UNITÀ 6: SCHERMI LCD (DISPLAY A CRITALLI LIQUIDI)

OBIETTIVI

Come potete immaginare c'è un'ampia gamma di periferiche di input e output digitali. Fino ad ora abbiamo usato solo quelle più comuni, semplici pulsanti, come dispositivi per l'impostazione dei livelli logici "1" e "0" e LED e diodi per la loro rappresentazione. Ma ora è il momento di parlare di altre periferiche digitali più importanti e meglio conosciute. In questa unità, quindi, andiamo a dare un'occhiata allo schermo LCD come una periferica di output: consente di visualizzare qualsiasi tipo di informazione di output, inclusi numeri, lettere e simboli.

SEZIONE TEORICA

- SCHERMI LCD
- CARATTERISTICHE
- CARATTERISTICHE GRAFICHE
- LA LIBRERIA DI LiquidCrystal.h
- LA MEMORIA DEI DATI DI EEPROM

SEZIONE PRATICA

- COLLEGAMENTO AGLI SCHERMI LCD
- ESEMPIO 1: Hello world!
- ESEMPIO 2: Lo schermo
- ESEMPIO 3: Il cursore
- ESEMPIO 4: Lampeggio
- ESEMPIO 5: Direzione del testo
- ESEMPIO 6: Scorrimento
- ESEMPIO 7: Autoscorrimento
- ESEMPIO 8: Caratteri personalizzati
- ESEMPIO 9: Display
- ESEMPIO 10: Visualizzazione numeri interi
- ESEMPIO 11: Visualizzazione numeri virgola mobile
- ESEMPIO 12: Menu

- ESEMPIO 13: Scelta delle opzioni
- ESEMPIO 14: Un ultimo miglioramento

MATERIALI PRATICI

- Un Laptop o desktop computer
- Arduino IDE; dovrebbe includere materiale supplementare già installato e configurato
- Una scheda di controllo Arduino UNO
- un cavo USB

SOMMARIO

SEZIONE TEORICA

1. SCHERMI LCD

Questa è una periferica di output che non solo mostra numeri ma anche tutti i tipi di caratteri, testi, simboli e anche elementi di grafica semplici. Ha migliaia di usi e devi averlo visto tonnellate di volte. Li vedrai con diversi numeri di righe e caratteri per linea. Ci sono anche quelli con differenti retroilluminazioni a colori e caratteri di diversi colori e dimensioni. Tutti hanno il proprio controller che gestisce tutte le operazioni interne. La maggior parte degli schermi LCD è compatibile con la popolare Hitachi HD44780. Ecco perché non fa molta differenza rispetto a quale schermo si utilizzi: 2 righe da 16 caratteri l'una o 4 righe da 20 caratteri. Stiamo usando uno schermo LCD da 2 x 16. C'è una foto nella figura con una rappresentazione semplificata e uno schema elettrico. (Figura 1).

Figura 1

Uno schermo LCD è una periferica digitale. I suoi segnali possono essere collegati direttamente ai pin di input e output del controller. Sono divisi in tre gruppi:

- **Alimentazione**: i pin di input e output forniscono la tensione necessaria allo schermo e sono di solito + 5V GND e VCC. C'è anche un terzo pin, un VLC; fornisce una tensione alternata tra 0 V e +5 V che può essere utilizzata per regolare il contrasto dello schermo.
- **Controllo**: questi sono i segnali che decidono se la schermata riceva un comando o alcuni dati, se verrà letto o scritto dal controller o se la schermata è abilitata o meno.
- **Dati**: Il controller utilizza questi segnali per inviare comandi e dati appropriati allo schermo.

Il pin nº 1 è il primo a sinistra, diamo un'occhiata a dove è posizionato il pin 1. (Figura 2)

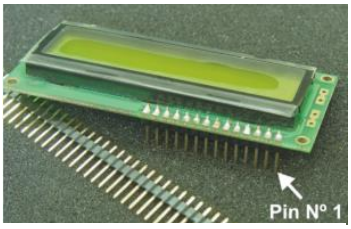

Figura 2

Diamo un'occhiata alla seguente tabella che contiene una descrizione di ogni pin.

PIN N°	NOME	TIPO	DESCRIZIONE
1	**Vss**	Tensione fornita	Tensione fornita terra (0 V)
2	**Vdd**	Tensione fornita	Tensione fornita positive +5 Vcc
3	**VLC**	Tensione fornita	Aggiustamento del contrasto: voltaggio alternante tra 0 and +5 Vcc.
4	**RS**	Input	Output dal controller Arduino. Scegli tra istruzioni e dati: RS=0 Arduino trasferisce istruzioni RS=1 Arduino trasferisce dati (codici ASCII)
5	**R/W**	Input	Output da Arduino. Legge e scrive controlli: R/W=0 Arduino esegue data write sullo schermo LCD R/W=1 Arduino esegue data read sullo schermo LCD
6	**E**	Input	Output da Arduino. Abilita lo schermo: E=0 schermo LCD disabilitato (ad alta impedenza) E=1 schermo LCD abilitato
7-14	**DB0:DB7**	Input/Output	Line bus di dati e istruzioni. Arduino trasferisce istruzioni o dati agli schermi secondo segnale RS. Le linee DB0:DB7 sono usate con una interfaccia a 8 bit Le linee DB4:DB7 sono usate con una interfaccia a 4 bit
15	**L+**	Tensione fornita	Tensione positive per luce di sfondo (+5 Vcc)
16	**L-**	Tensione fornita	Tensione negative per luce di sfondo (0 V)

Non intendiamo utilizzare i numeri di pin 15 e 16: sono facoltativi. È possibile che non appaiano nemmeno sullo schermo; solo quelli con luce di fondo li usano. Gli schermi sono tra le periferiche più potenti e versatili. Come abbiamo già detto, sono in grado di visualizzare tutti i tipi di messaggi composti da testo, numeri o simboli, nonché fornire una varietà di effetti di visualizzazione come i movimenti a sinistra o a destra, il tremolio, lo scorrimento, ecc. Uno schermo LCD ha un controller separato.

