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

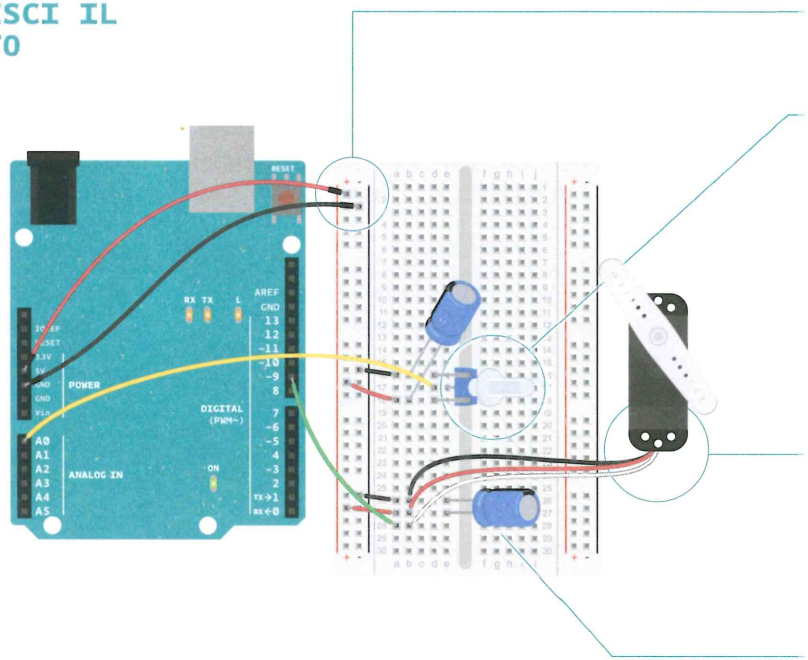


Fig. 1

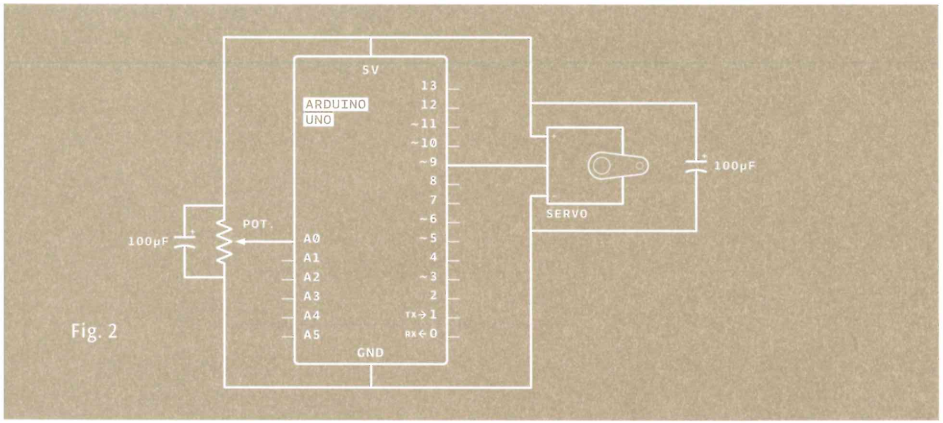


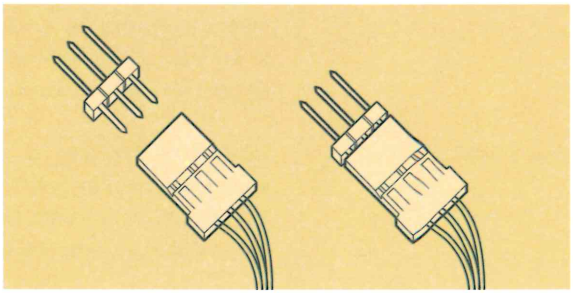
Fig. 2

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.



