ARDUINO
IL LIBRO DEI PROGETTI

ARDUINO - IL LIBRO DEI PROGETTI

AUTORI
I progetti e i testi sono di Scott Fitzgerald e Michael Shiloh
Con il contributo di Tom Igoe

DESIGN E ART DIRECTION
TODO
Giorgio Olivero, Mario Ciardulli, Vanessa Poli, Michelle Nebiolo
todo.to.it

FABBRICAZIONE DIGITALE E PROJECT MANAGEMENT
Officine Arduino Torino
Katia De Coi, Enrico Bassi

ADVISOR E SUPPORTER
Massimo Banzi, Gianluca Martino, Smart Projects

TEST DEI PROGETTI E REVISIONI DEL TESTO
Michael Shiloh, Michelle Nebiolo, Katia De Coi, Alessandro Buat, Federico Vanzati, David Mellis

TRADUZIONE E IMPAGINAZIONE DELLA VERSIONE ITALIANA
Luisa Castiglioni, pxess-office.co

GRAZIE
Un grande ringraziamento va all'intera comunità di Arduino per i continui contributi, il supporto e il feedback.
Un grazie speciale va al team di Fritzing: alcune illustrazioni di componenti elettronici usate nel libro sono state prese o modificate dal progetto open-source Fritzing (www.fritzing.org).
Un ringraziamento di cuore va a Paul Badger per la libreria CapacitiveSensor usata nel progetto 13.

Progettato, stampato e rilegato a Torino, Italia
Giugno 2013

INDICE

00

Tutti, ogni giorno, utilizzano la tecnologia. La maggior parte di noi lascia la programmazione agli ingegneri perché pensa che codice e elettronica siano complicati e difficili; in realtà, possono essere divertenti ed emozionanti. Grazie ad Arduino designer, artisti, hobbisti e studenti di tutte le età stanno imparando a creare cose che si accendono, si muovono, rispondono a persone, animali, piante e al resto del mondo.

Nel corso degli anni Arduino è stato utilizzato come "cervello" in migliaia di progetti, uno più creativo dell'altro. Una comunità mondiale di maker si è raccolta intorno a questa piattaforma open source, passando dal personal computer alla personal fabrication, e contribuendo a un nuovo mondo fatto di partecipazione, cooperazione e condivisione.

Arduino è aperto e semplice. È fondato su lezioni che abbiamo imparato insegnando ai nostri studenti: basta partire dal presupposto che imparare a utilizzare le tecnologie digitali è semplice e accessibile. Così l'elettronica e il codice diventano strumenti creativi che chiunque può utilizzare – come pennelli e colori. Questo libro ti guida attraverso le basi in maniera pratica attraverso progetti creativi per imparare facendo. Una volta apprese le basi, avrai una gamma di software e circuiti da utilizzare per creare qualcosa di bello e per inventare qualcosa di divertente.

BENVENUTO NEL MONDO DI ARDUINO!

ARDUINO RENDE FACILE PROGRAMMARE PICCOLI COMPUTER CHIAMATI MICROCONTROLLORI, CHE SONO CIÒ CHE RENDE INTERATTIVI GLI OGGETTI

Ogni giorno ne sei circondato: sono all'interno di timer, termostati, giocattoli, telecomandi, forni a microonde e anche in alcuni spazzolini da denti. Semplicemente svolgono un compito specifico e se quasi non li si nota – come spesso avviene – è perché lo stanno facendo bene. Sono stati programmati per sentire e controllare attraverso sensori e attuatori.

I sensori ascoltano il mondo fisico. Convertono l'energia in segnali elettrici che si dà loro quando si premono pulsanti, si agitano le braccia o si grida. Pulsanti e manopole sono sensori che si toccano con le dita, ma ci sono molti altri tipi di sensori.

Gli attuatori agiscono nel mondo fisico. Convertono l'energia elettrica in energia fisica, come la luce, il calore e il movimento.

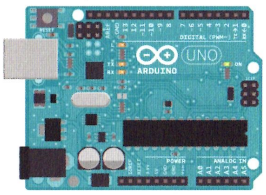
I microcontrollori ascoltano i sensori e parlano con gli attuatori. Decidono cosa fare in base al programma che scrivi.

I microcontrollori e l'elettronica sono solo lo scheletro dei tuoi progetti, però. Dovrai utilizzare competenze che probabilmente già possiedi per dare consistenza alle tue idee. Per esempio, in uno dei progetti che ti proponiamo, potrai costruire una freccia e collegarla a un motore e mettere entrambi in una scatola con una manopola per creare un indicatore con cui dire alla gente se sei occupato o no. In un altro, metterai alcune luci e un sensore di inclinazione su un cartoncino per fare una clessidra.

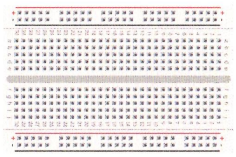
Arduino può rendere reattivo il tuo progetto, ma solo tu lo puoi fare bello. Ti forniremo alcuni suggerimenti su come farlo.

Arduino è stato progettato per ottenere risultati. Per realizzare questo obiettivo, abbiamo deciso di offrirti unicamente le nozioni di base su programmazione e elettronica necessarie per iniziare i progetti. Se desideri saperne di più su questi aspetti, ci sono molte buone guide disponibili. Forniremo un paio di riferimenti, altri si possono trovare online: arduino.cc/starterkit

PARTI NEL KIT

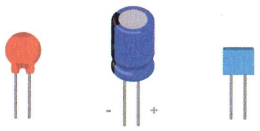


Arduino Uno - Il microcontrollore sarà al centro dei tuoi progetti. Si tratta di un semplice computer, ma uno con il quale non hai ancora avuto a che fare. Costruirai i circuiti e le interfacce per l'interazione e dirai al microcontrollore come comportarsi con gli altri componenti.

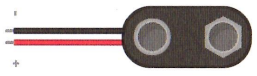


Breadboard - Una base su cui costruire i tuoi circuiti elettronici. È un pannello di connessione, con file di fori che consentono di collegare insieme cavi e componenti. Oltre il modello senza saldatura del kit, sono disponibili versioni che richiedono saldature (basette millefori).

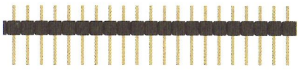
Cavo USB - Permette di collegare Arduino Uno al computer per la programmazione. Inoltre fornisce l'alimentazione ad Arduino per la maggior parte dei progetti presenti nel kit.



Condensatori - Questi componenti immagazzinano e rilasciano energia elettrica in un circuito. Quando la tensione del circuito è superiore di quella che viene immagazzinata nel condensatore, la corrente fluisce, dando una carica al condensatore. Quando la tensione del circuito è inferiore, la carica immagazzinata viene rilasciata. Spesso sono posizionati tra l'alimentazione e la massa vicino a un sensore o a un motore per compensare le fluttuazioni di tensione.

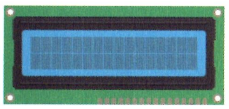


Connettore per batteria - Collega una batteria da 9V ai cavi di alimentazione, può essere facilmente collegato a una breadboard o ad Arduino.



Connettori a pettine - Questi piedini si infilano nei connettori femmina, come quelli sulla breadboard. Facilitano le connessioni.

Diodo - Assicura che l'elettricità scorra solo in una direzione. Utile quando nel circuito si ha un motore o un carico ad alta corrente/tensione. I diodi sono polarizzati: significa che è importante la direzione in cui sono inseriti nel circuito. Collocati in un modo, permettono il passaggio di corrente attraverso di essi. Inseriti nella direzione opposta, la bloccano. Il lato anodo si collega generalmente al punto di tensione più elevata nel circuito. Il catodo si collega al punto di minore tensione o a massa. Il catodo è solitamente contrassegnato da una striscia su un lato del componente.

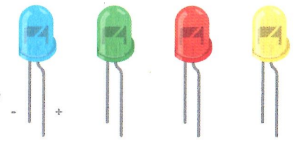


Display a cristalli liquidi (LCD) - Display alfanumerico o grafico a cristalli liquidi. Gli LCD sono disponibili in molte dimensioni, forme e stili. Nel kit ce n'è uno da 2 righe di 16 caratteri.

Filtri (rosso, verde, blu) - Filtrano diverse lunghezze d'onda della luce. Quando sono utilizzati in combinazione con le fotoresistenze, fanno reagire il sensore solo alla quantità di luce del colore filtrato.

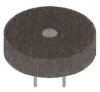
Fotoaccoppiatore - Consente di collegare due circuiti che non hanno l'alimentazione in comune. Internamente c'è un piccolo LED che, quando viene illuminato, attiva una fotocellula. Quando si applica tensione al piedino +, il LED si accende e l'interruttore interno si chiude. Le due uscite fungono da interruttore nel secondo circuito.

Fotoresistenza - (chiamata anche fotocellula). Componente la cui resistenza cambia in base alla quantità di luce che lo colpisce.



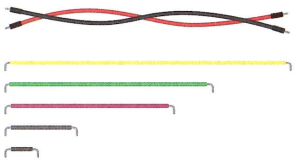
LED (Diodo Emettitore di Luce) - Un tipo di diodo che si illumina quando l'elettricità lo percorre. Come in tutti i diodi, l'elettricità scorre in una sola direzione attraverso questi componenti. Probabilmente li conosci come spie su molti dispositivi elettronici. L'anodo, che si collega all'alimentazione, è di solito il terminale più lungo, e il catodo il più corto.

Motore a corrente continua - Converte l'energia elettrica in energia meccanica quando la corrente elettrica viene applicata ai suoi terminali. Gli avvolgimenti all'interno del motore vengono magnetizzati quando vi fluisce la corrente. Questi campi magnetici attraggono e respingono magneti permanenti facendo ruotare l'albero. Se la direzione della corrente viene invertita, il motore gira in senso opposto.



Piezo - Un componente elettrico per rilevare le vibrazioni e produrre suoni.

Ponte H - Un circuito che permette di controllare la polarità della tensione applicata a un carico, in genere un motore. Il ponte H nel kit è un circuito integrato, ma potrebbe anche essere costruito con un certo numero di componenti separati.



Ponticelli - Da utilizzare per collegare tra loro i componenti sulla breadboard e ad Arduino.

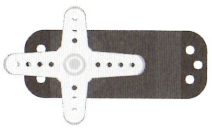
Potenziometro - Una resistenza variabile con tre piedini. Due dei piedini sono connessi alle estremità di una resistenza fissa. Il piedino centrale, o cursore, si muove lungo la resistenza, dividendola in due metà. Quando i piedini esterni del potenziometro sono collegati all'alimentazione e a massa, il piedino centrale fornisce una tensione proporzionale alla posizione della manopola.

Pulsanti - Interruttori che chiudono un circuito fintanto che vengono premuti. Si inseriscono facilmente nelle breadboard. Sono adatti per rilevare segnali acceso/spento.

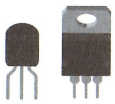
Resistenze - Resistono al flusso di energia elettrica in un circuito modificando di conseguenza la tensione e la corrente. I valori di resistenza sono misurati in Ohm (rappresentata dal carattere greco omega, Ω). Le strisce colorate delle resistenze indicano il loro valore (vedi tabella colori delle resistenze, a pagina 41).

Sensore di inclinazione - Un tipo di interruttore che si apre o si chiude a seconda del suo orientamento. Tipicamente sono cilindri cavi con all'interno una sfera di metallo che collega due contatti quando il sensore è in verticale.

Sensore di temperatura - Cambia la sua tensione di uscita in funzione della temperatura del componente. I piedini esterni si collegano all'alimentazione e a massa. La tensione sul piedino centrale è proporzionale alla temperatura.



Servomotore - Un tipo di motore che può ruotare solo di 180 gradi. Si controlla inviando impulsi elettrici da Arduino. Questi impulsi dicono al motore in quale posizione spostarsi.



Transistor - Un dispositivo con tre piedini che può funzionare da interruttore elettronico. Utile per il controllo di componenti ad alta tensione/alta corrente, come i motori. Un piedino si collega a massa, un altro al componente da controllare e il terzo ad Arduino. Quando il transistor riceve tensione sul piedino collegato ad Arduino chiude il circuito tra la massa e il carico.

TAVOLA DEI SIMBOLI

FILI COLLEGATI

TRANSISTOR

PULSANTE

FILI NON COLLEGATI

MOSFET

MOTORE

INTERRUTTORE DI INCLINAZIONE

FOTORESISTENZA

POTENZIOMETRO

RESISTENZA

DIODO

PIEZO

LED

CONDENSATORE

BATTERIA

CONDENSATORE POLARIZZATO

MASSA

In questo libro ti mostreremo i circuiti sia con illustrazioni realistiche sia con schemi elettrici.
Le illustrazioni ti daranno un'idea di che aspetto avrà la breadboard in una possibile implementazione del progetto. Gli schemi, invece, utilizzano i simboli per catturare l'essenza dei circuiti: presentano i componenti e i modi in cui sono connessi in forma chiara, concisa e non ambigua, ma non la loro organizzazione fisica. Schemi e simboli sono i modi con cui vengono rappresentati i circuiti elettronici. Quando esplorerai il mondo dell'elettronica scoprirai che alcuni libri e siti web forniscono solo schemi, quindi imparare a leggere i circuiti in questo modo è una preziosa competenza.
Qui trovi i simboli che useremo in tutto il libro.

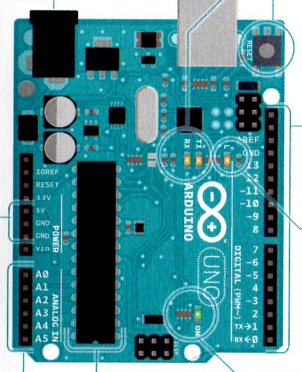
LA SCHEDA ARDUINO

Connettore di alimentazione
Serve per alimentare Arduino quando non è collegato a una porta USB. Vanno bene tensioni tra 7 e 12V.

Porta USB
Serve per alimentare l'Arduino Uno, caricare i tuoi sketch nella scheda e per comunicare con Arduino (via `Serial.println()`).

Pulsante di reset
Fa ripartire il microcontrollore ATmega.

LED TX e RX
Indicano le comunicazioni tra Arduino e il computer. Aspettati uno sfarfallio durante il caricamento dello sketch, così come durante la comunicazione seriale. Utile per il debug.



Piedini digitali
Utilizza questi piedini con `digitalRead()`, `digitalWrite()` e `analogWrite()`. Quest'ultimo funziona solo sui piedini con il simbolo PWM.

LED del piedino 13
L'unico attuatore integrato ad Arduino Uno. Oltre a essere l'oggetto del tuo primo sketch di lampeggio, questo LED è molto utile per il debug.

Piedini GND e 5V
Usa questi piedini per fornire un'alimentazione di 5V e massa al tuo circuito.

Microcontrollore ATmega
Il cuore di Arduino Uno.

Analog in
Usa questi piedini con `analogRead()`.

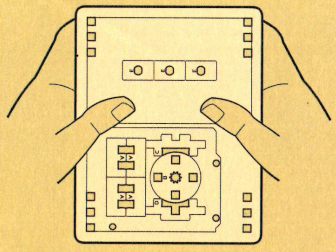
Power LED
Indica che Arduino sta ricevendo alimentazione. Utile per il debug.

Nel tuo Starter Kit hai trovato una base di legno pretagliata e facile da assemblare che renderà ancora più semplice il lavoro sui tuoi progetti – che siano o no parte di questo libro.

Per costruirla, togli la base di legno dalla scatola e segui le istruzioni a destra.

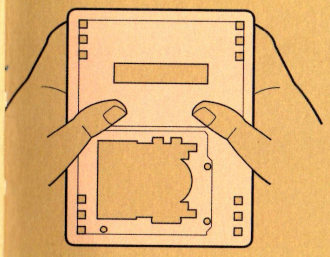
Fai attenzione a utilizzare solo le parti che vengono visualizzate, ma non perdere gli altri pezzi: ti serviranno per alcuni dei prossimi progetti.

Iniziamo!



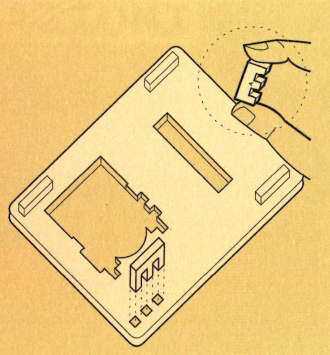
1

Prendi la base di legno e, con attenzione, separa i pezzi.



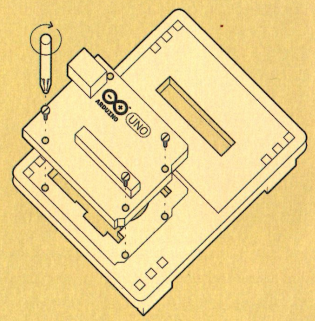
2

Continua fino a che hai separato tutti i pezzi.



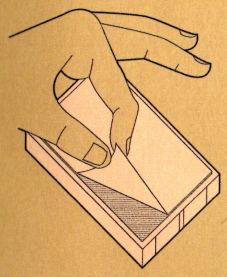
3

Metti i pezzi marchiati con una "A" nei buchi negli angoli, per creare i piedini della base.



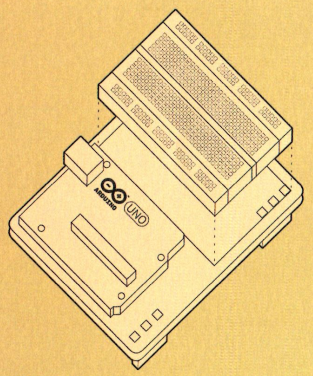
4

Fissa l'Arduino Uno alla base con 3 viti. Fai attenzione a non stringerle troppo.



5

Rimuovi accuratamente il foglio di protezione sul retro dalla breadboard.



6

Posiziona la breadboard sulla base di legno, accanto all'Arduino Uno.

COSE CHE TI SERVIRANNO

Batteria da 9V

Piccola sorgente di luce come una torcia elettrica

Materiale conduttivo come un foglio di alluminio o una maglia di rame

Carta colorata

Forbici

Un vecchio CD o DVD

Scotch e colla

Una scatola in cui poter fare dei buchi

Attrezzi di base come un cacciavite

Dispositivo elettronico alimentato a 9V
Qualsiasi dispositivo elettronico alimentato a batteria con almeno un interruttore o pulsante su cui puoi smanettare.

Saldatore
(necessario solo nel Progetto 15)

IMPOSTAZIONI

PRIMA DI INIZIARE A CONTROLLARE IL MONDO INTORNO A TE, DEVI SCARICARE L'IDE PER PROGRAMMARE LA TUA SCHEDA

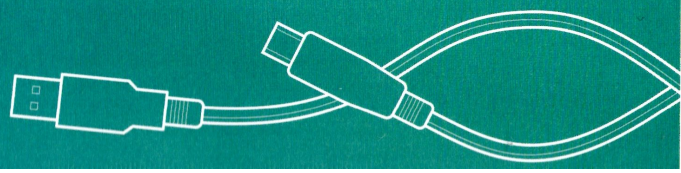
L'IDE di Arduino ti permette di scrivere programmi e caricarli in Arduino.

Scarica l'ultima versione dell'IDE da:
arduino.cc/download

Tieni la scheda Arduino e un cavo USB vicino al computer. Non collegarli ancora.

Segui le procedure indicate nelle prossime pagine per Windows, Mac OS X o Linux.

La versione online della guida è disponibile al link:
arduino.cc/guide



WINDOWS INSTALLAZIONE

Versione online
arduino.cc/windows

INSTRUZIONI PER: WINDOWS 7, VISTA, E XP

1. Terminato il download, fai doppio clic sul file "Install Arduino". Se appare una finestra di avviso di sicurezza, fai clic su "Esegui" o "Autorizza" e accetta il License Agreement. Clicca su "Next" per scegliere la cartella in cui installare l'IDE e su "Install". Alla fine del processo verranno creati i collegamenti all'IDE di Arduino sul Desktop e nel Menu Start.
2. Collega Arduino al computer con il cavo USB. La scheda viene automaticamente alimentata dalla connessione USB del computer e si accende il LED verde con la scritta ON. La scheda può essere anche alimentata da un alimentatore esterno.
3. Una volta collegata, Windows dovrebbe avviare automaticamente il processo di installazione del driver. Poiché il computer non è sempre in grado di trovare i driver autonomamente, dovrai indirizzarlo alla cartella corretta.
   Windows XP: Se ti viene chiesto di consentire a Windows Update la ricerca del software, seleziona "Si, solo in questa occasione" e poi "Installa da un elenco o percorso specifico";
   Windows Vista o 7: Se ti viene chiesto di installare il driver automaticamente o di cercarlo nel computer, su Windows 7, scegli la seconda opzione. Su Windows Vista, procedi direttamente al passaggio successivo selezionando la scelta raccomandata.
   Se l'installazione non dovesse avviarsi in automatico, clicca sul Menu Start e apri il pannello di controllo. Poi accedi alla gestione dispositivi seguendo questi percorsi:
   Windows XP: Passa alla visualizzazione classica -> Sistema -> Hardware -> Gestione Periferiche
   Windows Vista: Visualizzazione classica -> Gestione dispositivi
   Windows 7: Sistema e sicurezza -> Sistema -> Gestione dispositivi
4. Cerca il dispositivo Arduino sotto le categorie "Altre periferiche", "Altri dispositivi" o "Dispositivi sconosciuti" e, cliccando col tasto destro del mouse, seleziona "Aggiorna driver" o "Aggiornamento software driver".
5. Clicca su "Sfoglia", cerca la cartella di Arduino appena installata e seleziona la cartella "Drivers" (non la sottocartella "FTDI USB Drivers"). Premi "OK" e poi "Avanti" per continuare.
6. Windows ora installa il driver.
7. In "Gestione dispositivi", sotto la sezione "Porte (COM & LPT)", dovresti vedere una porta simile a "Arduino UNO (COM4)".

Congratulazioni! Hai installato l'IDE di Arduino nel tuo computer.

MAC OS X INSTALLAZIONE

Versione online
arduino.cc/mac

INSTRUZIONI PER: OS X 10.5 E SUCCESSIVE

1. Quando è termintao il download dell'IDE, fai doppio clic sul file .zip: espanderà l'applicazione Arduino.
2. Copia l'applicazione Arduino nella cartella delle applicazioni o dove vuoi installare il software.
3. Collega la tua Arduino al computer con il cavo USB. La scheda verrà automaticamente alimentata dalla connessione USB del computer e si accenderà il led verde con la scritta ON.
4. Non è necessario installare alcun driver per far funzionare correttamente la scheda.
5. A seconda della versione di OS X che stai usando, potrebbe aprirsi una finestra di dialogo che ti chiede di configurare le "Preferenze di sistema". Clicca sul bottone "Network" e poi su "Applica".
6. Nonostante venga visualizzata come un dispositivo "Non configurato", la Uno funzionerà ugualmente in maniera corretta. Esci dalle preferenze di sistema e...

Congratulazioni! Hai installato Arduino e sei pronto a realizzare i tuoi progetti.

LINUX INSTALLAZIONE

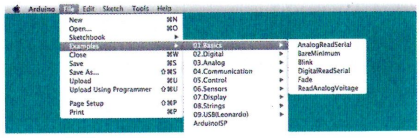
Se stai usando Linux, trovi le istruzioni a questo link:
arduino.cc/linux

COMUNICARE CON ARDUINO

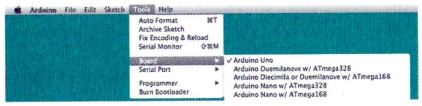
Ora che hai installato l'IDE di Arduino assicurati che il computer possa parlare con la scheda: è ora di verificare se puoi caricare un programma.

1. Fai doppio clic sull'applicazione di Arduino per aprirla. Se carichi l'IDE nella lingua sbagliata, puoi cambiarla nelle preferenze dell'applicazione. Cerca "Language Support" su questa pagina per i dettagli: ***arduino.cc/ide***

2. Vai allo sketch di esempio del LED che lampeggia ('sketch' è il modo con cui vengono chiamati i programmi di Arduino). Lo trovi sotto: ***FILE > EXAMPLES > 01.BASICS > BLINK***



3. Si dovrebbe aprire una finestra con del testo. Per ora lascia la finestra e seleziona la scheda sotto il menu: ***TOOLS > BOARD***



4. Scegli la porta seriale (COM) a cui è collegata Arduino dal menu ***TOOLS > SERIAL PORT***.

— *Su Windows.* Questa è probabilmente la COM con il numero più alto. Non succede niente se sbagli e se non funziona prova la successiva. Per scoprirlo, è possibile scollegare la scheda Arduino e riaprire il menu; la voce che scompare dovrebbe essere la scheda Arduino. Ricollega la scheda e seleziona la porta seriale.
— *Su Mac.* Dovrebbe essere qualcosa con dentro/dev/tty. usbmodem. Normalmente ce ne sono due; selezionane uno.

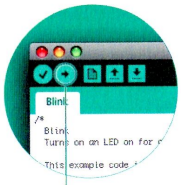


Fig. 1

5. Per caricare lo sketch Blink in Arduino, premi il pulsante ***UPLOAD*** nell'angolo in alto a sinistra della finestra. Vedi la Fig. 1.

6. Dovresti vedere una barra che indica il progresso dell'upload vicino all'angolo in basso a sinistra dell'IDE di Arduino e le luci TX e RX sulla scheda dovrebbero lampeggiare. Se l'upload è avvenuto con successo, l'IDE mostrerà il messaggio ***DONE UPLOADING***.

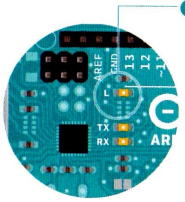


Fig. 2

7. Qualche secondo dopo la fine dell'upload, dovresti vedere il LED giallo accanto alla ***L*** iniziare a lampeggiare. Vedi la Fig. 2.
Se funziona, congratulazioni! Hai programmato con successo l'Arduino per far lampeggiare il suo LED!

Qualche volta la tua nuova Arduino è già programmata con lo sketch Blink, così non riesci a capire se sei veramente tu al comando. Se è questo il caso, cambia il tempo di `delay` mettendo 100 come numero tra parentesi e carica ancora lo sketch Blink. Ora il LED dovrebbe lampeggiare più velocemente.
Congratulazioni! Sei al comando! Ora vai al Progetto 01. (Non devi salvare nessun cambiamento che hai fatto.)

ALTRE INFORMAZIONI

Se hai problemi con i passaggi descritti in precedenza, guarda i suggerimenti di troubleshooting a questo link:
arduino.cc/trouble

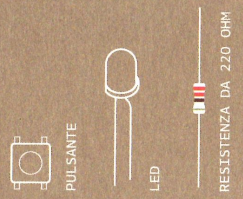
Mentre ti stai preparando a costruire i tuoi progetti, puoi guardare questa pagina per ulteriori informazioni sull'ambiente di programmazione di Arduino:
arduino.cc/ide

Potresti aver voglia di guardare anche:

— gli esempi per usare vari sensori e attuatori
arduino.cc/tutorial

— la reference per il linguaggio di Arduino
arduino.cc/examples

01



INGREDIENTI

CONOSCI I TUOI STRUMENTI

COSTRUIRAI UN SEMPLICE CIRCUITO CON ALCUNI PULSANTI, UN LED E UNA RESISTENZA

Scopri: la teoria elettrica di base, il funzionamento di una breadboard, i componenti in serie e in parallelo

Tempo: **30 MINUTI**
Livello: ■□□□□

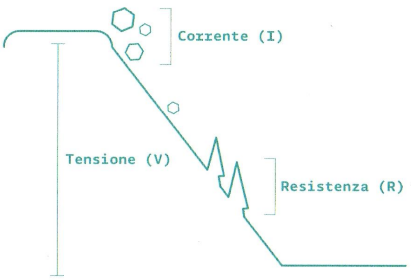
L'elettricità è un tipo di energia come il calore, la gravità o la luce. L'energia elettrica scorre lungo conduttori come i fili elettrici. Puoi convertire l'energia elettrica in altre forme di energia facendo cose interessanti, come accendere una luce o produrre dei suoni con un altoparlante.

I componenti che potresti usare per farlo, come gli altoparlanti o le lampadine, sono **trasduttori** elettrici. I trasduttori convertono gli altri tipi di energia in energia elettrica e viceversa. Ciò che converte le altre forme di energia in energia elettrica è spesso chiamato **sensore**, e ciò che converte l'energia elettrica in altre forme di energia è detto **attuatore**. Costruirai **circuiti** per far fluire l'elettricità attraverso vari componenti. I circuiti sono percorsi chiusi fatti da conduttori elettrici dotati di una fonte di energia (come una pila) e un carico che fa qualcosa di utile con l'energia.

In un circuito, l'elettricità scorre da un punto di maggior energia potenziale (solitamente chiamato alimentazione o +) a uno di minor energia potenziale. La massa (spesso rappresentata con un -) è generalmente il punto nel circuito con il potenziale elettrico più basso. Nei circuiti che costruirai, l'elettricità scorre in una sola direzione. Questo tipo di circuito si chiama a corrente continua (CC). Nei circuiti a corrente alternata (CA) l'elettricità cambia la sua direzione 50 o 60 volte al secondo (dipende dal luogo in cui vivi). Questo è il tipo di elettricità che arriva da una presa elettrica di casa.

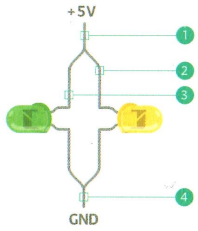
Ci sono alcuni termini da conoscere per lavorare con i circuiti elettrici. La **corrente** (misurata in Ampere, il cui simbolo è **A**) è la quantità di carica elettrica che attraversa un punto specifico di un circuito. La **tensione** (misurata in Volt, il cui simbolo è **V**) è la differenza in energia potenziale tra un punto e l'altro di un circuito. E infine, la **resistenza** (misurata in Ohm, il cui simbolo è Ω) rappresenta quanto un componente resiste al flusso di energia elettrica.

Per immaginare il circuito elettrico puoi pensare alla caduta di una roccia da una rupe come mostrato nella Fig. 1. Più alta è la rupe, più energia avranno le rocce quando colpiranno il fondo. L'altezza della rupe è come la tensione in un circuito: maggiore è la tensione della fonte di energia, più energia dovrai usare. Più rocce hai, più energia sarà trasportata giù dalla rupe. Il numero di rocce è come la corrente in un circuito elettrico. Le rocce attraversano cespugli lungo la discesa, perdendo energia; l'energia si esaurisce scontrandosi contro i cespugli. I cespugli sono come le resistenze in un circuito: si oppongono al flusso elettrico convertendolo in altre forme di energia.



