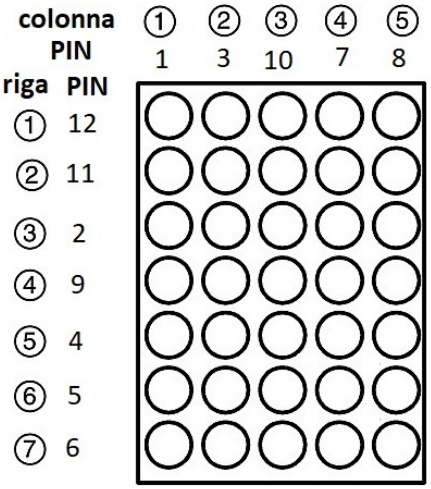
Programmazione con Arduino
dei display a LED a matrice di punti (display DOT)

Struttura dei display

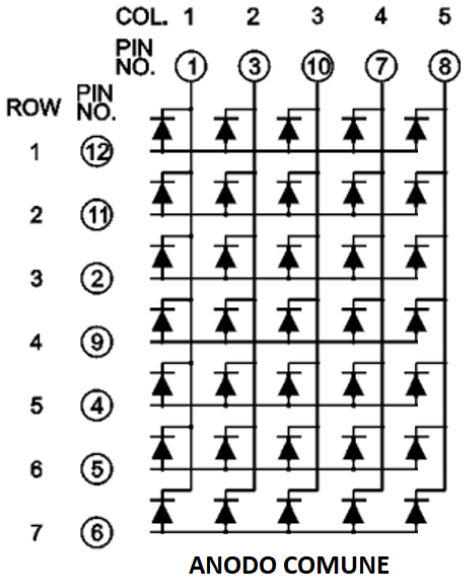
In figura viene riportata l'immagine di un display a LED a matrice di punti (display DOT) del tipo 5x7, ovvero caratterizzato dalla presenza di 5 colonne e 7 righe, con la relativa piedinatura; i LED del display sono in questo caso 35, 7 per ogni colonna.



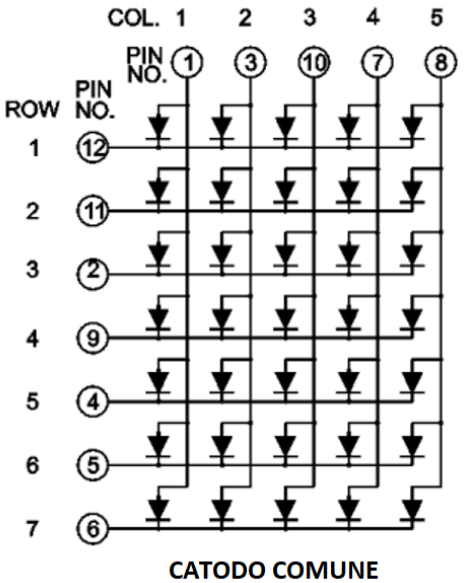
Come i display a 7 segmenti sono disponibili nelle versioni ad anodo comune e a catodo comune; gli schemi elettrici vengono mostrati in figura.

Per far accendere un LED appartenente ad un display ad anodo comune si devono collegare gli anodi a VCC; un LED si accende portando a GND l'anodo corrispondente.

Per far accendere un LED appartenente ad un display a catodo comune si devono collegare i catodi a GND; un LED si accende portando a VCC l'anodo corrispondente.