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.
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.
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.
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.
Leggi i valori del potenziometro	Nel `loop()`, leggi l'ingresso analogico e invia il valore al monitor seriale.
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.
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.

```
#include <Servo.h>

Servo myServo;

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

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

  Serial.begin(9600);
}

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

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

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

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

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.

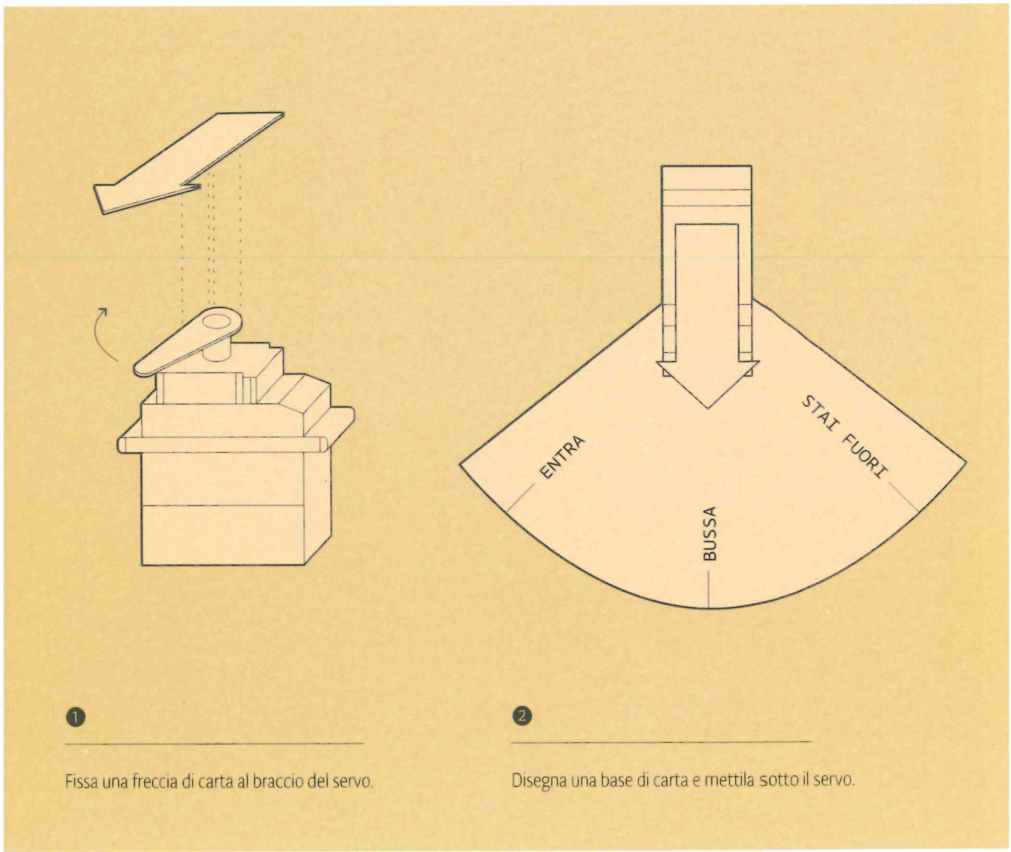
I servomotori sono normali motori con all'interno ingranaggi e un circuito. La meccanica all'interno fornisce un feedback al circuito, così è sempre consapevole della sua posizione. Anche se sembra che abbia una gamma limitata di movimenti, è possibile ottenerne una grande varietà con alcune parti meccaniche 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.

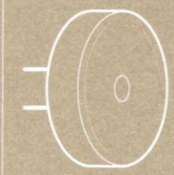
Ora che la parte di movimento è in funzione, è tempo di far sapere alle persone che sei disponibile ad aiutarli nei loro progetti o che vuoi essere lasciato in pace per pianificare la tua prossima creazione.

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.



PIEZO

FOTORESISTENZA

RESISTENZA DA 10 KILO OHM

THEREMIN COMANDATO DALLA LUCE

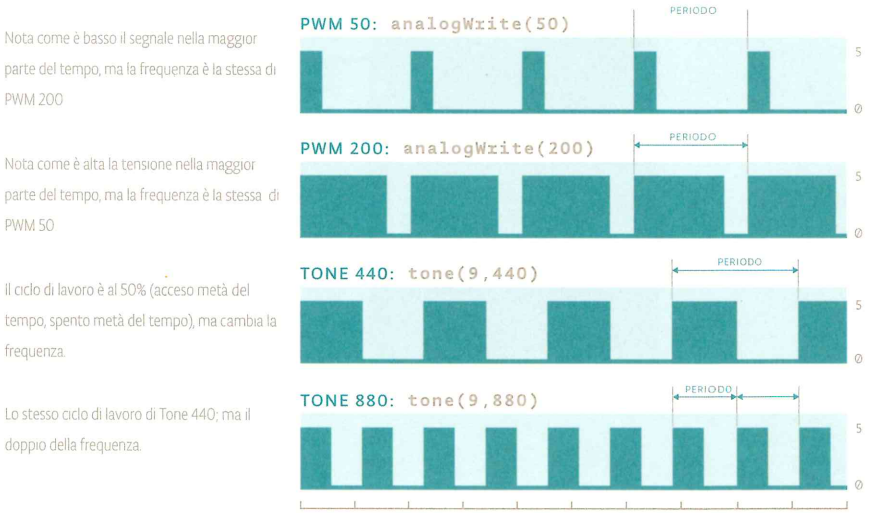
È ORA DI FARE RUMORE! USANDO UNA FOTORESISTENZA E UN PIEZO, POTRAI FARE IL TUO THEREMIN COMANDATO DALLA LUCE

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

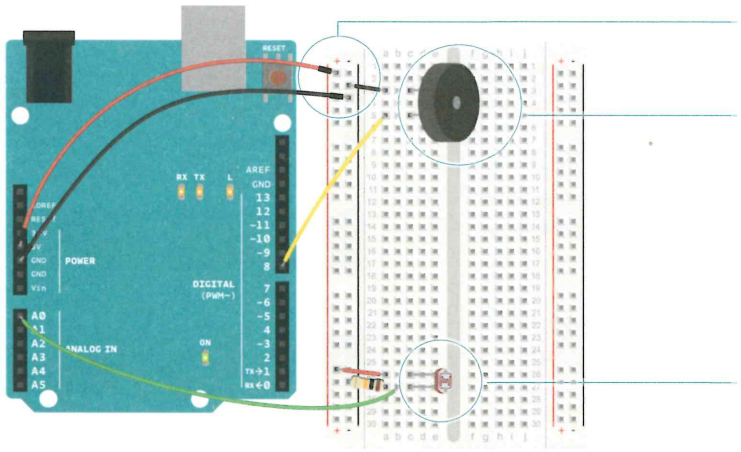


Fig. 2

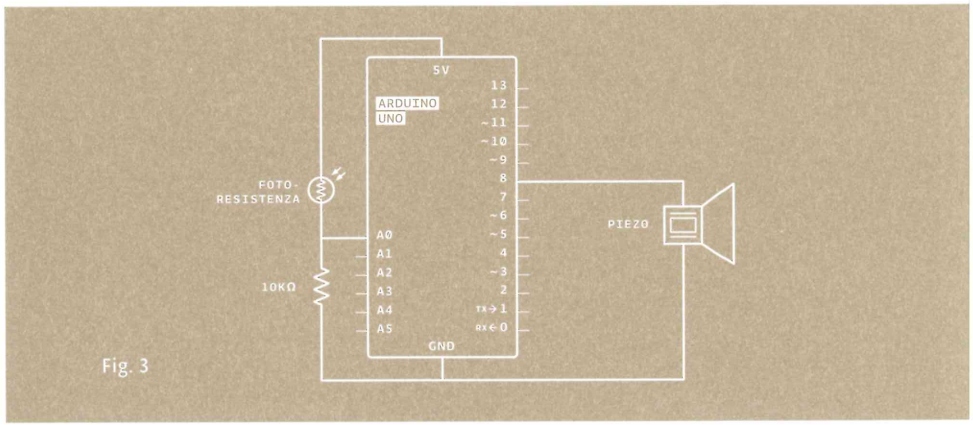


Fig. 3

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?

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.

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.

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.

Imposta la direzione del piedino digitale e accendilo

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

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.

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.

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.

```
int sensorValue;
int sensorLow = 1023;
int sensorHigh = 0;

const int ledPin = 13;

void setup() {

  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH);

  while (millis() < 5000) {

    sensorValue = analogRead(A0);
    if (sensorValue > sensorHigh) {
      sensorHigh = sensorValue;
    }
    if (sensorValue < sensorLow) {
      sensorLow = sensorValue;
    }
  }

  digitalWrite(ledPin, LOW);
}
```

Comando while()
arduino.cc/while

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.

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.