La caduta di una roccia come metafora del flusso di corrente.
Fig. 1

UN PAIO DI COSE SUI CIRCUITI



La corrente (1) = corrente (2) + corrente (3) = corrente (4).
Fig. 2

— In un circuito ci deve essere un percorso completo dalla fonte di energia (alimentazione) fino al punto di minor energia (massa). Se non c'è un percorso per far fluire l'energia, il circuito non funziona.

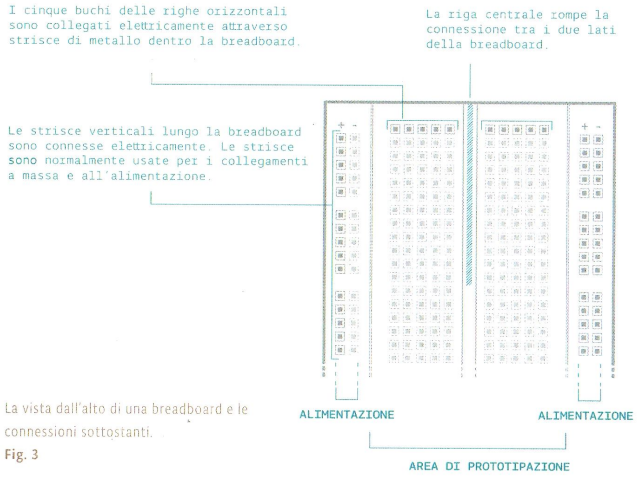
— Tutta l'energia elettrica viene utilizzata in un circuito dai suoi componenti. Ogni componente converte parte dell'energia in un'altra forma di energia. In ogni circuito, tutta la tensione viene convertita in un'altra forma di energia (luce, calore, suono, ecc).

— Il flusso di corrente in un punto specifico di un circuito è sempre lo stesso in arrivo e in uscita.

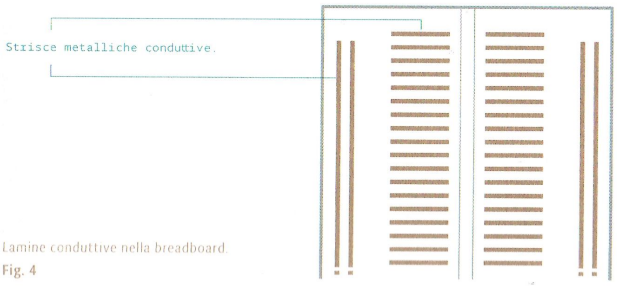
— La corrente elettrica cerca il percorso di minor resistenza verso terra. Dati due percorsi possibili, la maggior parte della corrente elettrica passa lungo il sentiero con meno resistenza. Se si ha una connessione che collega la potenza e la massa senza resistenza, si provoca un cortocircuito, e la corrente cerca di seguire questa strada. In un cortocircuito, la sorgente di alimentazione e i fili convertono l'energia elettrica in calore; possono provocare anche esplosioni. Se hai mai cortocircuitato una batteria e visto delle scintille, sai quanto possa essere pericoloso un cortocircuito.

COS'È UNA BREADBOARD?

La breadboard è il luogo dove costruirai i circuiti. Quella contenuta nel kit è senza saldature, perché non c'è bisogno di saldare nulla, è una specie di LEGO in forma elettronica. Le righe orizzontali e verticali della breadboard, come mostrato in Fig. 3, portano l'elettricità attraverso sottili connettori metallici sotto la plastica forata.



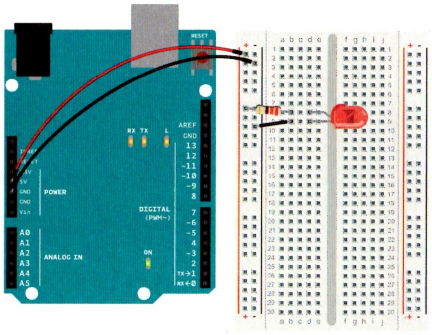
La vista dall'alto di una breadboard e le connessioni sottostanti.
Fig. 3



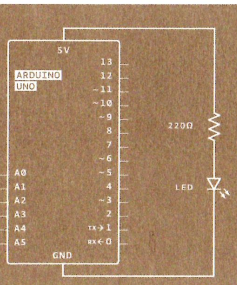
Lamine conduttive nella breadboard.
Fig. 4

SCHEMI ELETTRICI

Nel corso di questi progetti, avrai a che fare con due modi di rappresentare i circuiti: uno schema di montaggio (come nella Fig. 5), che rappresenta gli elementi del kit. L'altra è uno schema elettrico (come nella Fig. 6): un modo più astratto che mostra le relazioni tra i componenti di un circuito. Gli schemi elettrici non sempre mostrano dove sono posizionati i componenti l'uno rispetto all'altro, ma mostrano come sono collegati.

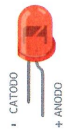


Schema di un circuito.
Fig. 5



Schema elettrico.
Fig. 6

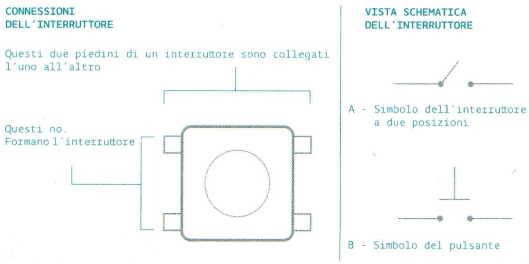
I TUOI PRIMI COMPONENTI



Un ***LED***, diodo emettitore di luce, è un componente che converte l'energia elettrica in luce. I LED sono componenti polarizzati, il che significa che permettono all'energia elettrica di fluire attraverso di loro in una sola direzione. Il piede più lungo sul LED, chiamato anodo, si collega all'alimentazione. Il piedino corto, il catodo, si collega a massa. Quando la tensione viene applicata all'anodo del LED, e il catodo è collegato a massa, il LED emette luce.

Una ***resistenza*** è un componente che resiste al flusso di energia elettrica (vedi l'elenco dei componenti per una spiegazione sulle strisce colorate sul lato). Converte parte dell'energia elettrica in calore. Se si mette una resistenza in serie con un componente come un LED, la resistenza consuma parte dell'energia elettrica e il LED riceve meno energia. Ciò consente di alimentare componenti con la quantità di energia di cui hanno bisogno. Si usa una resistenza in serie con il LED per evitare che riceva troppa tensione. Senza la resistenza, il LED è più luminoso per qualche istante, ma si brucia rapidamente.

Un ***interruttore*** interrompe il flusso di energia elettrica, spezzando il circuito quando è aperto. Quando un interruttore viene chiuso, completa il circuito. Ci sono molti tipi di interruttori. Quelli nel kit sono pulsanti perché sono chiusi solo quando è applicata una pressione.



L'interruttore.
Fig. 7

COSTRUISCI IL CIRCUITO

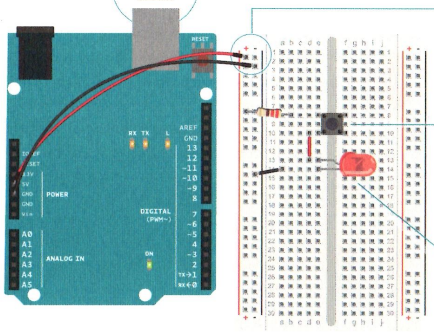


Fig. 8

Il tuo primo circuito interattivo, usando un interruttore, una resistenza e un LED. Arduino è solo la fonte di alimentazione per questo circuito; nei prossimi progetti, potrai collegare i suoi piedini di ingresso e di uscita per il controllo di circuiti più complessi.

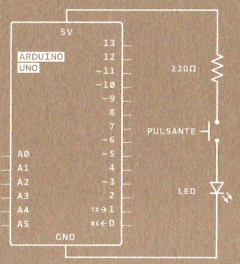


Fig. 9

Stai per utilizzare Arduino in questo progetto, ma solo come fonte di energia. Quando la colleghi a una porta USB o a una batteria da 9 volt, Arduino fornisce un'alimentazione di 5 volt tra il suo piedino 5V e il piedino GND. 5V = 5 volt: lo vedrai scritto in questo modo molte volte.

1. Se Arduino è collegato a una batteria o a un computer via USB, scollegalo prima di costruire il circuito!

2. Collega un filo rosso al piedino 5V sulla scheda Arduino e metti l'altra estremità in una delle linee lungo la tua breadboard. Collega la massa di Arduino alla linea adiacente al filo nero. È utile mantenere il colore del filo coerente (rosso per alimentazione, nero per massa) in tutto il circuito.

3. Ora che hai alimentazione sulla breadboard, monta l'interruttore a cavallo della fessura al centro della breadboard.

4. Utilizza una resistenza da 220 ohm per collegare l'alimentazione a un lato dell'interruttore. Le illustrazioni di questo libro utilizzano 4 fasce. Il kit può avere resistenze a 4 e 5 fasce. Utilizza la figura a lato per controllare quella giusta per questo progetto. Guarda pagina 41 per una spiegazione dettagliata dei codici colore delle resistenze.
Sull'altro lato del pulsante, collega l'anodo (piedino lungo) del LED. Con un filo collega il catodo (piedino corto) del LED a massa. Quando sei pronto, collega il cavo USB ad Arduino.

USALO

Una volta che è tutto pronto, premi il pulsante. Si dovrebbe accendere il LED. Congratulazioni, hai appena costruito un circuito! Quando sarai stanco di premere il pulsante per accendere la luce, sarà il momento di movimentare le cose e aggiungere un secondo pulsante.

Metterai i componenti sulla breadboard in serie e in parallelo.
I componenti in serie sono posizionati uno dopo l'altro.
I componenti in parallelo sono uno di fianco all'altro.

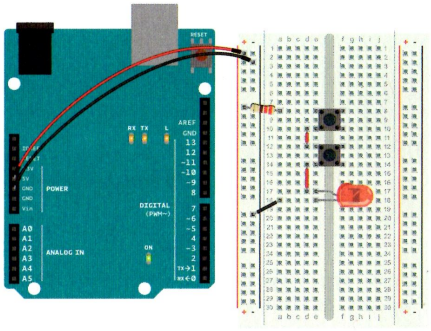
Questi due elementi sono in serie

Circuito in serie

I COMPONENTI IN SERIE SONO UNO DOPO L'ALTRO

Una volta tolta l'alimentazione aggiungi un pulsante accanto all'altro già sulla breadboard. Collegali in serie come mostrato in Fig. 10. Connetti l'anodo (piedino lungo) del LED al secondo interruttore. Connetti il catodo del LED a massa. Alimenta ancora l'Arduino: ora, per accendere il LED, devi schiacciare entrambi i pulsanti. Essendo in serie, hanno bisogno di essere chiusi entrambi per completare il circuito.

RIMUOVI SEMPRE L'ALIMENTAZIONE PRIMA DI INSERIRE QUALCOSA NEL TUO CIRCUITO



I due pulsanti sono in serie. Significa che sono percorsi dalla stessa corrrente elettrica; entrambi devono essere premuti per accendere il LED.
Fig. 10

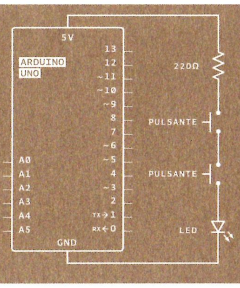
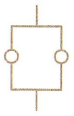


Fig. 11

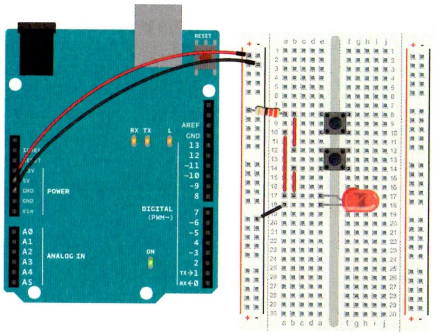


Questi due elementi sono in parallelo

Circuito in parallelo

I COMPONENTI IN PARALLELO SONO UNO DI FIANCO ALL'ALTRO

Ora che sai tutto sui circuiti in serie, è tempo di dedicarsi a quelli in parallelo. Lascia dove sono l'interruttore e il LED, ma rimuovi il filo tra i due pulsanti. Collega entrambi i pulsanti alla resistenza. Connetti le altre estremità dei pulsanti al LED, come mostrato nella Fig. 12. Ora, premendo uno dei due pulsanti, il circuito è completato e la luce si accende.



Questi due interruttori sono in parallelo. Significa che la corrente elettrica si divide tra di loro. Se uno dei pulsanti viene premuto, il LED si accende.
Fig. 12

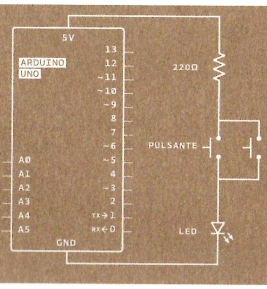


Fig. 13

CAPIRE LA LEGGE DI OHM

Puoi usare questo cerchio per ricordarti il rapporto fra tensione, corrente e resistenza. Metti il dito sopra uno dei tre, e vedi come si relaziona agli altri due.

V = I * R

I = V / R

R = V / I

Corrente, tensione e resistenza sono collegati. Cambiando uno di questi valori in un circuito, si condizionano gli altri. Il rapporto tra di loro è conosciuto come Legge di Ohm, dal nome dello scienziato che la scoprì, Georg Simon Ohm.

TENSIONE (V) = CORRENTE (I) * RESISTENZA (R)

Quando misuri l'amperaggio (corrente) in un circuito che stai costruendo, i valori sono nell'ordine dei milliampere, che sono millesimi di un ampere.

Nel circuito della Fig. 5, stai fornendo 5 volt. Il resistore ha 220 ohm di resistenza. Per trovare la corrente usata dal LED, sostituisci i valori nell'equazione. Dovresti avere 5=I*220. Dividendo entrambe le parti dell'equazione per 220, dovrebbe risultare che I = .023 cioè 23 millesimi di un ampere o 23 milliampere (23 mA) usati dal LED. Questo valore è il massimo che puoi usare in sicurezza con questi LED, ecco perché stai usando una resistenza da 220 ohm.

Puoi espandere il tuo progetto in molti modi, sia creando il tuo pulsante (due pezzi di alluminio con del filo funzionano bene), sia creando una combinazione di pulsanti e LED in parallelo e in serie. Cosa succede quando metti tre o quattro LED in serie? Cosa succede quando sono in parallelo? Perché si comportano in questo modo?

Un ***multimetro*** è uno strumento che misura la quantità di resistenza, corrente e tensione nel circuito. Sebbene non necessario in questi progetti, rappresenta uno strumento utile. Online trovi una buona descrizione su come usarlo all'indirizzo ***arduino.cc/multimeter***

Hai imparato le caratteristiche elettriche della tensione, della corrente e della resistenza costruendo un circuito sulla breadboard. Con alcuni componenti come i LED, le resistenze e i pulsanti, puoi creare semplici sistemi interattivi: se si preme un pulsante, la luce si accende. A questi fondamenti dell'elettronica si farà riferimento – e saranno ampliati – nei prossimi progetti.

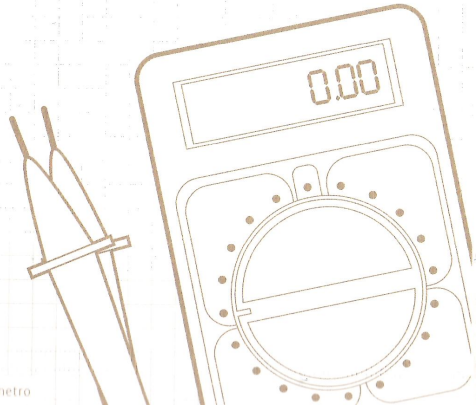


Fig. 14 – Un multimetro

02

INGREDIENTI

INTERFACCIA PER ASTRONAVE

ARDUINO STA PER DIVENTARE IL PROTAGONISTA DI UN FILM DI FANTASCIENZA

Scopri: input e output digitali, il tuo primo programma, le variabili

Tempo: **45 MINUTI**
Livello: ■□□□□

Basato sul progetto: **1**

Ora che hai acquisito le basi dell'elettronica, è tempo di passare a controllare cose e oggetti con Arduino. In questo progetto, costruirai qualcosa che potrebbe essere un'interfaccia di una astronave di un film di fantascienza degli anni Settanta. Realizzerai un pannello di controllo con un pulsante e delle luci che si accenderanno quando schiaccerai il pulsante. Puoi decidere se le luci significano "Inserire l'hyperdrive" o "Spara il laser!". Un LED verde sarà acceso finché non premerai un pulsante. Quando Arduino riceverà un segnale da un pulsante, la luce verde si spegnerà e le altre due luci inizieranno a lampeggiare.

I piedini digitali di Arduino possono leggere solo due stati: quando c'è o non c'è la tensione su un piedino di ingresso. Questo tipo di ingresso è chiamato digitale (o qualche volta binario, per via dei due stati). Questi stati vengono definiti come ALTO (`HIGH`) e BASSO (`LOW`). `HIGH` è come dire "c'è tensione qui!" e `LOW` significa "non c'è tensione su questo piedino!". Quando metti un piedino di `OUTPUT` a `HIGH`, usando un comando chiamatao `digitalWrite()`, lo stai accendendo. Se misuri la tensione tra il piedino e la massa, ottieni 5 volt. Quando metti un piedino di `OUTPUT` a `LOW`, lo stai spegnendo. Il piedino digitale di Arduino funziona sia come ingresso sia come uscita. Nel tuo codice, dovrai configurarli a seconda della funzione che vorrai dare loro. Quando i piedini sono output (uscita), puoi accendere componenti come i LED. Se configuri i piedini come input (ingresso), puoi verificare se è stato premuto o no un pulsante. Dato che i piedini 0 e 1 sono usati per comunicare con il computer, è meglio iniziare con il piedino 2.

COSTRUISCI IL CIRCUITO

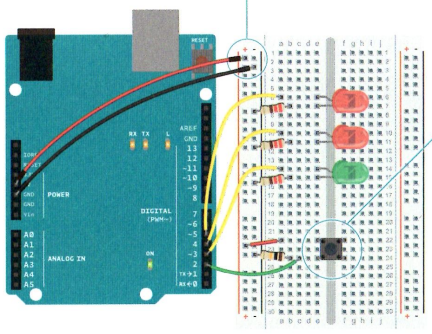


Fig. 1

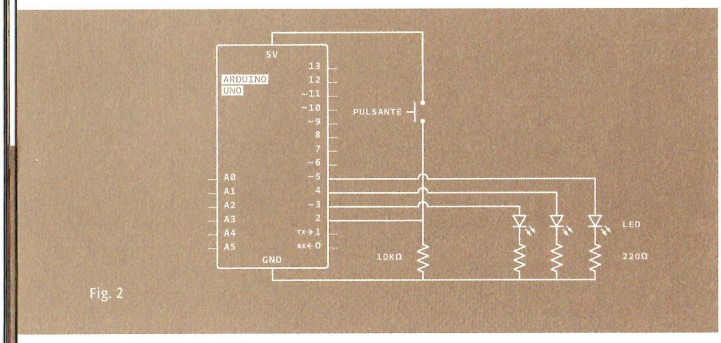
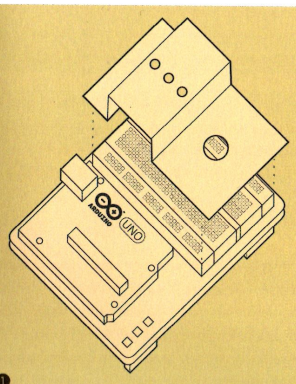


Fig. 2

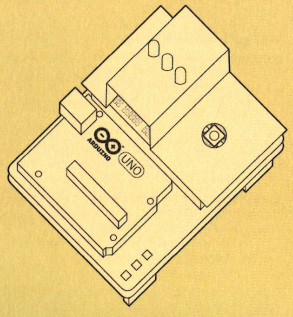
1. Collega la breadboard alle connessioni 5V e GND (massa) di Arduino, come nel progetto precedente. Posiziona due LED rossi e uno verde sulla breadboard. Connetti il catodo (piedino corto) di ciascun LED a massa attraverso una resistenza da 220 ohm. Connetti l'anodo (piedino lungo) del LED verde al piedino 3. Connetti gli anodi dei LED rossi rispettivamente ai piedini 4 e 5.

2. Posiziona l'interruttore sulla breadboard come nel progetto precedente. Connetti un lato all'alimentazione e l'altro lato al piedino digitale 2 di Arduino. Hai anche bisogno di aggiungere una resistenza da 10 kilo ohm da massa al piedino dell'interruttore che è collegato ad Arduino. Questa resistenza di 'pull-down' connette il piedino a massa quando il pulsante è aperto, così si legge **LOW** quando nessuna tensione proviene dal pulsante.

Puoi coprire la breadboard con il cartoncino del kit. O puoi decorarlo per realizzare il tuo personale sistema di lancio. Se le luci si accendono e si spengono non significa nulla di per sé, ma, quando le metti in un pannello di controllo e dai loro un senso, acquistano un significato maggiore. Cosa vuoi che significhi il LED verde? E il lampeggio del LED rosso? Sei libero di deciderlo!



1. Piega il cartoncino pre-tagliato come mostrato qui.



2. Metti il cartoncino piegato sulla breadboard. I tre LED e il pulsante aiutano a trovare la posizione.

IL CODICE

Alcune note prima di iniziare

Ogni programma Arduino ha due funzioni principali. Le funzioni sono parti di un programma del computer che eseguono comandi specifici. Le funzioni hanno nomi univoci e sono 'chiamate' quando servono. Le funzioni necessarie in un programma Arduino sono chiamate `setup()` e `loop()`. Queste funzioni devono essere dichiarate: significa che si deve dire ad Arduino cosa faranno. `setup()` e `loop()` sono indicate come vedi sulla destra.
In questo programma, stai per creare una variabile prima di entrare nella parte principale del programma. Le variabili sono nomi che dai a delle aree di memoria di Arduino così puoi tenere traccia di ciò che sta accadendo. Questi valori possono cambiare a seconda delle istruzioni del tuo programma. I nomi delle variabili dovrebbero essere descrittivi dei valori che stanno immagazzinando. Per esempio, una variabile chiamata `switchState` spiega cosa sta immagazzinando: lo stato di un pulsante. Invece, una variabile chiamata '`x`' non spiega nulla su ciò che contiene.

```
void setup(){
}

void loop(){
}
```

{ Parentesi graffe }
Il codice scritto all'interno di parentesi graffe è eseguito quando viene chiamata la funzione.

Iniziamo a programmare

Per creare una variabile, hai bisogno di dichiararne il *tipo*. Il tipo di dato `int` contiene un numero intero (chiamato anche un intero); che è un numero qualsiasi senza punti decimali. Quando dichiari una variabile, normalmente gli dai anche un valore iniziale. La dichiarazione della variabile deve finire con un punto e virgola (;).

```
1 int switchState = 0;
```

Configura la funzionalità dei piedini

Il `setup()` viene eseguito una volta soltanto all'accensione dell'Arduino. Qui è dove configuri i piedini digitali per renderli ingressi o uscite usando la funzione `pinMode()`.
I piedini connessi ai LED sono uscite (OUTPUT) e il pulsante un ingresso (INPUT).

```
2 void setup(){
3   pinMode(3,OUTPUT);
4   pinMode(4,OUTPUT);
5   pinMode(5,OUTPUT);
6   pinMode(2,INPUT);
7 }
```

Maiuscole, minuscole
Nel codice stai attento a distinguere le maiuscole e le minuscole. Per esempio, `pinMode` è il nome di un comando, ma pinmode produce un errore.

Crea la funzione loop

Il `loop()` viene eseguito continuamente dopo che è stato completato il `setup()`. Il `loop()` è il posto dove verifichi la tensione sugli ingressi e controlli le uscite. Per controllare la tensione su un input digitale, usa la funzione `digitalRead()` che rileva se c'è tensione sul piedino scelto. Per sapere quale piedino leggere, `digitalRead()` si aspetta un *parametro*. I parametri sono informazioni che dai alle funzioni dicendo loro come dovrebbero svolgere il loro lavoro. Per esempio, `digitalRead()` ha bisogno di un parametro: quale piedino leggere. Nel tuo programma, `digitalRead()` sta verificando lo stato del piedino 2 e

```
8 void loop(){
9   switchState = digitalRead(2);
10  // questo è un commento
```

Commenti
I commenti sono note che scrivi per te stesso, che il computer ignora. Per scrivere un commento, aggiungi due barre //. Il computer ignora tutto il contenuto presente dopo le due barre.

immagazzinando il valore nella variabile **switchState**. Se c'è tensione sul piedino quando viene chiamata **digitalRead()**, la variabile switchState ha il valore **HIGH** (o 1). Se non c'è tensione sul piedino, switchState prende il valore **LOW** (o 0).

L'istruzione if

Un comando **if()** nella programmazione confronta due cose e determina se il confronto è vero o falso ed esegue le azioni che gli indichi. Quando confronti due cose nella programmazione, devi usare due simboli dell'uguale ==. Se usi un segno solo, imposti il valore della variabile invece di confrontarlo.

Costruisci la tua astronave

digitalWrite() è il comando che ti permette di impostare un'uscita a 5V o 0V. **digitalWrite()** richiede due parametri: quale piedino controllare e a quale valore impostare il piedino, **HIGH** o **LOW**. Se vuoi accendere i LED rossi e spegnere il LED verde nella tua istruzione **if()**, il tuo codice sarà così.

```
11  if (switchState == LOW) {
12  // il pulsante non è premuto
13    digitalWrite(3, HIGH); // LED verde
14    digitalWrite(4, LOW);  // LED rosso
15    digitalWrite(5, LOW);  // LED rosso
16  }
```

Può essere utile trascrivere il flusso del programma in pseudocodice: un modo per descrivere cosa vuoi che il programma faccia in un linguaggio semplice, ma strutturato in modo che renda facile scrivere un vero programma. In questo caso stai determinando se `switchState` è `HIGH` (significa che il pulsante è premuto) o no. Se il pulsante è premuto, spegnerai il LED verde e accenderai quello rosso. Nel pseudocodice, il comando potrebbe essere così:

```
se switchState è LOW:
  accendi il LED verde
  spegni il LED rosso

se switchState è HIGH:
  spegni il LED verde
  accendi il LED rosso
```

Se ora esegui il tuo programma, le luci cambiano quando premi il pulsante. Questo è piuttosto chiaro, ma puoi aggiungere un tocco di complessità al programma per una reazione più interessante.

Hai detto ad Arduino cosa fare quando l'interruttore è aperto. Ora definisci cosa accade quando è chiuso. L'istruzione **if()** ha un componente opzionale **else** (in italiano oppure) che fa succedere qualcosa se la condizione originale non si è verificata. In questo caso, dato che hai verificato che il pulsante è **LOW**, scrivi il codice per la condizione **HIGH** dopo l'istruzione **else**.

Per far lampeggiare i LED rossi quando è premuto il pulsante, devi accendere e spegnere le luci nell'istruzione **else** che hai appena scritto.

Ora il tuo programma fa lampeggiare il LED rosso quando il pulsante è premuto.

Dopo aver impostato i LED in un determinato stato, devi mettere in pausa l'Arduino prima di riportarli allo stato precedente. Se non aspetti, le luci vanno avanti e indietro così velocemente che appaiono leggermente meno luminose ma non accese o spente. Questo perché l'Arduino esegue il suo **loop()** migliaia di volte ogni secondo e il LED si accende e spegne più velocemente di quanto riusciamo a percepirlo. La funzione **delay()** impedisce ad Arduino di eseguire istruzioni per un certo periodo di tempo. **delay()** richiede un parametro che determina il numero di millisecondi prima di eseguire le istruzioni successive. Ci sono 1000 millisecondi in un secondo. **delay(250)** fa pausa per un quarto di secondo.

```
17  else {  // il pulsante è premuto
18    digitalWrite(3, LOW);
19    digitalWrite(4, LOW);
20    digitalWrite(5, HIGH);
21    delay(250);  // aspetta un quarto di secondo
22    // cambia gli stati dei LED
23    digitalWrite(4, HIGH);
24    digitalWrite(5, LOW);
25    delay(250); // aspetta un quarto di secondo
26  }
27 } // torna indietro all'inizio del loop
```

USALA

Una volta che la tua Arduino è programmata, dovresti vedere accendersi la luce verde. Quando premi il pulsante, le luci rosse iniziano a lampeggiare e la luce verde si spegne. Prova a cambiare il tempo delle due funzioni `delay()`; nota cosa accade alle luci e come la risposta del sistema cambia in base alla velocità del lampeggio. Quando esegui un `delay()` nel programma, questo ferma tutte le altre funzionalità. Non avviene nessuna lettura di sensori fino a che questo periodo non è passato. Sebbene i ritardi siano spesso utili, quando realizzi il tuo progetto assicurati che non interferiscano inutilmente con l'interfaccia.

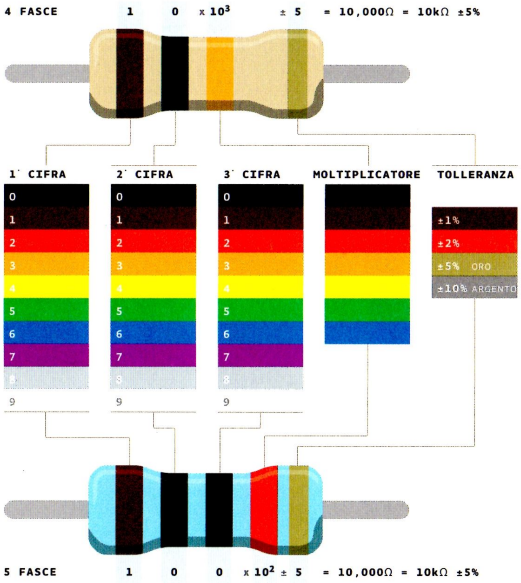
Come potresti far lampeggiare i LED rossi quando si avvia il programma? Come potresti creare un'interfaccia più grande o più complessa con LED e pulsanti per le tue avventure interstellari?

Quando inizi a creare un'interfaccia per il tuo progetto, pensa alle aspettative delle persone che lo useranno. Quando premono un pulsante, vorranno un feedback immediato? Dovrebbe esserci un ritardo tra le loro azioni e ciò che fa Arduino? Prova a metterti nei panni di un altro quando stai progettando e verifica che le tue aspettative siano adeguate alla realtà del progetto.

In questo progetto, hai creato il tuo primo programma con Arduino per controllare con un pulsante il comportamento di alcuni LED. Hai usato delle variabili, un'istruzione if()..., e funzioni per leggere lo stato di un ingresso e controllare delle uscite.

