleranza e altri parametri.

I componenti di precisione (ad esempio resistenze all'1%) possono avere 4 cifre (si aggiunge una quarta fascia).

Dato che i componenti possono essere sistemati nel posto sbagliato, è necessario verificarne sempre il corretto valore.

È inutile cercare di ottenere valori fuori standard collegando in serie o in parallelo più componenti: le tolleranze rendono illusori valori intermedi tra quelli della serie normalizzata.

I vari tipi di condensatore hanno caratteristiche e comportamento diverso. È importante scegliere il condensatore adatto in base all'impiego; alcune indicazioni sono riportate nel seguito.

<p>Condensatori ceramici multistrato</p> <p>Il valore è indicato da tre cifre, con significato analogo alle fasce colorate.</p> <p>Dimensioni ridotte e buon comportamento alle frequenze elevate ma scarsa precisione (20 % per i valori più alti). Adatti come bypass sulle alimentazioni 100 nF</p>	
<p>Condensatori a film plastico</p> <p>Hanno buona precisione, ma sono adatti per lavorare solo a frequenze relativamente basse (banda audio o pochi MHz)</p>	
<p>Condensatori elettrolitici</p> <p>Richiedono la presenza di una componente continua di polarità ben definita. Devono essere rispettate polarità e tensione di funzionamento.</p> <p>Capacità elevata con dimensioni ridotte; adatti in banda audio o per alimentatori.</p> <p>Precisione molto scarsa (+50 - 20 %)</p> <p>La foto a lato riporta esempi di condensatori elettrolitici in alluminio.</p>	
<p>Condensatori elettrolitici tantalio</p> <p>I condensatori elettrolitici al tantalio (a destra) hanno dimensioni ridotte e miglior comportamento alle frequenze elevate.</p>	

2) Piastre per montaggi

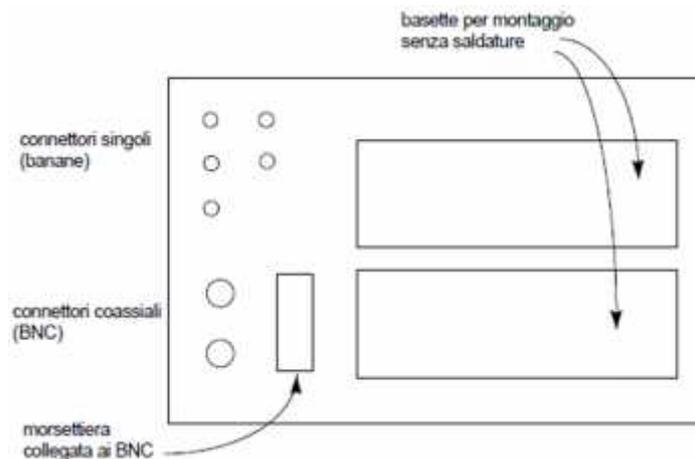


Figura 19 - Collegamenti esterni alla Breadboard

Le piastre a disposizione per i montaggi comprendono:

- boccole per cavi con banane (da usare per le alimentazioni)
- connettori coassiali tipo BNC (da usare per i segnali), riportate su una morsettiere.

Nelle basette con due BNC il morsetto rosso corrisponde al conduttore centrale, e il morsetto nero alla massa. Per le basette con tre BNC verificare le connessioni nella parte inferiore della basetta.

3) Basette per montaggi senza saldatura (Breadboard).

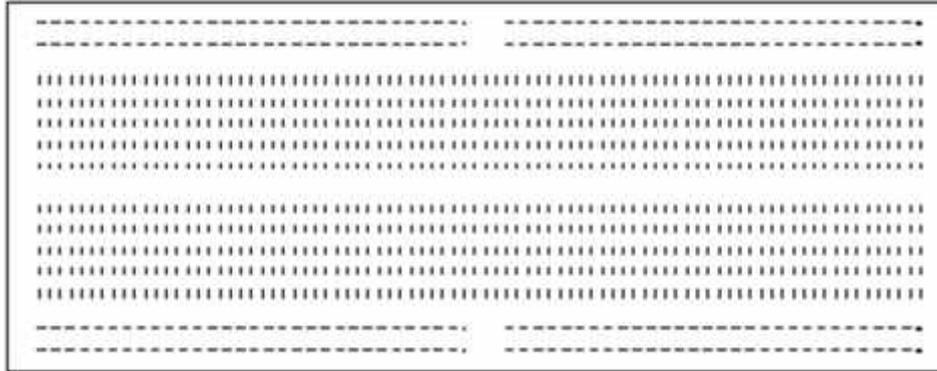


Figura 207 - Disposizione dei punti di contatto disponibili sulla Breadboard

Le interconnessioni sono realizzate da contatti a molla presenti sotto ogni foro della basetta (indicati nel disegno con trattini verticali o orizzontali).

I terminali di componenti o i fili inseriti nei fori vengono automaticamente collegati con quelli inseriti nei fori adiacenti.

Per evitare falsi contatti e altri malfunzionamenti conviene sempre verificare che la parte di conduttore da infilare nella basetta non sia ossidata (eventualmente ripulirla con le pinze/tronchesine fornite, o accorciare il conduttore). Evitare di inserire terminali attorcigliati o piegati.

Usando fili troppo sottili, oppure con tracce di isolante in prossimità degli estremi, può capitare di infilare tra i contatti la parte isolata; per evitare questo rischio togliere almeno 1 cm di isolante a ciascun estremo del conduttore.

4) Collegamenti esistenti tra i vari punti di connessione

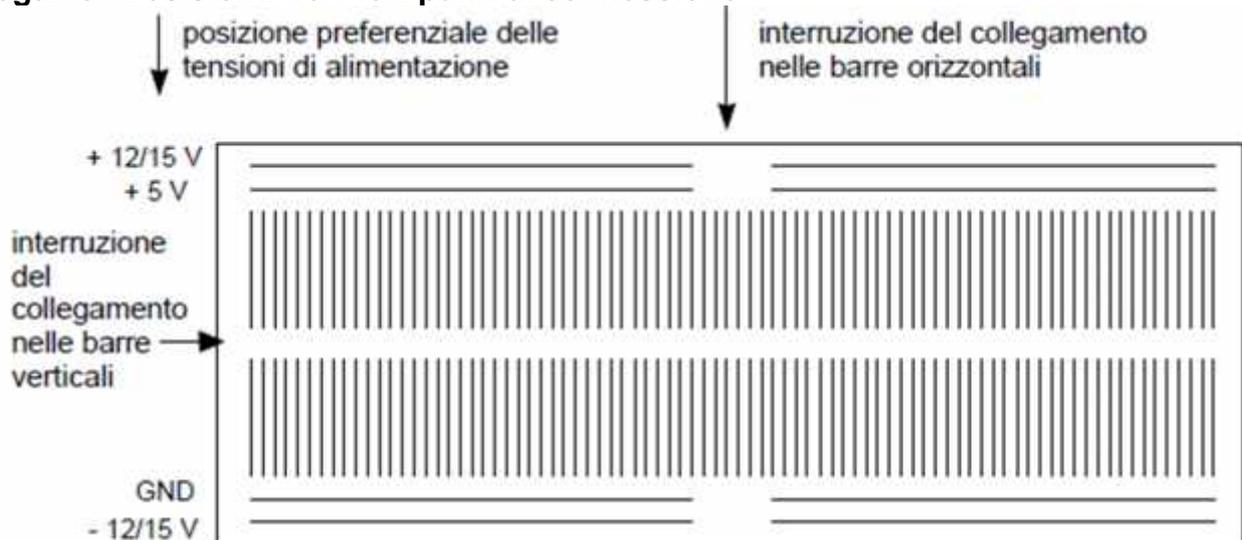


Figura 8 - Disposizione dei collegamenti interni alla Breadboard

Questo disegno indica i collegamenti tra i punti di inserzione dei componenti; sono disponibili:

- quattro barre orizzontali (interrotte al centro), da usare per massa e alimentazioni;
- due gruppi di strisce verticali, da usare per montare e collegare i vari componenti.

Usare le barre orizzontali per massa e alimentazioni (in ordine di valore; la più positiva in alto).
Usare le barre verticali per inserire e collegare i componenti attivi e passivi.

5) Contenitori Dual-In-Line

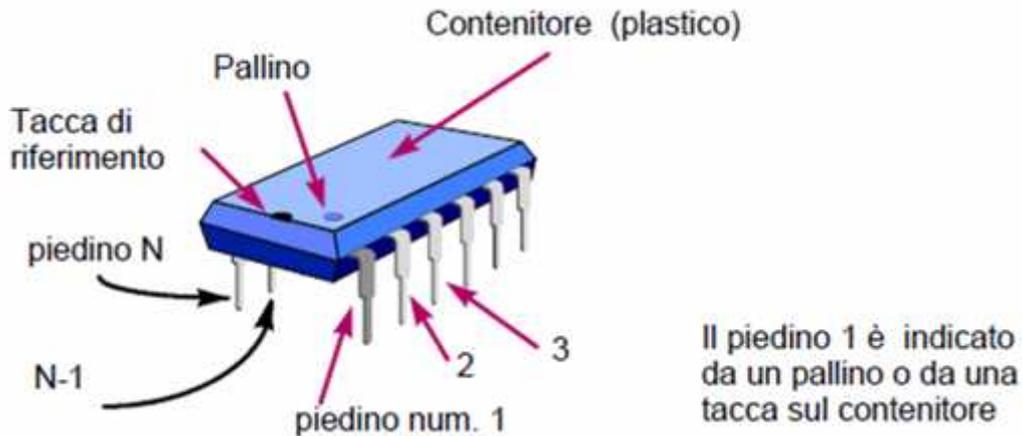


Figura 22 - Riconoscimento dei pin di un integrato DIL

Integrato in contenitore DIL

La maggior parte dei componenti attivi (circuiti integrati) usati nel laboratorio sono collocati in contenitori detti DIL (Dual-In-Line).

La numerazione dei piedini è sempre quella indicata in figura 22.

Osservando il componente dall'alto, la numerazione procede in verso antiorario.

Per i transistori in contenitore metallici (T05 e T018) la linguetta sul contenitore indica l'emettitore.

In tutte le altre situazioni, o per casi dubbi, fare riferimento alle caratteristiche del componente.

6) Montaggio dei componenti

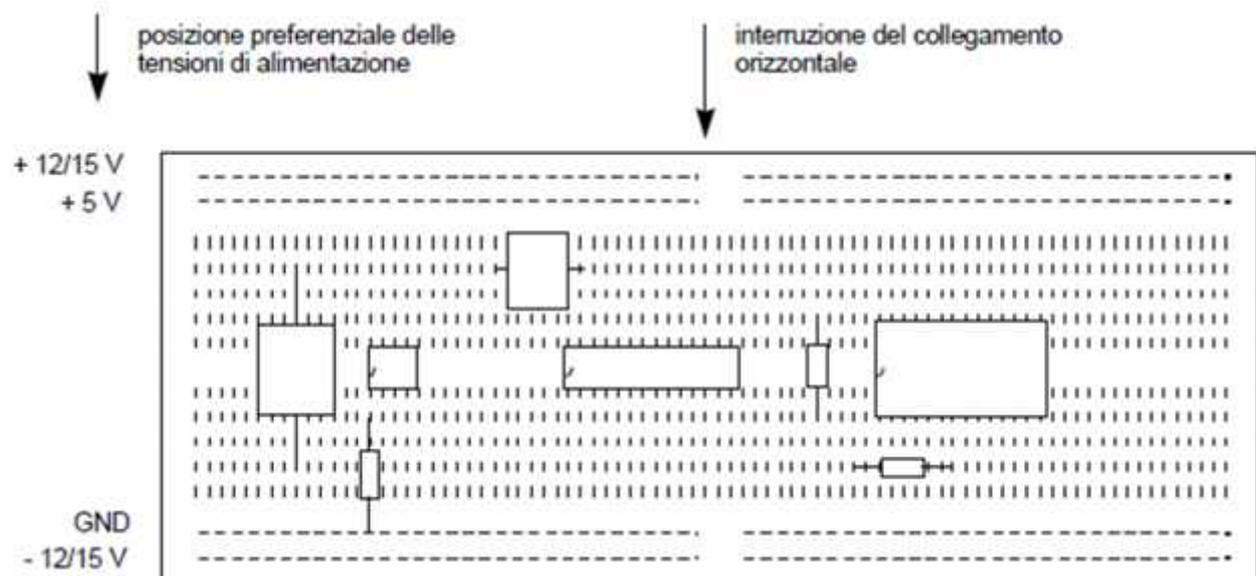


Figura 239 - Posizionamento dei componenti sulla Breadboard

Studiare il montaggio in modo da rispecchiare, per quanto possibile, la disposizione delle parti indicata nello schema elettrico.

