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

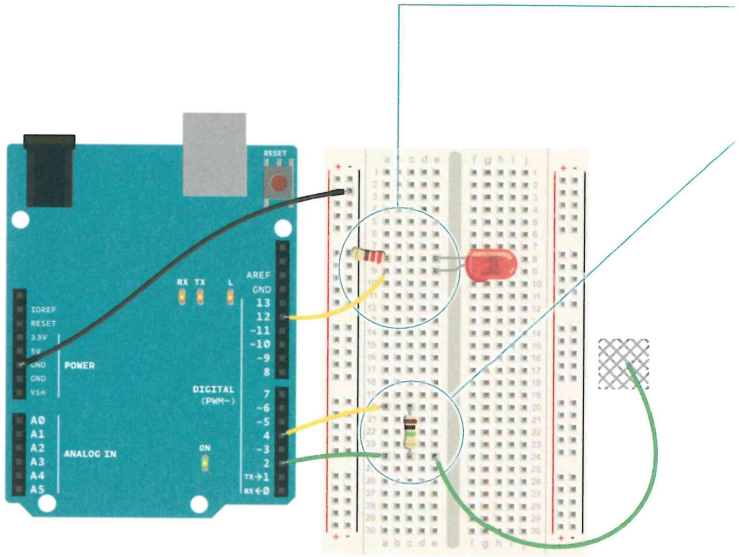


Fig. 1

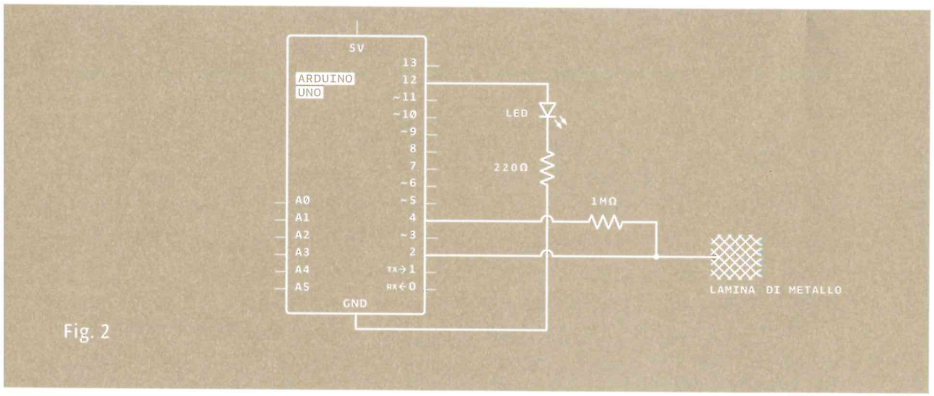


Fig. 2

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.

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.

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.

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

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.

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

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

int threshold = 1000;
const int ledPin = 12;

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

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

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

  delay(10);
}
```

USALA

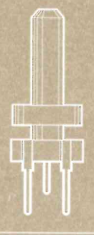
Dopo aver programmato Arduino, vorrai scoprire quali sono i valori del sensore quando viene toccato. Apri il monitor seriale e prendi nota del valore proveniente dal sensore quando non lo tocchi. Premi leggermente il filo che fuoriesce dalla breadboard. Il numero dovrebbe aumentare. Prova a premere con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

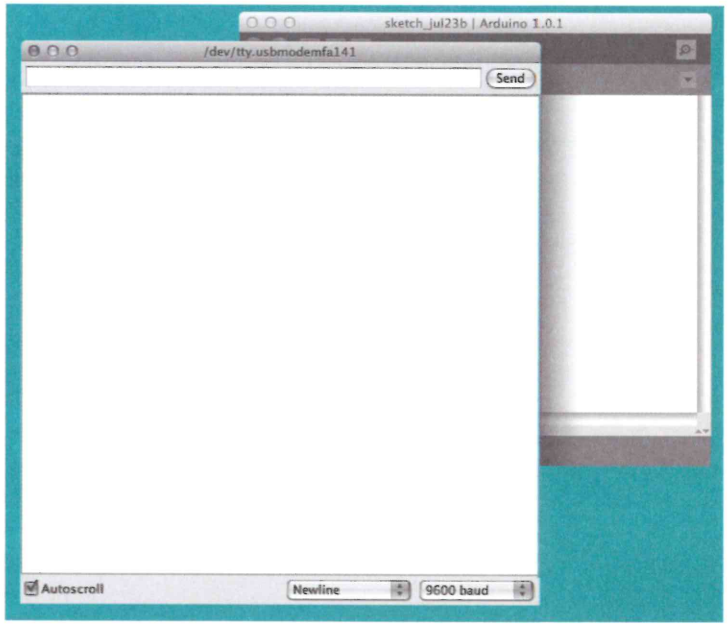


Fig. 1

COSTRUISCI IL CIRCUITO

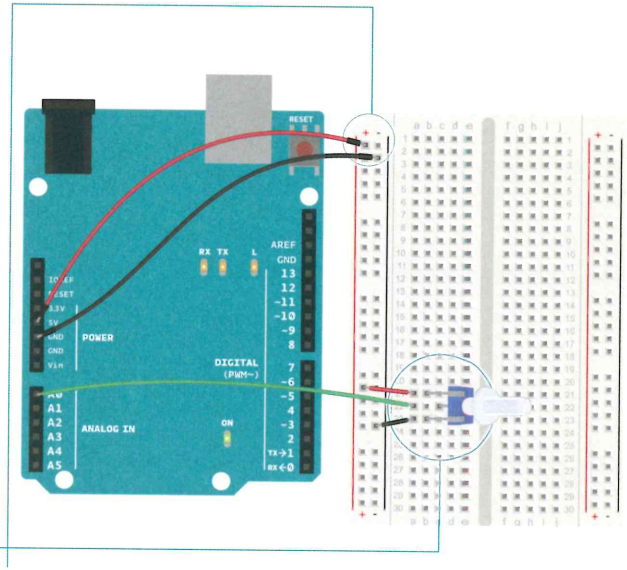


Fig. 2

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.

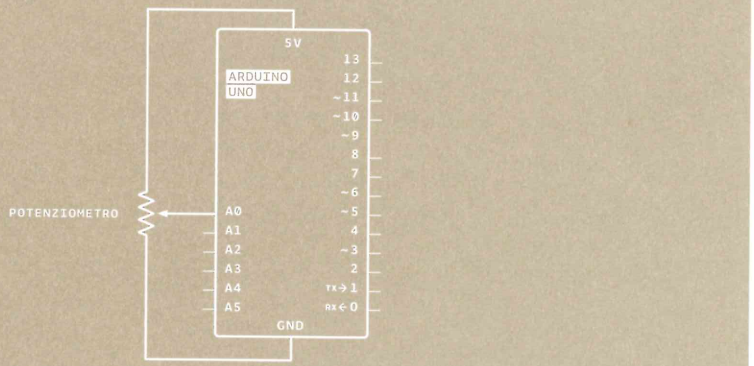


Fig. 3

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.

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.

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.

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.

Crea un oggetto per l'immagine

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

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

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

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

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

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