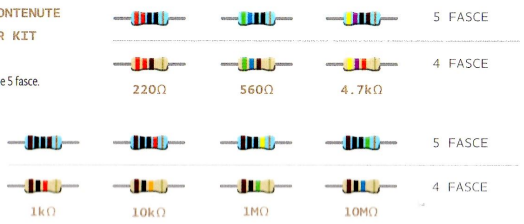
COME LEGGERE IL CODICE COLORI DELLE RESISTENZE

I valori delle resistenze sono espressi da anelli colorati secondo una norma sviluppata negli anni Venti quando era troppo complesso scrivere numeri su oggetti tanto piccoli. Ogni colore corrisponde a un numero, come vedi nella tabella sotto. Ogni resistenza ha 4 o 5 anelli. Nel tipo con 4 anelli, i primi due anelli indicano le prime due cifre del valore mentre il terzo indica il numero degli zero seguenti (tecnicamente rappresenta la potenza di dieci). L'ultimo anello specifica la tolleranza: nell'esempio qui sotto, l'oro indica che il valore della resistenza può essere 10 kilo ohm più o meno 5%.

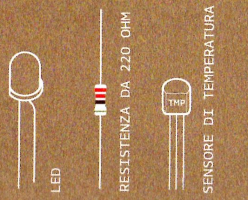


RESISTENZE CONTENUTE NELLO STARTER KIT

Troverai versioni con 4 e 5 fasce.



03



INGREDIENTI

AMOROMETRO

TRASFORMA ARDUINO IN UNA MACCHINA DELL'AMORE. USANDO UN INGRESSO ANALOGICO, POTRAI MISURARE QUANTO SEI CALDO VERAMENTE!

Scopri: ingresso analogico, monitor seriale

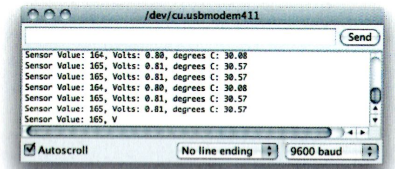
Tempo: **45 MINUTI**
Livello: ■■■■■

Basato sui progetti: **1, 2**

Pulsanti e interruttori sono una gran cosa, ma il mondo fisico è molto di più che non acceso e spento. Anche se Arduino è uno strumento digitale, può leggere informazioni da sensori analogici per misurare grandezze come la temperatura o la luce. Per farlo, utilizzerai il convertitore analogico-digitale (ADC) integrato nell'Arduino. I piedini di ingresso analogico A0-A5 restituiscono un valore compreso tra 0 e 1023 quando leggono delle tensioni comprese tra 0 e 5 volt.

Userai un ***sensore di temperatura*** per misurare quanto è calda la tua pelle. Questo componente produce una tensione che varia in base alla temperatura percepita. Ha tre piedini: uno che si connette a massa, uno che si connette all'alimentazione e il terzo che produce una tensione variabile per Arduino. Nello sketch di questo progetto, leggerai il valore del sensore e lo userai per accendere e spegnere i LED, indicando quanto sei caldo. Esistono molti modelli di sensori di temperatura. Questo, il TMP36, è comodo perché produce una tensione in uscita che è direttamente proporzionale alla temperatura in gradi Celsius.

L'ambiente di sviluppo integrato (IDE) di Arduino è dotato di uno strumento chiamato ***monitor seriale*** che ti consente di visualizzare dati provenienti dal microcontrollore. Usando il monitor seriale, è possibile ottenere informazioni sullo stato dei sensori e avere un'idea di ciò che sta accadendo nel circuito e nel codice.



Monitor seriale.
Fig. 1

COSTRUISCI IL CIRCUITO

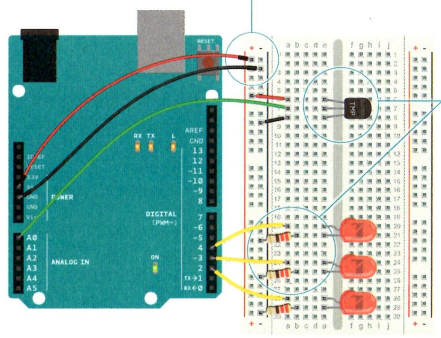


Fig. 2

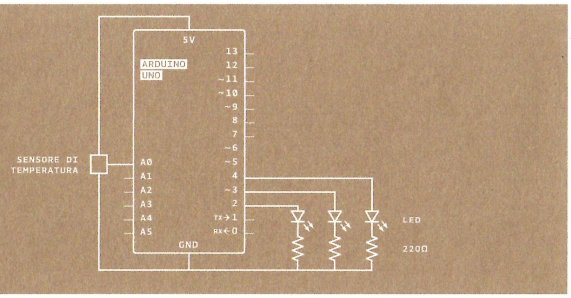


Fig. 3

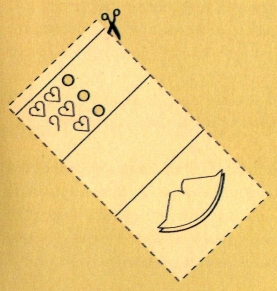
In questo progetto, prima di procedere devi misurare la temperatura della stanza. Lo si può fare anche attraverso calibrazione. È possibile usare un pulsante per impostare la temperatura di riferimento o prendendo un campione con Arduino prima di iniziare il `loop()` e usarlo come punto di riferimento. Il Progetto 06 scenderà nei dettagli di questo tema; puoi intanto guardare l'esempio di calibrazione che viene dato con il software di Arduino: ***arduino.cc/calibration***

1. Proprio come hai fatto nei progetti precedenti, collega la tua breadboard in modo da avere alimentazione e massa.

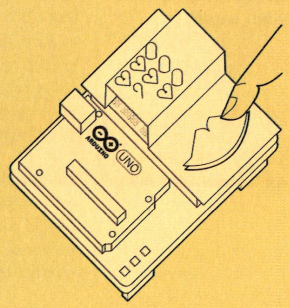
2. Collega a massa il catodo (il piedino corto) di ogni LED che stai usando attraverso una resistenza da 220 ohm. Collega gli anodi dei LED ai piedini dal 2 al 4. Questi saranno gli indicatori per il progetto.

3. Posiziona il TMP36 sulla breadboard con la parte piatta verso Arduino (l'ordine dei piedini è importante!) come mostrato in Fig. 2. Collega il piedino di sinistra della parte piatta all'alimentazione e il piedino destro a massa. Collega il piedino centrale al piedino A0 su Arduino. Questo è il piedino di ingresso analogico 0.

Crea un'interfaccia per il tuo sensore che permetta alle persone di interagirvi. Un ritaglio di carta a forma di mano è un buon indicatore. Se ti senti fortunato, crea un paio di labbra da baciare, guarda come si accendono le luci! Potresti etichettare i LED per dar loro un significato. Un LED potrebbe significare che sei un asociale, due LED che sei caldo e amichevole e tre che sei troppo caldo da gestire!



1. Ritaglia un pezzo di carta da posizionare sulla breadboard. Disegna delle labbra dove sarà messo il sensore e taglia alcuni cerchi per farci passare i LED.



2. Posiziona il ritaglio sulla breadboard in modo che le labbra coprano il sensore e che i LED fuoriescano dai buchi. Premi le labbra per vedere quanto sei caldo!

IL CODICE

Un paio di costanti utili

Le costanti sono simili alle variabili: ti permettono di dare un nome univoco agli elementi presenti nel programma, ma, al contrario delle variabili, non possono cambiare. Dai un nome all'ingresso analogico per riconoscerlo facilmente e crea un'altra costante per la temperatura di base. Per ogni 2 gradi al di sopra di questo riferimento, si accende un LED. Hai già visto i tipi di dati int, usali qui per identificare su quale piedino è collegato il sensore. La temperatura è immagazzinata come un *float*, cioè un numero decimale.

```
1 const int sensorPin = A0;
2 const float baselineTemp = 20.0;
```

Avvia la porta seriale alla velocità desiderata

Nel `setup()` usa un nuovo comando, `Serial.begin()`, che apre una connessione tra l'Arduino e il computer, così puoi vedere il valore proveniente dall'ingresso analogico sullo schermo del computer.
Il parametro **9600** è la velocità alla quale comunica l'Arduino: 9600 bit al secondo. Usa il monitor seriale dell'IDE di Arduino per visualizzare le informazioni che invii dal tuo microcontrollore. Quando apri il monitor seriale dell'IDE verifica che la velocità di trasmissione sia 9600.

```
3 void setup(){
4   Serial.begin(9600); // apri una porta seriale
```

Imposta le direzioni dei piedini digitali e spegnili

Il prossimo è un ciclo `for()` per impostare alcuni piedini come uscite. Sono i piedini che in precedenza hai collegato ai LED. Invece di dare loro nomi unici e digitare la funzione `pinMode()` per ognuno di essi, puoi usare un ciclo `for()` per impostarli velocemente. Questo è un trucco comodo se hai un gran numero di cose simili che vuoi ripetere in un programma. Di' al ciclo `for()` di andare dal piedino 2 al 4 in modo sequenziale.

```
5   for(int pinNumber = 2; pinNumber<5; pinNumber++){
6     pinMode(pinNumber,OUTPUT);
7     digitalWrite(pinNumber, LOW);
8   }
9 }
```

tutorial per il ciclo for()
arduino.cc/for

Leggi il sensore di temperatura

Nel `loop()`, usa una variabile locale chiamata `sensorVal` per immagazzinare le letture del sensore. Per ottenere il valore dal sensore, chiama `analogRead()` che richiede come parametro il numero del piedino dove leggere la tensione. Il valore, compreso tra 0 e 1023, rappresenta la tensione sul piedino.

```
10 void loop(){
11   int sensorVal = analogRead(sensorPin);
```

Invia i valori del sensore di temperatura al computer

La funzione `Serial.print()` manda informazioni dall'Arduino al computer collegato. Puoi vedere questa informazione con il monitor seriale. Se dai al `Serial.print()` come parametro un testo tra virgolette, questo invia il testo che hai digitato. Se dai una variabile come parametro, invia il valore di quella variabile.

```
12   Serial.print("Sensor Value: ");
13   Serial.print(sensorVal);
```

Converti la lettura del sensore in tensione

Anche con poche conoscenze di matematica, è possibile determinare la reale tensione sul piedino. La tensione è tra 0 e 5 volt e potrebbe essere decimale (per esempio, potrebbe essere 2,5 volt), perciò hai bisogno di immagazzinarla in un **float**. Crea una variabile chiamata **voltage** per memorizzare questo numero. Dividi **sensorVal** per 1024.0 e moltiplicalo per 5.0. Il nuovo numero rappresenta la tensione sul piedino.

```
14  // converti la lettura ADC in tensione
15  float voltage = (sensorVal/1024.0) * 5.0;
```

Come con il valore del sensore, invialo al monitor seriale.

```
16  Serial.print(", Volts: ");
17  Serial.print(voltage);
```

Converti la tensione in temperatura e manda il valore al computer

Se esamini la documentazione (*datasheet*) del sensore, trovi informazioni sui valori della tensione in uscita. I datasheet sono come manuali per i componenti elettronici. Sono stati scritti da ingegneri per altri ingegneri. Il datasheet di questo sensore spiega che una variazione di 10 millivolt del sensore è equivalente a un cambio di temperatura di 1 grado Celsius. Indica anche che il sensore può leggere temperature sotto gli 0 gradi. Per questo, devi creare una compensazione (*offset*) per i valori sotto gli 0 gradi. Se prendi la tensione, sottrai 0.5 e moltiplichi per 100, hai la temperatura accurata in gradi Celsius. Memorizza questo numero in una variabile **float** chiamata **temperature**.

Ora che hai la temperatura vera, manda il valore al monitor seriale. Visto che la variabile **temperature** è l'ultima cosa che stampi in questo ciclo, puoi usare un comando leggermente differente: **Serial.println()**.
Questo comando manda a capo il monitor seriale dopo l'invio del valore, aiutando a rendere più leggibili i dati visualizzati.

```
18  Serial.print(", degrees C: ");
19  // converti la tensione in temperatura
20  float temperature = (voltage - .5) * 100;
21  Serial.println(temperature);
```

Starter Kit dei datasheet
arduino.cc/kitdatasheets

Spegni i LED con una temperatura bassa

Con la temperatura vera puoi usare un'istruzione **if()...else** per accendere i LED. Usando la temperatura di riferimento come punto di partenza, accendi un LED per ogni aumento di temperatura di 2 gradi. Cerca diversi intervalli di valori mentre ti muovi sulla scala delle temperature.

```
22  if(temperature < baselineTemp){
23    digitalWrite(2, LOW);
24    digitalWrite(3, LOW);
25    digitalWrite(4, LOW);
```

Accendi un LED a una bassa temperatura

L'operatore && significa "e" (and), in senso logico. Puoi verificare più condizioni come se la temperatura è di 2 gradi più alta del riferimento e meno di 4 gradi sopra di essa.

```
26  }else if(temperature >= baselineTemp+2 &&
      temperature < baselineTemp+4){
27    digitalWrite(2, HIGH);
28    digitalWrite(3, LOW);
29    digitalWrite(4, LOW);
```

Accendi due LED a una temperatura media

Se la temperatura è tra 2 e 4 gradi sopra il riferimento, questo blocco di codice accende anche il LED sul piedino 3.

```
30  }else if(temperature >= baselineTemp+4 &&
      temperature < baselineTemp+6){
31    digitalWrite(2, HIGH);
32    digitalWrite(3, HIGH);
33    digitalWrite(4, LOW);
```

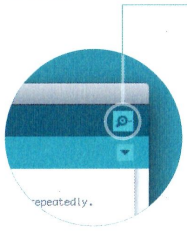
Accendi tre LED a un'alta temperatura

```
34  }else if(temperature >= baselineTemp+6){
35    digitalWrite(2, HIGH);
36    digitalWrite(3, HIGH);
37    digitalWrite(4, HIGH);
```

Il convertitore digitale-analogico non può leggere velocissimamente così dovresti inserire un leggero ritardo alla fine del tuo `loop()`. Se lo leggi troppo di frequente, i tuoi valori appaiono irregolari.

```
38  }
39  delay(1);
40 }
```

USALO



Con il codice caricato su Arduino, clicca l'icona del monitor seriale. Dovresti vedere un flusso di valori così formattati: `Sensor: 200, Volts: .70, degrees C: 17`

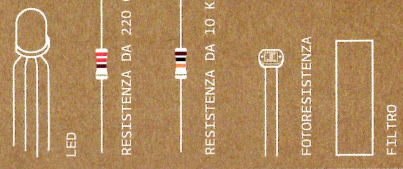
Metti le dita intorno al sensore quando è collegato alla breadboard e guarda cosa accade ai valori nel monitor seriale. Prendi nota della temperatura quando il sensore è lasciato all'aria aperta.

Chiudi il monitor seriale e imposta la costante del riferimento di temperatura al valore della temperatura ambiente. Carica ancora il codice e prova a tenere il sensore tra le dita. Mentre la temperatura aumenta, dovresti vedere accendersi i LED uno a uno. Congratulazioni!

Crea un'interfaccia per due persone per testare la loro compatibilità reciproca. Decidi cosa significa compatibilità e come la leggerai. Forse si sono tenuti la mano e hanno generato calore? Forse si sono abbracciati? Cosa ne pensi?

Espandendo i tipi di ingressi che puoi leggere, hai usato analogRead() e il monitor seriale per tenere traccia dei cambiamenti dentro Arduino. Ora è possibile leggere un gran numero di sensori analogici.

04



INGREDIENTI

LAMPADA MISCELA COLORI

USANDO UN LED A TRE COLORI E TRE FOTORESISTENZE, POTRAI CREARE UNA LAMPADA CHE CAMBIA COLORE IN BASE ALLE CONDIZIONI ESTERNE DI LUCE

Scopri: uscite analogiche, mappare valori

Tempo: **45 MINUTI**
Livello: ■■□□□

Basato sui progetti: **1, 2, 3**

Far lampeggiare i LED può essere divertente, ma che ne pensi di farli sfumare o miscelarne i colori? Ti potresti aspettare che si tratti solo di fornire meno tensione a un LED per fargli cambiare luminosità.

Arduino non può variare la tensione d'uscita dei suoi piedini che può essere solo di 5V. Quindi avrai bisogno di una tecnica chiamata modulazione di larghezza di impulso – in inglese ***Pulse Width Modulation (PWM)*** – per attenuare i LED. La modulazione di larghezza di impulso accende e spegne rapidamente i piedini di uscita in un determinato intervallo di tempo. Il cambiamento avviene più velocemente di quanto l'occhio umano possa percepire. È simile al modo in cui funziona il cinema: mostrare velocemente una serie di immagini fisse per creare l'illusione del movimento.

Quando fai passare velocemente il piedino da `HIGH` a `LOW`, è come se stessi cambiando la tensione. La percentuale di tempo in cui un piedino è `HIGH` in un periodo è chiamata *duty cycle* o ciclo di lavoro utile. Quando il piedino è `HIGH` per metà di un periodo e `LOW` per l'altra metà, il ciclo di lavoro è 50%. Un ciclo di lavoro più basso produce un LED più attenuato che un ciclo di lavoro più alto.

Arduino Uno ha sei piedini riservati al PWM ***(piedini digitali 3, 5, 6, 9, 10 e 11)***; sono identificati da ~ accanto al loro numero sulla scheda.

In questo progetto come ingressi userai delle ***fotoresistenze*** (sensori che cambiano la loro resistenza secondo la quantità di luce che li colpisce; sono conosciuti anche come fotocellule o fotorivelatori). Se ne colleghi una estremità ad Arduino, puoi misurare il cambiamento di resistenza controllando la tensione sul piedino.

COSTRUISCI IL CIRCUITO

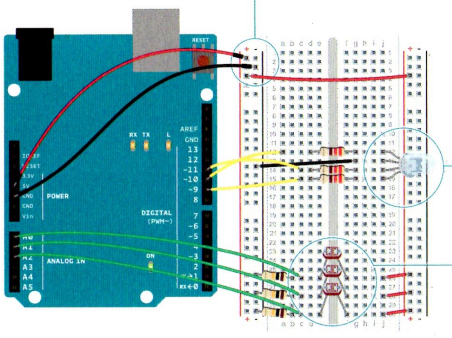


Fig. 1

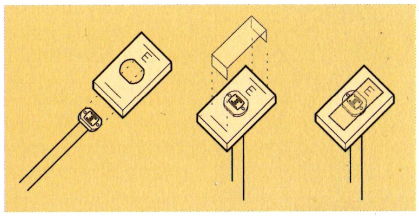


Fig. 2

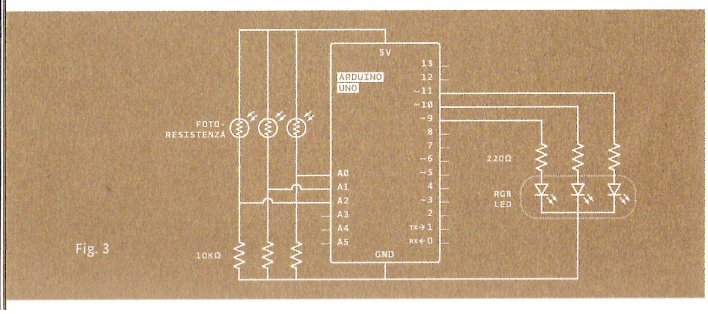


Fig. 3

1. Collega la breadboard così hai alimentazione e massa su entrambi i lati, come nei progetti precedenti.

2. Metti tre fotoresistenze sulla breadboard a cavallo del separatore centrale come in Fig. 1. Collega una estremità di ogni fotoresistenza all'alimentazione. Sull'altro lato, collega a massa una resistenza da 10 kilo ohm. Questa resistenza è in serie con la fotoresistenza, e insieme formano un partitore di tensione. La tensione nel punto in cui si incontrano è proporzionale al rapporto delle loro resistenze, secondo la legge di Ohm (vedi il Progetto 01 per maggiori dettagli sulla legge di Ohm). Al variare della luce che colpisce la fotoresistenza cambia la tensione nel punto di contatto. Sullo stesso lato della resistenza, collega le fotoresistenze agli ingressi analogici 0, 1 e 2 con un ponticello.

3. Prendi tre filtri colorati e posizionali ciascuno su una fotoresistenza. Metti il filtro rosso sulla fotoresistenza collegata all'A0, il verde su quella collegata all'A1 e il blu su quella collegata all'A2. Ognuno di questi filtri consente solo alla luce di una specifica lunghezza d'onda di arrivare al sensore. Il filtro rosso fa passare solo luce rossa, il filtro verde solo luce verde e il filtro blu solo luce blu. Questo permette di rilevare i livelli di colore nella luce che colpisce i sensori.

4. Il LED con 4 gambe è un LED RGB a catodo comune. Il LED contiene tre elementi separati – rosso, verde e blu – e una massa comune (il catodo). Creando una differenza di tensione tra il catodo e la tensione d'uscita dei piedini PWM dell'Arduino (che sono connesi agli anodi attraverso resistenze da 220 ohm), il LED sfuma nei tre colori. Prendi nota di quale sia il piedino più lungo sul LED, posizionalo sulla breadboard e collegalo a massa. Collega gli altri tre piedini alle uscite digitali 9, 10 e 11, tramite le resistenze da 220 ohm. Assicurati di collegare ogni LED al piedino PWM corretto, seguendo l'immagine a sinistra.

IL CODICE

Costanti utili

Imposta le costanti per i piedini che stai usando come ingressi e uscite, così puoi tenere traccia di quale sensore si accoppia con quale colore del LED. Usa `const int` come tipo di dato.

```
const int greenLEDPin = 9;
const int redLEDPin = 11;
const int blueLEDPin = 10;

const int redSensorPin = A0;
const int greenSensorPin = A1;
const int blueSensorPin = A2;
```

Variabili per immagazzinare le letture dei sensori e il livello di luce di ogni LED

Aggiungi variabili per i valori dei sensori in ingresso e per i valori di luminosità dei LED. Si può usare il tipo di dati `int` per tutte le variabili.

```
int redValue = 0;
int greenValue = 0;
int blueValue = 0;

int redSensorValue = 0;
int greenSensorValue = 0;
int blueSensorValue = 0;
```

Imposta la direzione dei piedini digitali e configura la porta seriale

Nel `setup()`, avvia la comunicazione seriale a 9600 bps. Come nell'esempio precedente, la usi per visualizzare i valori dei sensori tramite il monitor seriale. Inoltre, puoi vedere i valori mappati che utilizzi per sfumare il LED. Inoltre, definisci i piedini del LED come uscite con `pinMode()`.

```
void setup() {
  Serial.begin(9600);

  pinMode(greenLEDPin,OUTPUT);
  pinMode(redLEDPin,OUTPUT);
  pinMode(blueLEDPin,OUTPUT);
}
```

Leggi il valore di ogni sensore di luce

Nel `loop()` leggi i valori del sensore sui piedini A0, A1 e A2 con `analogRead()` e memorizza il valore delle variabili. Metti un breve `delay()` tra ogni `analogRead()` per dar tempo all'ADC (convertitore analogico-digitale) di fare il suo lavoro.

```
void loop() {
  redSensorValue = analogRead(redSensorPin);
  delay(5);
  greenSensorValue = analogRead(greenSensorPin);
  delay(5);
  blueSensorValue = analogRead(blueSensorPin);
```

Riporta le letture del sensore al computer

Riporta su una riga i valori del sensore.
"`\t`"è equivalente a premere il tasto "`tab`" sulla tastiera.

```
  Serial.print("Raw Sensor Values \t Red: ");
  Serial.print(redSensorValue);
  Serial.print("\t Green: ");
  Serial.print(greenSensorValue);
  Serial.print("\t Blue: ");
  Serial.println(blueSensorValue);
```

Converti le letture dei sensori

La funzione per modificare la luminosità dei LED tramite PWM è `analogWrite()`. Ha bisogno di due parametri: il piedino da modificare e un valore tra 0 e 255. Il secondo rappresenta il ciclo di lavoro che Arduino produce sul piedino specificato. Un valore di 255 imposta il piedino **HIGH** per tutto il tempo, portando alla massima luminosità il LED collegato. Un valore di 127 imposta il piedino **HIGH** per metà del periodo, rendendo il LED più sfumato. Il valore 0 imposta il piedino **LOW** per tutto il tempo, spegnendo il LED. Per convertire la lettura di un sensore da un valore tra 0 e 1023 a uno tra 0 e 255 per `analogWrite()`, dividi per 4 la lettura del sensore.

```
31  redValue = redSensorValue/4;
32  greenValue = greenSensorValue/4;
33  blueValue = blueSensorValue/4;
```

Riporta i livelli di luce del LED che sono stati calcolati

Invia i nuovi valori mappati su una riga separata.

```
34  Serial.print("Mapped Sensor Values \t Red: ");
35  Serial.print(redValue);
36  Serial.print("\t Green: ");
37  Serial.print(greenValue);
38  Serial.print("\t Blue: ");
39  Serial.println(blueValue);
```

Imposta i livelli di luce del LED

```
40  analogWrite(redLEDPin, redValue);
41  analogWrite(greenLEDPin, greenValue);
42  analogWrite(blueLEDPin, blueValue);
43 }
```

USALA

Quando hai programmato e collegato Arduino, apri il monitor seriale. Il LED è probabilmente di un colore biancastro, a seconda del colore predominante della luce nella tua stanza. Guarda i valori provenienti dai sensori nel monitor seriale: se sei in un ambiente con illuminazione stabile, il numero sarà probabilmente abbastanza consistente.

Spegni la luce nella stanza in cui stai lavorando e guarda cosa succede ai valori dei sensori. Con una torcia elettrica, illumina ogni sensore e nota come i valori cambiano nel monitor seriale e come il LED cambia colore. Quando le fotoresistenze sono coperte con un filtro, reagiscono solo alla luce di una certa lunghezza d'onda. Questo ti dà la possibilità di cambiare ogni colore in modo indipendente.

Avrai notato che il valore d'uscita della fotoresistenza non copre tutto l'intervallo da 0 a 1023. Per questo progetto va bene, ma per una spiegazione più dettagliata di come calibrare l'intervallo effettivo, vedi il Progetto 06.

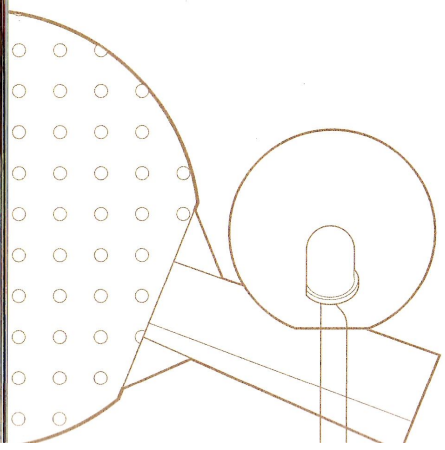
Probabilmente hai notato che la sfumatura dei LED non è lineare. Quando il LED è circa a metà luminosità, sembra che non diventi più luminoso. Questo perché i nostri occhi non percepiscono la luminosità in maniera lineare. La luminosità del LED dipende non solo dal livello di `analogWrite()`, ma anche dalla distanza della luce dal diffusore, la distanza dell'occhio dalla luce e la luminosità della luce rispetto al resto della luce nella stanza.

Come puoi usarla per sapere se fuori è una bella giornata mentre stai lavorando in casa? Quali altri tipi di sensori puoi utilizzare per controllare il colore del LED?

Progetto 04
Lampada miscela colori

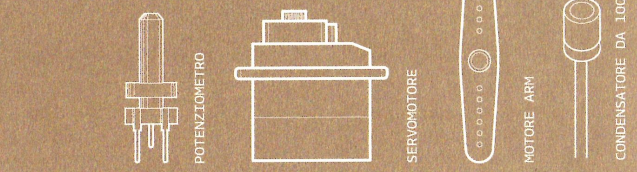
Il LED di per sé è abbastanza carino, ma non è una vera lampada. Tuttavia, ci sono vari modi per diffondere la luce e a farla somigliare a una lampadina tradizionale a incandescenza. Per esempio, una pallina da ping pong con un foro per il LED può diventare un bel diffusore. Altre possibilità sono quelle di coprire la luce con colla traslucida o levigare la superficie del LED. Non importa quale opzione scegli, si perderà almeno un po' di luminosità quando è diffusa, ma probabilmente sarà molto più gradevole.

Non più limitato al solo accendere e spegnere luci, ora hai il controllo sulla luminosità. analogWrite() è la funzione che ti permette di comandare in PWM i componenti collegati ai piedini 3, 5, 6, 9, 10, 11, variando il ciclo di lavoro.

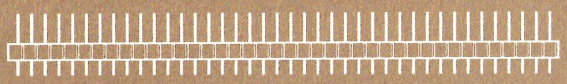


La palla da ping pong tagliata per ospitare il LED.
Fig. 4

05



CONNETTORE A PETTINE (3 piedini)



INGREDIENTI

INDICATORE D'UMORE

USA UN SERVOMOTORE PER CREARE UN INDICATORE MECCANICO PER RIVELARE DI QUALE UMORE SEI OGGI

Scopri: mappatura di valori, i servomotori, usare le librerie di Arduino

Tempo: **1 ORA**
Livello: ■■□□□

Basato sui progetti: **1, 2, 3, 4**

I ***servomotori*** sono un tipo speciale di motori che non ruotano in modo continuo, ma si muovono a una specifica posizione e ci stanno fino a che non dici loro di muoversi ancora. I servo normalmente ruotano solo di 180 gradi (metà di un cerchio). Combinando uno di questi motori con un piccolo oggetto di cartoncino, sarai in grado di far sapere alla gente se possono chiederti aiuto per il loro prossimo progetto oppure no.

Analogamente al modo in cui hai modulato a larghezza di impulso (PWM) i LED nel Progetto 04, i servomotori si aspettano degli impulsi che dicano loro in che modo muoversi. Gli impulsi arrivano sempre agli stessi intervalli di tempo, ma la larghezza varia tra 1000 e 2000 microsecondi. Anche se è possibile scrivere del codice per generare questi impulsi, il software di Arduino ha una libreria che ti permette di controllare facilmente il motore.

Dato che il servo ruota solo di 180 gradi e il tuo ingresso analogico oscilla tra 0 e 1023, hai bisogno di una funzione chiamata `map()` per cambiare la scala dei valori che arrivano dal potenziometro.

Uno degli aspetti più importanti della comunità di Arduino è rappresentato dalle persone di talento che estendono le sue funzionalità attraverso l'aggiunta di software. È possibile per tutti scrivere librerie che implementano le funzionalità di Arduino. Sono disponibili librerie per una gran varietà di sensori, attuatori e altri dispositivi che gli utenti hanno messo a disposizione della comunità. Una libreria software espande la funzionalità di un ambiente di programmazione. Il software di Arduino ha alcune librerie utili per lavorare con l'hardware o i dati. Una delle librerie incluse è progettata per i servomotori. Nel tuo codice, importando la libreria, tutte le sue funzionalità saranno disponibili per il tuo progetto.