Montare gli integrati tutti nello stesso verso (ad esempio con il pin 1 in basso a sinistra).

Per gli amplificatori operazionali usare preferibilmente componenti singoli in contenitore DIL 8 pin, che permettono di raggruppare il montaggio stadio per stadio.

Nel caso di circuiti complessi è consigliabile montare uno stadio (o una parte autonoma) per volta, e verificarne via via il funzionamento.

Può essere necessario disaccoppiare le alimentazioni verso massa con condensatori da 100nF (ceramici multistrato) collegati in prossimità dei piedini di alimentazione di ciascun integrato.

7) Collegamenti

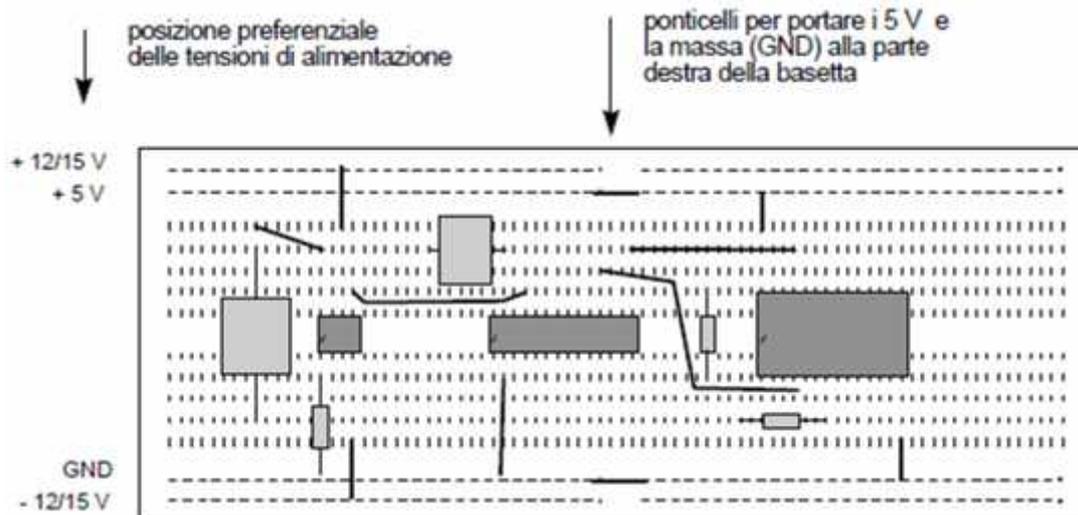


Figura 24 - Collegamenti tra i componenti

Usare solo il filo rigido fornito con la basetta;

Eseguire dei collegamenti diretti e brevi per quanto possibile;

Se occorre alimentare i circuiti nella parte destra della basetta, ripristinare con ponticelli il collegamento tra le barre orizzontali.

Esempio di montaggio ben fatto:

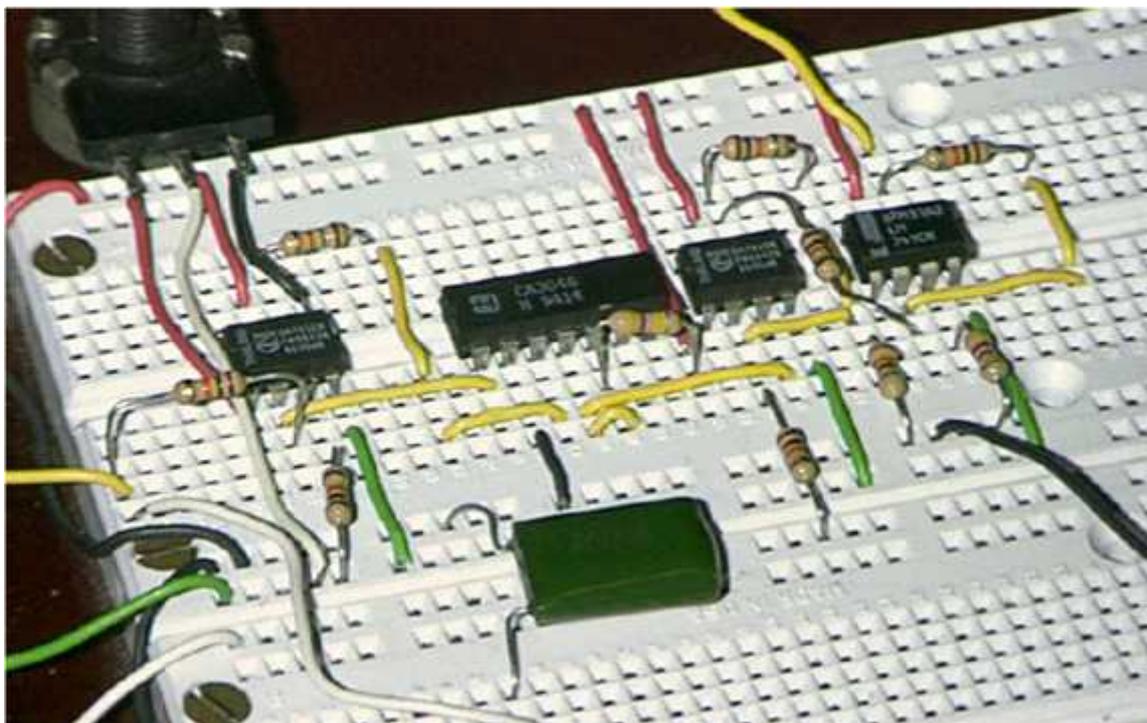


Figura 25 - Esempio di montaggio dei componenti e collegamenti corretti

Lo stesso circuito, montato ancora in modo accettabile (senza accorciare troppo i terminali dei componenti)

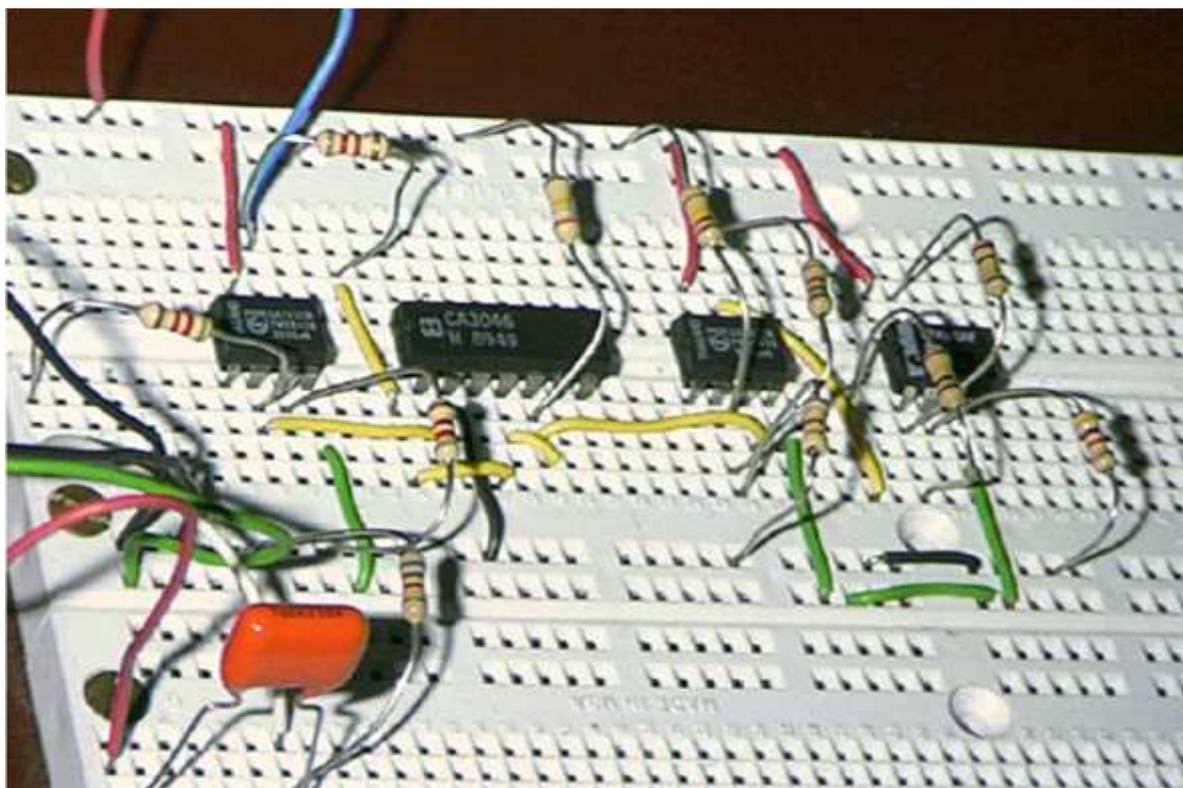


Figura 26 - Esempio di montaggio dei componenti e collegamenti accettabili

8) Situazioni da evitare

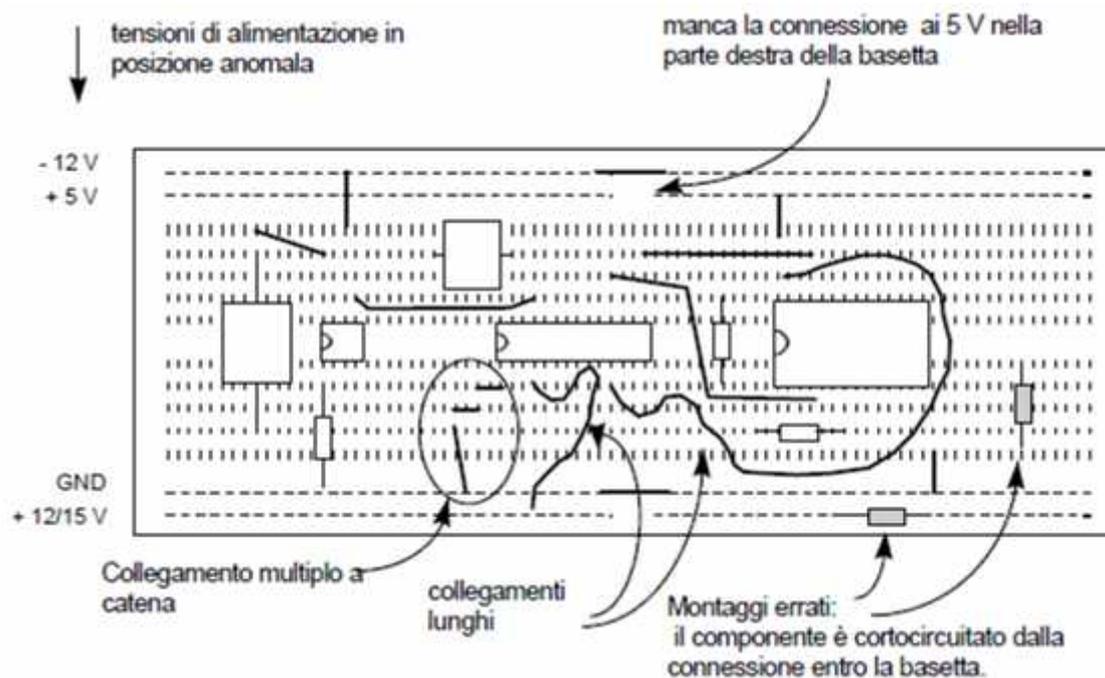


Figura 27 - Esempio di montaggio dei collegamenti non corretti

Evitare di usare le barre orizzontali per segnali diversi dalle alimentazioni, o di scambiare la posizione di queste rispetto a quella indicata.