```
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 }
```

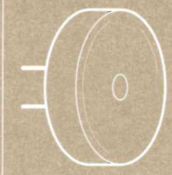
L'intervallo nella funzione `map()` che determina il suono è piuttosto ampio; prova a cambiare le frequenze per trovarne una giusta per il tuo stile musicale.

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.

INGREDIENTI

PULSANTE



PIEZO

RESISTENZA DA 10 KILO OHM

RESISTENZA DA 1 MEGA OHM

RESISTENZA DA 220 OHM

07

TASTIERA MUSICALE

CON ALCUNE RESISTENZE E PULSANTI COSTRUISCI UNA PICCOLA TASTIERA MUSICALE

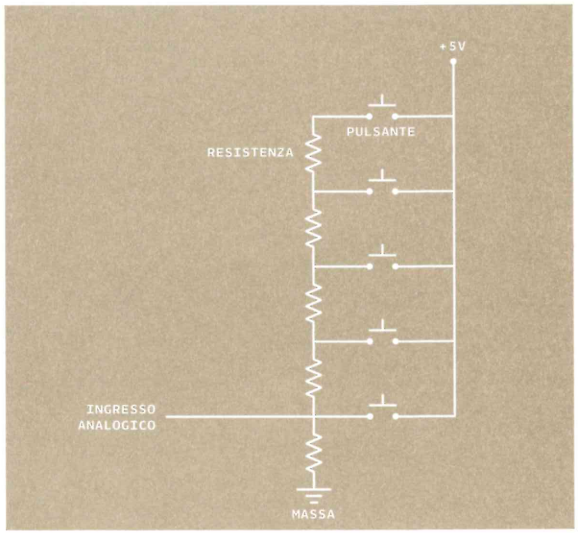
Scopri: la rete di resistenze a scala, gli array

Tempo: **45 MINUTI**
Livello: ■■■■□

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

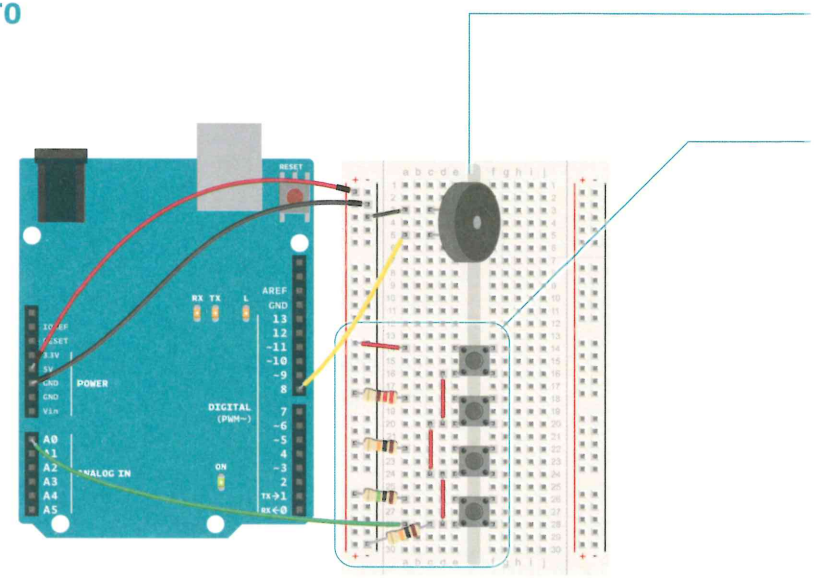
Anche se è possibile semplicemente collegare un numero di pulsanti a ingressi digitali per avere toni diversi, in questo progetto useremo una ***rete di resistenze a scala****.*

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

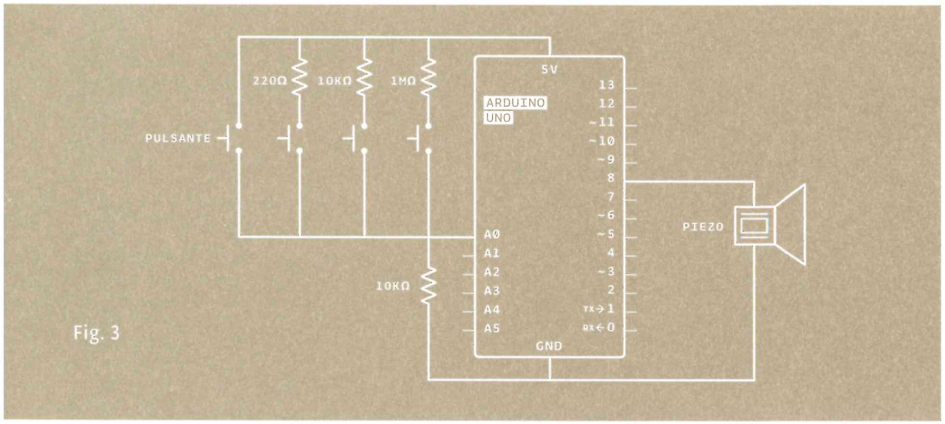
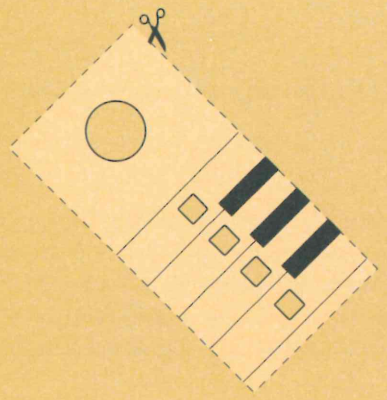


Fig. 3

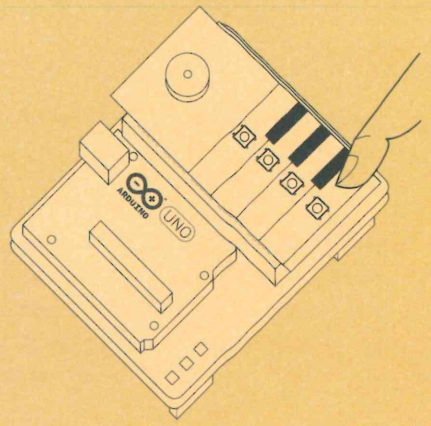
1. Collega la breadboard all'alimentazione e a massa come nei progetti precedenti. Collega un'estremità del piezo a massa. Collega l'altra estremità al piedino 8 di Arduino.

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.

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()`.

Comunicazione seriale

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

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)`.

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.

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

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

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

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

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

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

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 A0 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.

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.

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.

```
  else if(keyVal >= 990 && keyVal <= 1010){
    tone(8, notes[1]);
  }
  else if(keyVal >= 505 && keyVal <= 515){
    tone(8, notes[2]);
  }
  else if(keyVal >= 5 && keyVal <= 10){
    tone(8, notes[3]);
  }
```