COSTRUISCI IL CIRCUITO

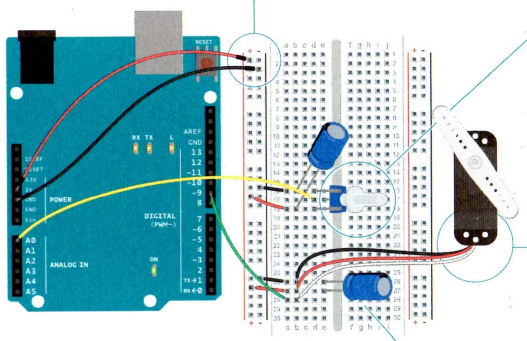


Fig. 1

1. Collega 5V e GND della scheda Arduino a un lato della tua breadboard.

2. Metti un potenziometro sulla breadboard e collega un lato a 5V e l'altro a massa. Un potenziometro è un partitore di tensione. Quando giri la manopola, cambi il rapporto di tensione tra il piedino in mezzo e l'alimentazione. Puoi leggere questo cambiamento su un ingresso analogico. Collega il piedino di mezzo al piedino analogico 0. Questo controlla la posizione del servomotore.

3. Il servo ha tre fili che fuoriescono da esso. Uno è l'alimentazione (rosso), uno è la massa (nero) e il terzo (bianco) è la linea di controllo che riceve le informazioni da Arduino. Collega tre piedini di un connettore a pettine al connettore del servo (vedi Fig. 3). Collega il connettore a pettine alla breadboard così ogni piedino è in una riga diversa. Collega 5V al filo rosso, massa al filo nero e il filo bianco al piedino 9.

4. Quando un servomotore inizia a muoversi, utilizza più corrente di prima. Questo causa un calo di tensione della scheda. Collegando un condensatore da 100 uf tra l'alimentazione e la massa accanto al connettore a pettine come mostrato in Fig. 1, puoi compensare i cambiamenti di tensione che avverranno. Puoi anche mettere un condensatore tra l'alimentazione e la massa del tuo potenziometro. Questi sono chiamati ***condensatori di disaccoppiamento*** perché riducono o separano i cambiamenti causati dai componenti nel resto del circuito. Fai attenzione a collegare il catodo alla massa (è il lato con una stringa nera sul lato) e l'anodo all'alimentazione. Se metti i condensatori al contrario, potrebbero esplodere.

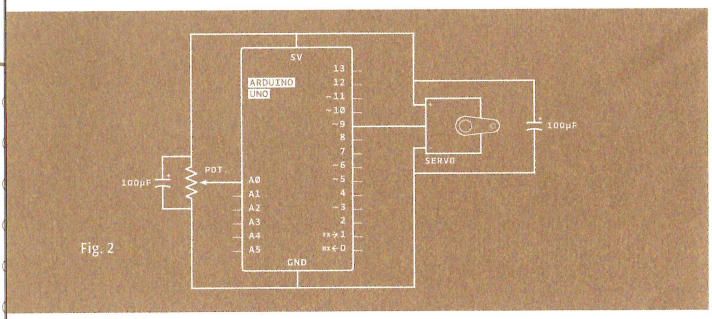
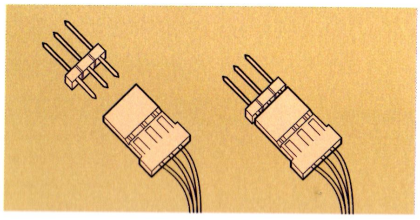


Fig. 2



Il tuo servomotore ha connettori femmina, così hai bisogno di aggiungere dei connettori a pettine per collegarlo alla breadboard

Fig. 3

IL CODICE

Importa la libreria

Per usare la libreria Servo, per prima cosa devi importarla: questo rende le funzioni dalla libreria disponibili per il tuo sketch.

```
#include <Servo.h>
```

Nota che le istruzioni #include non hanno il punto e virgola alla fine della riga.

Crea l'oggetto Servo

Per fare riferimento al Servo, hai bisogno di creare un'istanza della classe servo in una variabile: si chiama *oggetto*. Quando esegui questa operazione, gli stai dando un nome univoco che ha tutte le funzioni e le capacità che offre la libreria Servo. Da questo punto in poi, nel programma, ogni volta che fai riferimento a `myServo`, parlerai con l'oggetto Servo.

```
Servo myServo;
```

Dichiara le variabili

Crea una costante per dare un nome al piedino a cui è collegato il potenziometro e le variabili per contenere il valore di ingresso analogico e l'angolo al quale vuoi che il servo si muova.

```
int const potPin = A0;
int potVal;
int angle;
```

Associa l'oggetto Servo a un piedino di Arduino, avvia la porta seriale

Nel `setup()`, devi dire ad Arduino a quale piedino è collegato il tuo servo.

Includi una connessione seriale così puoi controllare i valori del potenziometro e vedere come vengono mappati alla posizione del servomotore.

```
void setup() {
  myServo.attach(9);

  Serial.begin(9600);
}
```

Leggi i valori del potenziometro

Nel `loop()`, leggi l'ingresso analogico e invia il valore al monitor seriale.

```
void loop() {
  potVal = analogRead(potPin);
  Serial.print("potVal: ");
  Serial.print(potVal);
```

Mappa i valori del potenziometro ai valori del servo

Per creare un valore utilizzabile per il servomotore dal tuo ingresso analogico, è più facile utilizzare la funzione `map()`. Questa funzione scala i numeri. In questo caso trasforma i valori tra 0 e 1023 in valori compresi tra 0 e 179. Richiede cinque parametri: il numero da scalare (qui è potVal), il valore minimo dell'ingresso (0), il valore massimo dell'ingresso (1023), il valore minimo dell'uscita (0) e il valore massimo dell'uscita (179). Memorizza il nuovo valore nella variabile angolo (`angle`). Quindi, invia il valore mappato al monitor seriale.

```
  angle = map(potVal, 0, 1023, 0, 179);
  Serial.print(", angle: ");
  Serial.println(angle);
```

Ruota il servo

Infine, è il momento di spostare il servo. Il comando `servo.write()` muove il motore alla posizione specificata. Alla fine del `loop()` metti un ritardo (`delay`) così il servo ha tempo di spostarsi nella nuova posizione.

```
  myServo.write(angle);
  delay(15);
}
```

USALO

Quando Arduino è stato programmato e acceso, apri il monitor seriale. Dovresti vedere una riga di valori simile a questa:

```
potVal : 1023, angle : 179
potVal : 1023, angle : 179
```

Quando muovi il potenziometro, dovresti vedere cambiare i numeri. Ancora più importante, dovresti vedere che il servomotore cambia posizione. Nota la relazione tra il valore di `potVal` e `angle` nel monitor seriale e la posizione del servo. Dovresti vedere risultati coerenti quando ruoti il potenziometro.

Una cosa bella di utilizzare i potenziometri come ingressi analogici è che ti forniscono l'intera gamma di valori compresi tra 0 e 1023. Questo li rende utili per testare progetti che utilizzano l'ingresso analogico.

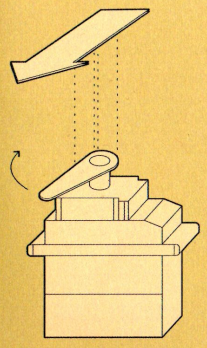
I servomotori sono normali motori con all'interno ingranaggi e un circuito. La meccanica all'interno fornisce un feedback al circuito, così è sempre consapevole della sua posizione. Anche se sembra che abbia una gamma limitata di movimenti, è possibile ottenerne una grande varietà con alcune parti meccaniche aggiuntive. Ci sono parecchie risorse che ne descrivono in dettaglio i meccanismi come ***robives.com/mechs*** e il libro ***Making Things Move*** di *Dustyn Roberts*.

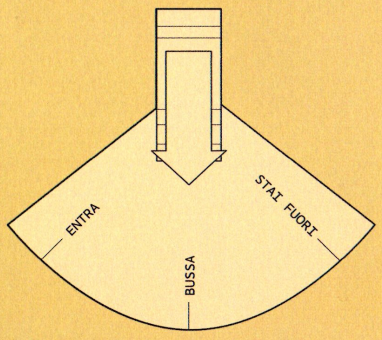
Il potenziometro non è l'unico sensore che è possibile utilizzare per il controllo del servo. Con la stessa configurazione fisica (una freccia che punta a una serie di indicatori differenti) e un sensore diverso, che tipo di indicatore puoi costruire? Come si potrebbe fare con la temperatura (come nell'amorometro)? Potresti determinare l'ora del giorno con una fotoresistenza? Come entra in gioco la mappatura di valori quando si usano questi tipi di sensori?

I servomotori possono facilmente essere controllati da Arduino usando una libreria, che è una raccolta di codice che estende un ambiente di programmazione. Qualche volta è necessario manipolare i valori mappandoli da una scala all'altra.

Ora che la parte di movimento è in funzione, è tempo di far sapere alle persone che sei disponibile ad aiutarli nei loro progetti o che vuoi essere lasciato in pace per pianificare la tua prossima creazione.

Con le forbici, taglia un pezzo di cartone a forma di freccia. Metti il servo a 90 gradi (controlla il valore angolare sul monitor seriale se non sei sicuro). Incolla la freccia in modo che sia orientata nella stessa direzione del corpo del motore. Ora dovresti essere in grado di ruotare la freccia di 180 gradi quando giri il potenziometro.
Prendi un pezzo di carta più grande del servo con la freccia attaccata e disegna su di esso un semicerchio. Da un lato del cerchio, scrivi "Stai fuori". Dall'altra parte, "Entra" e "Bussa" a metà dell'arco. Metti il servo con la freccia sopra il foglio. Congratulazioni, hai un modo per dire alle persone quanto sei impegnato con i tuoi progetti!

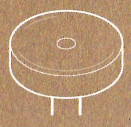




❶ Fissa una freccia di carta al braccio del servo.

❷ Disegna una base di carta e mettila sotto il servo.

06



PIEZO

FOTORESISTENZA

RESISTENZA DA 10 KILO OHM

INGREDIENTI

THEREMIN COMANDATO DALLA LUCE

È ORA DI FARE RUMORE! USANDO UNA FOTORESISTENZA E UN PIEZO, POTRAI FARE IL TUO THEREMIN COMANDATO DALLA LUCE

Scopri: produrre suoni con la funzione tone(), calibrazione di sensori analogici

Tempo: **45 MINUTI**
Livello: ■■■■□

Basato sui progetti: **1, 2, 3, 4**

Il *theremin* è uno strumento musicale che produce suoni in base ai movimenti delle mani del musicista. Probabilmente lo hai sentito in qualche film dell'orrore. Il theremin rileva la posizione delle mani del musicista in relazione a due antenne leggendo le variazioni di capacità sulle antenne, che sono collegate a un circuito analogico che crea il suono. Un'antenna controlla la frequenza del suono e l'altra controlla il volume. Sebbene Arduino non possa replicare esattamente i misteriosi suoni di questo strumento, è possibile emularli usando la funzione `tone()`. La Fig. 1 mostra le differenze tra gli impulsi emessi da `analogWrite()` e `tone()`. Questo permette a un trasduttore come uno speaker o un piezo di vibrare a velocità diverse.

Nota come è basso il segnale nella maggior parte del tempo, ma la frequenza è la stessa di PWM 200.

Nota come è alta la tensione nella maggior parte del tempo, ma la frequenza è la stessa di PWM 50.

Il ciclo di lavoro è al 50% (acceso metà del tempo, spento metà del tempo), ma cambia la frequenza.

Lo stesso ciclo di lavoro di Tone 440, ma il doppio della frequenza.

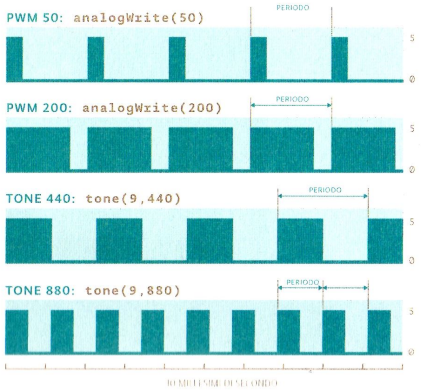


Fig. 1

Invece di misurare la capacità con Arduino, usa una fotoresistenza per rilevare la quantità di luce. Muovendo le mani sul sensore, cambia la quantità di luce che cade sulla fotoresistenza, come hai fatto nel Progetto 04. Il cambiamento di tensione sul piedino analogico determina la frequenza delle note. Collega le fotoresistenze ad Arduino usando un partitore di tensione come hai fatto nel Progetto 04. Probabilmente nei progetti precedenti hai notato che quando leggi questo circuito usando la funzione `analogRead()` le tue letture non coprono l'intervallo tra 0 e 1023. La resistenza fissa connessa a massa determina il valore più basso dell'intervallo e la luminosità della luce determina il valore massimo. Invece di accontentarsi di un intervallo limitato, calibra le letture del sensore prendendo il valore minore e maggiore e mappali alle frequenze sonore usando la funzione `map()` per ottenere un intervallo il più ampio possibile per il theremin. Questo aggiunge il vantaggio di adattare le letture del sensore quando sposti il circuito in un ambiente diverso, come una stanza con condizioni diverse di luce.

Un ***piezo*** è un piccolo elemento che vibra quando riceve elettricità. Quando si muove, sposta aria intorno a sé creando onde sonore.

COSTRUISCI IL CIRCUITO

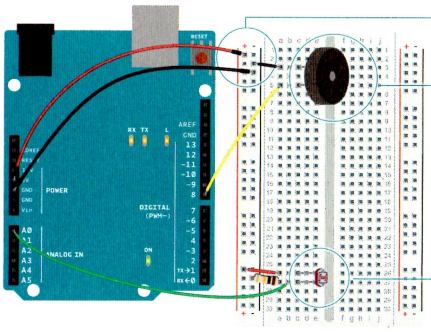


Fig. 2

1. Sulla breadboard, collega le linee esterne all'alimentazione e a massa.
2. Prendi il piezo e collegane un'estremità a massa e l'altra al piedino digitale 8 su Arduino.
3. Posiziona la fotoresistenza sulla breadboard, collegandone un'estremità a 5V. Collega l'altra estremità al piedino analogico 0 di Arduino e a massa attraverso una resistenza da 10 kilo ohm. Il circuito è lo stesso del partitore di tensione del Progetto 04.

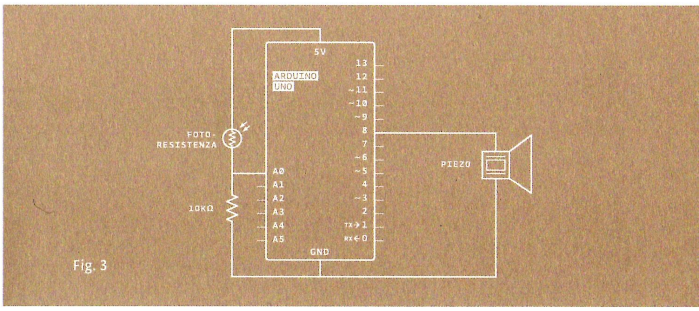


Fig. 3

I theremin tradizionali possono controllare la frequenza e il volume del suono. In questo esempio, controllerai solo la frequenza. Sebbene non si possa controllare il volume attraverso Arduino, è possibile cambiare manualmente il livello di tensione che giunge al piezo. Cosa accade se metti un potenziometro in serie con il piedino 8 e il piezo? E con un'altra fotoresistenza?

IL CODICE

Crea le variabili per calibrare il sensore

Crea una variabile per memorizzare il valore letto dalla fotoresistenza con `analogRead()`. Poi, crea variabili per i valori alto e basso. Imposta il valore iniziale nella variabile `sensorLow` a 1023 e il valore di `sensorHigh` a 0. Quando esegui il programma per la prima volta, confronta questi numeri alla lettura del sensore per trovare i reali valori massimi e minimi.

```
1 int sensorValue;
2 int sensorLow = 1023;
3 int sensorHigh = 0;
```

Dai un nome alla costante per l'indicatore di calibrazione

Crea una costante chiamata `ledPin`. Userai questo LED come indicatore del fatto che il sensore ha finito la calibrazione. Per questo progetto collegalo al piedino 13.

```
4 const int ledPin = 13;
```

Imposta la direzione del piedino digitale e accendilo

Nel `setup()`, cambia il `pinMode()` del ledPin a `OUTPUT` e accendi la luce.

```
5 void setup() {

6   pinMode(ledPin, OUTPUT);
7   digitalWrite(ledPin, HIGH);
```

Usa un ciclo while() per la calibrazione

Nei passi successivi occorre calibrare i valori minimi e massimi del sensore. Usa un'istruzione `while()` per eseguire un ciclo per 5 secondi. Il ciclo `while()` si esegue fino a che si verifica una certa condizione. In questo caso usa la funzione `millis()` per misurare il tempo. `millis()` riporta da quanti millesimi di secondo l'Arduino è acceso o resettato.

```
8   while (millis() < 5000) {
```

Comando while()
arduino.cc/while

Confronta i valori del sensore per la calibrazione

Nel ciclo, leggi il valore del sensore; se il valore è minore di `sensorLow` (inizialmente 1023), aggiorna questa variabile. Se è più grande di `sensorHigh` (inizialmente 0), aggiornalo.

```
9     sensorValue = analogRead(A0);
10    if (sensorValue > sensorHigh) {
11      sensorHigh = sensorValue;
12    }
13    if (sensorValue < sensorLow) {
14      sensorLow = sensorValue;
15    }
16  }
```

Indica che la calibrazione è finita

Quando sono passati 5 secondi, termina il ciclo `while()`. Spegni il LED attaccato al piedino 13. Usa i valori maggiore e minore del sensore appena registrato per calcolare la frequenza nella parte principale del programma.

```
17  digitalWrite(ledPin, LOW);
18 }
```

Leggi e immagazzina il valore del sensore

Nel `loop()`, leggi il valore su A0 e immagazzinalo in `sensorValue`.

Mappa il valore del sensore a una frequenza

Crea una variabile di nome `pitch`. Il valore di `pitch` viene calcolato a partire da `SensorValue`. Utilizza `sensorLow` e `sensorHigh` come i limiti per i valori in entrata. Come valori iniziali d'uscita, prova 50 e 4000. Questi numeri impostano l'intervallo di frequenze che genererà Arduino.

Suona la frequenza

Chiama la funzione `tone()` per riprodurre un suono. Ci vogliono tre argomenti: quale piedino suonare (in questo caso il piedino 8), quale frequenza suonare (determinata dalla variabile `pitch`) e per quanto tempo suonare la nota (per iniziare prova con 20 millesimi di secondo).

Chiama quindi la funzione `delay()` per 10 millesimi di secondo per creare un po' di stacco tra le note.

```
19 void loop() {
20   sensorValue = analogRead(A0);

21   int pitch =
       map(sensorValue,sensorLow,sensorHigh, 50, 4000);

22   tone(8,pitch,20);

23   delay(10);
24 }
```

USALO

Quando accendi Arduino, c'è una finestra di 5 secondi per calibrare il sensore. Per farlo, muovi la mano su e giù sopra la fotoresistenza modificando la quantità di luce che la raggiunge. Quanto più si replicano i movimenti che si prevede di fare mentre si suona lo strumento, migliore è la calibrazione.

Dopo 5 secondi, la calibrazione è completa e il LED su Arduino si spegne. Dovresti quindi sentire rumori provenienti dal piezo! Al variare della quantità di luce che cade sui sensori dovrebbe variare la frequenza prodotta dal piezo.

L'intervallo nella funzione `map()` che determina il suono è piuttosto ampio; prova a cambiare le frequenze per trovarne una giusta per il tuo stile musicale.

La funzione `tone()` opera in modo molto simile al PWM in `analogWrite()`, ma con una differenza significativa. In `analogWrite()` la frequenza è fissa; cambia il ciclo di lavoro degli impulsi in un dato periodo di tempo. Con `tone()` invi ancora impulsi, ma cambiando la loro frequenza. `tone()` manda sempre impulsi a un ciclo di lavoro del 50% (metà del tempo il piedino è acceso, l'altra metà del tempo è spento).

La funzione tone() ti offre la possibilità di generare frequenze diverse tramite un altoparlante o un piezo. Usando i sensori in un partitore di tensione, probabilmente non sarà possibile ottenere l'intera gamma di valori tra 0 e 1023. Calibrando i sensori, è possibile mappare gli ingressi a un intervallo adatto.

07

INGREDIENTI

TASTIERA MUSICALE

CON ALCUNE RESISTENZE E PULSANTI COSTRUISCI UNA PICCOLA TASTIERA MUSICALE

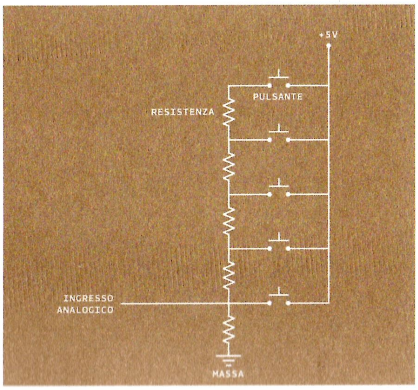
Scopri: la rete di resistenze a scala, gli array

Tempo: **45 MINUTI**
Livello: ■■■■■

Basato sui progetti: **1, 2, 3, 4, 6**

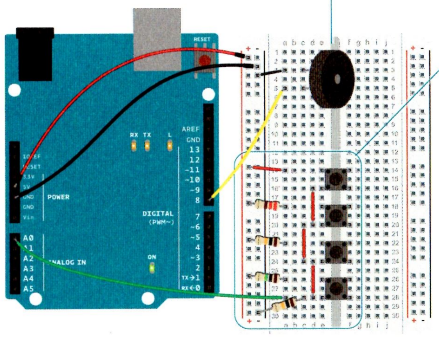
Anche se è possibile semplicemente collegare un numero di pulsanti a ingressi digitali per avere toni diversi, in questo progetto useremo una rete di resistenze a scala.

Questa permette di leggere più pulsanti usando un solo input analogico: è una tecnica utile se ti trovi a corto di ingressi digitali. Collega i pulsanti che sono connessi in parallelo al piedino analogico 0. La maggior parte di questi pulsanti è collegata all'alimentazione attraverso una resistenza. Quando premi ogni pulsante, un livello diverso di tensione viene applicato al piedino di ingresso. Se premi due bottoni insieme, hai un unico input basato sulla relazione tra due resistenze in parallelo.



Una rete di resistenze a scala e cinque pulsanti come ingresso analogico
Fig. 1

COSTRUISCI IL CIRCUITO



Questa disposizione di resistenze e pulsanti che alimenta un ingresso analogico è chiamata rete di resistenze a scala.
Fig. 2

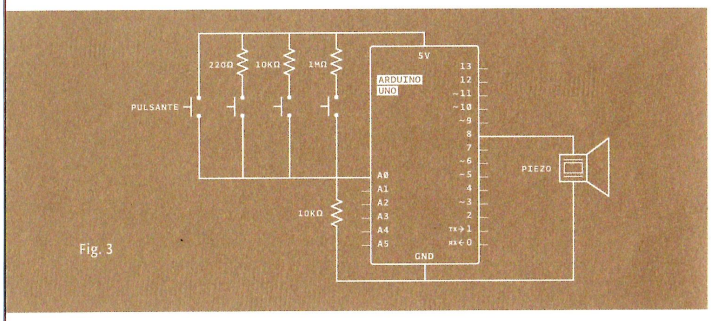
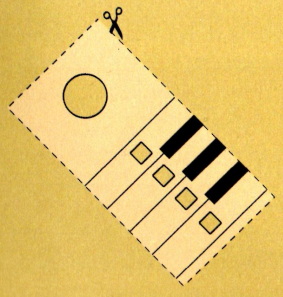


Fig. 3

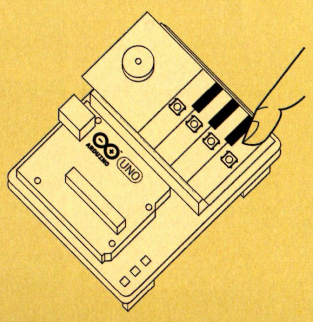
1. Collega la breadboard all'alimentazione e a massa come nei progetti precedenti. Collega un'estremità del piezo a massa. Collega l'altra estremità al piedino 8 di Arduino.

2. Posiziona i pulsanti sulla breadboard come mostrato nel circuito. Questa disposizione di resistenze e pulsanti che alimenta un ingresso analogico è chiamata rete di resistenze a scala. Collega la prima resistenza direttamente all'alimentazione. Collega il secondo, il terzo e il quarto pulsante all'alimentazione attraverso, rispettivamente, resistenze da 220 ohm, 10 kilo ohm e 1 mega ohm. Collega tutte le uscite dei pulsanti insieme in un unico punto. Collegalo a massa con una resistenza da 10 kilo ohm e collegalo anche al piedino analogico 0. Ognuna di queste resistenze si comporta da partitore di tensione.

Pensa a un contenitore per la tastiera. Mentre i vecchi sintetizzatori analogici avevano fili elettrici che spuntavano da tutte le parti, la tua tastiera è raffinata e digitale. Prepara un piccolo cartoncino per posizionare i pulsanti. Etichetta i tasti, così sai quale nota suona ogni tasto.



1. Disegna e taglia un pezzo di carta con buchi per i 4 pulsanti e il piezo. Decoralo per farlo assomigliare a una tastiera.



2. Metti il cartoncino sopra i pulsanti e il piezo. Divertiti con la tua creazione!

IL CODICE

L'array

In questo programma, hai bisogno di tenere una lista di frequenze che vuoi suonare quando schiacci ogni pulsante. Puoi iniziare con le frequenze per le note do, re, mi, fa centrali (262Hz, 294Hz, 330Hz e 349Hz). Per farlo, hai bisogno di un nuovo tipo di variabile chiamata array.

Un array è un modo per immagazzinare differenti valori che sono in relazione l'uno con l'altro, come le frequenze in una scala musicale, usando un solo nome. Sono uno strumento utile per avere accesso alle informazioni in modo veloce ed efficiente. Per dichiarare un array, inizia come faresti con una variabile, ma fai seguire al nome un paio di parentesi quadre: []. Dopo il segno uguale, metti gli elementi tra parentesi graffe.

Per leggere o cambiare gli elementi dell'array, fai riferimento ai singoli elementi usando il nome dell'array e l'indice di ciò che vuoi indirizzare. L'indice fa riferimento all'ordine nel quale compaiono gli elementi quando è creato l'array. Il primo elemento nell'array è 0, il secondo è 1 e così via.

```
int buttons[6];
// crea un array di 6 numeri interi

int buttons[0] = 2;
// assegna il valore 2 al primo elemento
//dell'array
```

Crea un array di frequenze

Crea un array di quattro note usando le frequenze indicate sopra. Rendi questo array una variabile globale dichiarandolo prima del `setup()`.

```
1 int notes[] = {262,294,330,349};
```

Comunicazione seriale

Nel `setup()`, inizia una comunicazione seriale con il computer.

```
2 void setup() {
3   Serial.begin(9600);
4 }
```

Leggi il valore analogico e invialo al monitor seriale

Nel `loop()`, dichiara una variabile locale per memorizzare il valore del piedino A0. Dato che ogni pulsante è collegato all'alimentazione tramite una resistenza di valore diverso, ognuno produce un diverso valore. Per vedere i valori, inviali al computer aggiungendo la riga `Serial.println(keyVal)`.

```
5 void loop() {
6   int keyVal = analogRead(A0);
7   Serial.println(keyVal);
```

Usa un'istruzione if()...else per determinare quale nota suonare

Usando un'istruzione `if()...else`, puoi assegnare ogni valore a un tono diverso. I numeri inclusi nel programma d'esempio sono valori approssimativi per queste resistenze. Siccome tutte le resistenze hanno una certa tolleranza, questi potrebbero non essere perfetti. Usa le informazioni dal monitor seriale per sistemarli se necessario.

```
8   if(keyVal == 1023){
9     tone(8, notes[0]);
10  }
```

Suona le note che corrispondono al valore analogico

Dopo ogni istruzione `if()`, chiama la funzione `tone()`. Il programma fa riferimento all'array per determinare quale frequenza suonare. Se il valore di AO corrisponde a una delle tue istruzioni `if()`, puoi dire ad Arduino di riprodurre un tono. È possibile che il tuo circuito sia un po' 'rumoroso' e i valori possono oscillare un po' mentre si preme un interruttore. Per adattarsi a queste variazioni, è buona norma verificare un breve intervallo di valori. Se si utilizza il confronto `"&&"`, è possibile controllare più condizioni per vedere se sono vere.

Se premi il primo pulsante, suona notes[0]. Se premi il secondo, suona notes[1] e se premi il terzo, suona notes[2]. Questo è uno dei momenti in cui gli array diventano veramenti utili.

```
11   else if(keyVal >= 990 && keyVal <= 1010){
12     tone(8, notes[1]);
13   }
14   else if(keyVal >= 505 && keyVal <= 515){
15     tone(8, notes[2]);
16   }
17   else if(keyVal >= 5 && keyVal <= 10){
18     tone(8, notes[3]);
19   }
```

Smetti di suonare il tono quando non è premuto nulla

Solo una frequenza può suonare su un piedino in un dato momento, quindi se premi più tasti, senti solo un suono.

Per interrompere le note quando nessun pulsante viene premuto, chiama la funzione **noTone()**, specificando il numero del piedino sul quale interrompere la riproduzione audio.

```
20   else{
21     noTone(8);
22   }
23 }
```

USALO

Se le resistenze sono vicine ai valori nel programma di esempio, quando premi i tasti dovresti sentire alcuni suoni dal piezo. In caso contrario, controlla il monitor seriale per assicurarti che ciascuno dei pulsanti sia in un intervallo che corrisponde alle note nella istruzione `if()...else`. Se stai ascoltando un suono stentoreo, prova ad aumentare un po' l'intervallo.

Premi più pulsanti insieme e vedi che tipo di valori ottieni sul monitor seriale. Usa questi nuovi valori per attivare anche più suoni. Prova con diverse frequenze per ampliare la tua produzione musicale. Trovi le frequenze delle note musicali su questa pagina: ***arduino.cc/frequencies***

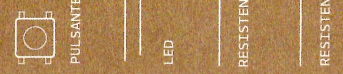
Se sostituisci i pulsanti e le scale di resistenze con sensori analogici, puoi usare le informazioni in più che ti forniscono per creare uno strumento più dinamico? Potresti usare il valore per cambiare la durata di una nota o, come nel progetto con il theremin, creare una scala di suoni.

La funzione `tone()` è divertente per generare suoni, ma ha alcune limitazioni. Può creare solo onde quadre, non sinusoidi o triangoli. Le onde quadre non sembrano davvero delle onde. Come hai visto nella Fig. 1 del Progetto 06, sono una serie di impulsi acceso/spento.