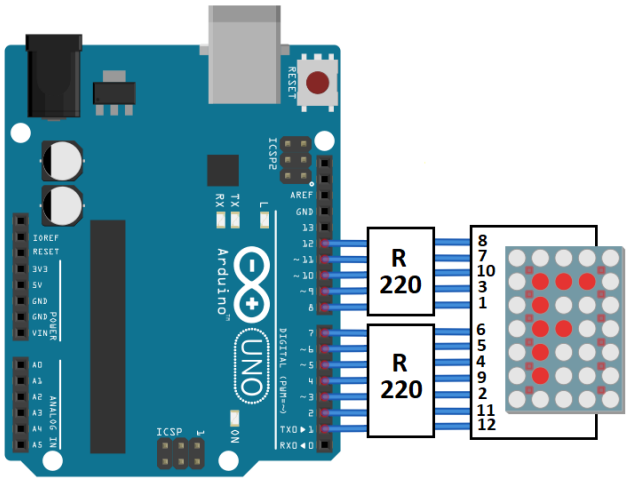
ANODO COMUNE



CATODO COMUNE

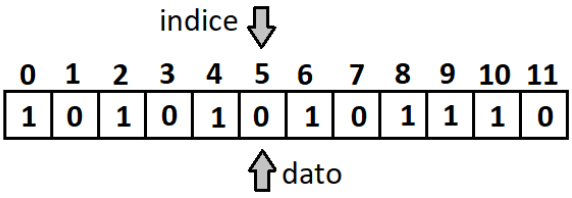
Schema circuitale

Nel seguito si suppone che i pin del display corrispondenti alle righe vengano collegati (tramite resistori di limitazione da 220 Ω ad esempio) ai pin da 1 a 7 della scheda Arduino, quelli corrispondenti alle colonne (sempre tramite resistori di limitazione) ai pin da 8 a 12.
Lo schema viene riportato in figura.

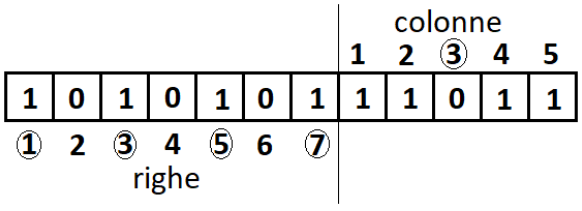


Programmazione

Per programmare questi display con Arduino utile ricorrere all'impiego degli *array* (strutture che contengono dati dello stesso tipo); gli elementi che appartengono ad un array si distinguono l'uno dall'altro utilizzando un indice.
Un array che deve ad esempio contenere 12 dati può essere rappresentato come in figura.



In questo caso si dice che la locazione di indice 0 contiene il dato 1, la locazione di indice 1 contiene il dato 0, la locazione di indice 2 il dato 1 e così via.
Se *N* è il numero delle locazioni, la prima locazione ha indice 0 mentre l'ultima *N*-1 (in questo caso quindi 11).
Nel caso del display 5x7 si può utilizzare un array con 12 locazioni; nelle prime 7 vengono selezionati, per ciascuna colonna, i LED che si devono accendere; nelle ultime 5 viene selezionata la colonna di appartenenza dei LED.
Si consideri ora come esempio un display a catodo comune.
Se si vogliono far accendere i LED appartenenti alle righe 1-3-5-7 della colonna 3 si deve definire quindi un array come quello riportato in figura.
La colonna viene selezionata con 0 in quanto il catodo corrispondente deve essere portato a GND; i LED da accendere vengono selezionati con 1 in quanto gli anodi corrispondenti devono essere portati a VCC.



In fase di programmazione un array come quello descritto (che è stato denominato in questo caso C3 per indicare l'accensione dei LED della terza colonna) deve essere dichiarato nel modo seguente.

```
int C3[12]={1,0,1,0,1,0,1,1,1,0,1,1};
```

E' evidente che all'array può anche essere assegnato un nome qualsiasi e che 12 corrisponde alla dimensione del medesimo.
Nell'esempio ora citato, se si ha a disposizione un display ad anodo comune, la dichiarazione dell'array si modifica nel modo seguente.

```
int C3[12]={0,1,0,1,0,1,0,0,0,1,0,0};
```

La colonna viene selezionata con 1 in quanto l'anodo corrispondente deve essere portato a VCC; i LED da accendere vengono selezionati con 0 in quanto i catodi corrispondenti devono essere portati a GND.
Se si utilizzano display a 5 colonne è evidente come sia necessario definire, per ciascun simbolo che si vuole visualizzare, i 5 array corrispondenti.
Se si utilizzano display a 10 colonne e 10 righe (10x10) si devono invece definire 10 array con dimensione 20 (le prime 10 locazioni riservate alle righe, le ultime 10 alle colonne).
Per ottenere un'immagine fissa del simbolo che si vuole visualizzare le colonne devono essere scansionate ad una ad una ad intervalli di 5 ms utilizzando un ciclo che va a leggere tutte le locazioni di ciascun array e, in relazione al contenuto delle medesime, porta a livello alto o basso i corrispondenti pin di Arduino.
Un esempio di codice è il seguente.

```
for(i=1;i<=12;i++){
digitalWrite(i,C1[i-1]);
}
delay(5);
```

Vengono di seguito proposti tre esempi di programmazione.