```
3 PImage logo;
```

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.

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.

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.

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.

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.

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.

```
4 int bgcolor = 0;

5 void setup() {

6   colorMode(HSB, 255);

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

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

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.

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.

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.

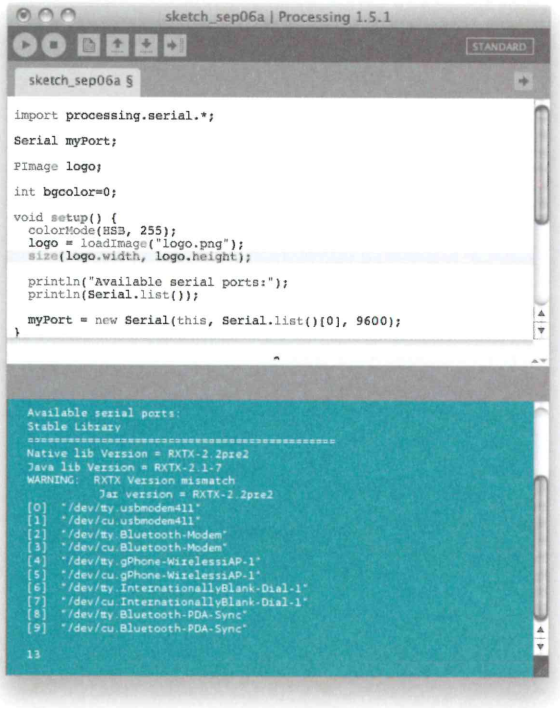
USALO

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.

```
void draw() {
  if (myPort.available() > 0) {
    bgcolor = myPort.read();
    println(bgcolor);
  }
  background(bgcolor, 255, 255);
image(logo, 0, 0);
}
```

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

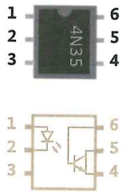
Scopri: il fotoaccoppiatore, la connessione con altri componenti