Evitare i rimandi multipli (collegamenti a catena). Questo vale specialmente per i nodi ad alta impedenza, per gli ingressi di amplificatori operazionali, per i collegamenti di massa e di alimentazione.

La foto nel seguito è un esempio di come NON si deve eseguire il montaggio (fili lunghi e incrociati, alimentazioni e massa con rinvii a catena, ...).

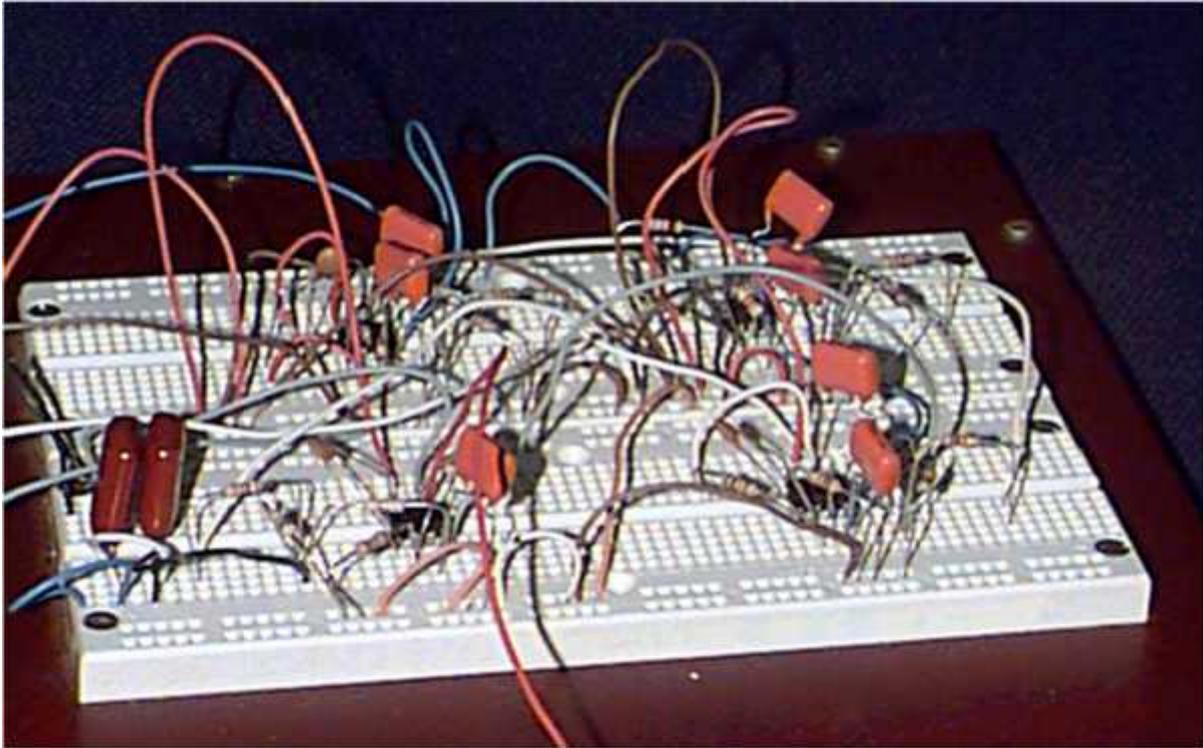


Figura 28 - Esempio di montaggio dei componenti e collegamenti non accettabili

L'elevatissima impedenza di ingresso dei circuiti CMOS determina che i relativi morsetti di input si portino ad un potenziale che dipende dai campi elettrici esterni. Oltre a modificare il corretto funzionamento, questa condizione può danneggiare il circuito integrato (ad esempio perché porta i circuiti logici ad operare in una zona lineare con una conseguente elevata dissipazione dell'integrato che aumentando la temperatura può determinare anche la distruzione fisica dello stesso). Pertanto gli ingressi dei circuiti **digitali CMOS non devono essere mai lasciati "aperti"** ovvero non collegati a massa o al positivo dell'alimentazione.

Gli ingressi degli integrati in tecnologia TTL possono essere lasciati "aperti" ovvero non collegati perché si portano automaticamente a livello logico alto (HIGH=1), ma conviene preferibilmente forzare l'input alto con una resistenza di pullup del valore di 10-100Kohm collegata tra l'input e il positivo dell'alimentazione a +5V.

Esercitazione N. 1

Nome progetto: **Blink_1.ino**

Specifiche del progetto:

Il candidato sulla base delle proprie conoscenze implementi un software in linguaggio C opportunamente commentato per la scheda Arduino, in grado di gestire un led collegato con l'anodo al pin 13 del micro e il catodo a massa.

In particolare si richiede di temporizzare l'accensione e spegnimento del led con un tempo di 0,2 secondi, ovvero per $T_{on} = 0,2 \text{ sec.} = 200 \text{ msec.}$ (si ricorda che 1 secondo è uguale a 1000 millisecondi) il led è acceso poi si spegne per altri $T_{off} = 0,2 \text{ secondi}$ e si ripete questo ciclo

all'infinito, quindi la frequenza di lampeggio del diodo led sarà pari a $f = 1 / (T_{on} + T_{off}) = 1 / (0,2 + 0,2) = 2,5$ Hertz

Effettuare sulla scheda Arduino i seguenti collegamenti:

- Collegare al pin 13 l'anodo del led LD1 (LED_A) con il catodo collegato verso massa (GND)

Si scarichi il file denominato “**Blink_1.ino**” dal sito “www.istitutoprimelevi.gov.it” quindi selezionare menù “**Studenti**” ed in seguito selezionare “**Progetti e lavori**”, tale file può essere memorizzato in qualsiasi parte del disco fisso, ad esempio sul desktop può andare altrettanto bene l'importante è ricordarsi la cartella di destinazione.

Lanciare il software Arduino con un doppio clic sull'icona relativa → Selezionare il menu “File” → selezionare “Open” → selezionare la cartella che si trova in C:\Arduino esempi” → selezionare “Apri” → selezionare il file appena scaricato sul desktop denominato “Blink_1.ino”, vedere figura 29.

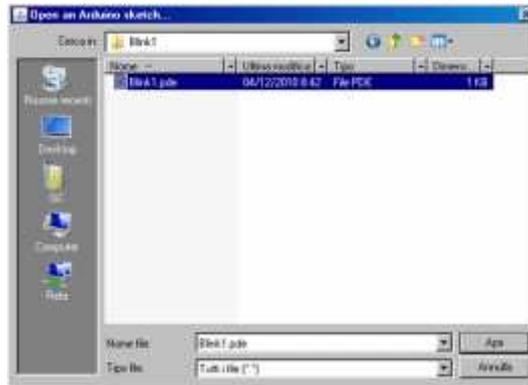


Figura 29 - Finestra per aprire il primo esempio

→ il software in linguaggio C visualizzato in figura 30 è il primo esempio di come si realizza una semplice lampeggio del diodo led interno (quello indicato in figura 31) oppure di un diodo led esterno alla scheda Arduino con l'anodo (A = terminale più lungo) collegato al pin 13 mentre il catodo (K = terminale più corto) collegato a GND (quello indicato in figura 32).

```

/*  I.I.S. Primo LEVI - Torino
    Esercizio N. 1  Progetto: Blink_1
    Autore: Questo e' un esempio di pubblico dominio
    Descrizione: Accendi il LED per 0,2 secondi, in seguito spegnilo
    per 0,2 secondi e ripeti il ciclo all'infinito.
    Data: 03/12/2010 */
void setup() // funzione di configurazione dei Input/Output
{ // inizializza il pin 13 come output, perche' e' collegato un LED
  pinMode(13, OUTPUT);
}
void loop() // programma principale (main) --> ciclo infinito (loop)
{
  digitalWrite(13, HIGH); // accendi il LED forzando un livello ALTO sul pin 13
  delay(200);             // aspetta 0,2 secondi
  digitalWrite(13, LOW);  // spegni il LED forzando un livello BASSO sul pin 13
  delay(200);             // aspetta 0,2 secondi
}

```

Figura 30 - Esercizio N. 1 – Accensione a intermittenza acceso / spento (on/off – effetto blink) di un led



Figura 31 - Led interno utilizzato nell'esempio N. 1

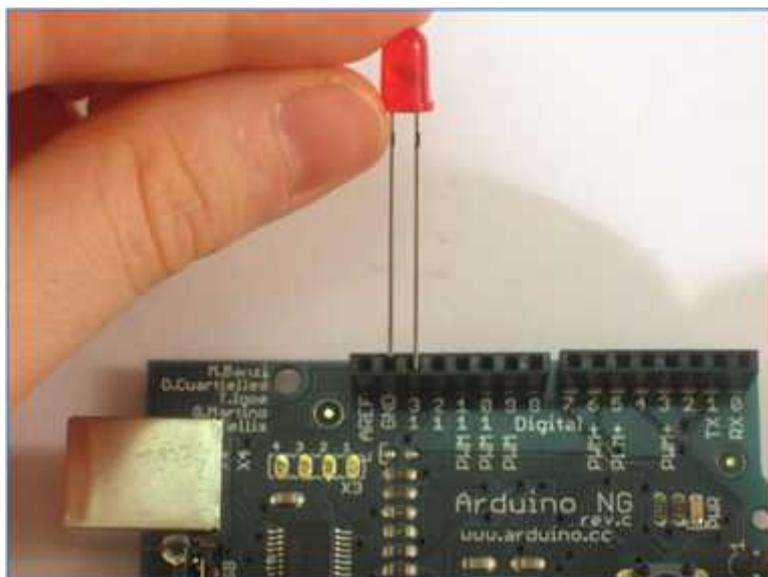


Figura 32 - Led esterno alla scheda da collegare per l'esempio N. 1

Se possedete una scheda Arduino Uno, il processo di caricamento è abbastanza veloce, basta avviare il caricamento del software. La scheda verrà automaticamente aggiornata con il nuovo software da scaricare. Procedere nel seguente modo per scaricare in nuovo software sulla scheda stessa → effettuare un clic sull'icona

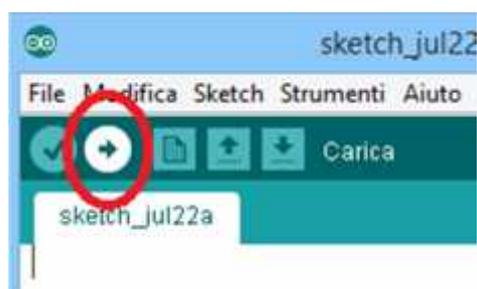
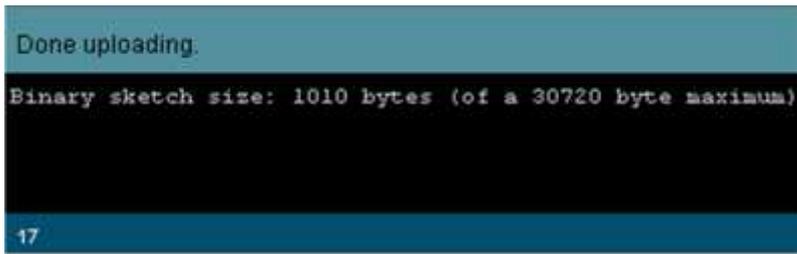


Figura 33 - Icona per effettuare l'"UPLOAD" sulla scheda

→ attendere qualche secondo per la compilazione e trasmissione e dopo sarà possibile visualizzare il seguente messaggio



```
Done uploading.
Binary sketch size: 1010 bytes (of a 30720 byte maximum)
17
```

Figura 34 - Fine download del software

→ Il LED interno e/o esterno inizierà a lampeggiare con un periodo di 2,5Hz. Come avrete notato il programma, appena scaricato sulla scheda Arduino, inizia subito a “lavorare”, cioè ad eseguire il software che è stato implementato e poi compilato.