```
  else{
    noTone(8);
  }
}
```

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.

INGREDIENTI

PULSANTE

LED

RESISTENZA DA 10 KILO OHM

RESISTENZA DA 220 OHM

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

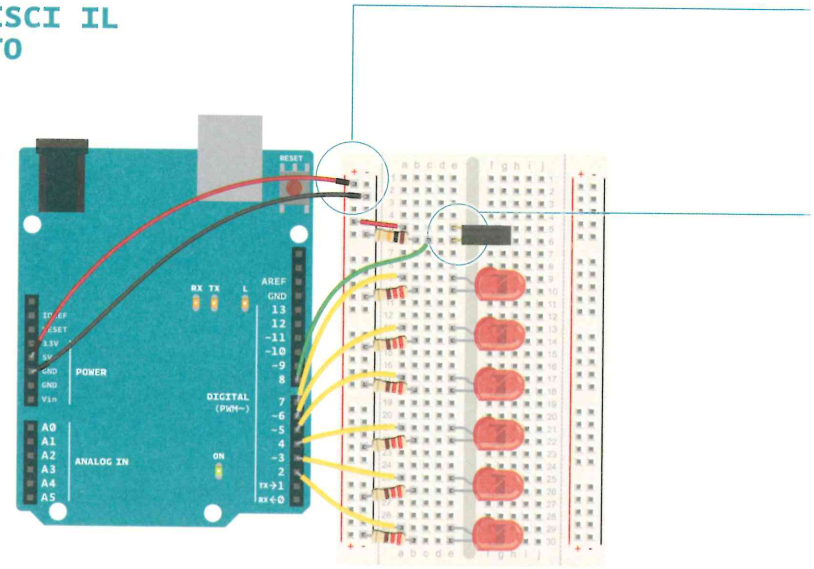


Fig. 1

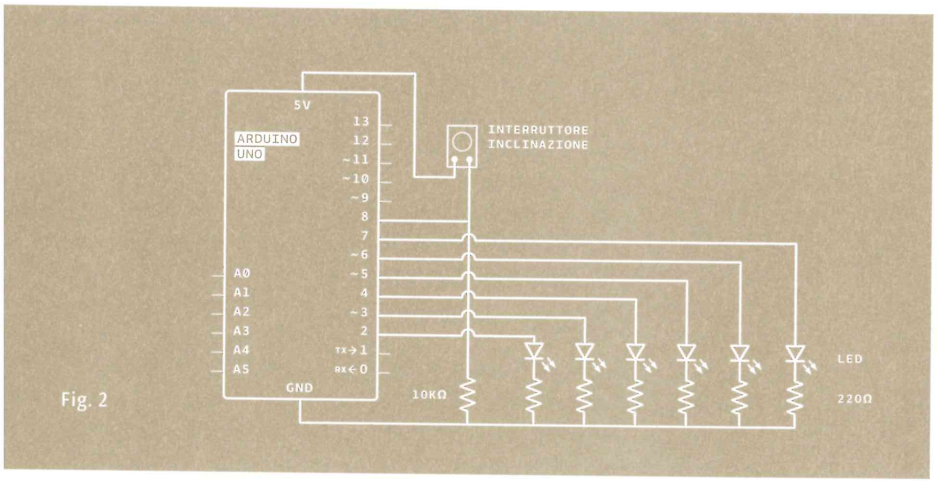


Fig. 2

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.

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.

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.

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.

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.

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`.

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`.

```
const int switchPin = 8;

unsigned long previousTime = 0;

int switchState = 0;
int prevSwitchState = 0;

int led = 2;

long interval = 600000;

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

  pinMode(switchPin, INPUT);
}

void loop(){
  unsigned long currentTime = millis();

  if(currentTime - previousTime > interval) {
    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.

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.

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`.

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`.

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.

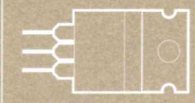
USALO

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.

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


    if(led == 7){
    }
  }

  switchState = digitalRead(switchPin);