Prima di creare il tuo gruppo musicale tieni presente alcune cose: si può suonare solo un tono alla volta e `tone()` interferisce con `analogWrite()` sui piedini 3 e 11.

Gli array sono utili per raggruppare tipi simili di informazioni; vi si accede da un numero di indice che si riferisce ai singoli elementi. Le scale di resistenze sono un modo semplice per aumentare gli ingressi digitali di un sistema collegandosi a un ingresso analogico.

08



INGREDIENTI

CLESSIDRA DIGITALE

IN QUESTO PROGETTO, COSTRUIRAI UNA CLESSIDRA DIGITALE CHE ACCENDE UN LED OGNI DIECI MINUTI. SAPRAI PER QUANTO TEMPO STAI LAVORANDO SUI TUOI PROGETTI USANDO IL TIMER INCORPORATO IN ARDUINO

Scopri: il tipo di dati long, creare un timer

Tempo: **30 MINUTI**
Livello: ■■■■■

Basato sui progetti: **1, 2, 3, 4**

Finora con Arduino, quando hai voluto che accadesse qualcosa in uno specifico intervallo di tempo, hai usato la funzione delay(): è pratica, ma un po' limitante. Quando Arduino chiama delay(), blocca il suo stato corrente per la durata del ritardo. Significa che non ci sono altri ingressi o uscite quando è in attesa. I ritardi non sono molto utili per tenere traccia del tempo. Se vuoi fare qualcosa ogni 10 secondi, avere un delay() di 10 secondi è piuttosto scomodo.

La funzione `millis()` aiuta a risolvere questi problemi. Tiene traccia del tempo in cui Arduino è funzionante in millesimi di secondo. L'hai usato precedentemente nel Progetto 06 quando hai creato un timer per la calibrazione.

Finora hai dichiarato le variabili come `int`. Un `int` (intero) è un numero a 16 bit, che contiene valori tra -32768 e 32767. Possono sembrare numeri grandi, ma Arduino conta 1000 volte al secondo con la funzione `millis()` e quindi esaurisci i numeri in poco tempo. Il tipo di dato `long` contiene un numero da 32 bit (tra -2147483648 e 2147483647). Dal momento che non è possibile tornare indietro per ottenere i numeri negativi, la variabile per immagazzinare il tempo `millis()` è chiamata `unsigned long`. Quando un tipo di dato è chiamato unsigned, è solo positivo. Questo ti permette di contare ancora di più. Un `unsigned long` può contare fino a 4294967295. C'è abbastanza spazio per far sì che la funzione `millis()` tenga traccia del tempo per almeno 50 giorni. Confrontando il valore attuale di `millis()` a un valore specifico, puoi vedere se è passato un certo periodo di tempo.

Quando capovolgi la clessidra, un interruttore di inclinazione cambierà il suo stato e prenderà il via un nuovo ciclo di accensione dei LED.

L'interruttore di inclinazione funziona come un normale interruttore nel senso che è un sensore acceso/spento. Qui lo userai come ingresso digitale.

Ciò che rende unico l'interruttore di inclinazione è che rileva l'orientamento. Normalmente ha una piccola cavità all'interno del corpo che contiene una sfera di metallo. Quando è inclinato in modo corretto, la sfera rotola su un lato della cavità e collega i due terminali che sono sulla breadboard chiudendo l'interruttore.

COSTRUISCI IL CIRCUITO

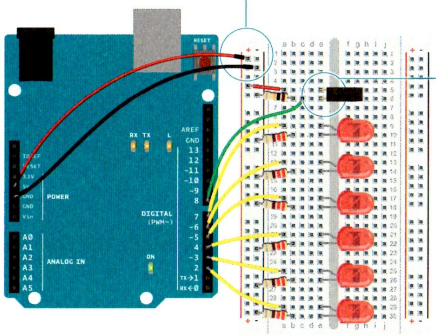


Fig. 1

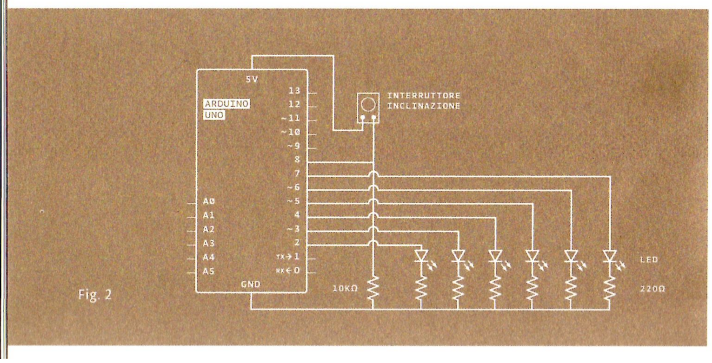


Fig. 2

1. Collega alimentazione e massa alla breadboard.
2. Collega l'anodo (la gamba più lunga) dei sei LED ai piedini digitali 2-7. Collega i LED a massa attraverso una resistenza da 220 ohm.
3. Collega una estremità dell'interruttore di inclinazione a 5V. Collega l'altra a una resistenza da 10 kilo ohm a massa. Collega il punto dove si incontrano al piedino digitale 8.

Non c'è bisogno che Arduino sia collegato al computer per funzionare. Prova a costruire un supporto con del cartoncino o del polistirolo e alimentare Arduino con una batteria per creare una clessidra portatile. Puoi creare una copertura con delle cifre a fianco delle luci.

Gli ***interruttori di inclinazione*** sono strumenti ottimi ed economici per determinare l'orientamento di qualcosa.
Gli ***accelerometri*** sono altri tipi di sensori di inclinazione, ma forniscono molte più informazioni. Sono anche significativamente più costosi. Se ti serve controllare se qualcosa è su o giù, un sensore di inclinazione è più che sufficiente.

IL CODICE

Dichiara una costante

Hai bisogno di alcune variabili globali nel tuo programma. Per iniziare, crea una costante chiamata `switchPin`. Questo è il nome del piedino al quale è connesso l'interruttore di inclinazione.

```
1 const int switchPin = 8;
```

Crea una variabile per memorizzare il tempo

Crea una variabile di tipo `unsigned long`. Questa memorizza il momento in cui è stato modificato l'ultimo LED.

```
2 unsigned long previousTime = 0;
```

Variabili per gli ingressi e le uscite

Crea una variabile per lo stato dell'interruttore e un'altra per memorizzare il precedente stato dell'interruttore. Userai queste due variabili per confrontare la posizione dell'interruttore da un ciclo al successivo.
Crea una variabile chiamata `led`, che verrà usata per verificare quale LED è il prossimo ad accendersi. Inizia con il piedino 2.

```
3 int switchState = 0;
4 int prevSwitchState = 0;

5 int led = 2;
```

Dichiara una variabile descrivendo l'intervallo tra gli eventi

L'ultima variabile da creare è l'intervallo di accensione di ogni LED. Questo è un tipo di dato `long`. In 10 minuti (il tempo tra l'accensione di ogni LED) ci sono 600000 millesimi di secondo. Se vuoi che il ritardo tra le luci sia più lungo o corto, questo è il numero da cambiare.

```
6 long interval = 600000;
```

Imposta la direzione dei tuoi piedini digitali

Nel `setup()`, hai bisogno di dichiarare come output i piedini dei LED 2-7. Un ciclo `for()` li dichiara tutti e sei come `OUTPUT` con solo 3 linee di codice. Hai bisogno anche di dichiarare switchPin come `INPUT`.

```
7 void setup() {
8   for(int x = 2;x<8;x++){
9     pinMode(x, OUTPUT);
10  }

11  pinMode(switchPin, INPUT);
12 }
```

Controlla da quanto tempo è stato avviato il programma

Quando inizia il `loop()`, misura la quantità di tempo in cui Arduino è stato in funzione con `millis()` e immagazzinalo in una variabile locale chiamata `currentTime`.

```
13 void loop(){
14   unsigned long currentTime = millis();
```

Valuta la quantità di tempo trascorso dal precedente loop()

Usando un'istruzione `if()`, controlla se è passato abbastanza tempo per accendere un LED. Sottrai `currentTime` da `previousTime` e controlla se è maggiore della variabile intervallo. Se sono passati 600000 millesimi di secondo (10 minuti), imposta la variabile `previousTime` al valore di `currentTime`.

```
15   if(currentTime - previousTime > interval) {
16     previousTime = currentTime;
```

Accendi un LED, prepara il prossimo

`previousTime` indica l'ultima volta in cui è stato acceso un LED. Quando hai impostato `previousTime`, accendi il LED e incrementa la variabile `led`. La prossima volta che passi l'intervallo di tempo, il LED successivo si accende.

```
17    digitalWrite(led, HIGH);
18    led++;
```

Verifica se tutte le luci sono accese

Aggiungi una istruzione `if` in più nel programma per controllare se il LED sul piedino 7 si è acceso. Non farci ancora nulla. Potrai decidere dopo cosa succede alla fine dell'ora.

```
19    if(led == 7){
20    }
21  }
```

Leggi il valore dell'interruttore

Dopo aver controllato il tempo, guarda se l'interruttore ha cambiato il suo stato. Scrivi il valore dell'interruttore nella variabile `switchState`.

```
22  switchState = digitalRead(switchPin);
```

Azzera le variabili se necessario

Con una istruzione `if()`, verifica se l'interruttore si trova in una posizione diversa da quella precedente. Il `!=` controlla se `switchState` è diverso da `prevSwitchState`. Se sono diversi, spegni i LED, riporta la variabile `led` al primo piedino e azzera il timer per i LED impostando `previousTime` al valore di `currentTime`.

```
23  if(switchState != prevSwitchState){
24    for(int x = 2;x<8;x++){
25      digitalWrite(x, LOW);
26    }

27    led = 2;
28    previousTime = currentTime;
29  }
```

Imposta lo stato corrente allo stato precedente

Alla fine del `loop()`, salva lo stato dell'interruttore in `prevSwitchState`, così puoi confrontarlo al valore di `switchState` nel `loop()` successivo.

```
30  prevSwitchState = switchState;
31}
```

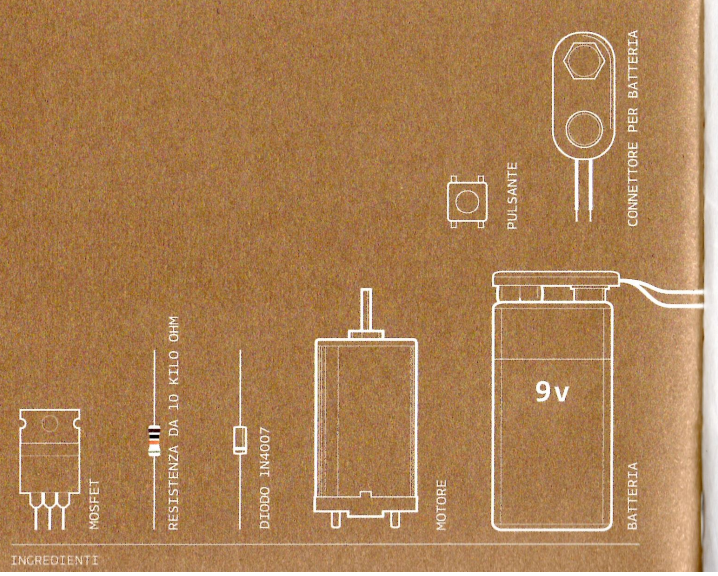
USALO

Una volta che hai programmato la scheda controlla l'ora su un orologio. Dopo che sono passati 10 minuti, il primo LED dovrebbe essersi acceso. Ogni 10 minuti si accenderà un altro LED. Alla fine di un'ora, tutte le 6 luci dovrebbero essere accese. Quando ribalti il circuito e fai cambiare stato all'interruttore di inclinazione, le luci si spegneranno e il timer ripartirà.

Quando l'orologio raggiunge un'ora e tutte le 6 luci sono accese, rimangono accese. Puoi pensare a un modo per attirare l'attenzione quando l'ora è passata? Suoni o il lampeggiare di luci sono entrambi buoni indicatori. La variabile led può essere controllata per vedere se tutte le luci sono accese; questo è un buon modo per catturare l'attenzione di qualcuno. Diversamente da una clessidra riempita di sabbia, le luci vanno su o giù a seconda dell'orientamento dell'interruttore. Riesci a capire come utilizzare la variabile switchState per indicare in quale direzione dovrebbero andare le luci?

Per misurare il tempo tra eventi, usa la funzione millis(). Dato che i numeri che genera sono più grandi di quelli che puoi memorizzare in un int, dovresti usare il tipo di dato unsigned long per immagazzinarne i valori.

09



INGREDIENTI

GIRANDOLA MOTORIZZATA

USA ARDUINO PER FAR RUOTARE UNA GIRANDOLA COLORATA UTILIZZANDO UN MOTORE

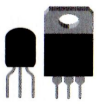
Scopri: transistor, forti carichi di corrente e di tensione

Tempo: **45 MINUTI**
Livello: ■■■□□

Basato sui progetti: **1, 2, 3, 4**

Per varie ragioni controllare motori con Arduino è molto più complicato che controllare i LED. Innanzitutto, i motori richiedono molta più corrente di quanta ne possano fornire i piedini d'uscita di Arduino e i motori possono generare corrente – attraverso un processo chiamato induzione – che può danneggiare il circuito. Comunque, i motori fanno muovere le cose, rendendo i progetti molto più interessanti. Ben vengano quindi le complicazioni!

Muovere le cose richiede molta energia; molta di più di quanto possa fornire Arduino. Alcuni motori hanno bisogno anche di una tensione più alta. All'inizio del movimento, e quando si ha un carico pesante attaccato, il motore assorbe quanta più energia possibile. Arduino può fornire solo 40 milliampere (mA) dai suoi piedini digitali, molto meno di quanto la maggior parte dei motori ha bisogno per lavorare.



I ***transistor*** sono componenti che permettono di controllare alti carichi di corrente e di tensione tramite la bassa corrente di uscita di un piedino di Arduino. Ce ne sono molti tipi diversi, ma tutti si basano sullo stesso principio. Puoi pensare a un transistor come a un interruttore digitale. Quando fornisci tensione a uno dei piedini del transistor, chiamato gate (porta), si chiude il circuito tra gli altri due piedini, chiamati source (sorgente) e drain (scarico). Così puoi accendere e spegnere un motore di elevata tensione o corrente con Arduino.

I ***motori*** sono dispositivi induttivi. L'induzione è un processo per cui una corrente elettrica variabile che scorre in un filo è in grado di generare un campo magnetico variabile. Quando si dà elettricità a un motore, una bobina di filo di rame avvolta strettamente all'interno del motore crea un campo magnetico. Questo campo fa girare l'asse, la parte che sporge dal motore.

È vero anche il contrario: un motore può generare elettricità quando l'albero motore viene fatto ruotare. Collega un LED ai due fili del motore, quindi fai girare l'albero con la mano. Se non succede niente, giralo dall'altra parte. Il LED si dovrebbe accendere. Hai appena realizzato un generatore con il tuo motore. Quando smetti di fornire energia elettrica al motore, questo continua a girare per inerzia. Girando, il motore genera una tensione nella direzione opposta alla corrente che hai fornito. Hai visto questo risultato quando il motore ha acceso il LED. Questa tensione inversa, a volte chiamata ***controtensione***, può danneggiare il transistor. Per questa ragione, dovresti mettere un diodo in parallelo al motore, così la controtensione passa attraverso il diodo, che permette all'elettricità di scorrere in una direzione, proteggendo il resto del circuito.

COSTRUISCI IL CIRCUITO

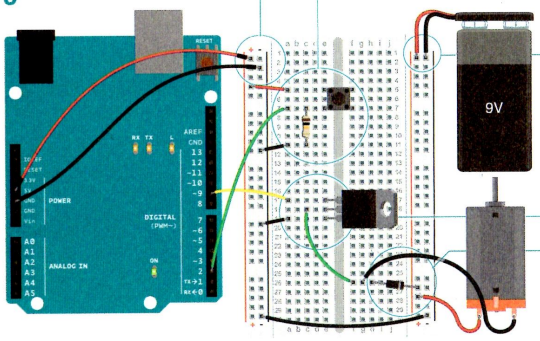


Fig. 1

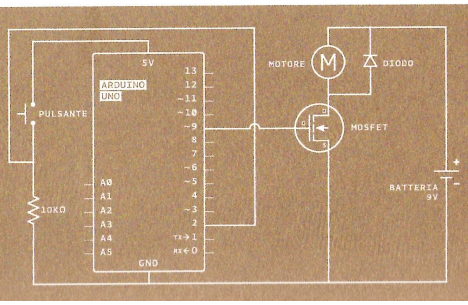


Fig. 2

1. Collega alimentazione e massa alla breadboard attraverso Arduino.

2. Aggiungi alla scheda un pulsante, connettendo un lato all'alimentazione e l'altro lato al piedino digitale 2 di Arduino. Aggiungi una resistenza pull-up da 10 kilo ohm a massa sul piedino di uscita dell'interruttore.

3. Quando usi circuiti con tensioni differenti, devi collegare tra loro le masse per creare una massa comune. Collega il connettore per batteria 9V alla breadboard. Collega la massa della batteria alla massa di Arduino sulla breadboard con un ponticello come mostrato in Fig. 1. Poi attacca l'estremità libera del motore all'alimentazione 9V.

4. Metti il transistor sulla breadboard. Verifica che la linguetta di metallo sia dalla parte opposta a te. Collega il piedino digitale 9 al piedino di sinistra del transistor. Questo piedino è chiamato **gate** (**porta**). Una tensione sul gate crea un collegamento tra gli altri due piedini. Collega un'estremità del motore al piedino centrale del transistor. Questo piedino è chiamato **drain** (**scarico**). Quando Arduino attiva il transistor, fornendo tensione al gate, questo piedino si collega al terzo piedino, chiamato **source** (**sorgente**). Collega source a massa.

5. Poi, collega l'alimentatore del motore al motore e alla breadboard. L'ultimo componente da aggiungere è il diodo. Il diodo è un componente polarizzato: va inserito nel circuito solo in una specifica direzione. Nota che il diodo ha una striscia su una estremità: è il polo negativo, o catodo, del diodo. L'altro è il polo positivo o anodo. Collega l'anodo del diodo a massa del motore e il catodo del diodo all'alimentazione del motore. Guarda la Fig. 1. Sembra al rovescio, e infatti lo è. Il diodo ti aiuta a prevenire ogni controtensione generata dal motore che rientra nel circuito. Ricorda, la controtensione viene generata nella direzione opposta alla tensione che fornisci.

Anche i LED sono diodi, nel caso ti stupissi del fatto che anche le loro estremità si chiamano anodi e catodi. Ci sono molti tipi di diodi, ma hanno tutti in comune un elemento: permettono alla corrente di fluire dall'anodo al catodo, ma non il contrario.

IL CODICE

Dai un nome alle costanti e alle variabili

Il codice è molto simile a quello che hai usato la prima volta per accendere un LED. Prima di tutto, imposta le costanti per il pulsante e i piedini del motore e una variabile chiamata `switchState` per memorizzare il valore dello switch (pulsante).

```
const int switchPin = 2;
const int motorPin = 9;
int switchState = 0;
```

Dichiara la direzione dei piedini

Nel `setup()`, dichiara il `pinMode()` dei piedini del motore (`OUTPUT`) e dell'interruttore (`INPUT`).

```
void setup() {
  pinMode(motorPin, OUTPUT);
  pinMode(switchPin, INPUT);
}
```

Leggi l'ingresso, attiva l'uscita se viene premuto

Il `loop()` è semplice. Controlla lo stato di `switchPin` con `digitalRead()`.

Se è premuto l'interruttore, imposta il motorPin a `HIGH`. Se non è premuto, imposta il piedino a `LOW`. Quando è `HIGH`, il transistor si attiva, completando il circuito. Quando è `LOW`, il motore è spento.

```
void loop(){
  switchState = digitalRead(switchPin);

  if (switchState == HIGH) {
    digitalWrite(motorPin, HIGH);
  }
  else {
    digitalWrite(motorPin, LOW);
  }
}
```

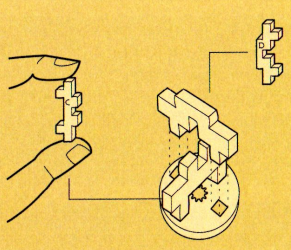
I motori hanno una tensione di esercizio ottimale. Lavorano anche con meno del 50% della tensione nominale e fino al 50% in più rispetto a quel valore. Se si varia la tensione, è possibile modificare la velocità di rotazione del motore. Non variarla troppo, però, o si brucia il motore.

I motori richiedono una particolare attenzione quando sono controllati da un microcontrollore; questo, infatti, non può fornire abbastanza corrente e/o tensione per alimentare un motore. Inoltre è bene utilizzare diodi per evitare di danneggiare il circuito.

I transistor sono dispositivi a stato solido, non hanno parti in movimento. Per questo motivo, è possibile accenderli e spegnerli molto rapidamente. Prova a collegare un potenziometro a un ingresso analogico e usalo per modulare la larghezza di impulso (PWM) del piedino che controlla il transistor. Cosa pensi che succeda alla velocità del motore se si varia la tensione? Utilizzando vari motivi sul disco, si possono ottenere diversi effetti visivi?

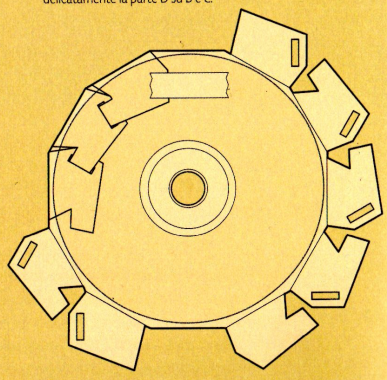
USALA

Assembla il mozzo per il CD come mostrato al punto 1 e attaccalo al motore come indicato al punto 2. Attacca al CD il cartoncino ritagliato come mostrato al punto 3. Incastra il CD al mozzo e assicuralo con un po' di colla. Fai una prova prima di procedere. Collega una pila da 9V al connettore per batteria. Alimenta Arduino con una USB. Quando premi l'interruttore sulla breadboard, il motore gira molto rapidamente.



1

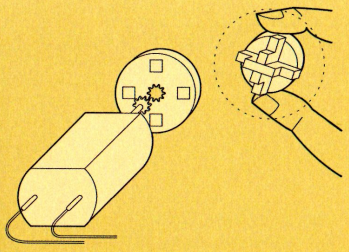
Incastra la parte C nella parte B, e quindi premi delicatamente la parte D su B e C.



3

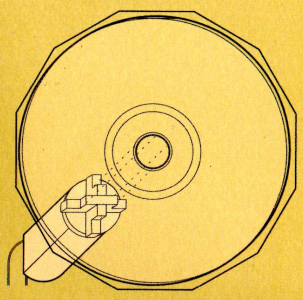
Inserisci il disco di carta sul CD e fissalo con le linguette sul retro.

Con il motore che gira così velocemente, puoi fare una girandola piuttosto grande. Fai attenzione a che non voli via e che non colpisca qualcuno negli occhi. Prova diversi motivi all'esterno per creare diversi effetti visivi.



2

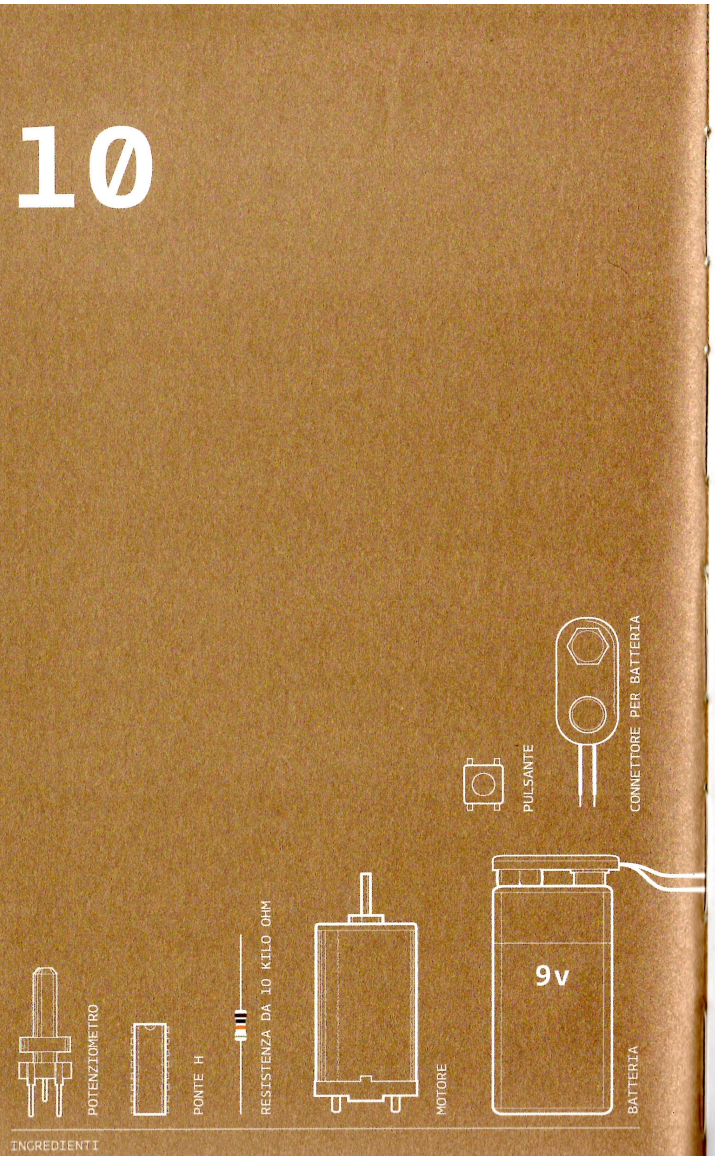
Premi delicatamente l'albero motore nel foro nella parte posteriore della parte B.



4

Fissa il CD alla croce formata dalle parti B e D. Utilizza un po' di colla per evitare che esca il CD.

10



ZOOTROPIO

CREA IMMAGINI IN MOVIMENTO CON ARDUINO COLLEGANDO UN MOTORE A UN PONTE H E AD ALCUNE IMMAGINI FISSE

Scopri: i ponti H

Tempo: **30 MINUTI**
Livello: ■■■■■

Basato sui progetti: **1, 2, 3, 4, 9**

Prima ancora di internet, della televisione e del cinema, le immagini in movimento erano state create con uno strumento chiamato zootropio. Lo zootropio crea l'illusione del movimento a partire da un gruppo di immagini fisse che si distinguono l'una dalle altre per poche differenze. Normalmente è un cilindro con delle fessure ai lati. Quando il cilindro gira e guardi attraverso le fessure, i tuoi occhi percepiscono le immagini fisse come animate. Le fessure aiutano a evitare che le immagini siano sfocate e la velocità con cui appaiono le immagini spiega perché sembrano muoversi. Originariamente, era fatto a mano o con un meccanismo a manovella.

In questo progetto, costruirai il tuo zootropio che anima una pianta carnivora. Creerai il movimento con un motore. Per rendere il sistema ancora più avanzato, aggiungerai un interruttore che consente di controllare la direzione, un altro per spegnerlo e accenderlo e un potenziometro per il controllo della velocità.

Nel progetto della girandola motorizzata avevi un motore da girare in una sola direzione. Se si dovesse invertire l'alimentazione e la massa del motore, il motore girerebbe nella direzione opposta. Non è molto pratico farlo ogni volta che si desidera far girare qualcosa in una direzione diversa, quindi dovrai utilizzare un componente chiamato ponte H per invertire la polarità del motore.

I ***ponti H*** sono componenti conosciuti come ***circuiti integrati (IC)***. I circuiti integrati sono componenti che contengono circuiti complessi in un contenitore molto piccolo. Questi possono contribuire a semplificare i circuiti più complessi inserendoli in un componente facilmente sostituibile. Per esempio, il ponte H che stai utilizzando in questo progetto contiene un certo numero di transistor. Per costruire il circuito contenuto all'interno del ponte H avresti bisogno di un'altra breadboard.

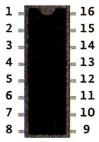


Fig. 1

Puoi accedere al circuito contenuto nel circuito integrato attraverso i piedini che escono dai lati. Circuiti integrati diversi hanno un diverso numero di piedini e non tutti sono utilizzati in ogni circuito. A volte è conveniente riferirsi ai piedini per numero piuttosto che con la funzione. Guardando un circuito integrato, la parte con la tacca è la parte in alto. Puoi identificare i numeri dei piedini contando dall'alto a sinistra in senso antiorario come in Fig. 1.

COSTRUISCI IL CIRCUITO

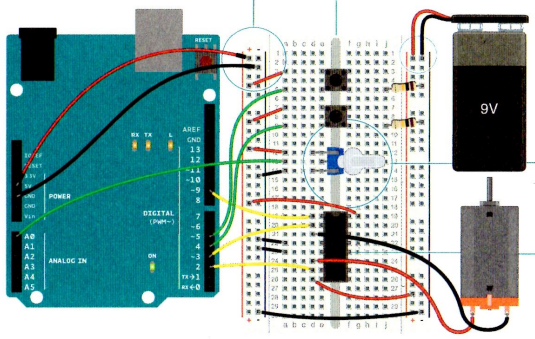


Fig. 2

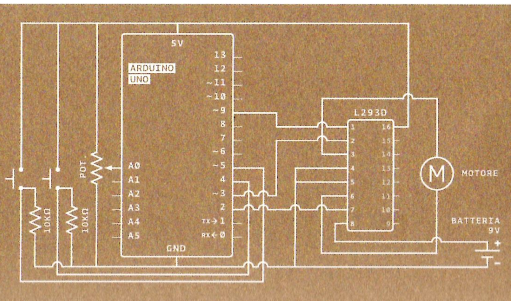


Fig. 3

1. Collega alimentazione e massa da un lato della breadboard ad Arduino.

2. Aggiungi 2 pulsanti alla breadboard collegando un lato di ciascuno all'alimentazione. Aggiungi una resistenza pull-down da 10 kilo ohm in serie con la massa sul piedino di uscita di entrambi gli interruttori. L'interruttore sul piedino 4 controlla la direzione, l'interruttore sul piedino 5 accende e spegne il motore.

3. Collega il potenziometro alla breadboard. Collega 5V da un lato e la massa dall'altro. Collega il piedino centrale all'input analogico 0 sull'Arduino. Questo serve per controllare la velocità del motore.

4. Metti il ponte H sulla breadboard in modo che sia al centro (vedi la Fig. 2 per il dettaglio della posizione). Collega il piedino 1 del ponte H al piedino digitale 9 di Arduino. Questo è il piedino di abilitazione sul ponte H. Quando riceve 5V accende il motore, quando riceve 0V spegne il motore. Usa questo piedino per la modulazione di larghezza di impulso del ponte H e regolare la velocità del motore.