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

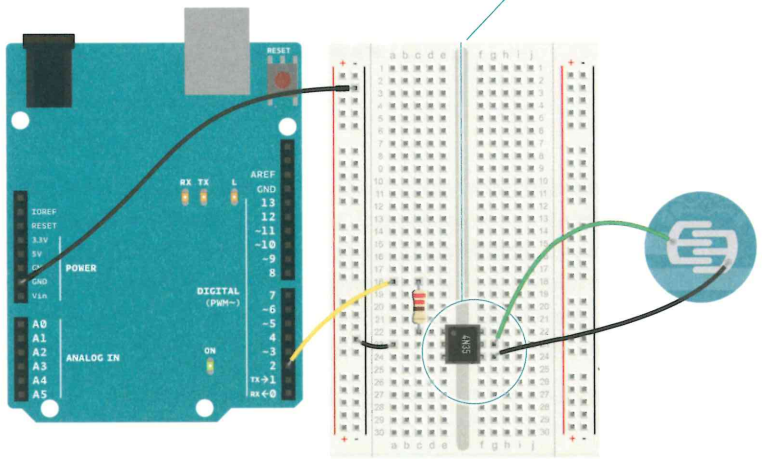


Fig. 1

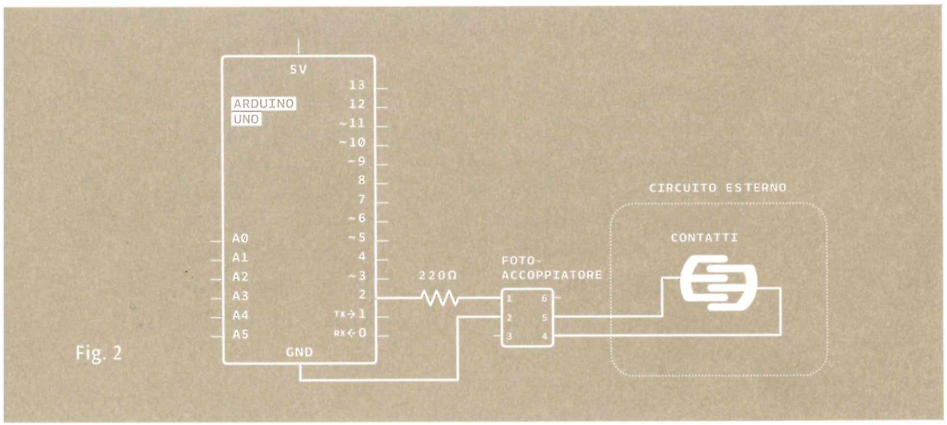


Fig. 2

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'interrutore 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.

Configura la direzione del piedino

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

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`.

Aspetta qualche secondo

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

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?

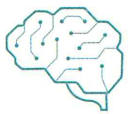
```
1 const int optoPin = 2;
```

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

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

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

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

Accelerometro -
Alimentazione -
Amperaggio (A o ampere) -
Analogico -
Anodo -
Array -
Argomento-
Attuatore -
Baud -
Binario -
Bit -
Booleano -
Buffer seriale -
Byte -
Calibrazione -
Capacità -
Carico -
Catodo -
Ciclo di lavoro -
Circuiti integrati (IC) -
Circuito -
Comunicazione seriale -
Condensatori di disaccoppiamento -
Conduttore -
Controtensione -
Convertitore analogico-digitale (Analog to Digital Converter ADC) -
Corrente -
Corrente alternata -
Corrente continua -
Cortocircuito -
Costante -
Datasheet -
Debugging -
Digitale -
Drain (scarico) -
Dual In-line Package (DIP) -
Elettricità -
Float (numero a virgola mobile) -
Fotoaccoppiatore -
Fotocellula -
Fotoresistenza -
Fototransistor -
Funzione -
Gate (porta) -
IDE -
Indice -
Induzione -
Int -
Interruttore -
Isolante -
Istanza -
LED a catodo comune -
Legge di Ohm -
Libreria -
Long -
Massa -
Microcontrollore -
Millesimo di secondo -
Modulazione di larghezza di impulso (Pulse Width Modulation PWM) -
Monitor seriale -
Oggetto -
Ohm -
Onda quadra -
Parallelo -
Parametro -
Partitore di tensione -
Periodo -
Polarizzato -
Processing -
Pseudocodice -
Resistenza -
Saldatura -
Sensore -
Serie -
Sketch -
Source (sorgente) -
Tensione -
Tipi di dato -
Transistor -
Trasduttore -
Unsigned -
USB -
Variabile -
Variabile globale -
Variabile locale -

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.

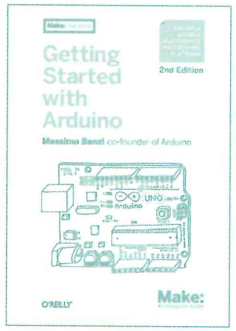
V

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.

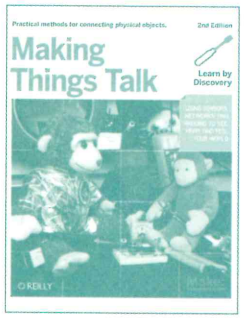
ALTRE LETTURE



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

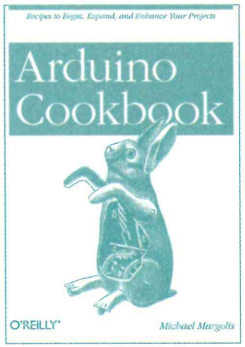
Getting Started with Processing di **Casey Reas** e **Ben Fry** [O'Reilly Media / Make, 2010]. Questa breve guida per l'ambiente di programmazione Processing spiega come programmare grafiche, suoni e contenuti multimediali.

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.

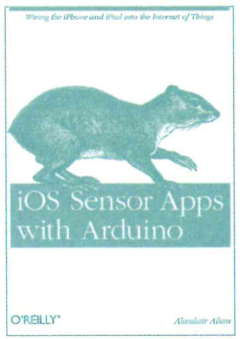
Getting Started with RFID di **Tom Igoe** [O'Reilly Media / Make, 2012]. Una breve introduzione all'utilizzo di identificazione a radiofrequenza con Arduino e Processing.



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.