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

    led = 2;
    previousTime = currentTime;
  }

  prevSwitchState = switchState;
}
```

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.



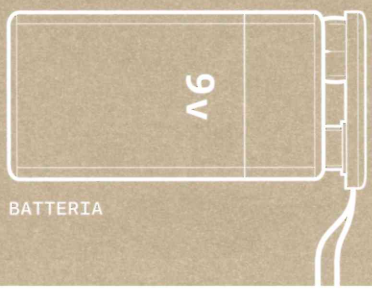
MOSFET

RESISTENZA DA 10 KILO OHM

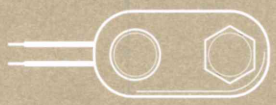
DIODO 1N4007

MOTORE

PULSANTE



BATTERIA



CONNETTORE PER BATTERIA

GIRANDOLA MOTORIZZATA

USA ARDUINO PER FAR RUOTARE UNA GIRANDOLA COLORATA UTILIZZANDO UN MOTORE

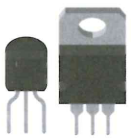
Scopri: transistor, forti carichi di corrente e di tensione

Tempo: **45 MINUTI**
Livello: ■■■□□

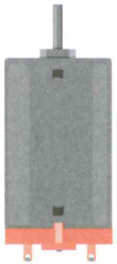
Basato sui progetti: **1, 2, 3, 4**

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

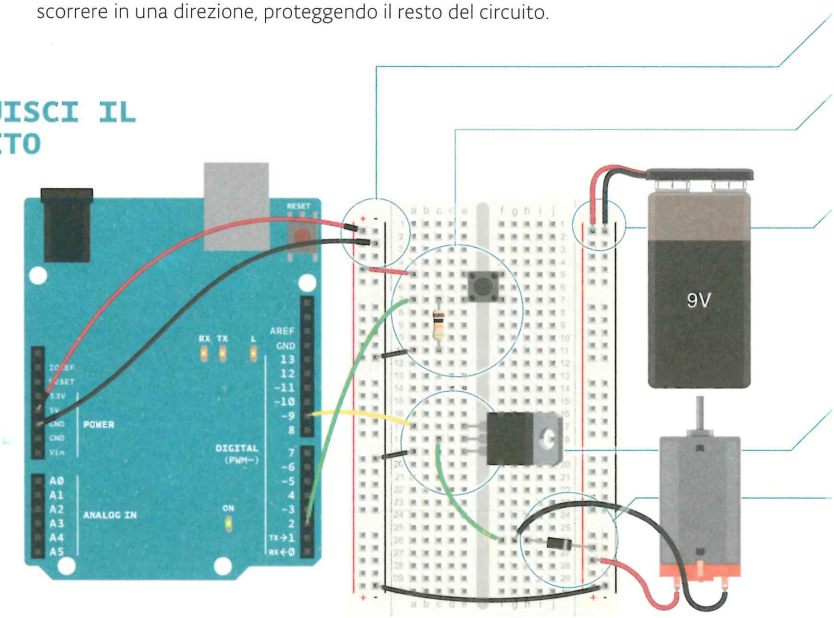


Fig. 1

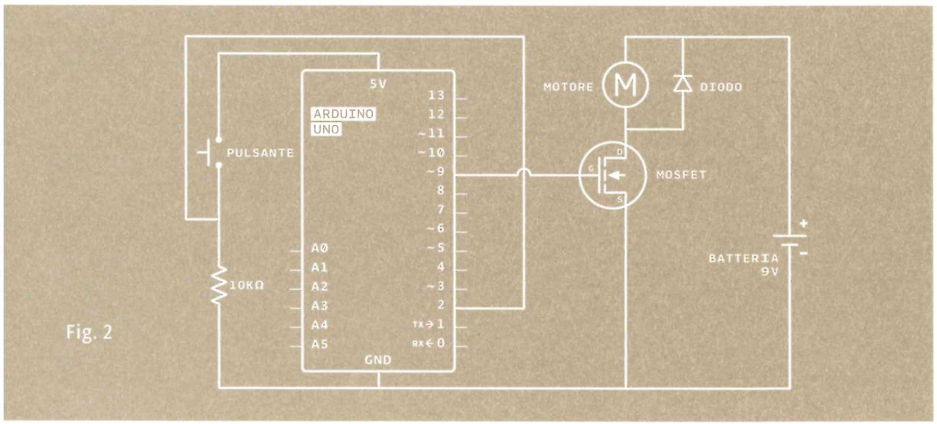


Fig. 2

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).

Dichiara la direzione dei piedini

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

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.

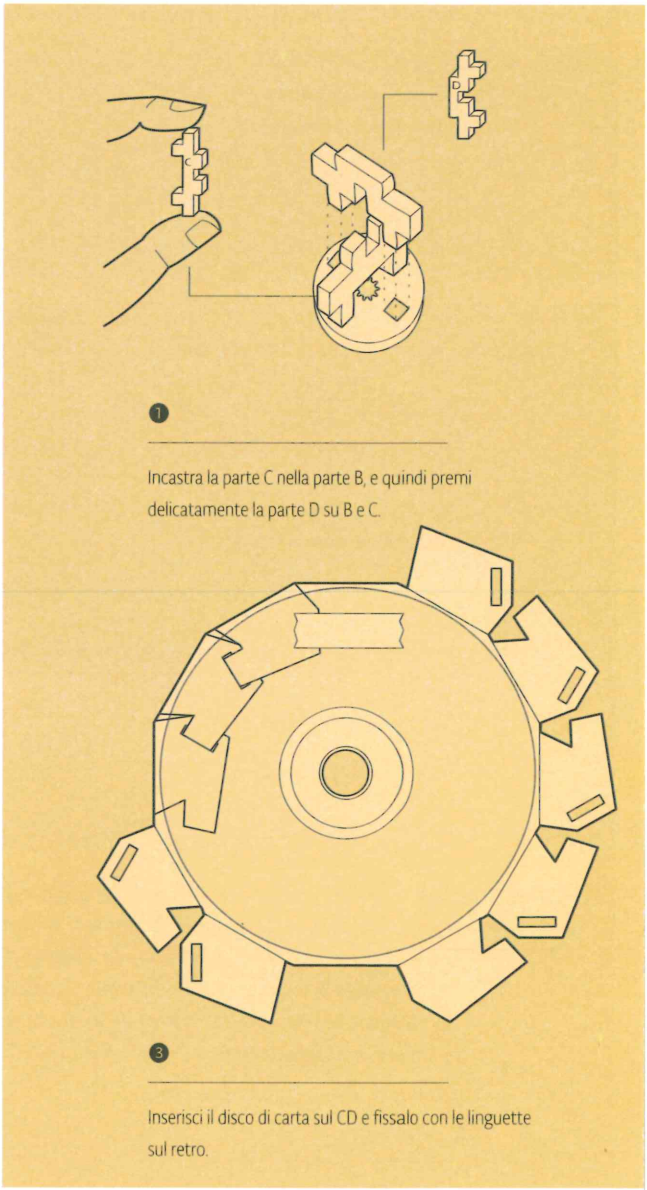
I motori richiedono una particolare attenzione quando sono controllati da un microcontrollore; questo, infatti, non può fornire abbastanza corrente e/o tensione per alimentare un motore. Inoltre è bene utilizzare diodi per evitare di danneggiare il circuito.

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

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

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

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

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.



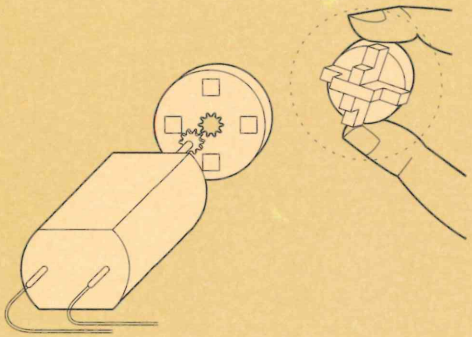
❶

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

❸

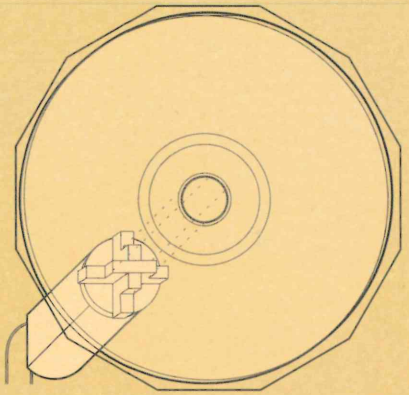
Inserisci il disco di carta sul CD e fissalo con le linguette sul retro.

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

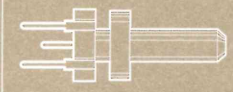
Premi delicatamente l'albero motore nel foro nella parte posteriore della parte B.



4

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

INGREDIENTI

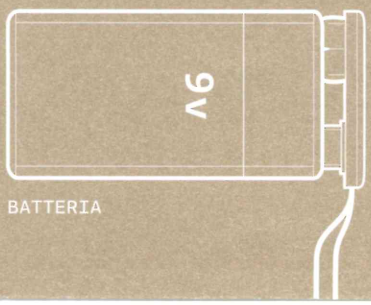


POTENZIOMETRO

PONTE H

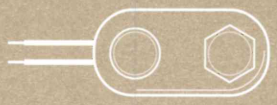
RESISTENZA DA 10 KILO OHM

MOTORE



BATTERIA

PULSANTE



CONNETTORE PER BATTERIA

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.

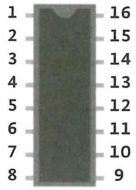


Fig. 1

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

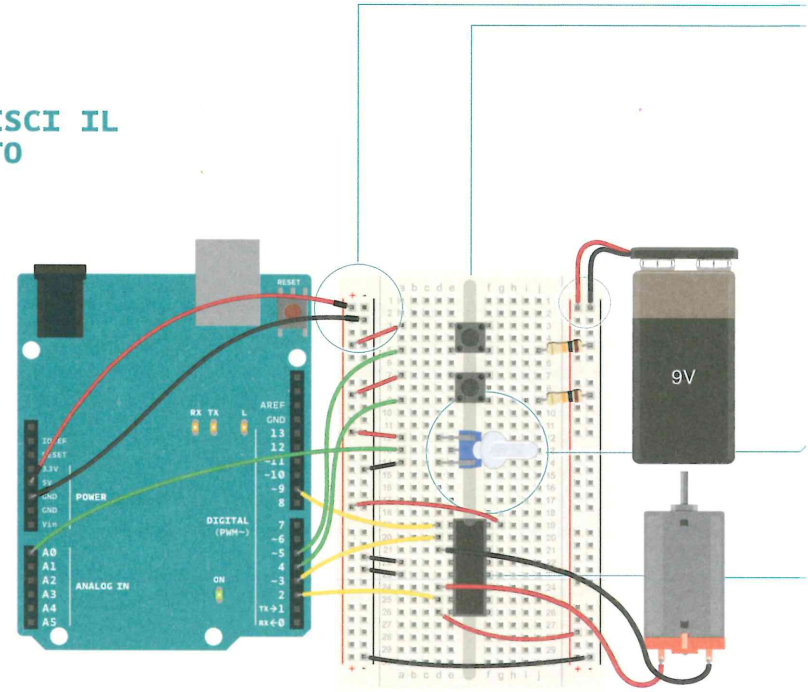


Fig. 2

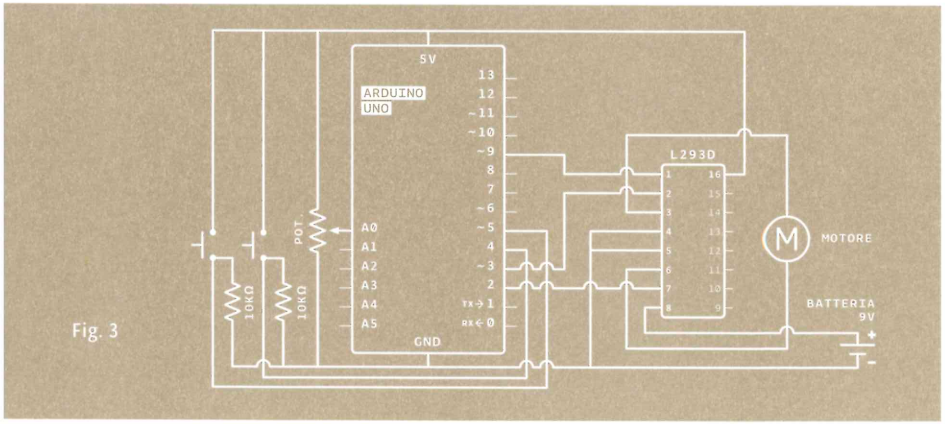


Fig. 3

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.

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.

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.

Dichiara i piedini digitali come input e output

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

Spegni il motore

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

Leggi il sensore

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

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

int onOffSwitchState = 0;
int previousOnOffSwitchState = 0;
int directionSwitchState = 0;
int previousDirectionSwitchState = 0;

int motorEnabled = 0;
int motorSpeed = 0;
int motorDirection = 1;

void setup(){
  pinMode(directionSwitchPin, INPUT);
  pinMode(onOffSwitchStateSwitchPin, INPUT);
  pinMode(controlPin1, OUTPUT);
  pinMode(controlPin2, OUTPUT);
  pinMode(enablePin, OUTPUT);

  digitalWrite(enablePin, LOW);
}

void loop(){
  onOffSwitchState =
    digitalRead(onOffSwitchStateSwitchPin);
  delay(1);
  directionSwitchState =
    digitalRead(directionSwitchPin);
  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.

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`.

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`.

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.

Salva lo stato attuale per il prossimo loop()

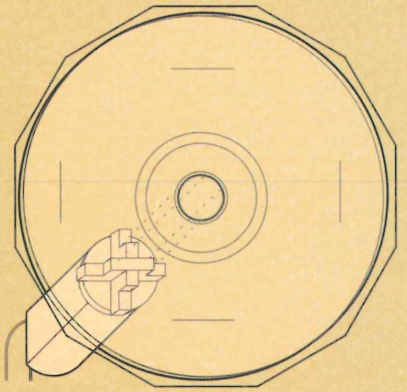
Prima di uscire dal `loop()`, salva lo stato attuale degli interruttori come stato precedente per la prossima parte del programma.

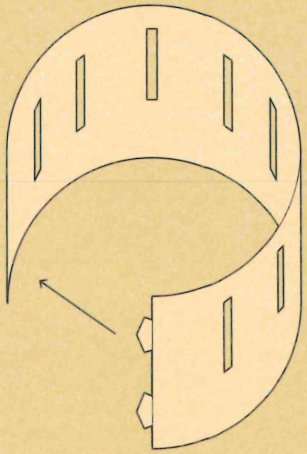
```
27  if(onOffSwitchState != previousOnOffSwitchState){
28    if(onOffSwitchState == HIGH){
29      motorEnabled = !motorEnabled;
30    }
31  }