5. Collega il piedino 2 del ponte H al piedino digitale 3 di Arduino. Collega il piedino 7 al piedino digitale 2. Questi piedini servono per comunicare con il ponte H, comunicandogli in quale direzione girare. Se il piedino 3 è `LOW` e il piedino 2 è `HIGH`, il motore gira in una direzione. Se il piedino 2 è `LOW` e il piedino 3 è `HIGH`, il motore gira nella direzione opposta. Se entrambi i piedini sono `HIGH` o `LOW` nello stesso tempo, il motore si ferma.

6. Il ponte H si alimenta dal piedino 16 (vedi Fig. 1); collegalo a 5V. I piedini 4 e 5 vanno entrambi a massa.

7. Collega il motore ai piedini 3 e 6 del ponte H. Questi due piedini si accendono o si spengono a seconda dei segnali che mandi ai piedini 2 e 7.

8. Inserisci il connettore della batteria (senza la batteria collegata!) alla massa sulla breadboard. Collega la massa da Arduino alla massa della batteria. Collega il piedino 8 del ponte H all'alimentazione della batteria. Questo è il piedino dal quale il ponte H alimenta il motore. Assicurati di non avere collegate tra loro le linee di alimentazione 9V e 5V. Devono essere separate, solo le masse devono essere collegate tra loro.

IL CODICE

Crea le costanti

Crea le costanti per i piedini di ingresso e uscita.

```
1 const int controlPin1 = 2;
2 const int controlPin2 = 3;
3 const int enablePin = 9;
4 const int directionSwitchPin = 4;
5 const int onOffSwitchStateSwitchPin = 5;
6 const int potPin = A0;
```

Crea le variabili per ricordare lo stato del programma

Usa le variabili per memorizzare i valori degli ingressi. Potrai fare la rilevazione del cambio di stato per entrambi gli interruttori, confrontando lo stato da un loop all'altro, come nel progetto della clessidra. Oltre a memorizzare lo stato attuale, è necessario registrare il precedente stato di ogni interruttore.

```
7 int onOffSwitchState = 0;
8 int previousOnOffSwitchState = 0;
9 int directionSwitchState = 0;
10 int previousDirectionSwitchState = 0;
```

Crea le variabili per il controllo del motore

`motorDirection` tiene traccia della direzione in cui gira il motore e `motorPower` tiene traccia se il motore sta girando oppure no.

```
11 int motorEnabled = 0;
12 int motorSpeed = 0;
13 int motorDirection = 1;
```

Dichiara i piedini digitali come input e output

Nel `setup()`, imposta la direzione di ogni piedino di ingresso e uscita.

```
14 void setup(){
15   pinMode(directionSwitchPin, INPUT);
16   pinMode(onOffSwitchStateSwitchPin, INPUT);
17   pinMode(controlPin1, OUTPUT);
18   pinMode(controlPin2, OUTPUT);
19   pinMode(enablePin, OUTPUT);
```

Spegni il motore

Imposta il piedino di abilitazione a `LOW` per iniziare, in modo che il motore non giri subito.

```
20   digitalWrite(enablePin, LOW);
21 }
```

Leggi il sensore

Nel tuo `loop()`, leggi lo stato dell'interruttore acceso/spento e immagazzinalo nella variabile `onOffSwitchState`.

```
22 void loop(){
23   onOffSwitchState =
       digitalRead(onOffSwitchStateSwitchPin);
24   delay(1);
25   directionSwitchState =
       digitalRead(directionSwitchPin);
26   motorSpeed = analogRead(potPin)/4;
```

Controlla se il sensore acceso/spento è cambiato

Se c'è una differenza tra lo stato attuale e precedente dell'interruttore e l'interruttore è attualmente **HIGH**, imposta la variabile `motorPower` a 1. Se è **LOW**, imposta la variabile a 0. Leggi i valori del pulsante di direzione e del potenziometro. Conserva i valori nelle loro rispettive variabili.

```
27  if(onOffSwitchState != previousOnOffSwitchState){
28    if(onOffSwitchState == HIGH){
29      motorEnabled = !motorEnabled;
30    }
31  }
```

Controlla se la direzione è cambiata

Controlla se il pulsante di direzione è in una posizione differente rispetto a prima. Se è diversa, cambia la variabile di direzione del motore. Ci sono solo 2 modi in cui il motore può girare, alterna così la variabile tra i due stati. Per raggiungere questo obiettivo utilizza l'operatore di inversione in questo modo: `motorDirection =!motorDirection`.

```
32   if (directionSwitchState !=
     previousDirectionSwitchState) {
33   if (directionSwitchState == HIGH) {
34     motorDirection = !motorDirection;
35    }
36  }
```

Cambia i piedini per far girare il motore nella giusta direzione

La variabile `motorDirection` determina in quale direzione il motore sta girando. Per determinare la direzione, imposta i piedini di controllo, uno **HIGH** e l'altro **LOW**. Quando `motorDirection` cambia, inverte gli stati dei piedini di controllo.

Se il pulsante di direzione viene premuto, gira il motore nella direzione opposta invertendo lo stato del `controlPin`.

```
37  if (motorDirection == 1) {
38    digitalWrite(controlPin1, HIGH);
39    digitalWrite(controlPin2, LOW);
40  }

41  else {
42    digitalWrite(controlPin1, LOW);
43    digitalWrite(controlPin2, HIGH);
44  }
```

Modula il motore se è attivo

Se la variabile `motorEnabled` è 1, imposta la velocità del motore usando la funzione `analogWrite()` per modulare a larghezza di impulso il piedino. Se `motorEnabled` è 0, spegni il motore impostando il valore `analogWrite` a 0.

```
45  if (motorEnabled == 1) {
46    analogWrite(enablePin, motorSpeed);
47  }
48  else {
49    analogWrite(enablePin, 0);
50  }
```

Salva lo stato attuale per il prossimo loop()

Prima di uscire dal `loop()`, salva lo stato attuale degli interruttori come stato precedente per la prossima parte del programma.

```
51  previousDirectionSwitchState =
      directionSwitchState;
52  previousOnOffSwitchState = onOffSwitchState;
53 }
```

USALO

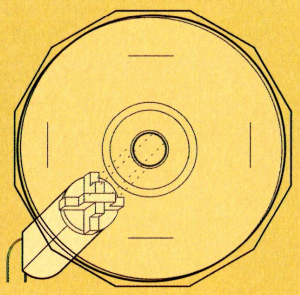
Dopo aver verificato che il circuito funziona come previsto, scollega la batteria e il cavo USB dal circuito.

Collega Arduino al computer. Collega la batteria al connettore. Quando premi l'interruttore acceso/spento, il motore dovrebbe iniziare a girare. Se giri il potenziometro, dovrebbe accelerare e rallentare. Premendo il tasto acceso/spento un'altra volta si ferma il motore. Prova a premere il tasto di direzione e verifica che il motore gira in entrambe le direzioni. Inoltre, se ruoti la manopola del potenziometro dovresti vedere la velocità del motore aumentare o diminuire a seconda del valore che sta inviando.

Per costruire il tuo zootropio, prendi la girandola che hai utilizzato nel progetto 09 e il ritaglio con le fessure verticali che è incluso nel kit. Una volta che il CD è saldamente collegato all'albero motore, collega tutto. Tieni il progetto in alto, in modo da poter guardare attraverso le fessure (ma assicurati che il CD sia fissato ma non troppo vicino al motore). Dovresti vedere la sequenza di immagini fisse "in movimento"! Se lo zootropio sta andando troppo velocemente o troppo lentamente, ruota la manopola del potenziometro per regolare la velocità dell'animazione.

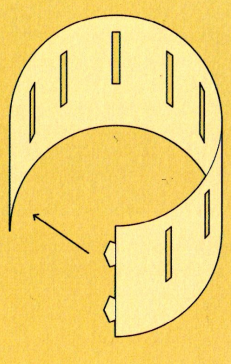
Prova a premere l'interruttore che cambia la direzione per vedere come appare l'animazione riprodotta al contrario. Lo zootropio e le immagini fornite nel kit sono solo il punto di partenza: prova a sperimentare con le tue animazioni, utilizzando il ritaglio come riferimento.

Inizia con una immagine semplice. Individua un punto fisso in essa, e fai piccoli cambiamenti in ogni frame. Cerca gradualmente di tornare all'immagine originale in modo che l'animazione prosegua in un ciclo continuo.



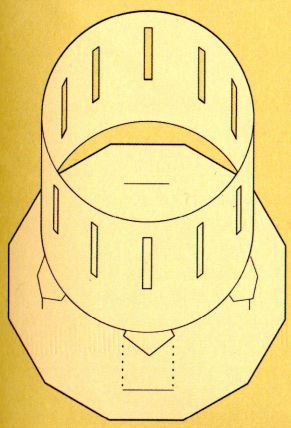
❶

Fissa il CD sulla base di legno. Aggiungi un po' di colla per assicurarti che non si stacchi quando si avvia il motore.



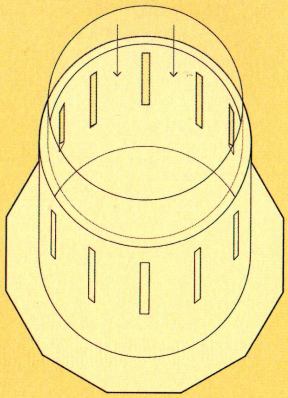
❷

Usa le linguette per chiudere il cartone, formando un cerchio.



❸

Inserisci le quattro linguette nella base dello zootropio.



❹

Inserisci la striscia di carta con le immagini nello zootropio.

Gli zootropi funzionano grazie a un fenomeno chiamato "persistenza della visione" (in inglese abbreviato con l'acronimo POV): l'illusione del movimento che si crea quando i nostri occhi osservano le immagini fisse, con minime variazioni in rapida successione. Se ricerchi online "POV Display", trovi molti progetti che utilizzano questo effetto, molti dei quali con i LED e con Arduino.

Fai una base per sostenere il motore. Una scatola di cartone di piccole dimensioni con un foro potrebbe funzionare come base, lasciando le mani libere di giocare con gli interruttori e le manopole. In questo modo sarà più facile mostrare il tuo lavoro a tutti.

Con un po' di lavoro, puoi fare in modo che il tuo zootropio funzioni anche in situazioni di scarsa luminosità. Collega un LED e una resistenza a uno dei piedini di uscita digitali liberi. Aggiungi anche un secondo potenziometro e collegalo a un ingresso analogico. Metti la luce in modo che illumini le immagini. Utilizzando l'ingresso analogico per misurare il tempo dei lampeggi del LED, prova e riprova in modo che la spia lampeggi quando la fessura è davanti ai tuoi occhi. Questo potrebbe richiedere un po' di lavoro con le manopole, ma l'effetto che ne risulta è davvero spettacolare!

11

SFERA DI CRISTALLO

CREA UNA SFERA DI CRISTALLO PER PREDIRE IL FUTURO

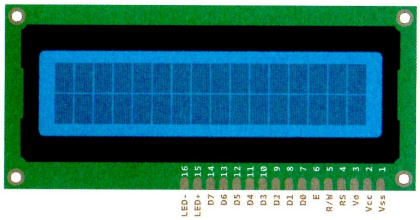
Scopri: i display LCD, istruzioni switch/case, random()

Tempo: **1 ORA**
Livello: ■■■■■

Basato sui progetti: **1, 2, 3**

Le sfere di cristallo possono aiutare a "prevedere" il futuro. Fai una domanda alla sfera onnisciente, e capovolgila per rivelare la risposta. Le risposte saranno predeterminate, ma puoi scriverci ciò che più ti piace. Utilizzerai Arduino per scegliere tra un totale di 8 risposte. L'interruttore di inclinazione nel kit ti aiuterà a simulare l'azione di scuotere la sfera per ottenere le risposte.

Il display LCD può essere utilizzato per visualizzare caratteri alfanumerici. Quello nel kit ha 16 colonne e 2 righe, per un totale di 32 caratteri. Ci sono tante connessioni sulla scheda: queste sono utilizzate per l'alimentazione e la comunicazione, così lo schermo sa cosa scrivere. Non sarà necessario collegarle tutte. Guarda la Fig. 1 per i piedini da collegare.



I piedini sullo schermo LCD che sono usati nel progetto e le etichette

Fig. 1

COSTRUISCI IL CIRCUITO

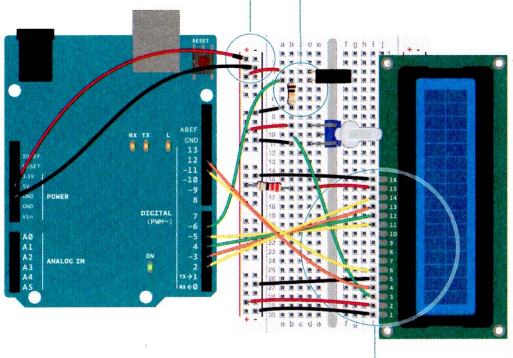


Fig. 2

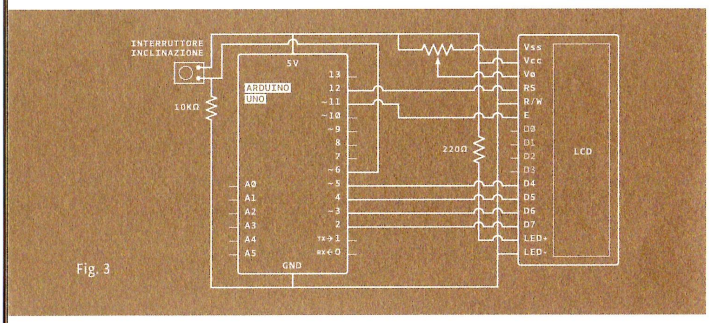


Fig. 3

In questo schema la disposizione dei piedini dell'LCD non corrisponde all'ordine fisico illustrato in Fig. 2. In uno schema, i piedini sono riorganizzati secondo un raggruppamento logico per rendere lo schema più chiaro possibile. Questo crea un po' di confusione ai neofiti.

Il circuito non è complesso, ma ci sono un sacco di fili. Fai attenzione che il cablaggio sia corretto.

1. Collega alimentazione e massa a un lato della breadboard.

2. Posiziona l'interruttore di inclinazione sulla breadboard e collega un filo a 5V. Collega l'altro lato a massa attraverso una resistenza da 10 kilo ohm e al piedino 6 di Arduino. Lo stai collegando come ingresso digitale, proprio come hai fatto in diversi altri progetti.

3. Il piedino register select (RS) controlla dove i caratteri compaiono sullo schermo. Il piedino read/write (R/W) mette lo schermo in modalità lettura o scrittura. In questo progetto userai la modalità scrittura. L'abilitazione (E) dice all'LCD che riceverà un comando. I piedini dei dati (D0-D7) sono usati per mandare i dati dei caratteri allo schermo. Ne userai solo 4 (D4-D7). Infine, c'è una connessione per regolare il contrasto del display. Userai un potenziometro per controllarlo.

4. La libreria LiquidCrystal che viene fornita con il software Arduino gestisce la comunicazione con questi piedini e semplifica il processo di scrittura del software per visualizzare i caratteri. I due piedini esterni dell'LCD (Vss e LED-) devono essere collegati a massa. Poi, collega il piedino R/W a massa. Questo mette lo schermo in modalità scrittura. L'alimentazione dell'LCD (Vcc) deve collegarsi direttamente a 5V. Il piedino LED+ sullo schermo si collega all'alimentazione tramite una resistenza da 220 ohm.

5. Collega: il piedino 2 di Arduino a LCD D7, il piedino 3 di Arduino a LCD D6, il piedino 4 di Arduino a LCD D5, il piedino 5 di Arduino a LCD D4. Questi sono i piedini dei dati che dicono allo schermo quali caratteri mostrare.

6. Collega E dello schermo al piedino 11 di Arduino. RS sull'LCD va collegato al piedino 12. Questo piedino permette di scrivere sull'LCD.

7. Metti il potenziometro sulla breadboard, collegandone un piedino all'alimentazione e l'altro a massa. Il piedino centrale dovrebbe essere collegato a V0 dell'LCD. Questo ti permette di regolare il contrasto dello schermo.

Prepara la libreria LiquidCrystal

Per prima cosa, devi importare la libreria `LiquidCrystal`. Poi, avvia la libreria in un modo simile a come hai fatto con la libreria Servo, dicendogli quali piedini verranno utilizzati per comunicare.

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

Ora che hai impostato la libreria, crea alcune variabili e costanti. Crea una costante per memorizzare il numero di piedino dell'interruttore, una variabile per lo stato attuale dell'interruttore, una variabile per lo stato precedente dell'interruttore e un'altra per scegliere quale risposta mostrerà lo schermo.

```
const int switchPin = 6;
int switchState = 0;
int prevSwitchState = 0;
int reply;
```

Imposta il piedino dell'interruttore con la funzione `pinMode()` nel `setup()`. Avvia la libreria dell'LCD e digli quanto è largo lo schermo.

```
void setup() {
  lcd.begin(16, 2);
  pinMode(switchPin,INPUT);
```

La descrizione della libreria LCD
arduino.cc/lcdlibrary

Stampa la tua prima riga

Ora è tempo di scrivere un breve benvenuto alla sfera di cristallo. La funzione `print()` scrive sullo schermo LCD. Stai per scrivere le parole "Interroga" sulla riga superiore dello schermo. Il cursore è già automaticamente posizionato all'inizio della riga superiore.

```
  lcd.print("Interroga");
```

Muovi il cursore

Per scrivere sulla riga successiva, devi dire allo schermo dove muovere il cursore. Le coordinate della prima colonna sulla seconda riga sono 0,1 (ricordiamo che i computer partono da zero. 0,0 è la prima colonna della prima riga). Usa la funzione `lcd.setCursor()` per muovere il cursore nel posto giusto e digli di scrivere "la sfera!".

```
  lcd.setCursor(0, 1);
  lcd.print("la sfera!");
}
```

Ora, quando parte il programma, compare "Interroga la sfera!" sul tuo schermo.
Nel `loop()`, controlla l'interruttore e metti il valore nella variabile `switchState`.

```
void loop() {
  switchState = digitalRead(switchPin);
```

Scegli una risposta a caso

Usa un'istruzione `if()` per determinare se l'interruttore è in una posizione differente rispetto a prima. Se ora è LOW, è tempo di scegliere una risposta a caso.
La funzione `random()` restituisce un numero in base al parametro che hai fornito. Per iniziare, avrai un totale di 8 risposte possibili. Ogni volta che è chiamata l'istruzione random(8), darà un numero tra 0 e 7. Immagazzina il numero nella variabile `reply`.

```
  if (switchState != prevSwitchState) {
    if (switchState == LOW) {
      reply = random(8);
```

La descrizione di random
arduino.cc/random

Pulisci lo schermo con la funzione `lcd.clear()`, che riporta il cursore alla posizione 0,0: la prima colonna nella prima riga dell'LCD. Stampa la riga "La sfera dice:" e prepara il cursore per la risposta.

```
19    lcd.clear();
20    lcd.setCursor(0, 0);
21    lcd.print("La sfera dice:");
22    lcd.setCursor(0, 1);
```

Predire il futuro

L'istruzione `switch()` esegue diversi pezzi di codice a seconda del parametro che specifichi. Ognuno di questi pezzi è chiamato `case`. L'istruzione `switch()` controlla il valore della variabile `reply`; qualunque sia il suo valore determina quale case viene eseguito.

Nelle istruzioni case, il codice è lo stesso, ma i messaggi sono differenti. Per esempio, nel case 0 il codice dice `lcd.print` ("Sì"). Dopo la funzione `lcd.print()` c'è un altro comando: `break`. Dice ad Arduino dov'è la fine del `case`. Quando raggiunge `break`, salta alla fine dell'istruzione switch. Crea un totale di 8 istruzioni case: quattro risposte positive, due negative e due che ti chiedono di provare ancora.

```
23    switch(reply){
24      case 0:
25      lcd.print("Sì");
26      break;
27      case 1:
28      lcd.print("Probabile");
29      break;
30      case 2:
31      lcd.print("Certo");
32      break;
33      case 3:
34      lcd.print("Bene");
35      break;
36      case 4:
37      lcd.print("Forse");
38      break;
39      case 5:
40      lcd.print("Chiedi ancora");
41      break;
42      case 6:
43      lcd.print("Improbabile");
44      break;
45      case 7:
46      lcd.print("No");
47      break;
48    }
49  }
50 }
```

Descrizione di Switch Case
arduino.cc/switchcase

L'ultima cosa da fare nel `loop()` è assegnare il valore di `switchState` alla variabile `prevSwitchState`. Questo ti permette di tracciare i cambiamenti dell'interruttore la prossima volta che viene eseguito il loop.

```
    prevSwitchState = switchState;
}
```

USALA

Per usare la sfera magica, alimenta Arduino. Controlla lo schermo e assicurati che dica "Interroga la sfera!" Se non vedi la scritta, prova a girare il potenziometro, che regola il contrasto dello schermo.

Fai una domanda alla sfera di cristallo e prova a ribaltare l'interruttore. Dovresti ricevere una risposta alla tua domanda. Se la risposta non ti piace, chiedi ancora.

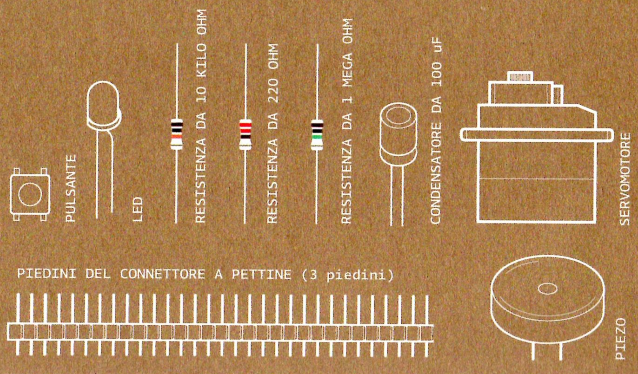
Prova ad aggiungere le tue parole all'istruzione `print()`, ma devi essere consapevole del fatto che puoi usare solo 16 caratteri per riga. Puoi anche provare ad aggiungere altre risposte. Assicurati, quando aggiungi i case supplementari, di modificare il numero di opzioni che casualmente popoleranno la variabile reply.

Gli LCD funzionano modificando le proprietà elettriche di un liquido posto tra i vetri polarizzati. Il vetro permette solo ad alcuni tipi di luce di passare attraverso. Quando il liquido tra i vetri viene caricato elettricamente, inizia a formarsi in un stato semi-solido. Questo nuovo stato ruota in una direzione diversa rispetto al vetro polarizzato, impedendo alla luce di passare attraverso, creando così i caratteri che vedi sullo schermo.

Le funzioni che abbiamo visto qui per modificare il testo dello schermo LCD sono abbastanza semplici. Una volta fatta pratica, guarda le altre funzioni della libreria. Prova a far scorrere il testo o aggiornalo continuamente. Per saperne di più su come funziona la libreria LiquidCrystal, visita il sito: ***arduino.cc/lcd***

Un display LCD consente di visualizzare il testo su uno schermo, utilizzando la libreria LiquidCrystal. Con le istruzioni switch...case controlla il flusso dei programmi confrontando una variabile con dei valori specifici.

12



INGREDIENTI

KNOCK LOCK

CREA LA TUA SERRATURA SEGRETA PER TENERE LONTANI GLI OSPITI INDESIDERATI!

Scopri: input con un piezo, scrivere le funzioni

Tempo: **1 ORA**
Livello: ■■■■■

Basato sui progetti: **1, 2, 3, 4, 5**

Il piezo, che hai usato per riprodurre suoni nei progetti theremin e tastiera musicale, può anche essere utilizzato come dispositivo di ingresso. Quando è collegato a 5V, il sensore può rilevare le vibrazioni misurabili dagli ingressi analogici di Arduino. Avrai bisogno di collegare una resistenza di alto valore (come 1 mega ohm) come riferimento a massa per far funzionare tutto correttamente.

Quando il piezo viene premuto contro una superficie solida in grado di vibrare, come un tavolo in legno, Arduino può percepire l'intensità del colpo. Utilizzando queste informazioni puoi contare quante volte è stato bussato in un intervallo prestabilito. Nel codice è possibile monitorare il numero di colpi e vedere se corrispondono alle tue impostazioni.

Un interruttore ti permetterà di bloccare il motore in posizione. Alcuni LED ti daranno lo stato: un LED rosso indica che la scatola è chiusa a chiave, un LED verde indica che è sbloccata e un LED giallo permette di sapere se è stato ricevuto un colpo valido.

Scriverai anche una funzione che ti consente di sapere se il colpo è troppo forte o troppo leggero. Scrivere funzioni consente di risparmiare tempo riutilizzando il codice invece di riscriverlo molte volte. Le funzioni possono avere dei parametri e restituire valori. In questo caso, fornirai alla funzione il volume del colpo. Se è nell'intervallo giusto, incrementi una variabile.

È possibile costruire anche solo il circuito, ma è molto più divertente se lo si utilizza come strumento per chiudere a chiave qualcosa. Se hai una scatola di legno o di cartone puoi tagliarci dei fori e utilizzare il servomotore per aprire e chiudere un chiavistello, tenendo le persone lontane dalle tue cose.

COSTRUISCI IL CIRCUITO

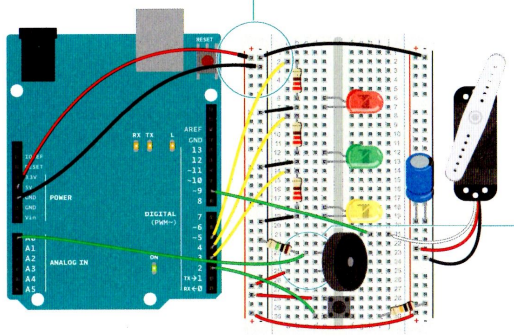


Fig. 1

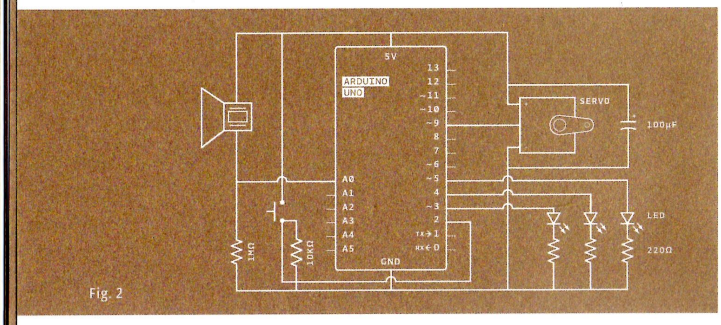


Fig. 2

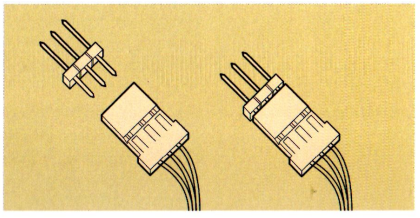
Ci sono molte connessioni sulla scheda, assicurati di tenere traccia di come sono collegate le cose.

1. Collega l'alimentazione e la massa su entrambi i lati della breadboard. Posiziona il pulsante sulla breadboard e collega una estremità a 5V. Collega l'altro lato dell'interruttore a massa attraverso una resistenza da 10 kilo ohm. Collega questo punto di incrocio al piedino digitale 2 di Arduino.

2. Collega i fili dal piezo alla breadboard. Collega un filo all'alimentazione. Se il piezo ha un filo rosso o uno contrassegnato con un "+", è quello da collegare all'alimentazione. Se il piezo non indica la polarità, allora si può collegare in entrambi i modi. Collega l'altra estremità del piezo al piedino analogico 0 di Arduino. Colloca una resistenza da 1 mega ohm tra la massa e l'altro filo. Valori di resistenza più bassi rendono il piezo meno sensibile alle vibrazioni.

3. Collega i LED, connettendo i catodi (piedino corto) a massa, e mettendo una resistenza da 220 ohm in serie con gli anodi. Attraverso le loro rispettive resistenze, collega il LED giallo al piedino digitale 3 di Arduino, il LED verde al piedino digitale 4 e il LED rosso al piedino digitale 5.

4. Inserisci i piedini del connettore a pettine nel connettore del servomotore (vedi la Fig. 3). Collega il filo rosso all'alimentazione e quello nero a massa. Metti un condensatore elettrolitico da 100 uF tra l'alimentazione e la massa per compensare eventuali irregolarità nella tensione e assicurati di aver collegato correttamente la polarità del condensatore. Collega il cavo dei dati del servo al piedino 9 di Arduino.