N.B. → Se durante la fase di compilazione del software si verifica un **errore** il programma si arresta e indica nella finestra in basso la tipologia di errore riscontrato. In figura 23 è stato simulato un errore di sintassi scrivendo in modo errato l’istruzione “**dilay(200);**” invece di quella corretta “**delay(200);**”

```
void loop() // programma principale (main) --> ciclo infinito (loop)
{
  digitalWrite(13, HIGH); // accendi il LED forzando un livello ALTO sul pin 13
  delay(200);           // aspetta 0,2 secondi
  digitalWrite(13, LOW); // spegni il LED forzando un livello BASSO sul pin 13
  dilay(200);          // aspetta 0,2 secondi
}
```



```
<
'dilay' was not declared in this scope

Blink_1_ino.ino: In function 'void loop()':
Blink_1_ino:16: error: 'dilay' was not declared in this scope
16
```

Figura 35 – Esempio di Errore di sintassi incontrato e rilevato durante la compilazione

Esercitazione N. 2

Nome progetto: **Digital_Read_Led_1.ino**

Specifiche del progetto:

Il candidato sulla base delle proprie conoscenze implementi un software in linguaggio C opportunamente commentato per la scheda Arduino, in grado di gestire un pulsante di tipo n.a. (normalmente aperto) che accende e spegne un led.

In particolare se viene premuto il pulsante di tipo n.a. denominato S1 e per tutto il tempo verrà premuto, il led LD1 si dovrà accendere mentre il led si spegnerà appena il pulsante viene rilasciato.

Effettuare sulla basetta Breadboard esterna alla scheda Arduino i seguenti collegamenti:

- Collegare al pin 13 l’anodo del led LD1 (LED_A) con il catodo collegato verso massa (GND)

- Collegare al pin 7 un terminale del pulsante n.a. denominato S1 (PULS_A) con l'altro terminale collegato a massa (GND)
- Collegare al pin 7 un terminale della resistenza di pull-up da 10Kohm – 1/4w – 5% (marrone – nero – arancio – oro) mentre l'altro terminale della resistenza dovrà essere collegato al +5V.

Si scarichi il file denominato “**Digital_Read_Led_1.ino**” dal sito “www.istitutoprimumolevi.gov.it” quindi selezionare menù “**Studenti**” ed in seguito selezionare “**Progetti e lavori**”, tale file può essere memorizzato in qualsiasi parte del disco fisso, ad esempio sul desktop può andare altrettanto bene l'importante è ricordarsi la cartella di destinazione.

Lanciare il software Arduino con un doppio clic sull'icona relativa → Selezionare il menu “File” → selezionare “Open” → selezionare la cartella che si trova in C:\Arduino esempi” → selezionare “Apri” → selezionare il file appena scaricato sul desktop denominato “Digital_Read_Led_1.ino”.

Il circuito da utilizzare è estremamente semplice quanto in apparenza “inutile”. Ha solo uno scopo didattico perché non ha senso “sprecare” un microcontrollore per accendere un led tramite un pulsante. In particolare si potranno determinare le seguenti condizioni di funzionamento:

Azione sul Pulsante di tipo n.a. (S1) collegato al pin 7 tramite una resistenza di pull-up del valore di 10Kohm (vedere figura 39)	Stato del led interno e/o esterno (LD1 se esterno deve essere collegato al pin 13)
Pulsante Premuto	Led Acceso
Pulsante Non Premuto	Led Spento

Il pulsante di tipo n.a. (figura 36 – 37 – 38) è un componente che cortocircuita i due contatti collegati al resto del circuito quando viene premuto, quindi permette il passaggio della corrente.

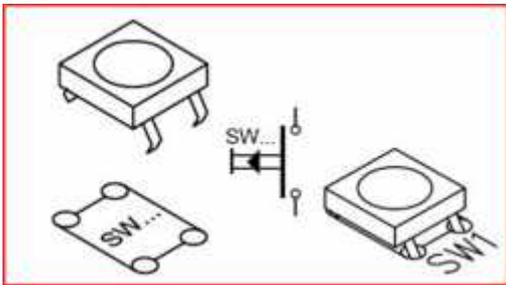


Figura 36 - Pulsante n.a. per montaggio su circuito stampato o Breadboard

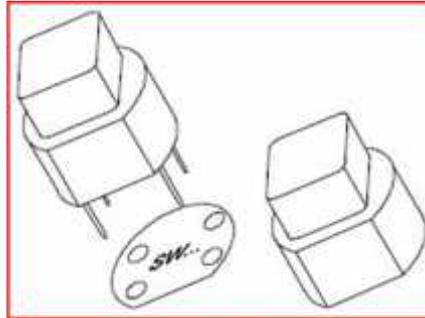


Figura 37 - Pulsante n.a.

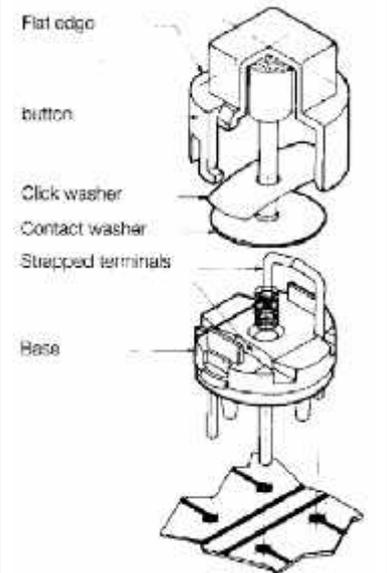


Figura 38 - Vista interna di un comune pulsante

Se non è premuto i due contatti forniscono una elevatissima resistenza (circuito aperto) ovvero la corrente che lo attraversa è uguale a zero ampere.

Esiste anche il pulsante di tipo n.c (normalmente chiuso) può essere collegato alla piastra del micro con una differente configurazione (vedere figura 39 e 40).

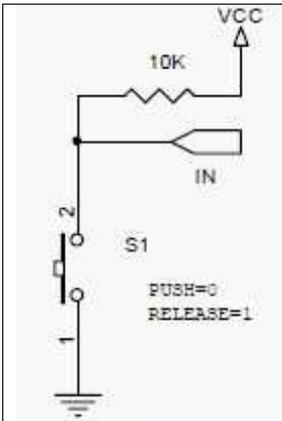


Figura 39 - Pulsante normalmente aperto (N.A.)

Circuito con resistenza di pull-up (tipica da 10Kohm) per collegare un pulsante di tipo N.A. oppure N.O. (*normaly open*) ad un pin del microcontrollore configurato come INPUT.

Pulsante premuto → livello logico acquisito sul pin del micro → 0 → livello LOW

Pulsante rilasciato → livello logico acquisito sul pin del micro → 1 → livello HIGH

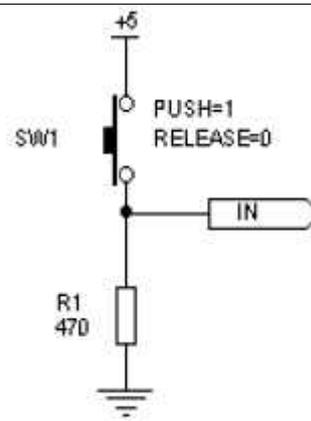


Figura 40 - Pulsante normalmente chiuso (N.C.)

Circuito con resistenza di pull-down per collegare un pulsante di tipo N.C. (*normaly close*) ad un pin del microcontrollore configurato come INPUT.

Pulsante premuto → livello logico acquisito sul pin del micro → 0 → livello LOW

Pulsante rilasciato → livello logico acquisito sul pin del micro → 1 → livello HIGH

Procediamo con i collegamenti alla scheda Arduino. Il primo collegamento va da un contatto del pulsante ad uno dei due terminali del resistore di pull-up (in questo caso 10 KOhm). Il secondo collegamento deve essere effettuato sempre dal nodo precedente, ovvero si deve collegare il resistore da 10Kohm e il pulsante, anche al pin 7 della scheda Arduino. Il terzo collegamento si effettua partendo dal terminale libero della resistenza da 10Kohm che deve essere collegato al +5V dell'alimentazione. Il quarto collegamento necessario è quello di utilizzare il contatto ancora libero del pulsante per collegarlo alla massa denominata GND (ground) della piastra Arduino (vedere figura 40 per i collegamenti necessari).

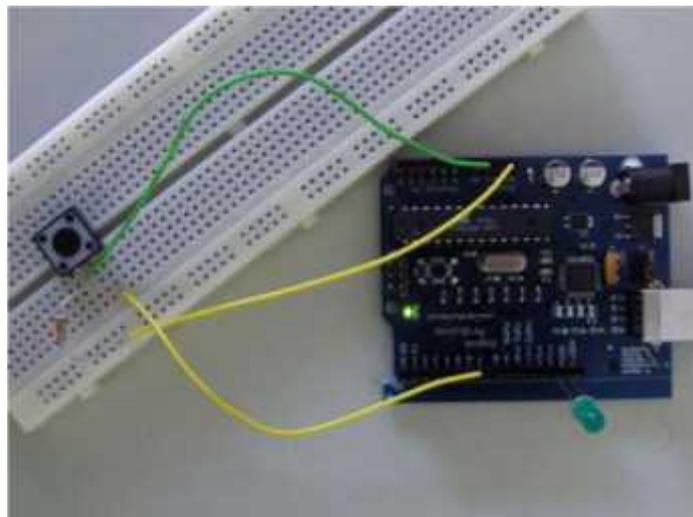


Figura 41 - Collegamenti alla scheda Arduino del pulsante S1 di tipo n.a. e del led verde LD1

Quando il pulsante è aperto (non premuto) non vi è alcuna connessione tra i due contatti del pulsante, in questo modo il pin 7 risulta collegato a +5V (attraverso la resistenza di pull-up che “spinge” ovvero “forza” un livello logico alto), e in tale condizione si dovrà leggere un livello digitale uguale a “1” ovvero LIVELLO HIGH = ALTO. Quando il pulsante viene chiuso (premuta), si ottiene un cortocircuito tra i due contatti o terminali, che collegano il pin 7 direttamente a massa (GND), in tal

modo si otterrà una lettura dell'input sul pin 7 coincidente con uno "0" ovvero LIVELLO LOW = BASSO.

```

/* I.I.S. Primo LEVI – Torino Esercizio N. 2
Progetto: Digital Read Led 1
Autore: Questo è un esempio di pubblico dominio
Descrizione: Lettura di un input digitale (pulsante collegato al pin 7)
con accensione/spegnimento di un led (pin 13) e ripetizione del ciclo all'infinito.
Data: 03/12/2010 */
int led = 13; // definizione della variabile "led" utilizzata per scrivere sul pin 13
int pulsante = 7; // definizione della variabile "pulsante" utilizzata per leggere sul pin 7
int lettura_pulsante = 0; // definizione della variabile "lettura_pulsante" utilizzata per
// memorizzare lo stato del pulsante
void setup() // funzione di inizializzazione del INPUT/OUTPUT
{
  pinMode(led, OUTPUT); // inizializza il pin 13 come OUTPUT collegato al led
  pinMode(pulsante, INPUT); // Inizializza il pin 7 come INPUT collegato al pulsante n.a.
}
void loop() // programma principale (main) --> ciclo infinito (loop)
{ // acquisisci il valore dell'input pin 7 nella variabile "lettura_pulsante"
  lettura_pulsante = digitalRead(pulsante);
  // verifica se il pulsante è premuto (condizione VERA = pulsante n.a. NON PREMUTO)
  if (lettura_pulsante == HIGH)
  { // possibile altra istruzione alternativa alla precedente --> if (lettura_pulsante == )
    digitalWrite(led, HIGH); // spegni il LED collegato al pin 13 della scheda Arduino
  }
  else
  {
    digitalWrite(led, LOW); // accendi il LED collegato al pin 13 della scheda Arduino
  }
}
}

```

Figura 10 - Esercizio N. 2 - Accensione di un led controllato da un pulsante di tipo n.a.