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

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

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

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

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.





1

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

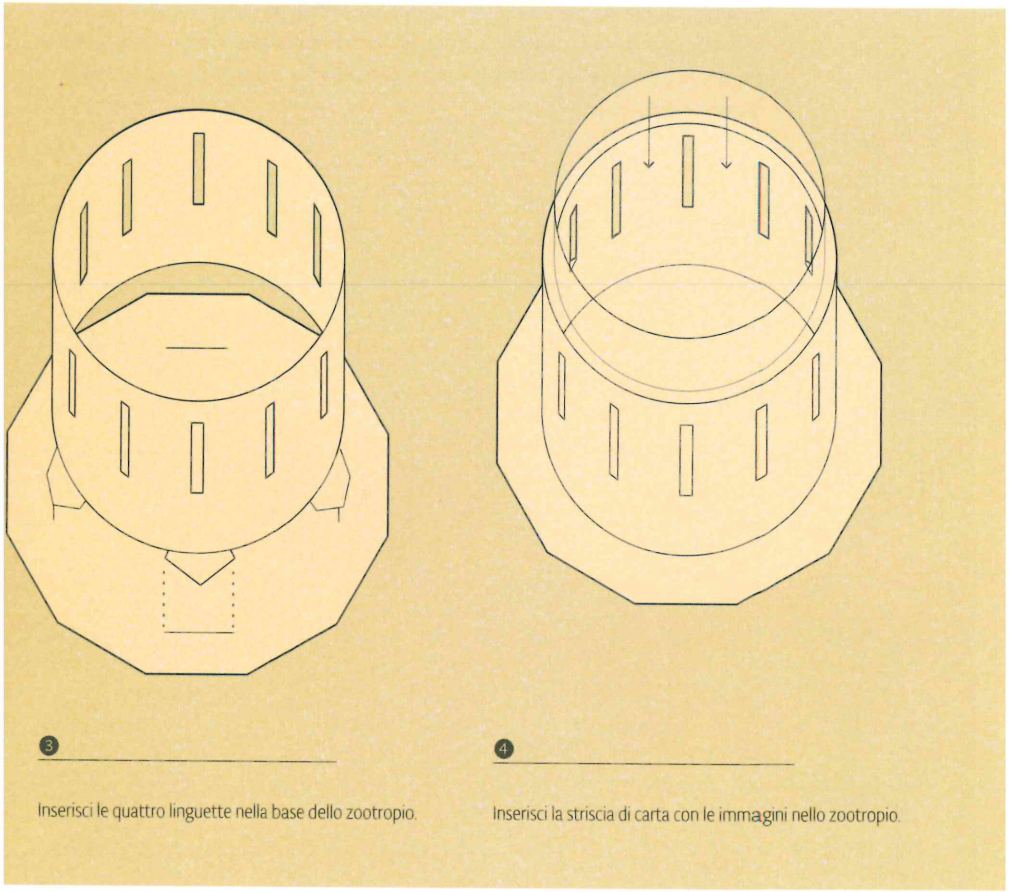
2

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

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.



3

Inserisci le quattro linguette nella base dello zootropio.

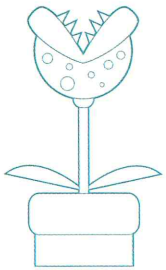
4

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.

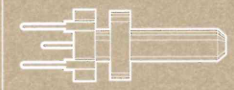
Con un po' di lavoro, puoi fare in modo che il tuo zootropio funzioni anche in situazioni di scarsa luminosità. Collega un LED e una resistenza a uno dei piedini di uscita digitali liberi. Aggiungi anche un secondo potenziometro e collegalo a un ingresso analogico. Metti la luce in modo che illumini 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!