Il servomotore è dotato di un connettore femmina, quindi hai bisogno di aggiungere piedini maschi per collegarlo alla breadboard.
Fig. 3

IL CODICE

La libreria Servo

Come nel progetto 'Indicatore d'umore', è necessario importare la libreria `Servo` e creare un'istanza per utilizzare il motore.

```
#include <Servo.h>
Servo myServo;
```

Costanti utili

Crea le costanti per dare un nome ai tuoi ingressi e uscite.

```
const int piezo = A0;
const int switchPin = 2;
const int yellowLed = 3;
const int greenLed = 4;
const int redLed = 5;
```

Variabili per memorizzare i valori dell'interruttore e del piezo

Crea le variabili per memorizzare i valori dell'interruttore e del piezo.

```
int knockVal;
int switchVal;
```

Soglie di rilevamento

Crea delle costanti da usare come soglie per i livelli minimo e massimo dei colpi.

```
const int quietKnock = 10;
const int loudKnock = 100;
```

Variabili per lo stato di blocco e il numero dei colpi

La variabile `locked` ti consente di sapere se la serratura è chiusa o no. Il `boolean` è un tipo di dato che può solo essere vero (1) o falso (0). Dovresti iniziare con la serratura sbloccata.
L'ultima variabile globale contiene il numero di colpi validi che hai rilevato.

```
boolean locked = false;
int numberOfKnocks = 0;
```

Imposta la direzione dei piedini digitali e avvia il servo e la porta seriale

Nel `setup()`, attacca il servo al piedino 9.
Imposta i piedini del LED come uscite e i piedini dell'interruttore come ingressi.

```
void setup(){
  myServo.attach(9);
  pinMode(yellowLed, OUTPUT);
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
  pinMode(switchPin, INPUT);
  Serial.begin(9600);
```

Sblocco

Avvia la comunicazione seriale con il computer in modo da poter controllare il volume del colpo, lo stato corrente del blocco e quanti colpi mancano allo sblocco.
Accendi il LED verde, muovi il servo nella posizione di sblocco e stampa lo stato attuale sul monitor seriale indicando che il circuito è nella posizione sbloccata.

```
  digitalWrite(greenLed, HIGH);
  myServo.write(0);
  Serial.println("The box is unlocked!");
}
```

Controlla l'interruttore

Nel `loop()`, per prima cosa verifica se la scatola è bloccata o no. Questo determina ciò che accade nel resto del programma. Se è bloccata, leggi il valore dell'interruttore.

```
void loop(){
  if(locked == false){
    switchVal = digitalRead(switchPin);
```

Blocco

Se l'interruttore è chiuso (lo stai premendo), modifica la variabile `locked` a `true`, indicando che il blocco è innestato. Spegni il LED verde e accendi il LED rosso. Se non hai acceso il monitor seriale, questo è un utile feedback visivo per informarti dello stato del blocco. Sposta il servo in posizione di blocco e stampa un messaggio sul monitor seriale che indica che la scatola è bloccata. Aggiungi un ritardo in modo che la serratura abbia tempo per cambiare posizione.

```
28    if(switchVal == HIGH){
29      locked = true;
30      digitalWrite(greenLed,LOW);
31      digitalWrite(redLed,HIGH);
32      myServo.write(90);
33      Serial.println("The box is locked!");
34      delay (1000);
35    }
36  }
```

Controlla il sensore del colpo

Se la variabile `locked` è vera, ed è attivo il blocco, leggi l'intensità della vibrazione del piezo e memorizzala in `knockVal`.

```
37  if(locked == true){
38    knockVal = analogRead(piezo);
```

Conta solo i colpi validi

L'istruzione successiva verifica se hai meno di tre colpi validi e se ci sono vibrazioni sul sensore. Se questi sono entrambi veri, controlla per vedere se il colpo corrente è valido o no e incrementa la variabile `numberOfKnocks`. Qui chiami la funzione `checkForKnocks()`. Scriverai la funzione una volta che hai finito il loop(), ma sai già che stai per chiedere se si tratta di un colpo valido perciò passa `knockVal` come parametro. Dopo aver verificato il risultato della funzione, stampa il numero di colpi di cui hai ancora bisogno.

```
39    if(numberOfKnocks < 3 && knockVal > 0){
40      if(checkForKnock(knockVal) == true){
41        numberOfKnocks++;
42      }
43      Serial.print(3-numberOfKnocks);
44      Serial.println("more knocks to go");
45    }
```

Sblocco

Controlla se hai tre o più colpi validi. In questo caso, cambia la variabile `locked` su falso e sposta il servo nella posizione di sblocco. Attendi pochi millesimi di secondi per dargli tempo di muoversi e modifica lo stato dei LED verde e rosso. Stampa un messaggio di stato sul monitor seriale, indicando che la scatola è sbloccata.

Chiudi l'istruzione `else` e il `loop()` con una coppia di parentesi graffe.

```
46    if(numberOfKnocks >= 3){
47      locked = false;
48      myServo.write(0);
49      delay(20);
50      digitalWrite(greenLed,HIGH);
51      digitalWrite(redLed,LOW);
52      Serial.println("The box is unlocked!");
53    }
54  }
55 }
```

Definisci una funzione per verificare la validità dei colpi

È ora di scrivere la funzione `checkForKnock()`. Quando la stai scrivendo, hai bisogno di indicare se restituirà un valore o no. Se non hai intenzione di restituire un valore, lo dichiari come tipo `void`, simile alle funzioni `loop()` e `setup()`. Se vuoi restituire un valore, è necessario dichiarare quale tipo avrà (`int`, `long`, `float`, etc.). In questo caso, verifica se il colpo è valido (vero) o no (falso). Dichiara la funzione come tipo `boolean`.

```
56 boolean checkForKnock(int value){
```

Questa particolare funzione verificherà un numero (la tua variabile `knockVal`) per vedere se è valido o no. Per passare questa variabile alla funzione, crea un parametro quando dichiari la funzione.

Controlla la validità dei colpi

Nella funzione, ogni volta che ti riferisci a `value` conterrà il numero ricevuto come argomento dal programma principale. A questo punto `value` è impostato al valore della variabile `knockVal`. Controlla se `value` è maggiore del colpo debole e inferiore del colpo forte.

```
57  if(value > quietKnock && value < loudKnock){
```

Indica se il colpo è valido

Se il valore è tra questi due, è un bussare valido. Fai lampeggiare una volta il LED giallo e stampa il valore del colpo sul monitor seriale.

```
58    digitalWrite(yellowLed, HIGH);
59    delay(50);
60    digitalWrite(yellowLed, LOW);
61    Serial.print("Valid knock of value ");
62    Serial.println(value);
```

La funzione restituisce true

Per informare il programma principale dell'esito del confronto, utilizza il comando `return`, che termina anche la funzione: una volta eseguito, si ritorna al programma principale.

```
63    return true;
64  }
```

Indicare colpi non validi; la funzione restituisce false

Se `value` è fuori dall'intervallo previsto, stampa il valore sul monitor seriale e la funzione restituisce false.

```
65  else {
66    Serial.print("Bad knock value ");
67    Serial.println(value);
68    return false;
69  }
```

Chiudi la funzione con un'altra parentesi.

```
70 }
```

USALO

Quando colleghi il circuito ad Arduino, apri il monitor seriale. Dovresti vedere accendersi il LED verde e il servo si sposta nella posizione di sblocco.

Il monitor seriale dovrebbe stampare "The box is unlocked!". Probabilmente senti che il piezo fa un piccolo clic quando viene alimentato.

Prova a bussare delicatamente e forte per vedere quale intensità del colpo attiva la funzione. Saprai che sta funzionando quando il LED giallo lampeggia e il monitor seriale ti dice che hai ricevuto un colpo valido e il suo valore. Ti permette anche di sapere il numero di colpi che mancano per sbloccare la scatola.

Una volta raggiunto il giusto numero di colpi, la luce rossa si spegne, la luce verde si accende, il servo si sposta di 90 gradi e il monitor seriale ti informa che il blocco è disinnestato.

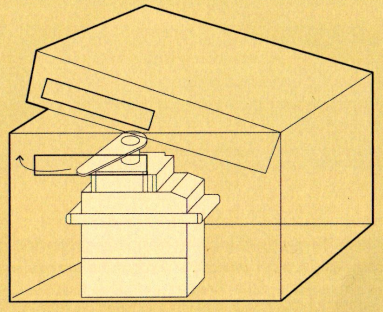
I valori per il colpo ideale possono variare da quelli dell'esempio. Questo dipende da un certo numero di variabili, come il tipo di superficie su cui è fissato il sensore e come è fissato. Utilizzando il monitor seriale e l'esempio AnalogInSerialOut nell'IDE di Arduino, trovi un valore del colpo adatto alla tua configurazione. Qui trovi una spiegazione dettagliata di questo esempio: ***arduino.cc/analogtoserial***

Se inserisci il progetto in una scatola, è necessario fare i fori per i LED e l'interruttore. Avrai anche bisogno di fare un fermo in cui inserire il servomotore. È utile avere un foro per farci passare il cavo USB per scoprire il grado di sensibilità ai colpi del nuovo ambiente. Potresti aver bisogno di riorganizzare la breadboard e Arduino o saldare i LED e l'interruttore per renderli accessibili dall'esterno del contenitore.
La saldatura è un processo di unione di due o più componenti metallici con una lega che viene fusa sulla giunzione. Se non hai mai saldato prima, chiedi a qualcuno che ha esperienza di aiutarti o fai un po' di pratica sul filo di scarto prima di tentare con i componenti di questo progetto. La saldatura è una connessione permanente, quindi assicurati di poter hackerare ciò che stai usando. Guarda ***arduino.cc/soldering*** per una buona spiegazione di come saldare.

Scrivere le proprie funzioni non solo permette di controllare il flusso del codice più facilmente, ma aiuta anche a mantenerlo leggibile quando i progetti diventano sempre più grandi. Nel corso del tempo, più codice scrivi e più troverai funzioni che è possibile riutilizzare in diversi progetti, rendendo il processo più veloce e vicino al tuo stile di programmazione.

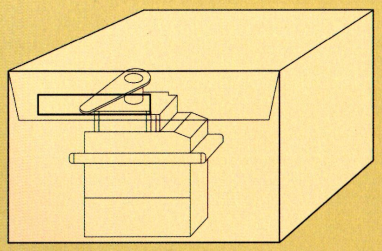
Questo esempio conta semplicemente il giusto numero di colpi, non importa quanto tempo è necessario. Si può fare un progetto più complesso creando un timer con la funzione `millis()`. Usa il timer per identificare se il colpo accade in un determinato periodo di tempo. Guarda di nuovo il progetto della clessidra digitale per un esempio di come funziona un timer. Non sei limitato a trovare i colpi in un intervallo specifico. Puoi rilevare schemi complessi di colpi in base alla quantità di vibrazioni e di tempo. C'è una serie di esempi online su come effettuare questa operazione, cerca "Arduino knock lock" per scoprirli.

Gli elementi piezo possono essere utilizzati come ingressi quando sono collegati come divisori di tensione con una resistenza di valore elevato. Progettare una funzione è un modo semplice per scrivere codice che può essere riutilizzato per compiti specifici.



❶

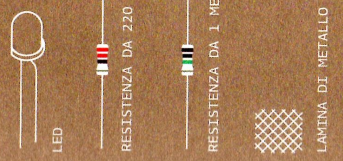
Taglia 2 fori nella scatola: uno su un lato e uno sul coperchio. Posiziona il servo nella scatola in modo che il braccio possa muoversi dentro e fuori i fori quando la scatola è chiusa.



❷

Fissa il servo in una posizione con del nastro adesivo in modo che il braccio possa ruotare facilmente attraverso la fessura che hai fatto.

13



INGREDIENTI

LAMPADA EMOTIVA

CREERAI UNA LAMPADA CHE SI ACCENDE E SI SPEGNE QUANDO TOCCHI UN PEZZO DI MATERIALE CONDUTTIVO

Scopri: installare librerie fornite da altri, creare un sensore di tocco

Tempo: **45 MINUTI**
Livello: ■■■■■

Basato sui progetti: **1, 2, 5**

Per questo progetto utilizzerai la libreria CapacitiveSensor di Paul Badger, che permette di misurare la capacità elettrica del tuo corpo.

La capacità misura la quantità di carica elettrica che un corpo è in grado di immagazzinare. La libreria controlla due piedini di Arduino (uno è un emettitore, l'altro un ricevitore) e misura il tempo a loro richiesto per raggiungere lo stesso stato. Questi piedini sono collegati a un oggetto metallico, come un foglio di alluminio. Quando ti avvicini all'oggetto, il tuo corpo è in grado di assorbire parte della carica, allungando i tempi necessari ai due piedini per raggiungere lo stesso stato.

Preparare la libreria

La versione più recente della libreria CapacitiveSensor è qui: ***arduino.cc/capacitive***. Scarica il file sul tuo computer e decomprimilo. Apri la cartella che contiene gli sketch di Arduino (solitamente è nella cartella "Documenti"). Nella cartella, crea una nuova cartella chiamata "libraries". Posiziona qui la cartella CapacitiveSensor e riavvia il software Arduino.
Clicca sul menu File>Examples nel software di Arduino, e vedrai una nuova voce "CapacitiveSensor". La libreria che hai aggiunto include un progetto di esempio. Apri CapacitiveSensorSketch e compilalo. Se non compare un avviso di errore, è stato installato correttamente.

Per maggiori informazioni sulle librerie:
arduino.cc/en/Reference/Libraries

COSTRUISCI IL CIRCUITO

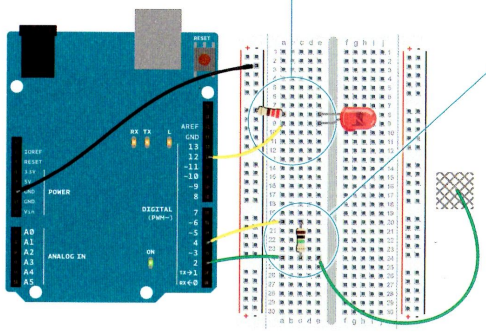


Fig. 1

1. Collega un LED al piedino 12 e collega il catodo a massa attraverso la resistenza da 220 ohm.

2. Collega i piedini digitali 2 e 4 alla breadboard. Collega i due piedini con una resistenza da 1 mega ohm. Nella stessa riga del piedino 2, inserisci un filo lungo (almeno 8-10 cm) che fuoriesce dalla breadboard, non collegato a nulla all'altra estremità. Questo diventa il sensore di tocco.

In questo progetto non c'è bisogno di fornire 5V alla breadboard. Il piedino digitale 4 fornisce la corrente al sensore.

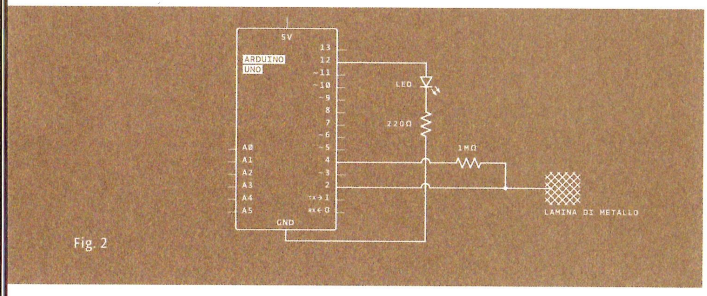


Fig. 2

Proprio come con altri progetti con i LED, diffonderne la luce li rende molto più interessanti. Palline da ping pong, piccoli paralumi di carta o di plastica, tutto ciò che hai a portata di mano può andar bene per diffondere la luce.

Puoi nascondere il sensore dietro qualcosa di solido e continua a funzionare. La capacità può essere misurata attraverso materiali non conduttivi come il legno e la plastica. L'aumento della superficie del sensore con una superficie conduttiva più grande lo rende più sensibile; prova a collegare un foglio di alluminio o una rete di rame al filo. Potresti fare una base per la lampada con cartone, legno sottile o con un panno e rivestire la superficie interna con un foglio collegato al filo del sensore. L'intera base della lampada si comporterebbe quindi come un sensore di tatto. Aggiorna la soglia variabile nel codice quando fai queste modifiche per garantire ancora un risultato affidabile.

IL CODICE

Importa la libreria CapacitiveSensor

All'inizio del tuo programma, includi la libreria CapacitiveSensor. Si fa come con una libreria nativa Arduino, come la libreria `Servo` nei progetti precedenti. Crea un'istanza della libreria. Quando la utilizzi, dici all'istanza quali piedini sono utilizzati per inviare e ricevere informazioni. In questo caso il piedino 4 alimenta il sensore di materiale conduttivo attraverso la resistenza, e il piedino 2 è quello che misura.

```
#include <CapacitiveSensor.h>
CapacitiveSensor capSensor = CapacitiveSensor(4,2);
```

Imposta la soglia

Crea una variabile per la soglia di rilevamento alla quale la lampada si accende. Puoi modificare questo valore dopo aver testato la funzionalità del sensore.
Poi definisci il piedino collegato al LED.

```
int threshold = 1000;
const int ledPin = 12;
```

Nella funzione `setup`(), apri una connessione seriale a 9600 bps per vedere i valori letti dal sensore. Inoltre, rendi il ledPin un `OUTPUT`.

```
void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
}
```

Rileva il tocco

Nella funzione `loop`(), crea una variabile di tipo long per memorizzare il valore del sensore. La libreria legge il valore del sensore usando il comando `CapacitiveSensor()` che richiede un parametro che identifica il numero di campioni che vuoi leggere. Se leggi solo pochi campioni, puoi vedere molte variazioni nel sensore. Se prendi troppi campioni, potresti introdurre un ritardo leggendo molte volte il sensore. 30 campioni è un buon valore di partenza. Stampa il valore del sensore sul monitor seriale.

```
void loop() {
  long sensorValue = capSensor.capacitiveSensor(30);
  Serial.println(sensorValue);
```

Controlla la lampada

Con un'istruzione `if`()...`else`, controlla se il valore del sensore è più alto della soglia. Se lo è, accendi il LED; altrimenti spegnilo.

```
  if(sensorValue > threshold) {
    digitalWrite(ledPin, HIGH);
  }
  else {
    digitalWrite(ledPin, LOW);
  }
```

Poi aggiungi un piccolo `delay`() prima di finire il `loop`().

```
  delay(10);
}
```

USALA

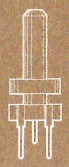
Dopo aver programmato Arduino, vorrai scoprire quali sono i valori del sensore quando viene toccato. Apri il monitor seriale e prendi nota del valore proveniente dal sensore quando non lo tocchi. Premi leggermente il filo che fuoriesce dalla breadboard. Il numero dovrebbe aumentare. Prova a premere con maggiore fermezza e vedi se cambia.

Una volta che hai un'idea della gamma di valori che stai ricevendo dal sensore, torna allo sketch e modifica la soglia variabile a un numero che è maggiore del valore del sensore quando non viene toccato, ma inferiore al suo valore quando viene premuto. Carica lo sketch con il nuovo valore. La luce dovrebbe accendersi quando tocchi il filo e spegnersi quando viene lasciato. Se non riesci ad accendere la luce, prova a ridurre ancora un po' la soglia.

Probabilmente hai notato che i valori del sensore cambiano a seconda di quanta parte del tuo dito è stata in contatto con il conduttore. Si può sfruttare questo fenomeno per ottenere altre interazioni con il LED? Che ne pensi di più sensori usati per alzare e abbassare l'intensità della luce? Se inserisci una resistenza di valore diverso tra i piedini 2 e 4 cambierà la sensibilità. È utile per l'interfaccia?

Librerie scritte da altri, come la CapacitiveSensor di Paul Badger sono strumenti utili per ampliare le funzionalità di Arduino. Una volta installate, si comportano in modo simile alle librerie che fanno parte del software di base.

14



POTENZIOMETRO

INGREDIENTI

MODIFICA IL LOGO DI ARDUINO

USANDO LA COMUNICAZIONE SERIALE, UTILIZZERAI ARDUINO PER CONTROLLARE UN PROGRAMMA SUL TUO COMPUTER

Scopri: la comunicazione seriale con il computer, Processing

Tempo: **45 MINUTI**
Livello: ■■■■■

Basato sui progetti: **1, 2, 3**

Hai fatto cose molto interessanti con il mondo fisico, ora è il momento di controllare il computer con Arduino. Quando programmi Arduino, stai aprendo una connessione tra il computer e il microcontrollore. Puoi utilizzare questa connessione per inviare i dati avanti e indietro ad altre applicazioni.

Arduino ha un chip che converte la comunicazione USB del computer alla comunicazione seriale che utilizza Arduino. La comunicazione seriale significa che i due computer, l'Arduino e il PC, si scambiano bit di informazioni serialmente o uno dopo l'altro nel tempo. Per la comunicazione seriale, i computer devono mettersi d'accordo sulla velocità con cui parlano l'un l'altro. Probabilmente hai notato che quando utilizzi il monitor seriale c'è un numero nell'angolo in basso a destra della finestra. Quel numero, 9600 bit al secondo, o baud, è lo stesso del valore che hai dichiarato con `Serial.begin()`. Questa è la velocità alla quale Arduino e il computer si scambiano i dati. Un bit è la più piccola quantità di informazioni che un computer è in grado di capire. Hai usato il monitor seriale per guardare i valori degli ingressi analogici; userai un metodo simile per trasferire i valori in un programma che stai per scrivere in un ambiente di programmazione chiamato **Processing**. Processing è basato su Java e l'ambiente di programmazione Arduino è basato su quello di Processing. Sembrano molto simili, così dovresti sentirti come a casa.

Prima di iniziare il progetto, scarica l'ultima versione di Processing dal sito ***processing.org***. Può essere utile guardare i tutorial "Getting Started" e "Overview" al link ***processing.org/learning***. Questo ti aiuterà a familiarizzare con Processing prima di iniziare a scrivere il software per comunicare con Arduino.

Il modo più efficiente di inviare dati tra Arduino e Processing è usare la funzione `Serial.write()` in Arduino. Questa funzione è simile alla `Serial.print()` che hai già utilizzato in quanto invia le informazioni a un computer collegato, ma, invece di inviare le informazioni leggibili dall'uomo come numeri e lettere, invia i valori compresi tra 0 e 255 sotto forma di byte non elaborati. Questo limita i valori che Arduino può inviare, ma consente uno scambio rapido di informazioni.

Sia sul computer che su Arduino, c'è una cosa chiamata buffer seriale che parcheggia le informazioni fino a quando non vengono lette da un programma. Invierai byte dal buffer seriale di Arduino a quello di Processing. Processing quindi leggerà i byte dal buffer. Mano a mano che il programma legge le informazioni dal buffer seriale, libera lo spazio per ulteriori informazioni.

Quando si utilizza la comunicazione seriale tra dispositivi e programmi, è importante che entrambe le parti non solo sappiano quanto velocemente comunicheranno, ma anche quello che si aspettano. Quando incontri qualcuno, probabilmente ti aspetti un "Ciao!". Se invece ti dicono qualcosa come "Il gatto è peloso", è probabile che sarai preso in contropiede. Con il software, è necessario ottenere un accordo da entrambe le parti su ciò che viene inviato e ricevuto.

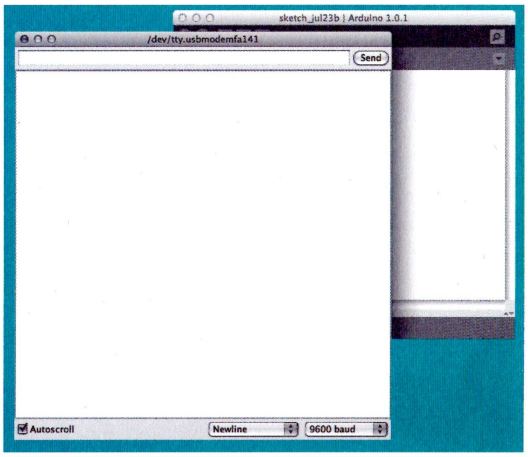


Fig. 1

COSTRUISCI IL CIRCUITO

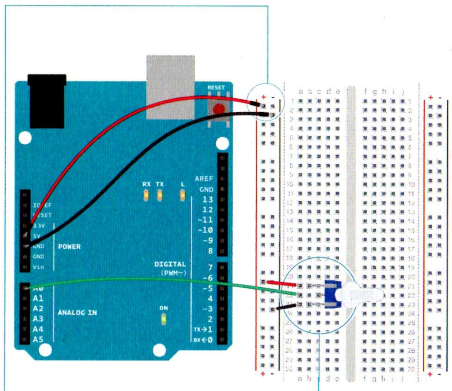


Fig. 2

1. Collega massa e alimentazione alla breadboard.
2. Collega ogni estremità del potenziometro alla massa e all'alimentazione. Collega il piedino centrale al piedino analogIn 0.

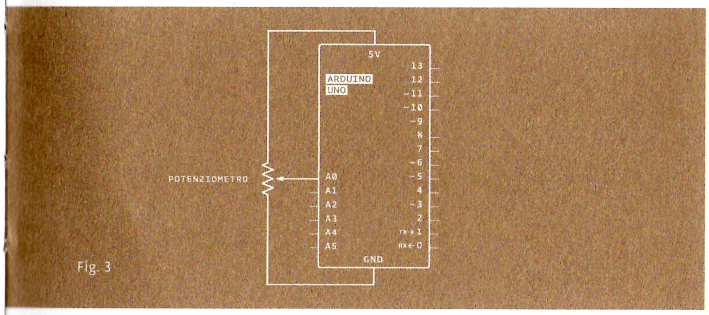


Fig. 3

IL CODICE DI ARDUINO

Apri una connessione seriale

In primo luogo, programma Arduino. Nel `setup()`, inizia la comunicazione seriale, così come hai fatto in precedenza per mostrare i valori di un sensore collegato. Il programma Processing che scrivi ha la stessa velocità di comunicazione seriale di Arduino.

```
1 void setup() {
2   Serial.begin(9600);
3 }
```

Invia il valore del sensore

Nel `loop()`, stai per usare il comando `Serial.write()` per inviare informazioni tramite la connessione seriale. `Serial.write()` può inviare solo valori tra 0 e 255. Per assicurarti che stai mandando solo valori compresi in questo intervallo, dividi la lettura analogica per 4.

```
4 void loop() {
5   Serial.write(analogRead(A0)/4);
```

Stabilizza l'ADC

Dopo l'invio del byte, attendi un millesimo di secondo per consentire all'ADC di stabilizzarsi. Carica il programma in Arduino, poi mettilo da parte mentre scrivi il tuo sketch di Processing.

```
6   delay(1);
7 }
```

IL CODICE DI PROCESSING

Importa la libreria seriale e crea l'oggetto

Il linguaggio di Processing è simile a quello di Arduino, ma ci sono abbastanza differenze per cui varrebbe la pena guardare i tutorial e la guida "Getting Started" citata prima per familiarizzare con questo linguaggio.
Apri un nuovo sketch di Processing. Processing, al contrario di Arduino, non conosce le porte seriali a meno che non venga inclusa una libreria esterna. Importa quindi la libreria seriale.
Devi creare un'istanza dell'oggetto seriale, come hai fatto in Arduino con la libreria Servo. Usa questo oggetto dal nome univoco ogni volta che desideri utilizzare la connessione seriale.

```
1 import processing.serial.*;
2 Serial myPort;
```

Crea un oggetto per l'immagine

Per usare immagini in Processing, devi creare un oggetto che contiene l'immagine e dargli un nome.

```
3 PImage logo;
```


ORA SALVA E CHIUDI L'IDE DI ARDUINO E APRI PROCESSING.

Variabile per immagazzinare il colore di sfondo

Crea una variabile che contiene il colore di sfondo del logo di Arduino. Il logo è un file png e ha la trasparenza, in modo che sia possibile vedere il cambiamento di colore dello sfondo.

```
4 int bgcolor = 0;
```

Processing ha una funzione **`setup()`**, proprio come Arduino. Apri la connessione seriale e dai al programma un paio di parametri che verranno utilizzati durante l'esecuzione.

```
5 void setup() {
```

Imposta la modalità colore

Puoi modificare il modo in cui Processing lavora con le informazioni sul colore. In genere, funziona con colori in modalità RGB (rosso, verde, blu). Questo è simile al colore che hai miscelato nel progetto 04, quando hai usato valori compresi tra 0 e 255 per modificare il colore di un LED RGB. In questo programma, utilizza invece una modalità di colore chiamata HSB, acronimo di Hue, Saturation e Brightness (tonalità, saturazione e luminosità). Potrai modificare la tonalità quando giri il potenziometro. **`colorMode()`** richiede due parametri: il tipo di modalità e il valore massimo che può aspettarsi.

```
6   colorMode(HSB, 255);
```

Carica l'immagine

Per caricare l'immagine di Arduino nello sketch, leggila nell'oggetto logo che hai creato in precedenza. Quando fornisci l'URL di un'immagine, Processing la scarica quando esegui il programma. Con la funzione **`size()`**, dici a Processing quanto grande deve essere la finestra sullo schermo. Se usi **`logo.width`** e **`logo.height`** come istruzioni, lo sketch si ridimensiona automaticamente alle dimensioni dell'immagine.

```
7   logo = loadImage("http://arduino.cc/logo.png");
8   size(logo.width, logo.height);
```

Stampa le porte seriali disponibili

Processing ha la capacità di stampare i messaggi di stato usando il comando **`println()`**. Usandolo con la funzione **`Serial.list()`**, ottieni un elenco di tutte le porte seriali del computer quando il programma viene avviato. Lo userai quando hai finito di programmare per vedere su quale porta è collegata Arduino.

```
9   println("Available serial ports:");
10  println(Serial.list());
```

Crea l'oggetto porta seriale

Devi fornire a Processing le informazioni sulla connesione seriale. Per dare all'oggetto seriale **`myPort`** le informazioni necessarie, il programma ha bisogno di una istanza dell'oggetto seriale. I parametri che si aspetta sono: con quale applicazione parla, con quale porta seriale comunica e a quale velocità.

```
11  myPort =
      new Serial(this, Serial.list()[0], 9600);
12 }
```

L'attributo **this** dice a Processing che stai per usare la connesione seriale in questa stessa applicazione. L'istruzione **Serial.list()[0]** specifica quale porta seriale stai usando. **Serial.list()** contiene un array di tutti i dispositivi seriali collegati. Il parametro **9600** dovrebbe esserti familiare: definisce la velocità alla quale comunica il programma.

La funzione **draw()** è analoga al **loop()** di Arduino in quanto si ripete continuamente. Qui è dove si disegna sulla finestra del programma.

```
13 void draw() {
```

Leggi i dati di Arduino dalla porta seriale

Controlla se ci sono informazioni da Arduino. Il comando **myPort.available()** indica se c'è qualcosa nel buffer seriale. Se ci sono byte, leggi il valore nella variabile **bgcolor** e stampalo nella finestra di debug.

```
14   if (myPort.available() > 0) {
15     bgcolor = myPort.read();
16     println(bgcolor);
17   }
```

Imposta lo sfondo dell'immagine e visualizza l'immagine

La funzione **background()** imposta il colore della finestra. Ha tre parametri. Il primo è la tonalità, il secondo la luminosità e l'ultimo la saturazione. Usa la variabile **bgcolor** come valore della tonalità e imposta la luminosità e la saturazione al valore massimo, 255.
Disegna il logo con il comando **image()**. Devi dire a **image()** cosa disegnare e le coordinate dove iniziare a disegnare nella finestra. 0,0 è in alto a sinistra: incomincia da lì.

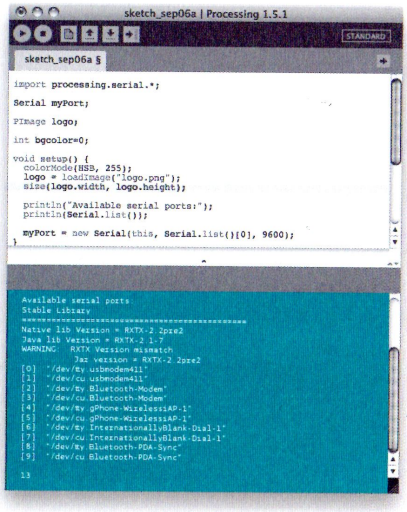
```
18   background(bgcolor, 255, 255);
19 image(logo, 0, 0);
20 }
```

USALO

Collega Arduino e apri il monitor seriale. Ruota la manopola del potenziometro. Dovresti vedere dei caratteri quando la giri. Il monitor seriale si aspetta caratteri ASCII, non byte grezzi. ASCII è l'informazione codificata per rappresentare il testo nel computer. Quello che vedi nella finestra è il monitor seriale che cerca di interpretare i byte come ASCII.

Quando usi **Serial.println()**, mandi informazioni formattate per il monitor seriale.
Quando usi **Serial.write()**, come nell'applicazione che stai eseguendo ora, stai inviando informazioni grezze. Programmi come Processing possono capire questi byte grezzi.