Esercitazione N. 3

Nome progetto: **DigitalReadSerial_1.ino**

Specifiche del progetto:

Il candidato sulla base delle proprie conoscenze implementi un software in linguaggio C opportunamente commentato per la scheda Arduino, in grado di gestire un pulsante di tipo n.a. (normalmente aperto) e un led con la trasmissione dei valori presenti sull'input del micro all'interfaccia seriale RS232.

Effettuare sulla basetta Breadboard esterna alla scheda Arduino i seguenti collegamenti:

- Collegare al pin 13 l'anodo del led LD1 (LED_A) con il catodo collegato verso massa (GND)
- Collegare al pin 2 un terminale del pulsante n.a. denominato S1 (PULS_A) con l'altro terminale collegato a massa (GND)
- Collegare al pin 2 un terminale della resistenza di pull-up da 10Kohm – 1/4w – 5% mentre l'altro terminale della resistenza dovrà essere collegato al +5V.

Questa esercitazione è simile alla precedente con la possibilità di fornire la lettura del pulsante tramite la visualizzazione sul Personal Computer con un apposito software che utilizza l'interfaccia seriale USB.

```

/* I.I.S. Primo LEVI - Torino
Esercizio N. 3          Data: 03/12/2010
Progetto: DigitalReadSerial_1   Autore: Questo è un esempio di pubblico dominio
Descrizione: Lettura di un input digitale (pulsante collegato al pin7)
con stampa del livello logico sulle porta seriale e ripetizione del ciclo all'infinito. */
void setup() // funzione di inizializzazione della seriale RS232
{
  pinMode(7, INPUT); // inizializza il pin 7 della scheda Arduino come INPUT (PULSANTE)
  digitalWrite(7, HIGH); // settaggio per la resistenza interna di pull-up da 10KOhm
  pinMode(13, OUTPUT); // inizializza il pin 13 della scheda Arduino come OUTPUT (LED)
  Serial.begin(9600); // inizializza la seriale RS232 con 9600 baud, 8 bit dati, nessuna parità e 1 bit di stop
}
void loop() // programma principale (main) --> ciclo infinito (.oop)
{
  int pulsante = digitalRead(7); // acquisisci il valore dell'input pin 7 nella variabile 'pulsante'
  if (pulsante == 0; // verifica se il pulsante è premuto (condizione VERA - pulsante n.a. PREMUTO)
  {
    Serial.print("Pulsante PREMUTO collegato al pin 7 --> Livello:");
    Serial.println(pulsante, DEC); // stampa sulla seriale il valore dell'input collegato al pulsante (pin 7)
    digitalWrite(13, HIGH); // accendi il LED forzando un livello ALTO sul pin 13
  }
  else // altrimenti se il pulsante non è premuto (condizione FALSA - pulsante n.a. NON PREMUTO)
  {
    Serial.print("Pulsante NON PREMUTO collegato al pin 7 --> Livello: "); // stampa sulla seriale
    Serial.println(pulsante, DEC); // stampa sulla seriale il valore dell'input collegato al pulsante (pin 7)
    digitalWrite(13, LOW); // spegni il LED forzando un livello BASSO sul pin 13
  }
}
}

```

Figura 43 - Esercizio N. 3

Azionando il pulsante S1 si può ottenere una finestra di output sulla seriale (da attivare con un

clic sull'icona ) simile a quella visualizzata in figura 44.

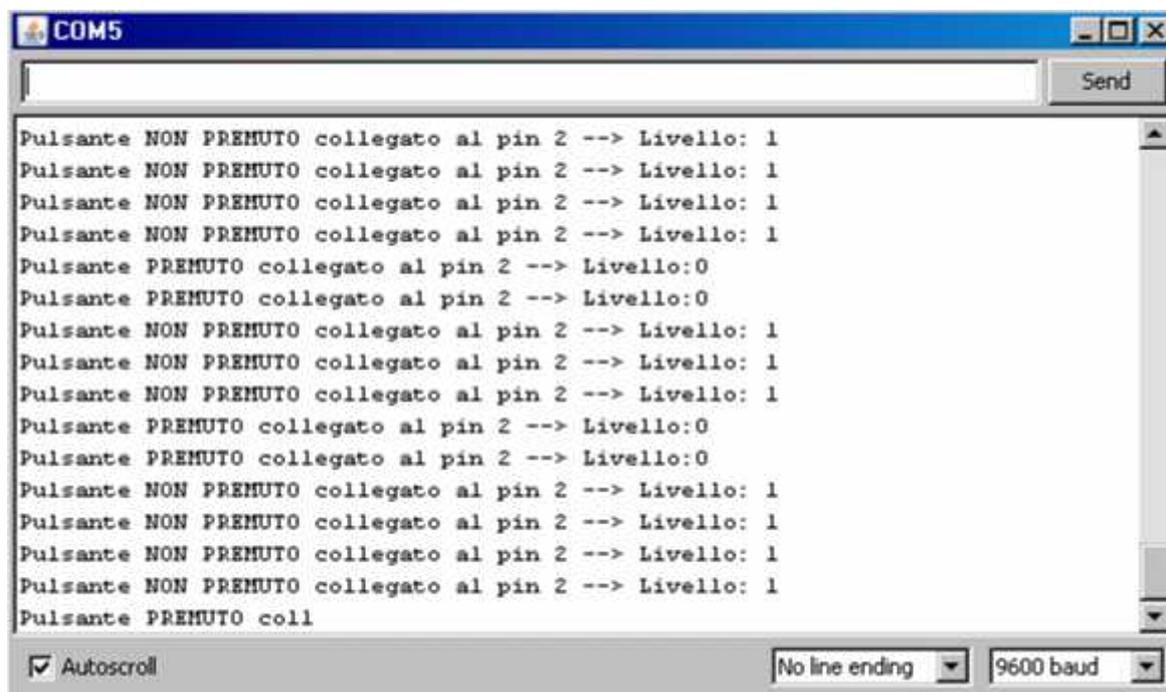


Figura 11 - Finestra del Serial Monitor (interfaccia seriale RS232 per stampare sul video del Personal Computer e/o inserire dei dati dalla tastiera)

Esercitazione N. 4

Nome progetto: **Alfabeto.ino**

Specifiche del progetto:

Il candidato sulla base delle proprie conoscenze implementi un software in linguaggio C opportunamente commentato per la scheda Arduino, in grado di effettuare la trasmissione dei caratteri ASCII compresi tra lo "0" (zero) e il carattere "Z" sull'interfaccia seriale RS232.

```

/* I.I.S. Primo LEVI - Torino
   Esercizio N. 4   Progetto: Alfabeto
   Autore: G. Carpignano
   Descrizione: effettuare la trasmissione dei caratteri ASCII compresi tra lo "0" (zero)
                e il carattere "Z" sull'interfaccia seriale RS232.
   Data: 03/12/2010 */
char carattere; // variabile per memorizzare un carattere ASCII
long i; // variabile per memorizzare loop di ritardo di circa 2 secondi
void setup() // funzione di inizializzazione della seriale RS232
{ // inizializza la seriale RS232 con 9600 baud, 8 bit dati, nessuna parità e 1 bit di stop
  Serial.begin(9600);
}
void loop() // programma principale (main) --> ciclo infinito (loop)
{
  // ciclo con inizio dal valore 30 Hex (coincidente con il carattere "0" zero)
  // al valore 5A Hex = 5B Hex - 1 (coincidente con il carattere "Z")
  for (carattere = 0x30; carattere < 0x5B; carattere++)
  {
    Serial.print(carattere); // trasmissione sulla seriale del carattere in codice ASCII
  }
  for (i=0; i<2000000; i++); // ciclo di ritardo di circa 2 secondi (loop)
  Serial.print("\n"); // trasmissione sulla seriale del carattere "new line"
                        // (nuova linea) in codice ASCII
}

```

Figura 45 - Esercizio N. 4

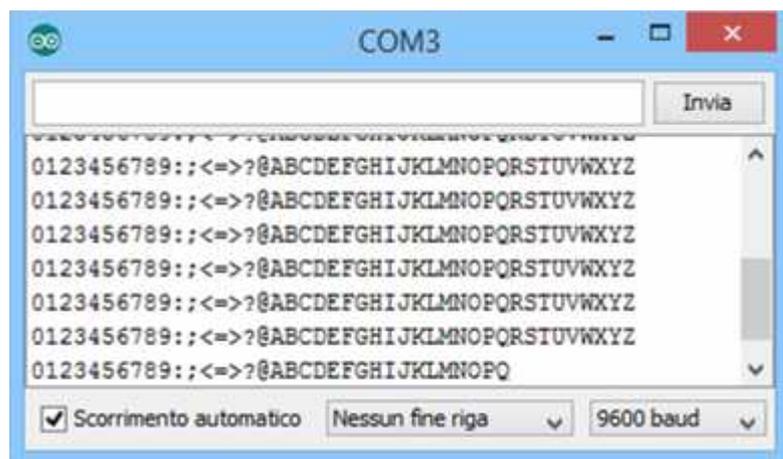


Figura 46 – Videata dell'esercizio N. 4

Esercitazione N. 5

Nome progetto: **stampa_formati.ino**

Specifiche del progetto:

Il candidato sulla base delle proprie conoscenze implementi un software in linguaggio C opportunamente commentato per la scheda Arduino, in grado di effettuare la trasmissione del carattere ASCII "!" (punto esclamativo) nei vari formati utilizzando l'interfaccia seriale RS232 (USB). Inoltre deve memorizzare e stampare il valore 211 nei vari formati di stampa disponibili: decimale, binario, esadecimale e ottale.

```

/* I.I.S. Primo LEVI - Torino Progetto: stampa_formati
   Descrizione: effettuare la trasmissione del carattere ASCII "!" nei vari formati
                utilizzando l'interfaccia seriale RS232 (USB). Memorizza e stampa il
                valore 211 nei vari formati: decimale, binario, esadecimale e ottale.
   Data: 03/12/2010 */
int test = 33; // valore numerico della tabella ASCII
int numero_decimale = 211; // memorizza nella variabile un valore in formato decimale
int numero_binario = B11010011; // memorizza nella variabile un valore in formato binario
int numero_esadecimale = 0xD3; // memorizza nella variabile un valore in formato esadecimale
int numero_ottale = 0323; // memorizza nella variabile un valore in formato ottale
void setup() // funzione di inizializzazione della seriale RS232
{ // inizializza la seriale RS232 con 9600 baud, 8 bit dati, nessuna parità e 1 bit di stop
  Serial.begin(9600);
}
void loop() // programma principale (main) --> ciclo infinito (loop)
{
  Serial.println(test); // stampa i caratteri "33". Di default è il valore DECIMALE
  Serial.write(test); // stampa il carattere ascii "!"
  Serial.println(); // stampa un CR (Carriage Return) e un LF (Line Feed).
  Serial.println(test, DEC); // stampa i caratteri "33".
  Serial.println(test, HEX); // stampa i caratteri "D3". Valore in esadecimale (base 16)
  Serial.println(test, OCT); // stampa i caratteri "323". Valore in ottale (base 8);
  Serial.println(test, BIN); // stampa i caratteri "110011". Valore in binario (base 2)
  Serial.print("Valore decimale 211 coincide in decimale: ");
  Serial.println(numero_decimale); // stampa num. decimale 211
  Serial.print("Valore binario 11010011 coincide in decimale: ");
  Serial.println(numero_binario); // stampa num. decimale 211
  Serial.print("Valore esadecimale 0xD3 coincide in decimale: ");
  Serial.println(numero_esadecimale); // stampa num. decimale 211
  Serial.print("Valore ottale 0323 coincide in decimale: ");
  Serial.println(numero_ottale); // stampa num. decimale 211
  while(1); // ciclo infinito per bloccare il programma
}

```

Figura 47 - Esercizio N. 5

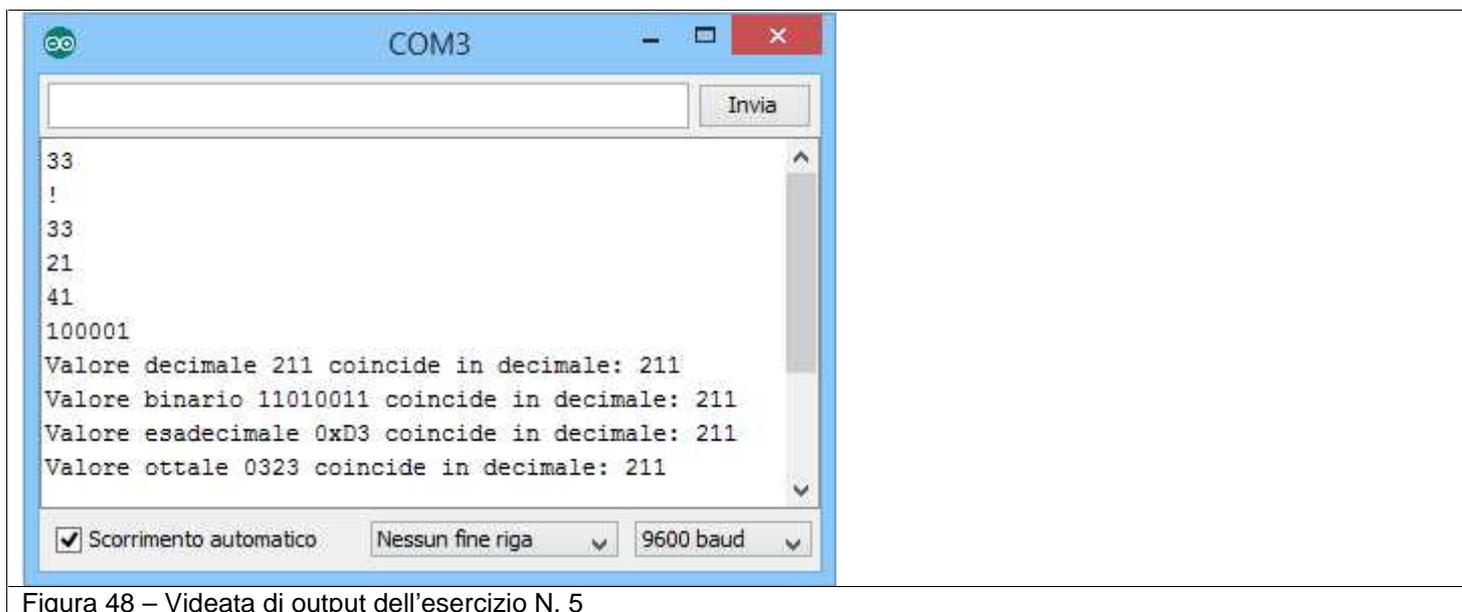


Figura 48 – Videata di output dell'esercizio N. 5

Come risulta evidente dal software l'istruzione "**Serial.write(test);**" produce un output sulla seriale di un solo carattere ASCII che corrisponde al "!", ovvero al punto esclamativo. Questo è coincidente con il valore decimale "33" nella tabella ASCII.

Esercitazione N. 6

Nome progetto: **Max_count_int.ino**

Specifiche del progetto:

Il candidato sulla base delle proprie conoscenze implementi un software in linguaggio C opportunamente commentato per la scheda Arduino, in grado di effettuare la stampa del superamento della massima capacità di conteggio di una variabile di tipo "int". Le variabili di tipo "int" possono memorizzare i numeri interi compresi tra -32768 e +32767.

```

/* I.I.S. Primo LEVI - Torino          Data: 10/07/2014
   Progetto: Max_count_int.           Autore: G. Carpignano
   Descrizione: Superamento della massima capacità di conteggio di una variabile di tipo INT
Le variabili di tipo "int" possono memorizzare i numeri interi compresi tra -32768 e +32767 */
void setup() // funzione di inizializzazione della seriale USB
{
  Serial.begin(9600); // inizializza la seriale a 9600 baud
}
void loop() // programma principale (main) --> ciclo infinito (loop)
{
  int x = 32767; // definisci X come variabile di tipo "int" con valore 32767
  Serial.println("Valore iniziale memorizzato della variabile X: ");
  Serial.println(x, DEC); // stampa in decimale il numero 32767
  x++; // corrisponde all'incremento unitario ovvero x = x + 1
  Serial.println("\nIncrementa di uno la variabile X");
  Serial.println("\nValore finale memorizzato della variabile X: ");
  Serial.println(x, DEC); // stampa in decimale il numero -32768
  x = -32768; // valore iniziale della variabile x
  Serial.println("\nNuovo valore iniziale memorizzato della variabile X: ");
  Serial.println(x, DEC); // stampa in decimale il numero -32768
  x--; // corrisponde al decremento unitario ovvero x = x - 1
  Serial.println("\nDecrementa di uno la variabile X");
  Serial.println("\nValore finale memorizzato della variabile X: ");
  Serial.println(x, DEC); // stampa in decimale il numero 32767
  while(1); // esegui una sola volta il software
}

```

Figura 49 - Esercizio N. 6



Figura 50 – Videata di output dell'esercizio N. 6

Esercitazione N. 7

Nome progetto: **ipotenusa.ino**

Specifiche del progetto:

Il candidato sulla base delle proprie conoscenze implementi un software in linguaggio C per la scheda Arduino opportunamente commentato, in grado di calcolare l'ipotenusa di un triangolo rettangolo.

In formula si avrà: $h = \sqrt{a^2 + b^2}$

```

/* I.I.S. Primo LEVI - Torino   Progetto: ipotenusa   Autore: Giorgio Carpignano
   Descrizione: Il candidato sulla base delle proprie conoscenze implementi un software in linguaggio C per
   la scheda Arduino opportunamente commentato, in grado di calcolare l'ipotenusa di un triangolo rettangolo.
   a^2 + b^2 = h^2           da cui si ottiene: h = radice_quadrata(a^2 + b^2)           Data: 01/03/2012   */
float a = 3.1; // definisci il lato a uguale a 3.1 cm (variabile con la virgola)
float b = 4.1; // definisci il lato b uguale a 4.1 cm (variabile con la virgola)
float h; // definisci l'ipotenusa come variabile con la virgola
void setup() // funzione di inizializzazione viene eseguita una sola volta all'inizio
{ // Inizializza la seriale RS232 con 9600 baud, 8 bit dati, nessuna parità e 1 bit di stop
  Serial.begin(9600);
  Serial.println("Calcolo dell'ipotenusa");
  Serial.print("Lato a = ");
  Serial.println(a);
  Serial.print("Lato b = ");
  Serial.println(b);
  // la funzione sqrt() calcola la radice quadrata, mentre pow() calcola la potenza di un numero
  h = sqrt( pow(a, 2) + pow(b, 2)); // calcolo ipotenusa
  Serial.print("Ipotenusa h = ");
  Serial.println(h);
}

void loop() // programma principale (main) --> ciclo infinito (loop)
{
  // non eseguire nulla nel ciclo infinito
}

```

Figura 51 – Esercizio N. 7

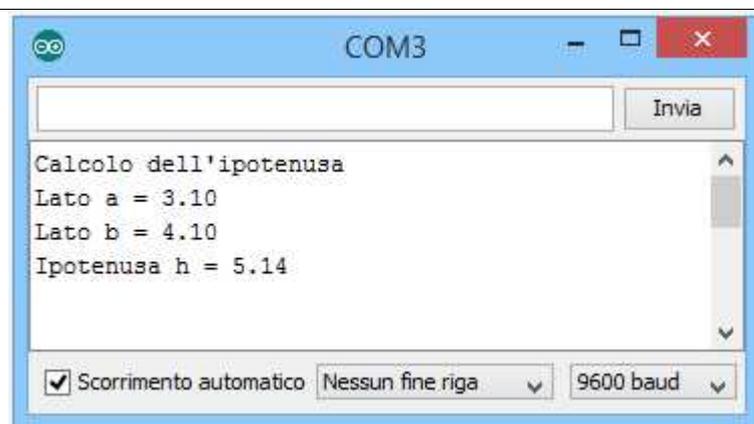


Figura 52 – Videata di output dell'esercizio N. 7

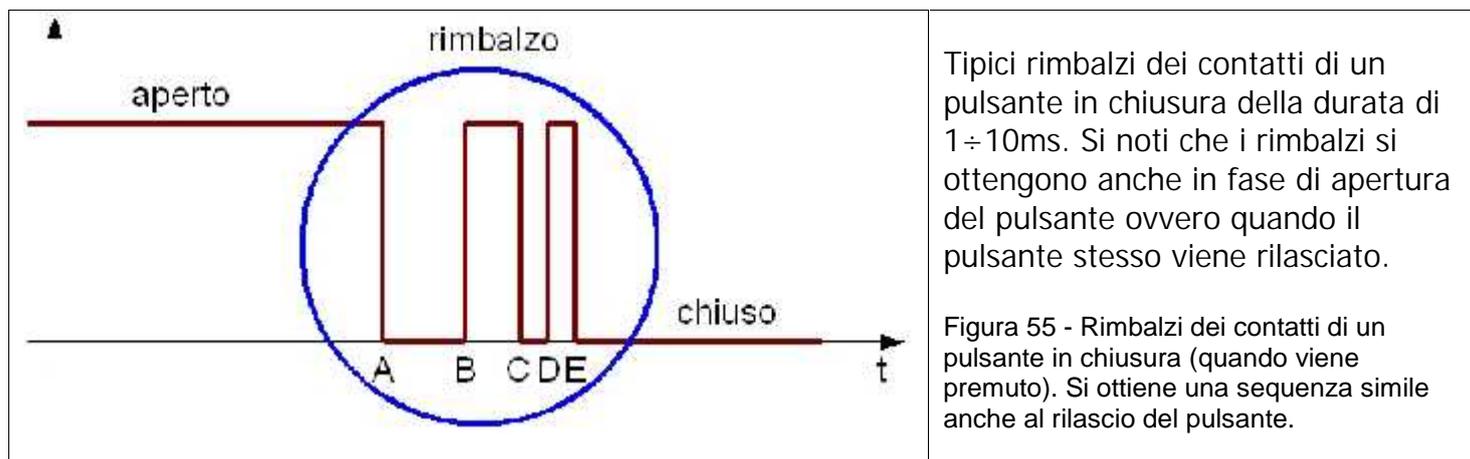
Esercitazione N. 8

Nome progetto:

Pulsante_Antiribalzo.ino

Specifiche del progetto:

Il candidato sulla base delle proprie conoscenze implementi un software in linguaggio C opportunamente commentato per la scheda Arduino, in grado di gestire un pulsante di tipo n.a. (normalmente aperto) e un led con la trasmissione dei valori presenti sull'input del micro all'interfaccia seriale RS232 eliminando i rimbalzi dei contatti sia in fase di chiusura che in fase di apertura del pulsante.