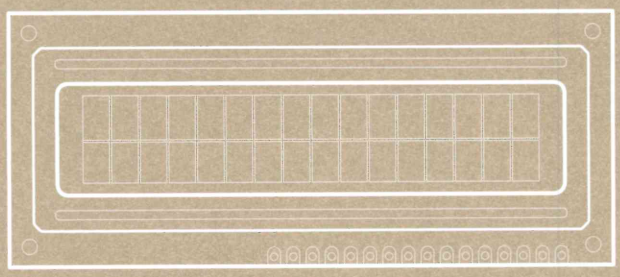
PULSANTE

RESISTENZA DA 10 KILO OHM

RESISTENZA DA 220 OHM



POTENZIOMETRO



SCHERMO LCD

SFERA DI CRISTALLO

CREA UNA SFERA DI CRISTALLO PER PREDIRE IL FUTURO

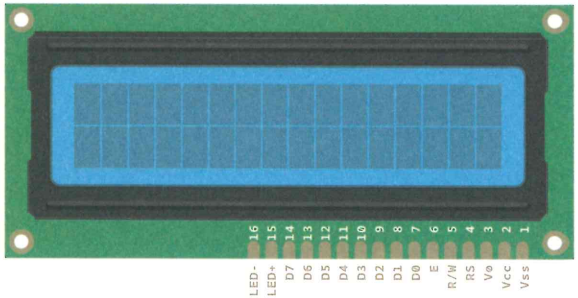
Scopri: i display LCD, istruzioni switch/case, random()

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

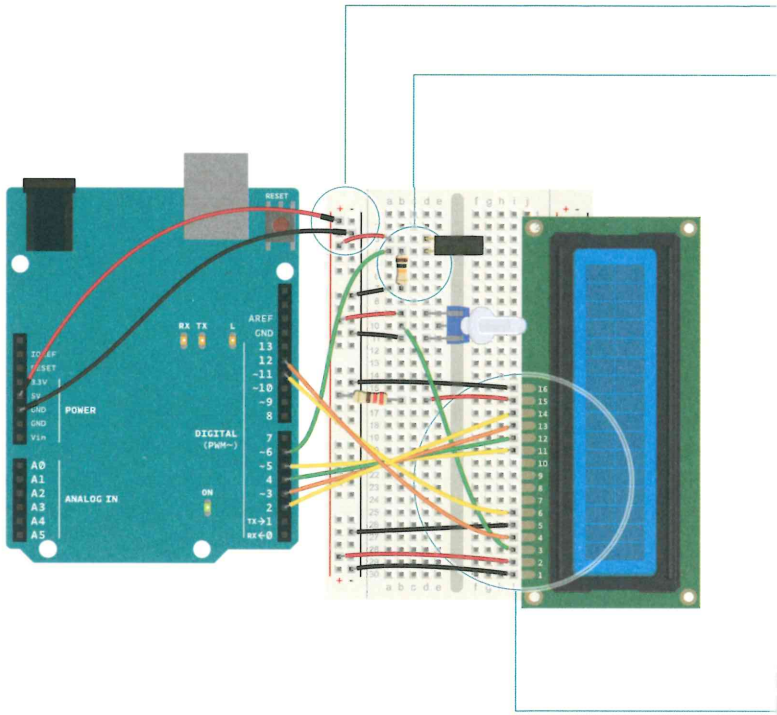


Fig. 2

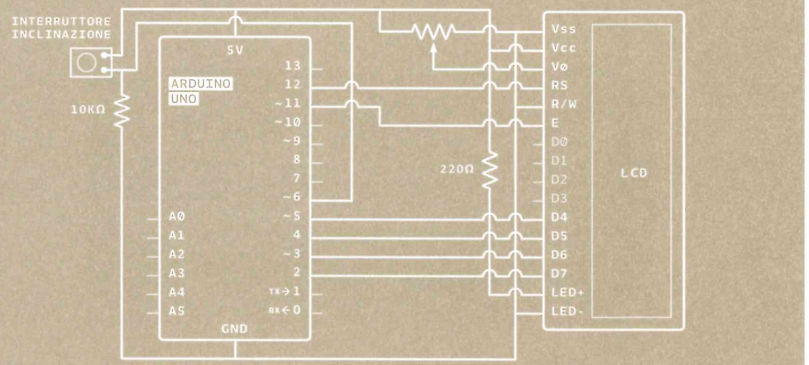


Fig. 3

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.

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.

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

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.

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!".

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

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`.

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

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

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

  lcd.print("Interroga");

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

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

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

La descrizione della libreria LCD
arduino.cc/lcdlibrary

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.

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.

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.

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

    switch(reply){
      case 0:
      lcd.print("Si");
      break;
      case 1:
      lcd.print("Probabile");
      break;
      case 2:
      lcd.print("Certo");
      break;
      case 3:
      lcd.print("Bene");
      break;
      case 4:
      lcd.print("Forse");
      break;
      case 5:
      lcd.print("Chiedi ancora");
      break;
      case 6:
      lcd.print("Improbabile");
      break;
      case 7:
      lcd.print("No");
      break;
    }
  }
}