Esercitazione 1_Verifica di funzionalità di un display

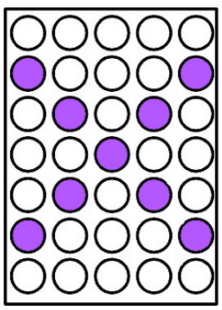
Scopo dell'esercitazione è di verificare la funzionalità di un display accendendone tutti i segmenti.
Il codice del programma, opportunamente commentato, viene di seguito riportato.

```
// esercitazione 1
// prova di funzionalità di un display DOT a catodo comune
int i;  // variabile contatore
// array LED tutti accesi
int C1[12]={1,1,1,1,1,1,1,0,1,1,1,1}; // array prima colonna
int C2[12]={1,1,1,1,1,1,1,1,0,1,1,1}; // array seconda colonna
int C3[12]={1,1,1,1,1,1,1,1,1,0,1,1}; // array terza colonna
int C4[12]={1,1,1,1,1,1,1,1,1,1,0,1}; // array quarta colonna
int C5[12]={1,1,1,1,1,1,1,1,1,1,1,0}; // array quinta colonna
```

```
void setup() {
for(i=1;i<=12;i++){
pinMode(i,OUTPUT);   // pin definiti come uscite
}
}
void loop() {
// colonna 1
for(i=1;i<=12;i++){
digitalWrite(i,C1[i-1]);
}
delay(5); // tempo di attesa
// colonna 2
for(i=1;i<=12;i++){
digitalWrite(i,C2[i-1]);
}
delay(5); // tempo di attesa
// colonna 3
for(i=1;i<=12;i++){
digitalWrite(i,C3[i-1]);
}
delay(5); // tempo di attesa
// colonna 4
for(i=1;i<=12;i++){
digitalWrite(i,C4[i-1]);
}
delay(5); // tempo di attesa
// colonna 5
for(i=1;i<=12;i++){
digitalWrite(i,C5[i-1]);
}
delay(5); // tempo di attesa
}
```

Esercitazione 2_ Visualizzazione del simbolo X

Scopo dell'esercitazione è la visualizzazione del simbolo X.
I LED che si devono accendere vengono visualizzati in figura.

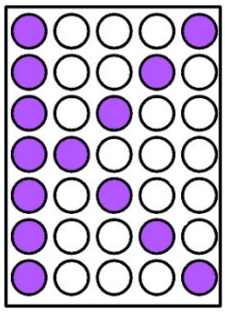


Il codice del programma, opportunamente commentato, viene di seguito riportato.

```
// esercitazione 2
// display DOT catodo comune
// visualizzazione del simbolo  X
int i;  // variabile contatore
// // array simbolo X
int C1[12]={0,1,0,0,0,1,0,0,1,1,1,1}; // array prima colonna
int C2[12]={0,0,1,0,1,0,0,1,0,1,1,1}; // array seconda colonna
int C3[12]={0,0,0,1,0,0,0,1,1,0,1,1}; // array  terza colonna
int C4[12]={0,0,1,0,1,0,0,1,1,1,0,1}; //  array quarta colonna
int C5[12]={0,1,0,0,0,1,0,1,1,1,1,0}; // array quinta colonna
void setup() {
for(i=1;i<=12;i++){
pinMode(i,OUTPUT);   // pin definiti come uscite
}
}
void loop() {
// colonna 1
for(i=1;i<=12;i++){
digitalWrite(i,C1[i-1]);
}
delay(5); // tempo di attesa
// colonna 2
for(i=1;i<=12;i++){
digitalWrite(i,C2[i-1]);
}
delay(5); // tempo di attesa
// colonna 3
for(i=1;i<=12;i++){
digitalWrite(i,C3[i-1]);
}
delay(5); // tempo di attesa
// colonna 4
for(i=1;i<=12;i++){
digitalWrite(i,C4[i-1]);
}
delay(5); // tempo di attesa
// colonna 5
for(i=1;i<=12;i++){
digitalWrite(i,C5[i-1]);
}
delay(5); // tempo di attesa
}
```

Esercitazione 3_ Visualizzazione a intermittenza del simbolo K

Scopo dell'esercitazione è la visualizzazione del simbolo K alternata allo spegnimento del display.
I LED che si devono accendere vengono visualizzati in figura.