La durata del rimbombo dipende principalmente dai seguenti fattori:

1. il tipo di contatto;
2. la presenza di una molla di richiamo;
3. la forza con cui viene azionato;
4. la presenza di archi elettrici;

Per le applicazioni a microcontrollore si può considerare concluso un rimbombo quando il contatto assume valore stabile entro un periodo di tempo compreso tra 1 e 10 ms.

Gli stati intermedi (figura 48) che si manifestano in successione durante la chiusura del contatto, rischiano di essere acquisiti dal microcontrollore fornendo un'informazione non corretta al software che gestisce l'applicazione; per questo motivo il rimbombo va eliminato. Per evitare di aggiungere componenti esterni al microcontrollore (fatto salvo un eventuale resistore di pull-up o pull-down, qualora non si possano utilizzare quelli interni al micro), il metodo che verrà illustrato si basa esclusivamente sull'utilizzo di tecniche software.

Esercitazione N. 8

Nome progetto: **2Pulsanti+2Led.ino**

Specifiche del progetto:

Il candidato sulla base delle proprie conoscenze implementi un software in linguaggio C opportunamente commentato per la scheda Arduino, in grado di gestire i 2 pulsanti di tipo n.a. (normalmente aperto) e i 2 led.

Effettuare sulla basetta Breadboard i seguenti collegamenti:

- Collegare al pin 7 l'anodo del led LD1 (LED_A) con il catodo collegato verso massa (GND)
- Collegare al pin 8 l'anodo del led LD2 (LED_B) con il catodo collegato verso massa (GND)
- Collegare al pin 3 un terminale del pulsante n.a. denominato S1 (PULS_A) con l'altro terminale collegato a massa (GND)
- Collegare al pin 4 un terminale del pulsante n.a. denominato S2 (PULS_B) con l'altro terminale collegato a massa (GND)
- Collegare al pin 3 un terminale della resistenza di pull-up da 10Kohm – 1/4w – 5% mentre l'altro terminale della resistenza dovrà essere collegato al +5V.
- Collegare al pin 4 un terminale della resistenza di pull-up da 10Kohm – 1/4w – 5% mentre l'altro terminale della resistenza dovrà essere collegato al +5V.

In particolare se viene premuto il pulsante di tipo n.a. denominato S1 (PULS_A) e per tutto il tempo che verrà premuto, il corrispondente led LD1 (LED_A) si dovrà accendere mentre si spegnerà appena il pulsante viene rilasciato.

Analoga procedura per quanto riguarda il pulsante S2 (PULS_B) che accenderà o spegnerà il led LD2 (LED_B).

Si tenga presente che inizialmente i due led LD1 e LD2 dovranno essere spenti e che i rimbalzi dei contatti non sono da considerare.

Inoltre nel caso di pressione simultanea dei due pulsanti si consideri l'accensione di un solo led relativo alla pressione del primo pulsante effettivamente acquisito dal software.

Il montaggio dei componenti avviene su una Breadboard esterna alla scheda Arduino.

```

/* I.I.S. Primo LEVI – Torino   Progetto: Digital_Read_2led_2puls   Autore: Giorgio Carpinone
   Descrizione: Il candidato sulla base delle proprie conoscenze implementa un software in linguaggio C per
   la scheda Arduino opportunamente commentato, in grado di:
   1) Se viene premuto il pulsante P1 e per tutto il tempo che rimane premuto si deve accendere il led LD1.
   2) Se viene premuto il pulsante P2 e per tutto il tempo che rimane premuto si deve accendere il led LD2.
   3) All'accensione iniziale i due led LD1 e LD2 devono essere spenti.
   4) Se vengono premuti contemporaneamente i due pulsanti P1 e P2 si devono accendere entrambi i led LD1 e LD2
   con una frequenza di 10 Hz per tutto il tempo che rimangono contemporaneamente premuti. Data: 01/03/2012 */
byte pin_led_1 = 7; // definizione della variabile utilizzata per scrivere sul pin 7 (led 1)
byte pin_led_2 = 8; // definizione della variabile utilizzata per scrivere sul pin 8 (led 2)
byte pin_puls_1 = 3; // definizione della variabile utilizzata per leggere sul pin 3 (pulsante 1)
byte pin_puls_2 = 4; // definizione della variabile utilizzata per leggere sul pin 4 (pulsante 2)

void setup() { // funzione di inizializzazione dei INPUT/OUTPUT
  {
    pinMode(pin_led_1, OUTPUT); // inizializza il pin 7 come OUTPUT collegato alla L1
    pinMode(pin_led_2, OUTPUT); // inizializza il pin 8 come OUTPUT collegato alla L2
    digitalWrite(pin_led_1, LOW); // spegni la lampada 1
    digitalWrite(pin_led_2, LOW); // spegni la lampada 2
    pinMode(pin_puls_1, INPUT); // inizializza il pin 3 come INPUT collegato al pulsante n.a.
    pinMode(pin_puls_2, INPUT); // inizializza il pin 4 come INPUT collegato al pulsante n.a.
  }
}

void loop() // programma principale (main) --> ciclo infinito (loop)
{
  if((digitalRead(pin_puls_1) || digitalRead(pin_puls_2)) == LOW)
  { // se entrambi i pulsanti P1 e P2 sono premuti accendi i led con una frequenza di 10Hz
    digitalWrite(pin_led_1, HIGH); // accendi led 1
    digitalWrite(pin_led_2, HIGH); // accendi led 2
    delay(50); // ritardo di 50 millisecondi
    digitalWrite(pin_led_1, LOW); // spegni led 1
    digitalWrite(pin_led_2, LOW); // spegni led 2
    delay(50); // ritardo di 50 millisecondi
  }
  else // altrimenti controlla ogni singolo pulsante
  {
    if(digitalRead(pin_puls_1) == LOW)
    { // se pulsante 1 e' premuto
      digitalWrite(pin_led_1, HIGH); // accendi led 1
    }
    else // se pulsante 1 non e' premuto
    {
      digitalWrite(pin_led_1, LOW); // spegni led 1
    }
    if(digitalRead(pin_puls_2) == LOW)
    { // se pulsante 2 e' premuto
      digitalWrite(pin_led_2, HIGH); // accendi led 2
    }
    else // se pulsante 2 non e' premuto
    {
      digitalWrite(pin_led_2, LOW); // spegni led 2
    }
  }
}

```

```

}
}
} // fine loop()

```

Figura 56 – Esercizio N. 8

Esercitazione N. 9

Nome progetto: **Led_con_n_flash_controllato_da_tastiera_PC.ino**

Specifiche del progetto:

Il candidato sulla base delle proprie conoscenze implementi un software in linguaggio C opportunamente commentato per la scheda Arduino, in grado di effettuare l'accensione e lo spegnimento di un led da tastiera del Personal Computer tramite interfaccia seriale. Il led è collegato con l'Anodo al pin 13 e il catodo a massa e viene acceso un numero di volte compreso tra "1" e "9" digitando il corrispondente numero sulla tastiera del PC.

P.S. → utilizzare il "Serial Monitor" settato a 9600 baud e digitare il carattere seguito da un carattere "INVIO" oppure effettuando un clic sul pulsante denominato "Invia".

Si ricorda che il PC invia sulla seriale fittizia (USB) solo codici ASCII, quindi il carattere maiuscolo "A" verrà trasmesso sulla seriale e viene ricevuto dal codice presente sulla scheda Arduino come un codice 0x41 (valore espresso in esadecimale per il linguaggio C) che corrisponde al valore decimale 65. Pertanto il codice ASCII corrispondente al numero "0" è uguale a "0x30" in esadecimale (48 in decimale), mentre il numero "9" corrisponde a "0x39".

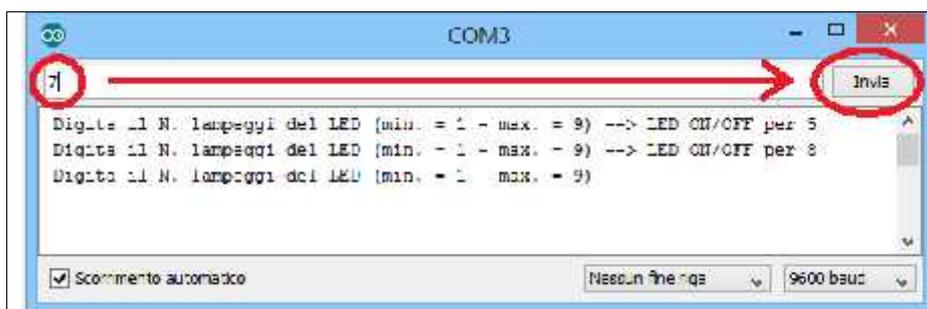


Figura 57 – Visualizzazione sul terminale dell'esercizio N. 9

```

/* I.I.S. Primo LEVI - Torino                               Data: 14/12/2012
Progetto: Led_con_n_flash_controllato_da_tastiera_PC      Autore: G. Carpignano
Descrizione:
Software per effettuare l'accensione e lo spegnimento di un led da tastiera del Personal Computer
tramite interfaccia seriale. Il led e' collegato con l'Anodo al pin 13 e il catodo a massa e viene
acceso un numero di volte compreso tra 1 e 9 digitando il corrispondente numero sulla tastiera del PC.
P.S. utilizzare il "Serial Monitor" settato a 9600 baud e digitare il carattere seguito
da un carattere INVIO oppure effettuando un clic sul pulsante denominato "Send".
Si ricorda che il PC invia sulla seriale fittizia (USB) solo codici ASCII, quindi il carattere
maiuscolo "A" verra' trasmesso sulla seriale viene ricevuto dal codice presente sulla scheda
Arduino come un codice 0x41 (valore espresso in esadecimale per il linguaggio C) che corrisponde
al valore decimale 65. */
int led = 13; // il led e' collegato con l'Anodo sul pin 13 e il Catodo a GND.
int leggi_byte;
void setup()
{
  pinMode(led, OUTPUT); // configura il pin 13 come output
  Serial.begin(9600); // inizializza la seriale a 9600 baud
  Serial.print(" Digita il N. lampeggi del LED (min. = 1 - max. = 9)");
}
void loop()
{
  if (Serial.available() > 0) // se e' presente sul buffer della seriale un carattere ASCII
  {
    leggi_byte = Serial.read(); // acquisisci il carattere della seriale e memorizzalo
    if ((leggi_byte > '0') && (leggi_byte <= '9')) // se il valore letto e' > 0 e <= 9
    {
      leggi_byte = leggi_byte - '0'; // converti valore da ASCII a valore numerico
      Serial.print(" --> LED ON/OFF per ");
      Serial.println(leggi_byte, DEC); // ritrasmetti il numero sulla seriale
      // ripeti la sequenza di accensione/spegnimento per il numero inserito da tastiera del PC
      for (int i = 0; i < leggi_byte; i++)
      {
        digitalWrite(led, HIGH); // accendi il led
        delay (500); // ritardo di 0,5 secondi
        digitalWrite(led, LOW); // spegni il led
        delay (500); // ritardo di 0,5 secondi
      }
      Serial.print(" Digita il N. lampeggi del LED (min. = 1 - max. = 9)");
    }
  }
}

```

Figura 58 – Esercizio N. 9

Esercitazione N. 10

Nome progetto: **Quiz.ino**

Specifiche del progetto:

Il candidato sulla base delle proprie conoscenze implementi un software in linguaggio C opportunamente commentato per la scheda Arduino, in grado di ottenere un "indicatore di primo evento" per utilizzatori di un quiz. Si consideri 4 concorrenti che possiedono ognuno un pulsante da premere il più velocemente possibile dopo la lettura di una domanda. La persona che preme per prima il pulsante determina l'accensione del led relativo ed escluderà l'accensione dei led dei concorrenti. Si richiede inoltre un pulsante per spegnere tutti i led.