  prevSwitchState = switchState;
}
```

Descrizione di Switch Case

arduino.cc/switchcase

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.

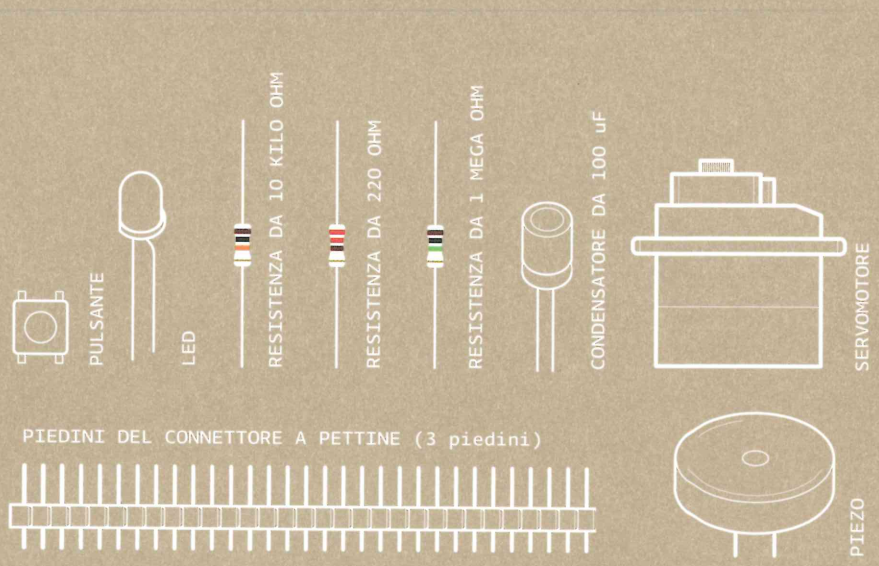
Prova ad aggiungere le tue parole all'istruzione `print()`, ma devi essere consapevole del fatto che puoi usare solo 16 caratteri per riga. Puoi anche provare ad aggiungere altre risposte. Assicurati, quando aggiungi i case supplementari, di modificare il numero di opzioni che casualmente popoleranno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

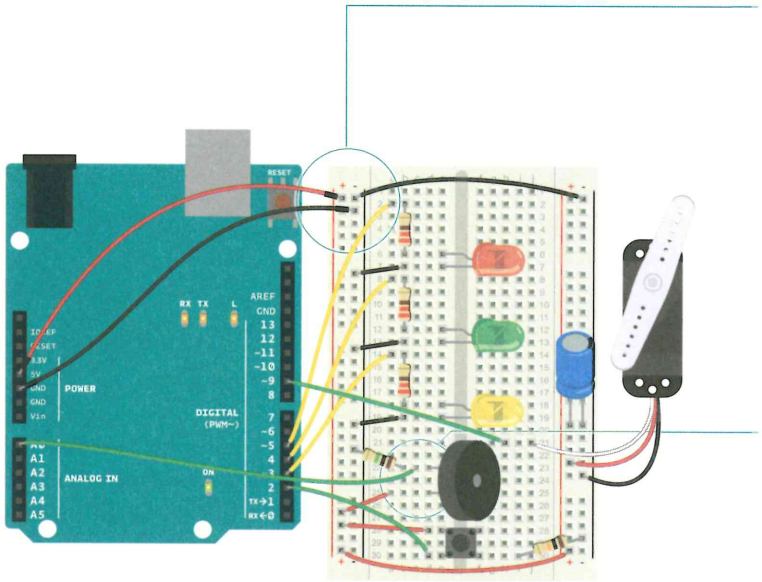


Fig. 1

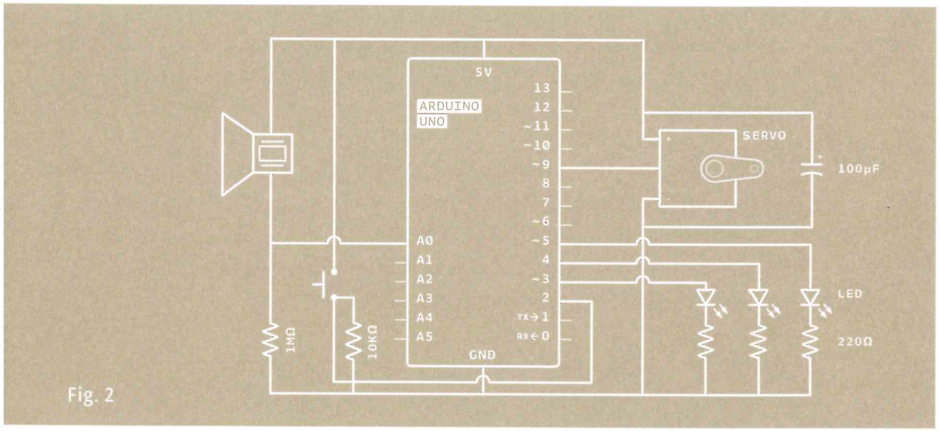


Fig. 2

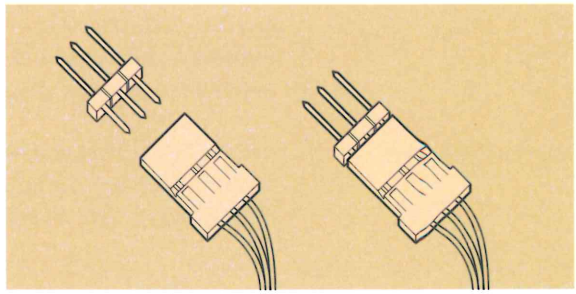
Ci sono molte connessioni sulla scheda, assicurati di tenere traccia di come sono collegate le cose.

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.

Costanti utili

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

Variabili per memorizzare i valori dell'interruttore e del piezo

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

Soglie di rilevamento

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

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.

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.

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.

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.

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

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

int knockVal;
int switchVal;

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

boolean locked = false;
int numberOfKnocks = 0;


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

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

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.

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`.

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.

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.

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`.

```
    if(switchVal == HIGH){
      locked = true;
      digitalWrite(greenLed,LOW);
      digitalWrite(redLed,HIGH);
      myServo.write(90);
      Serial.println("The box is locked!");
      delay (1000);
    }
  }

  if(locked == true){
    knockVal = analogRead(piezo);

    if(numberOfKnocks < 3 && knockVal > 0){
      if(checkForKnock(knockVal) == true){
        numberOfKnocks++;
      }
      Serial.print(3-numberOfKnocks);
      Serial.println("more knocks to go");
    }

    if(numberOfKnocks >= 3){
      locked = false;
      myServo.write(0);
      delay(20);
      digitalWrite(greenLed,HIGH);
      digitalWrite(redLed,LOW);
      Serial.println("The box is unlocked!");
    }
  }
}

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.

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.

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.

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.

Chiudi la funzione con un'altra parentesi.

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

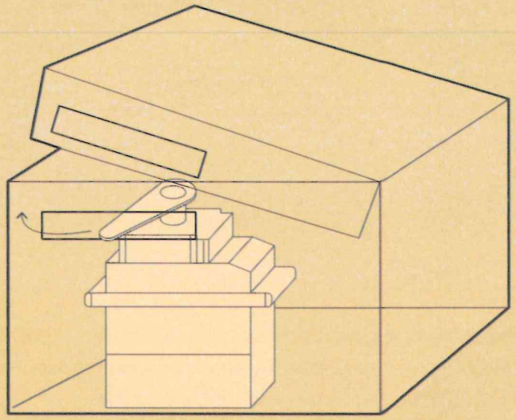
```
57  if(value > quietKnock && value < loudKnock){
58    digitalWrite(yellowLed, HIGH);
59    delay(50);
60    digitalWrite(yellowLed, LOW);
61    Serial.print("Valid knock of value ");
62    Serial.println(value);
63    return true;
64  }
65  else {
66    Serial.print("Bad knock value ");
67    Serial.println(value);
68    return false;
69  }
70 }
```

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.



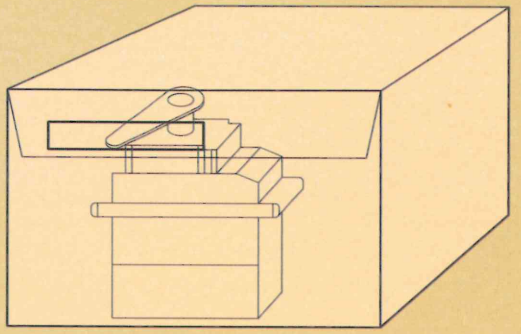
❶

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.

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.



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.

INGREDIENTI

LED

RESISTENZA DA 220 OHM

RESISTENZA DA 1 MEGA OHM

LAMINA DI METALLO

13