Chiudi il monitor seriale. Esegui lo sketch di Processing premendo il tasto freccia nell'IDE di Processing. Guarda la finestra di output di Processing. Si dovrebbe vedere un elenco simile alla figura sottostante.



Questa è una lista di tutte le porte seriali del tuo computer. Se usi Mac OS X, cerca una stringa che dica qualcosa del tipo "/dev/tty.usbmodem411", sarà probabilmente il primo elemento nella lista. Su Linux, appare come "/dev/ttyUSB0", o qualcosa di simile. Per Windows, appare come una porta COM, la stessa che usi quando programmi la scheda. Il numero di fronte è l'indice dell'array `Serial.list()[]`. Cambia il numero nel tuo sketch di Processing in modo che corrisponda alla porta corretta sul computer.

Riavvia lo sketch di Processing. Quando il programma inizia a girare, ruota il potenziometro collegato ad Arduino. Dovresti veder cambiare il colore dietro il logo di Arduino. Dovresti anche vedere i valori nella finestra di Processing. Quei numeri corrispondono ai byte grezzi che hai inviato da Arduino.

Una volta che hai smanettato a più non posso, prova a sostituire il potenziometro con un sensore analogico. Trova qualcosa di interessante per controllare il colore. Che sensazione ti dà l'interazione? Probabilmente è diversa rispetto a usare un mouse o la tastiera; ti sembra naturale?

Quando utilizzi la comunicazione seriale, solo un'applicazione alla volta può parlare con Arduino. Quindi, se stai eseguendo uno sketch di Processing che è collegato ad Arduino, non sarai in grado di caricare un nuovo sketch di Arduino né di utilizzare il monitor seriale fino a quando non avrai chiuso l'applicazione attiva.

Con Processing e gli altri ambienti di programmazione, puoi controllare suoni, immagini e video sul computer, in modo interessante e innovativo. Se ti interessa la possibilità di controllare i contenuti sul computer, prenditi del tempo per sperimentare con Processing. Ci sono diversi esempi di comunicazione seriale sia in Processing sia nell'IDE di Arduino che ti aiuteranno a sperimentare ancora.

La comunicazione seriale permette ad Arduino di comunicare con programmi sul computer. Processing è un ambiente di programmazione open source sul quale è basato l'IDE di Arduino. È possibile controllare uno sketch di Processing con Arduino tramite la comunicazione seriale.

15

FOTOACCOPPIATORE

RESISTENZA DA 220 OHM

INGREDIENTI

HACKERARE PULSANTI

CONTROLLA ALTRI COMPONENTI CHE TI CIRCONDANO. USANDO SEMPLICI CIRCUITI, PUOI "PREMERE" PULSANTI CON IL TUO ARDUINO

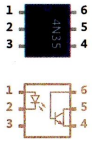
Scopri: il fotoaccoppiatore, la connessione con altri componenti

Tempo: **45 MINUTI**
Livello: ■■■■■

Basato sui progetti: **1, 2, 9**

Attenzione: non sei più un principiante se fai questo progetto. Aprirai un dispositivo elettronico e lo modificherai. Annullerai la garanzia del dispositivo e, se non stai attento, potresti danneggiarlo. Assicurati che hai familiarità con tutti i concetti di elettronica presenti nei progetti precedenti prima di provare questo esempio. Ti consigliamo di utilizzare oggetti economici così non ti dispiacerà creare danni con i tuoi primi progetti, finché non sviluppi esperienza e fiducia in te stesso.

Sebbene Arduino sia in grado di controllare molte cose, a volte è più facile usare strumenti che sono stati creati per scopi specifici. Forse desideri controllare un televisore o un lettore musicale o guidare una macchina telecomandata. La maggior parte dei dispositivi elettronici ha un pannello di controllo con dei pulsanti e molti di questi pulsanti possono essere manipolati in modo che tu possa "premerli" con Arduino. Controllare suoni registrati è un buon esempio. Se vuoi registrare e riprodurre un suono, ci vorrebbe un gran lavoro per farlo con Arduino. È molto più facile prendere un piccolo dispositivo che registra e riproduce l'audio e sostituire i suoi pulsanti con le uscite controllate da Arduino.



I ***fotoaccoppiatori*** sono circuiti integrati che consentono di controllare un circuito da un altro senza alcun collegamento elettrico tra i due. All'interno di un fotoaccoppiatore c'è un LED e un rilevatore di luce. Quando il LED del fotoaccoppiatore è acceso da Arduino, il rilevatore di luce chiude un interruttore interno. L'interruttore è collegato a due piedini di uscita (4 e 5) del fotoaccoppiatore. Quando l'interruttore interno è chiuso, i due piedini di uscita sono collegati. Quando l'interruttore è aperto, non sono collegati. In questo modo, è possibile chiudere interruttori su altri apparecchi senza collegarli ad Arduino.

Lo schema di questo esempio serve per controllare un modulo di registrazione digitale che permette di registrare e riprodurre 20 secondi di suono, ma le premesse di base valgono per qualsiasi dispositivo dotato di un pulsante da accedere. Anche se è possibile realizzare questo esempio senza alcuna saldatura, questa rende certamente le cose più facili. Per ulteriori informazioni sulla saldatura, vedi pag. 134.

COSTRUISCI IL CIRCUITO

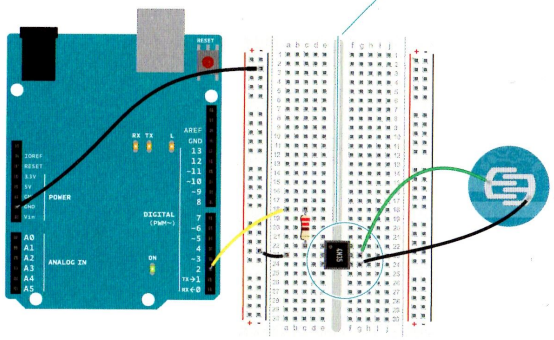


Fig. 1

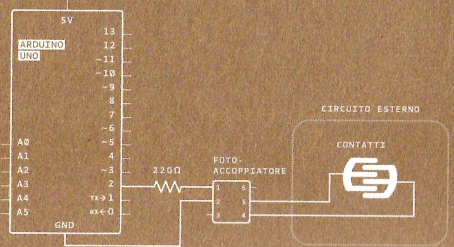


Fig. 2

1. Collega la massa alla breadboard attraverso Arduino.
2. Metti il fotoaccoppiatore sulla breadboard in modo che sia nel mezzo (guarda il diagramma del circuito).
3. Collega il piedino 1 del fotoaccoppiatore al piedino 2 di Arduino in serie con una resistenza da 220 ohm (ricorda che stai alimentando un LED interno e non vuoi che bruci). Collega il piedino 2 del fotoaccoppiatore a massa.
4. Sulla scheda principale del modulo sonoro ci sono alcuni componenti elettrici come un pulsante di riproduzione. Per controllarlo, devi rimuovere il pulsante. Capovolgi il circuito e trova le linguette che tengono il pulsante in posizione. Piega indietro leggermente le linguette e rimuovi il pulsante dalla scheda.
5. Sotto il pulsante ci sono due piastre metalliche. Questo è tipico di molti dispositivi elettronici a pulsanti. Le due "forchette" di questo modello sono i due lati dell'interruttore. Quando premi il pulsante un disco di metallo al suo interno collega queste due forchette.
6. Quando le forchette sono collegate, l'interruttore sul circuito è chiuso. Chiudi l'interruttore con il fotoaccoppiatore. Questo metodo funziona solo se uno dei due lati dell'interruttore del pulsante è collegato a massa sul dispositivo. Se non sei sicuro, prendi un multimetro e misura la tensione tra una delle due forchette sul dispositivo. Fallo con il dispositivo acceso, quindi fai attenzione a non toccare alcun punto della scheda. Quando hai scoperto quale forchetta è la massa, disconnetti l'alimentazione dal tuo dispositivo.
7. Poi, collega un filo a ciascuna delle piastre metalliche. Se saldi questi fili, fai attenzione a non unire i due lati del pulsante. Se non saldi e usi del nastro adesivo, assicurati che la connessione sia stabile o l'interruttore non si chiuderà. Assicurati che nessuno dei due fili si colleghi all'altra forchetta o l'interruttore sarà sempre chiuso.
8. Collega i due fili ai piedini 4 e 5 del fotoaccoppiatore. Collega il lato dell'interruttore a massa al piedino 4 del fotoaccoppiatore. Collega l'altra forchetta al piedino 5 del fotoaccoppiatore.

IL CODICE

Crea una costante

La parte più divertente del progetto è il circuito e il fotoaccoppiatore. Il codice è simile al primo progetto che hai fatto con Arduino. Stai per riprodurre il suono ogni 20 secondi impostando il piedino 2 su `HIGH`.
Crea una costante per il piedino di controllo del fotoaccoppiatore.

```
1 const int optoPin = 2;
```

Configura la direzione del piedino

Nel `setup()`, rendi il piedino del fotoaccoppiatore un output.

```
2 void setup(){
3   pinMode(optoPin, OUTPUT);
4 }
```

Alza e abbassa il piedino

Il `loop()` imposta optoPin ad `HIGH` per pochi millesimi di secondo, abbastanza perché il fotoaccoppiatore chiuda l'interruttore sull'apparecchio. Poi optoPin diventa `LOW`.

```
5 void loop(){
6   digitalWrite(optoPin, HIGH);
7   delay(15);
8   digitalWrite(optoPin, LOW);
```

Aspetta qualche secondo

Aspetta 21 secondi che il messaggio sia riprodotto per intero prima di iniziare di nuovo il `loop()`.

```
9   delay(21000);
10 }
```

USALO

Collega la batteria al registratore di suoni. Premi e tieni premuto il pulsante di registrazione sul dispositivo. Mentre stai tenendo premuto il pulsante, puoi registrare l'audio nel microfono. Usa la tua voce, il gatto o le pentole e padelle in cucina per fare un po' di rumore (ma stai attento con il gatto).

Dopo aver registrato un suono che ti piace, alimenta Arduino con il cavo USB. Dovresti sentire la tua registrazione. Se hai registrato per 20 secondi, il suono dovrebbe ricominciare pochi istanti dopo la sua conclusione.

Nel programma prova a sperimentare con differenti suoni e tempi di attivazione diversi con il `delay()`.
Se attivi l'interruttore mentre un suono viene riprodotto, si ferma. Come puoi approfittare di questo per creare sequenze particolari di suoni?

I circuiti integrati sono presenti in quasi tutti i dispositivi elettronici. Il chip di grandi dimensioni da 28 piedini su Arduino è un circuito integrato che ospita il cervello della scheda. Ci sono altri circuiti integrati che lo supportano con la comunicazione e l'alimentazione. Il fotoaccoppiatore e il principale chip di Arduino sono entrambi chip ***Dual In-line Package (DIP)***. Questi sono il tipo più usato dagli hobbisti perché sono facilmente inseribili in una breadboard e non devono essere saldati in modo permanente per essere utilizzati.

Il progetto di esempio gioca solo con l'audio a intervalli regolari. Come puoi incorporare gli input dai progetti precedenti per attivare questi suoni? Quali altre cose alimentate a batteria hai in giro per casa che hanno bisogno di un Arduino per essere controllate? Questa tecnica per controllare un dispositivo elettronico con un fotoaccoppiatore, collegandolo ai due lati di un interruttore, può essere usata con molti altri dispositivi. Quali altri apparecchi vuoi controllare?

I fotoaccoppiatori sono in grado di controllare i dispositivi che si trovano su un circuito diverso. I due circuiti sono separati elettricamente uno dall'altro all'interno del componente.

A/Z

GLOSSARIO

CI SONO MOLTI NUOVI TERMINI CHE HIA IMPARATO IN QUESTI PROGETTI. LI ABBIAMO RACCOLTI QUI COME RIFERIMENTO

A

Accelerometro **-** Un sensore che misura l'accelerazione. Qualche volta è utilizzato per rilevare l'orientamento o l'inclinazione.

Alimentazione **-** Una fonte di energia, di solito una batteria, un trasformatore o anche la porta USB del computer. È disponibile in molte varietà, come stabilizzata o meno, corrente alternata (CA) o corrente continua (CC). Di solito la tensione viene specificata insieme con la corrente massima che l'alimentatore è in grado di fornire prima di guastarsi.

Amperaggio (A o ampere) **-** La quantità di carica elettrica che attraversa un punto specifico di un circuito. L'amperaggio descrive la corrente che scorre attraverso un conduttore, come un filo elettrico.

Analogico **-** Qualcosa che può variare nel tempo liberamente.

Anodo **-** Il polo positivo di un condensatore o diodo (ricordati che il LED è un tipo di diodo).

Argomento **-** Un tipo di dati fornito a una funzione come input. Per esempio, digitalRead() per sapere quale piedino leggere richiede un parametro intero che indica il numero del piedino.

Array **-** Nella programmazione, è un gruppo di variabili identificate da un solo nome, e vi si accede tramite un numero di indice.

Attuatore **-** Un componente che trasforma l'energia elettrica in qualcosa che è percepibile nel mondo fisico. I motori sono tipi di attuatori.

B

Baud - Abbreviazione di "bit al secondo": indica la velocità con cui i computer comunicano tra loro.

Binario - Un sistema con solo due stati, come vero/falso o acceso/spento.

Bit - La quantità minima di informazione che un computer può inviare o ricevere. Ha due stati, 0 e 1.

Booleano - Un tipo di dato che indica se qualcosa è vero o falso.

Buffer seriale - Un'area di memoria del computer e del microcontrollore in cui vengono parcheggiate le informazioni ricevute nella comunicazione seriale fino a quando non vengono lette da un programma.

Byte - 8 bit di informazione. Un byte può contenere un numero tra 0 e 255.

C

Calibrazione - Una procedura per migliorare i risultati di un circuito o di un programma modificando determinati valori o componenti. Nei progetti Arduino, è spesso usata quando i sensori danno nel mondo reale valori diversi in circostanze diverse, per esempio la quantità di luce su una fotoresistenza. La calibrazione può essere automatica, come nel Progetto 06, o manuale, come nel Progetto 03.

Capacità - L'attitudine di qualcosa a conservare una carica elettrica. Questa carica può essere misurata con la libreria Capacitive Sensor, come abbiamo visto nel Progetto 13.

Carico - Un dispositivo che trasforma l'energia elettrica in qualcosa d'altro, come la luce, il calore o il suono.

Catodo - Il polo di un condensatore o di un diodo che normalmente è collegato a massa.

Ciclo di lavoro - Un rapporto che indica la quantità di tempo in cui un componente è acceso in un certo periodo. Quando si utilizza un valore di PWM di 127 (su un totale di 256), si sta creando un ciclo di lavoro del 50%.

Circuiti integrati (IC) - Un circuito che è stato creato su un piccolo pezzo di silicio e incapsulato in plastica (o resina epossidica). Piedini sporgenti dal corpo consentono di interagire con il circuito interno. Molto spesso siamo in grado di fare buon uso di un circuito integrato sapendo solo cosa collegare ai piedini senza dover capire come funziona internamente.

Circuito - Un percorso circolare che parte da un alimentatore, attraversa un carico e poi rientra attaverso l'altro polo dell'alimentatore. La corrente scorre in un circuito solo se è chiuso, cioè, se i percorsi di uscita e di ritorno sono entrambi continui o chiusi. Se un percorso viene interrotto, o aperto, allora la corrente non fluirà attraverso il circuito.

Comunicazione seriale - Il modo in cui Arduino comunica con il computer e con altri dispositivi. Comporta l'invio di un bit di informazione alla volta in sequenza. Arduino ha un convertitore USB seriale, che gli permette di comunicare con i dispositivi che non dispongono di una porta seriale dedicata.

Condensatori di disaccoppiamento - Condensatori che vengono utilizzati per ammorbidire i picchi e le cadute di tensione. Sono spesso posizionati vicino a un sensore o a un attuatore.

Conduttore - Qualcosa che permette all'elettricità di scorrere, per esempio un filo elettrico.

Controtensione - Tensione che si oppone alla corrente che l'ha creato. Può essere generata da motori che rallentano. Ciò può danneggiare i circuiti, per questo spesso si usano i diodi in combinazione con i motori.

Convertitore analogico-digitale (Analog to Digital Converter ADC) - Un circuito che converte una tensione analogica in un numero digitale che la rappresenta. Questo circuito è parte del microcontrollore ed è collegato ai piedini di ingresso analogico A0-A5. Convertire una tensione analogica in un numero digitale richiede un po' di tempo, perciò la funzione analogRead() produce un breve ritardo nel programma.

Corrente - Il flusso di carica elettrica che attraversa un circuito chiuso. Si misura in ampere.

Corrente alternata - Un tipo di corrente in cui l'elettricità cambia direzione periodicamente. Questa è l'elettricità fornita dalle prese di casa.

Corrente continua - Un tipo di corrente che scorre sempre nella stessa direzione. Tutti i progetti nel kit usano la corrente continua.

Cortocircuito - Un contatto diretto tra l'alimentazione e la massa crea un cortocircuito e blocca il tuo circuito. In alcuni casi potrebbe danneggiare l'alimentatore o parti del circuito e in rari casi potrebbe provocare un incendio.

Costante - Un identificatore il cui valore non può essere cambiato durante l'esecuzione di un programma.

D

Datasheet - Un documento scritto da ingegneri per altri ingegneri che descrive le caratteristiche dei componenti elettronici. Le informazioni tipiche in un datasheet comprendono la massima tensione e la corrente richiesti da un componente oltre a una spiegazione del suo funzionamento.

Debugging - La procedura di verifica di un circuito o di un programma per trovare errori (noti anche come 'bug'), fino a quando non si ottiene il comportamento desiderato.

Digitale - Un sistema a valori discreti. Arduino è un tipo di dispositivo digitale, conosce solo due stati discreti, acceso e spento.

Drain (scarico) - Il piedino a cui si collega il carico da comandare.

Dual In-line Package (DIP) - Un tipo di contenitore per circuiti integrati che permette ai componenti di essere facilmente inseriti in una breadboard.

E

Elettricità - Un tipo di energia generata da cariche elettriche. Puoi utilizzare i componenti elettronici per trasformare l'elettricità in altre forme di energia, come la luce e il calore.

F

Float (numero a virgola mobile) - Un tipo di dato per esprimere numeri con la virgola.

Fotoaccoppiatore - Conosciuto anche come optoisolatore, fotoisolatore, fotointerruttore e optointerruttore. Un LED e un fototransistor vengono accoppiati all'interno di un contenitore sigillato. Il LED è posizionato per illuminare il fototransistor, in modo che quando il LED è acceso il fototransistor condurrà. Utilizzato per fornire un elevato grado di

isolamento perché non vi è collegamento elettrico in comune tra l'ingresso e l'uscita.

Fotocellula - Un dispositivo che converte la luce in energia elettrica.

Fotoresistenza - Un componente in cui la resistenza varia a seconda della quantità di luce che lo colpisce.

Fototransistor - Un transistor che è controllato dalla luce invece che dalla corrente.

Funzione - Un blocco di codice che esegue un compito specifico.

G

Gate (porta) - Il piedino di un transistor che è collegato ad Arduino. Quando il gate è attivato applicando 5V, chiude la giunzione tra drain e source completando il circuito a cui è collegato.

I

IDE - 'Integrated Development Environment' (ambiente di sviluppo integrato). L'IDE di Arduino è l'applicazione che si usa per scrivere il software da caricare su Arduino. Esso contiene tutte le funzioni che Arduino può capire. Altri ambienti di programmazione, come Processing, hanno il loro proprio IDE.

Indice - Il numero fornito a un array che indica l'elemento a cui fai riferimento. I computer iniziano a contare da 0 invece di 1: perciò per accedere al terzo elemento di un array denominato tones è necessario scrivere tones[2].

Induzione - La procedura di utilizzare un campo magnetico per creare energia elettrica.

Int - Un tipo di dato che contiene un numero tra -32,768 e 32,767.

Interruttore - Un componente in grado di aprire o chiudere un circuito elettrico. Ci sono molti tipi di interruttori; quelli nel kit sono 'momentanei': chiudono il circuito solo se vengono premuti.

Isolante - Qualcosa che impedisce all'energia elettrica di fluire. Materiali conduttivi come i fili sono spesso coperti di isolanti come la gomma.

Istanza - Una copia di un oggetto software. Hai utilizzato istanze della libreria Servo nei Progetti 05 e 12; in ogni caso, hai creato un'istanza della libreria Servo da utilizzare nel progetto.

L

LED a catodo comune - Tipi di LED che hanno colori diversi in uno stesso componente, con un catodo e diversi anodi.

Legge di Ohm - Un'equazione matematica che illustra il rapporto tra resistenza, corrente e tensione. Di solito viene indicata come V (tensione) = I (corrente) x R (resistenza).

Libreria - Pezzo di codice che espande le funzionalità di un programma. Nel caso delle librerie Arduino, consentono la comunicazione con un particolare componente hardware o sono utilizzate per manipolare dati.

Long - Un tipo di dato che contiene un numero molto grande, da -2,147,483,648 a 2,147,483,647.

M

Massa - Il punto di un circuito in cui l'energia potenziale elettrica è pari a 0. Senza massa, l'elettricità non avrebbe modo di fluire nel circuito.

Microcontrollore - Il cervello di Arduino: è un piccolo computer che si può programmare per ascoltare, elaborare e visualizzare informazioni dell'ambiente circostante.

Millesimo di secondo - Un 1/1.000 di secondo. Arduino esegue i suoi programmi così velocemente che delay() e le altre funzioni basate sul tempo si misurano in millesimi di secondo.

Modulazione di larghezza di impulso (Pulse Width Modulation PWM) - Un metodo per simulare un'uscita analogica quando si utilizza un dispositivo digitale. La modulazione di larghezza di impulso fa accendere e spegnere un piedino a velocità molto elevata. Il rapporto tra il tempo di accensione e il tempo di spegnimento determina il valore del risultato analogico simulato.

Monitor seriale - Uno strumento parte dell'IDE di Arduino che permette l'invio e la ricezione di dati seriali da e per un'Arduino collegata. Guarda il set di funzioni Serial().

O

Oggetto - Un'istanza di una libreria. Quando usi la libreria Servo, se crei un'istanza chiamata myServo, myServo sarà l'oggetto.

Ohm - Unità di misura della resistenza; è rappresentata dal simbolo omega (Ω).

Onda quadra – Un tipo di forma d'onda che viene identificata dal fatto di avere solo due stati, acceso e spento. Quando viene utilizzata per generare dei toni, può suonare come un ronzio.

P

Parallelo - I componenti collegati tra gli stessi due punti in un circuito sono in parallelo. Componenti in parallelo hanno sempre la stessa tensione su di essi.

Parametro - Quando si dichiara una funzione, un parametro denominato fa da ponte tra le variabili locali nella funzione e gli argomenti che riceve quando viene chiamata la funzione.

Partitore di tensione - Un tipo di circuito che fornisce in uscita una tensione che è una frazione di quella applicata all'ingresso. Stai costruendo un partitore di tensione quando combini una fotoresistenza con una resistenza fissa per fornire un valore di ingresso analogico. Un potenziometro è un altro esempio di un partitore di tensione.

Periodo - Uno specifico intervallo di tempo nel quale succede qualcosa. Quando cambia il periodo, varia la frequenza alla quale si verificherà qualcosa.

Polarizzato - I terminali dei componenti polarizzati (ad esempio LED o condensatori) hanno funzioni diverse e, quindi, devono essere collegati nel modo giusto. I componenti polarizzati collegati nel modo sbagliato potrebbero non funzionare, essere danneggiati o danneggiare altre parti del circuito. I componenti non polarizzati (come le resistenze) possono essere collegati in entrambi i modi.

Processing - Un ambiente di programmazione basato sul linguaggio Java. È usato come strumento per insegnare i concetti di programmazione, e per realizzare prodotti finiti. L'IDE di Arduino è basato su quello di Processing, quindi anche questo ti risulta familiare. Inoltre, Processing e Arduino condividono una filosofia e un obiettivo simile: la creazione di strumenti

open source che permettono a persone non tecniche di lavorare con hardware e software.

Pseudocodice - Una via di mezzo tra la scrittura in un linguaggio di programmazione e l'utilizzo di un linguaggio naturale. Durante la creazione di pseudocodice, è utile scrivere brevi frasi descrittive.

R

Resistenza - Indica la tendenza di un materiale a opporsi al passaggio della corrente elettrica. In particolare, la resistenza può essere calcolata con la legge di Ohm: R = V / I.

S

Saldatura - Il metodo per realizzazione una connessione elettrica scogliendo una lega saldante sui componenti elettrici o fili che devono essere collegati. Crea un collegamento robusto tra i componenti.

Sensore - Un componente che misura una forma di energia (come la luce, il calore o l'energia meccanica) e la converte in energia elettrica, che Arduino può capire.

Serie - I componenti sono in serie quando sono connessi in cascata. La corrente che li attraversa è la stessa, e la tensione cade su ciascun componente a seconda della resistenza di ognuno.

Sketch - Il nome che viene dato ai programmi scritti con l'IDE di Arduino.

Source (sorgente) - Il piedino su un transistor che è collegato a massa. Quando il gate viene alimentato, source e drain sono collegati, completando il circuito in cui il transistor è inserito.

T

Tensione - Misura dell'energia potenziale, che spingerebbe una carica all'interno di un circuito chiuso.

Tipi di dato - Un sistema di classificazione che determina quali valori può contenere una particolare costante, una variabile o un array. Int, float, long e boolean sono tutti tipi che possono essere utilizzati in Arduino.

Transistor - Un dispositivo elettronico (di solito) a tre terminali che può agire come un amplificatore o un interruttore. Una tensione o una corrente di controllo applicata a un terminale controlla una (di solito) maggiore tensione o corrente tra una diversa coppia di terminali. I tipi di transistor più comuni sono il transistor a giunzione bipolare (BJT) e il transistor MOSFET. Spesso viene usato per permettere a una piccola corrente fornita da un'Arduino (limitata a 40 mA) di controllare una corrente sostanzialmente più grande, come quelle di cui hanno bisogno motori, relè o lampade a incandescenza. A seconda di come sono costruiti, i transistor MOSFET sono a canale N o P, elemento che determina il modo in cui devono essere collegati.

Trasduttore - Qualcosa che cambia una forma di energia in un'altra.

U

Unsigned - Un termine usato per descrivere alcuni tipi di dati, che indica che non possono rappresentare un numero negativo. È utile avere un numero unsigned se hai bisogno di contare solo in una direzione. Per esempio, quando si tiene traccia del tempo con millis(), si consiglia di utilizzare il tipo di dato unsigned long.

USB - Universal Serial Bus. Porta generica che è standard sulla maggior parte dei computer di oggi. Con un cavo USB, è possibile programmare e alimentare Arduino tramite una connessione USB.

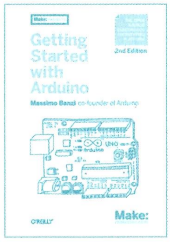
V

Variabile - Un'area nella memoria del computer o del microcontrollore per la memorizzazione delle informazioni necessarie in un programma. Le variabili memorizzano valori che possono cambiare durante l'esecuzione del programma. Il tipo di variabile dipende dal tipo di informazione che si desidera memorizzare e dalla sua dimensione massima. Per esempio un byte può memorizzare fino a 256 valori diversi, ma un int può memorizzare fino a 65.536 valori diversi. Le variabili possono essere locali di un particolare blocco di codice o globali per un intero programma.

Variabile globale - Una variabile a cui si può accedere in qualsiasi punto all'interno del programma. Viene dichiarata prima della funzione setup().

Variabile locale - Un tipo di variabile che viene utilizzato per un breve lasso di tempo, poi dimenticato. Una variabile dichiarata all'interno del setup() di un programma è locale: dopo che è terminato il setup(), Arduino si dimentica che la variabile sia mai esistita.

ALTRE LETTURE



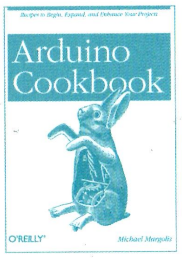
Getting Started with Arduino, 2° Edizione di **Massimo Banzi** [O'Reilly Media / Make, 2011]. La migliore introduzione a Arduino.

Getting Started with Processing di **Casey Reas** e **Ben Fry** [O'Reilly Media / Make, 2010]. Questa breve guida per l'ambiente di programmazione Processing spiega come programmare grafiche, suoni e contenuti multimediali.

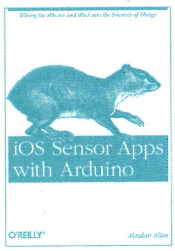
Making Things Talk, 2° Edizione di **Tom Igoe** [O'Reilly Media / Make, 2011]. Scritto per utenti Arduino più esperti, questo libro offre molte tecniche per la comunicazione tra Arduino e altri dispositivi su internet, e non solo.

Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction di **Daniel Shiffman** [Morgan Kaufman, 2009]. Una approfondita introduzione alla programmazione usando Processing, per beginners di tutte le età.

Getting Started with RFID di **Tom Igoe** [O'Reilly Media / Make, 2012]. Una breve introduzione all'utilizzo di identificazione a radiofrequenza con Arduino e Processing.



The Arduino Cookbook, 2° Edizione di **Michael Margolis** [O'Reilly Media / Make, 2011]. Questo libro ha molte grandi ricette per come utilizzare Arduino in modi più avanzati.



Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists di **Dustyn Roberts** [McGraw-Hill, 2010]. Una grande risorsa sulla costruzione di meccanismi mobili per l'interfaccia con i tuoi progetti.

Make: Electronics di **Charles Platt** [O'Reilly Media / Make, 2009]. Scritta brillantemente, è l'introduzione per l'elettronica adatta a chiunque. Arduino non è utilizzata in questo libro, ma è un testo prezioso per capire meglio l'elettronica.

iOS Sensor Apps with Arduino di **Alasdair Allan** [O'Reilly Media / Make, 2011]. Con questa breve guida, imparerai come collegare un sensore esterno a un dispositivo iOS e farli parlare tra loro tramite Arduino. Potrai anche creare un'applicazione iOS che analizza i valori del sensore che riceve e traccia le misure conseguenti, il tutto in tempo reale.

Arduino ti aiuta a realizzare cose che interagiscono con te. Che tu sia uno scienziato o un poeta, che tu abbia 10 o 90 anni, vogliamo permetterti di costruire grandi progetti con computer ed elettronica.

Le parti contenute nel kit e i progetti sono spiegati fin nei minimi dettagli. Arduino può rendere interattivi i tuoi progetti. Sta a te renderli belli.

ARDUINO

ARDUINO — IL LIBRO DEI PROGETTI

ARDUINO IL LIBRO DEI PROGETTI