Effettuare sulla basetta Breadboard i seguenti collegamenti:

- Collegare al pin 6 l'anodo del led LD1 (LED_UTENTE_A) con il catodo collegato verso massa (GND)
- Collegare al pin 7 l'anodo del led LD2 (LED_UTENTE_B) con il catodo collegato verso massa (GND)
- Collegare al pin 8 l'anodo del led LD3 (LED_UTENTE_C) con il catodo collegato verso massa (GND)
- Collegare al pin 9 l'anodo del led LD4 (LED_UTENTE_D) con il catodo collegato verso massa (GND)
- Collegare al pin 2 un terminale del pulsante n.a. denominato S1 (PULS_UTENTE_A) con l'altro terminale collegato a massa (GND)
- Collegare al pin 3 un terminale del pulsante n.a. denominato S2 (PULS_UTENTE_B) con l'altro terminale collegato a massa (GND)
- Collegare al pin 4 un terminale del pulsante n.a. denominato S3 (PULS_UTENTE_C) con l'altro terminale collegato a massa (GND)
- Collegare al pin 5 un terminale del pulsante n.a. denominato S4 (PULS_UTENTE_D) con l'altro terminale collegato a massa (GND)
- Collegare al pin 2 un terminale della resistenza di pull-up da 10Kohm – 1/4w – 5% mentre l'altro terminale della resistenza dovrà essere collegato al +5V.
- Collegare al pin 3 un terminale della resistenza di pull-up da 10Kohm – 1/4w – 5% mentre l'altro terminale della resistenza dovrà essere collegato al +5V.
- Collegare al pin 4 un terminale della resistenza di pull-up da 10Kohm – 1/4w – 5% mentre l'altro terminale della resistenza dovrà essere collegato al +5V.
- Collegare al pin 5 un terminale della resistenza di pull-up da 10Kohm – 1/4w – 5% mentre l'altro terminale della resistenza dovrà essere collegato al +5V.

In particolare si ricorda che non viene utilizzato nessun altro pulsante per il “reset” del circuito perché verrà sfruttato il pulsante di RESET della scheda Arduino per spegnere tutti i led.

```

/* I.I.S. Primo LEVI - Torino
   Data: 13/12/2010
   Progetto: Quiz      Autore: G. Carpianno
   Descrizione:
   Software per un "indicatore di primo evento" per utilizzatori di un quiz.
   Si consideri 4 concorrenti che possiedono ognuno un pulsante da premere
   il più velocemente possibile dopo la lettura di una domanda. La persona
   che preme per prima il pulsante determina l'accensione del led relativo
   ed escluderà l'accensione del led dei concorrenti. Si richiede inoltre un
   pulsante per spegnere tutti i led.  */

int led_utente_a = 5; // led dell'UTENTE A collegato al pin 6
int puls_utente_a = 2; // pulsante dell'UTENTE A da collegare sul pin 2
int led_utente_b = 7; // led dell'UTENTE B collegato al pin 7
int puls_utente_b = 3; // pulsante dell'UTENTE B da collegare sul pin 3
int led_utente_c = 8; // led dell'UTENTE C collegato al pin 8
int puls_utente_c = 4; // pulsante dell'UTENTE C da collegare sul pin 4
int led_utente_d = 9; // led dell'UTENTE D collegato al pin 9
int puls_utente_d = 5; // pulsante dell'UTENTE D da collegare sul pin 5

byte lettura_utenti = 0; // definizione della variabile "lettura_utenti" utilizzata per
// memorizzare lo stato di tutti gli utenti
byte memoria_utente = 0; // resetta il flag del primo evento dell'utente che ha premuto per primo

void setup() // funzione di inizializzazione dei INPUT/OUTPUT
{

```

```

pinMode(puls_utente_a, INPUT); // inizializza il pin 2 come INPUT collegato al pulsante n.a. dell'UTENTE A
pinMode(led_utente_b, OUTPUT); // inizializza il pin 7 come OUTPUT collegato al led UTENTE B
pinMode(puls_utente_b, INPUT); // inizializza il pin 3 come INPUT collegato al pulsante n.a. dell'UTENTE B
pinMode(led_utente_c, OUTPUT); // inizializza il pin 8 come OUTPUT collegato al led UTENTE C
pinMode(puls_utente_c, INPUT); // inizializza il pin 4 come INPUT collegato al pulsante n.a. dell'UTENTE C
pinMode(led_utente_d, OUTPUT); // inizializza il pin 9 come OUTPUT collegato al led UTENTE D
pinMode(puls_utente_d, INPUT); // inizializza il pin 5 come INPUT collegato al pulsante n.a. dell'UTENTE D
}
//
void loop()
{
  // leggi tutti i pulsanti degli utenti in contemporanea e memorizza in un solo byte
  lettura_utenti = 2*3*digitalRead(puls_utente_d) - 2*2*digitalRead(puls_utente_c)
  + 2*1*digitalRead(puls_utente_b) + 2*0*digitalRead(puls_utente_a);
  switch(lettura_utenti)
  {
    case 0x0E: // solo l'UTENTE A ha premuto il pulsante (in 00001111 Binario)
      if (memoria_utente == 0)
      {
        digitalWrite(led_utente_a, HIGH); // accendi il led dell'UTENTE A
        memoria_utente = 1; // setta il flag del primo evento dell'utente che ha premuto per primo
      }
      break;
    case 0x0D: // solo l'UTENTE B ha premuto il pulsante (in 00001101 Binario)
      if (memoria_utente == 0)
      {
        digitalWrite(led_utente_b, HIGH); // accendi il led dell'UTENTE B
        memoria_utente = 1; // setta il flag del primo evento dell'utente che ha premuto per primo
      }
      break;
    case 0x0B: // solo l'UTENTE C ha premuto il pulsante (in 00001011 Binario)
      if (memoria_utente == 0)
      {
        digitalWrite(led_utente_c, HIGH); // accendi il led dell'UTENTE C
        memoria_utente = 1; // setta il flag del primo evento dell'utente che ha premuto per primo
      }
      break;
    case 0x07: // solo l'UTENTE D ha premuto il pulsante (in 00000111 Binario)
      if (memoria_utente == 0)
      {
        digitalWrite(led_utente_d, HIGH); // accendi il led dell'UTENTE D
        memoria_utente = 1; // setta il flag del primo evento dell'utente che ha premuto per primo
      }
      break;
    default: // nessun pulsante premuto da parte degli utenti (pulsanti di tipo n.a.) oppure più pulsanti premuti
      // non effettuare nulla
      break;
  }
}

```

Figura 59 – Esercizio N. 10

Esercitazione N. 11

Nome progetto: **stringhe_1.ino**

Specifiche del progetto:

Il candidato sulla base delle proprie conoscenze implementi un software in linguaggio C opportunamente commentato per la scheda Arduino, in grado di ottenere la stampa del testo "Ciao1", "Ciao2", etc. o della stringa sulla seriale USB. Il carattere "\0" è corrispondente al carattere di fine della stringa.

Le stringhe devono essere terminate con un byte di fine stringa (valore zero).

L'inserimento del byte a zero è automatico nel caso in cui si utilizzano le doppie virgolette.

Occorre però ricordarsi di aggiungere un byte nel dimensionamento dell'array.

Si noti la stampa doppia della stringa di testo “Ciao4” che viene prima ottenuta con l'istruzione “**Serial.write(testo4);**” ed in seguito con l'istruzione “**Serial.print(testo4);**”. La stampa con l'istruzione “**Serial.print(testo5);**” non risulta possibile a causa del tipo di variabile utilizzata, cioè della variabile “**byte**” invece della variabile “**char**”.

```

/*  I.I.S. Primo LEVI - Torino
    Progetto: stringhe_1.pde    Autore: G. Carpignano
    Descrizione: Stampa del testo "Ciao1", "Ciao2", etc. o stringa sulla seriale USB.
    Il carattere "\0" e' corrispondente alla fine della stringa.
    Data: 03/02/2012  */
char testo1[6] = {'C', 'i', 'a', 'o', '1', '\0'}; // stringa con terminazione
char testo2[] = "Ciao2"; // stringa senza terminazione (viene chiusa in automatico dal programma)
char testo3[6] = "Ciao3"; // stringa senza terminazione (viene chiusa in automatico dal programma)
char testo4[] = {67, 105, 97, 111, 52, 0}; // stringa con terminazione (valori decimali)
byte testo5[] = { 'C', 'i', 'a', 'o', '5', '\0'}; // stringa con terminazione
void setup() // funzione di configurazione dei Input/Output
{ // inizializza la seriale RS232 con 9600 baud
  Serial.begin(9600);
}
void loop() // programma principale (main) --> ciclo infinito (loop)
{
  Serial.write(testo1); // stampa dell'intera stringa
  Serial.println(); // stampa di un CR (carriage Return) e LF (line Feed)
  Serial.write(testo2); // stampa dell'intera stringa
  Serial.println(); // stampa di un CR (carriage Return) e LF (line Feed)
  Serial.write(testo3); // stampa dell'intera stringa
  Serial.println(); // stampa di un CR (carriage Return) e LF (line Feed)
  Serial.write(testo4); // stampa dell'intera stringa
  Serial.println(); // stampa di un CR (carriage Return) e LF (line Feed)
  Serial.print(testo4); // stampa dell'intera stringa
  Serial.println(); // stampa di un CR (carriage Return) e LF (line Feed)
  for(int x=0; x<5; x++) // ciclo con il numero di caratteri da stampare
  {
    Serial.write(testo5[x]); // stampa del singolo carattere ASCII
  }
  while (1); // loop infinito (blocca il micro)
}

```

Figura 60 – Esercizio N. 11

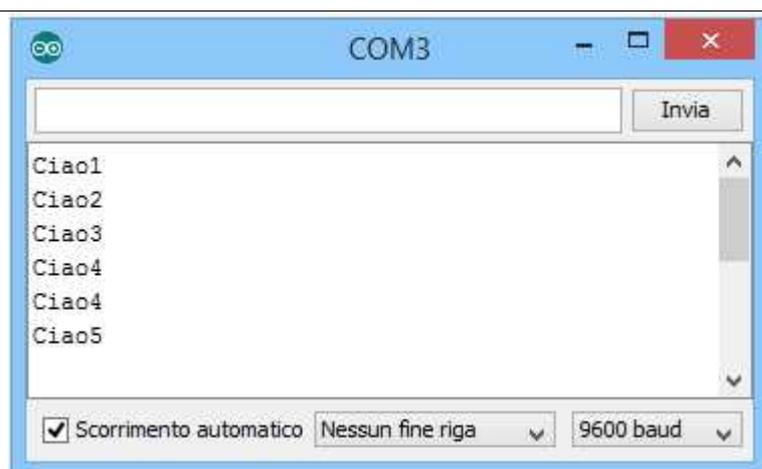


Figura 61 – Visualizzazione sul terminale dell'esercizio N. 11