Il codice del programma, opportunamente commentato, viene di seguito riportato.
Il programma principale è costituito dalla sequenza di due istruzioni che richiamano i due sottoprogrammi che consentono la visualizzazione del simbolo K alternata allo spegnimento del display.
Sono previste due variabili contatore.
Il valore finale assegnato alla variabile k (in questo caso 35) determina in pratica il tempo di persistenza sul display del simbolo K e il tempo in cui il medesimo rimane spento.

```
// esercitazione 3
// display DOT catodo comune
// visualizzazione a intermittenza del simbolo K
int i,k;  // variabili contatore
// array simbolo K
int C1[12]={1,1,1,1,1,1,1,0,1,1,1,1}; // array prima colonna
int C2[12]={0,0,0,1,0,0,0,1,0,1,1,1}; // array seconda colonna
int C3[12]={0,0,1,0,1,0,0,1,1,0,1,1}; // array terza colonna
int C4[12]={0,1,0,0,0,1,0,1,1,1,0,1}; // array quarta colonna
int C5[12]={1,0,0,0,0,0,1,1,1,1,1,0}; // array quinta colonna
// array display spento
int C1s[12]={0,0,0,0,0,0,0,0,1,1,1,1}; // array prima colonna
int C2s[12]={0,0,0,0,0,0,0,1,0,1,1,1}; // array seconda colonna
int C3s[12]={0,0,0,0,0,0,0,1,1,0,1,1}; // array terza colonna
int C4s[12]={0,0,0,0,0,0,0,1,1,1,0,1}; // array quarta colonna
int C5s[12]={0,0,0,0,0,0,0,1,1,1,1,0}; // array quinta colonna
void setup() {
for(i=1;i<=12;i++){
pinMode(i,OUTPUT);    // pin definiti come uscite
}
}
void loop() {
simboloK();  // sottoprogramma per la visualizzazione del simbolo K
spento();  // sottoprogramma per lo spegnimento del display
}
void simboloK(){
for(k=1;k<=35;k++){   // determina il tempo di persistenza del simbolo sul display
// colonna 1
for(i=1;i<=12;i++){
digitalWrite(i,C1[i-1]);
}
delay(5); // tempo di attesa
// colonna 2
```

```
for(i=1;i<=12;i++){
digitalWrite(i,C2[i-1]);
}
delay(5); // tempo di attesa
// colonna 3
for(i=1;i<=12;i++){
digitalWrite(i,C3[i-1]);
}
delay(5); // tempo di attesa
// colonna 4
for(i=1;i<=12;i++){
digitalWrite(i,C4[i-1]);
}
delay(5); // tempo di attesa
// colonna 5
for(i=1;i<=12;i++){
digitalWrite(i,C5[i-1]);
}
delay(5); // tempo di attesa
}
}
void spento(){
for(k=1;k<=35;k++){    )) // determina il tempo in cui il display rimane spento
// colonna 1
for(i=1;i<=12;i++){
digitalWrite(i,C1s[i-1]);
}
delay(5); // tempo di attesa
// colonna 2
for(i=1;i<=12;i++){
digitalWrite(i,C2s[i-1]);
}
delay(5); // tempo di attesa
// colonna 3
for(i=1;i<=12;i++){
digitalWrite(i,C3s[i-1]);
}
delay(5); // tempo di attesa
// colonna 4
for(i=1;i<=12;i++){
digitalWrite(i,C4s[i-1]);
}
delay(5); // tempo di attesa
// colonna 5
for(i=1;i<=12;i++){
digitalWrite(i,C5s[i-1]);
}
delay(5); // tempo di attesa
}
}
```