Lo schermo LCD che si sta usando e che è incluso nel kit dei materiali suggeriti appare nella figura precedente. Questo schermo comprende due righe da 16 caratteri ciascuna (2 x 16). E' utile saldare una striscia di 14/16 pin maschi; questo renderà più facile inserirli nella scheda del modulo di pratica più tardi. Sarete in grado di rendere i collegamenti con il controller molto più rapidamente.

2. LE CARATTERISTICHE

La comunicazione tra Arduino e gli schermi LCD è realizzata sostanzialmente tramite i pin digitali DB0-DB7. Puoi usare otto pin o, come nel nostro caso, solo quattro. Questo è chiamato “lavorare con un'interfaccia a 4 bit”. Arduino invia comandi o istruzioni allo schermo attraverso i pin. Lo schermo può quindi eseguire una serie di effetti di visualizzazione: scorrimento, sfarfallio, eliminazione, posizionamento del cursore, ecc… Invia anche i codici dei caratteri ASCII che si desidera visualizzare. Questi sono codici a 8 bit. Se si utilizza un'interfaccia a 8 bit, Arduino necessita solo di un singolo trasferimento per visualizzare ciascun carattere; un'interfaccia a 4 bit richiede due trasferimenti per visualizzare ciascun carattere. È un po' più lento, ma si usano meno cavi di connessione. La figura che segue mostra un esempio del set di caratteri che visualizzano gli schermi LCD; questo set è stabilito dal produttore. La memoria ROM interna contiene la definizione di ciascuno dei caratteri e può variare tra diversi modelli e versioni.

I quattro bit più leggeri, B3:B0, sono rappresentati nelle righe a sinistra. I quattro più pesanti, B7:B4, sono rappresentati nella parte superiore della tabella nelle colonne in codice binario. Tutto quello che dovete fare per codificare un carattere è selezionarlo e quindi trovare quale colonna o riga è inserita. Ecco un esempio: il carattere 'F' si trova nella colonna 4 (0100) e nella riga 6 (0110). Il suo codice binario è quindi 0100 0110 (0x46) che corrisponde esattamente al codice ASCII del carattere "F".

Abbiamo già detto che non tutte le schermate hanno lo stesso set di caratteri. Dipende dal produttore, dal modello, dalla versione ecc. Tuttavia, i codici corrispondenti ai caratteri ASCII standard sono comuni a tutte le schermate. Questi sono quelli delle colonne 2 (0010) a 7 (0111).

Figura 3

3. CARATTERI GRAFICI

È possibile creare un massimo di otto caratteri grafici di 5 x 8 punti o "pixel". Ogni carattere è numerato da 0 a 7 e necessita di un totale di 8 byte da definire. Lo schermo LCD dispone di una memoria RAM interna denominata CGRAM per eseguire questa operazione. Una volta definiti, è possibile visualizzarli inviando il numero corrispondente al carattere (tra 0 e 7). Non dimenticate: i caratteri grafici sono memorizzati nella memoria RAM. Se si scollega il sistema si perdono e Arduino deve definirli di nuovo.

I caratteri grafici sono definiti inserendo i byte in posizioni successive della memoria CGRAM; i pattern binari dei byte definiscono ogni carattere. Il CGRAM è una memoria volatile in grado di memorizzare complessivamente 64 byte. Un carattere di 5 x 8 punti richiede 8 byte per definirlo. Quindi è possibile definire fino a otto caratteri diversi (8 x 8) in qualsiasi modo.

La seguente tabella contiene quattro caratteri definiti da 5 x 8, ad esempio. Sono inseriti nelle prime 32 posizioni del CGRAM. Il primo va nelle posizioni da 0 a 7, il secondo nelle posizioni da 8 a 15 e così via.

Ogni bit di ciascuno dei byte che hanno un valore di "1" attiva il relativo punto o pixel sullo schermo LCD. Viene visualizzato il primo carattere grafico del CGRAM inviando il numero 0 come se fosse un codice ASCII. Il secondo carattere viene visualizzato inviando il numero 1 e così via con tutti quelli che hai definito.

Tutto quello che devi fare per definirli è creare un array o una matrice per ogni carattere. Quindi, il contenuto di ogni array viene copiato sullo schermo CGRAM utilizzando la funzione **crateChar()**. Devi creare lo stesso numero di array e di caratteri; ce ne sono quattro in questo esempio:

1. byte heart[8] = {10,21,17,17,17,10,4,0};
2. byte smile[8] = {B0,B01010,B0,B00100,B00000,B10001,B01110,B0};
3. byte letter_ñ [8] = {31,0,24,27,17,17,17,0};
4. byte letter_ú [8] = {2,4,17,17,17,19,13,0};

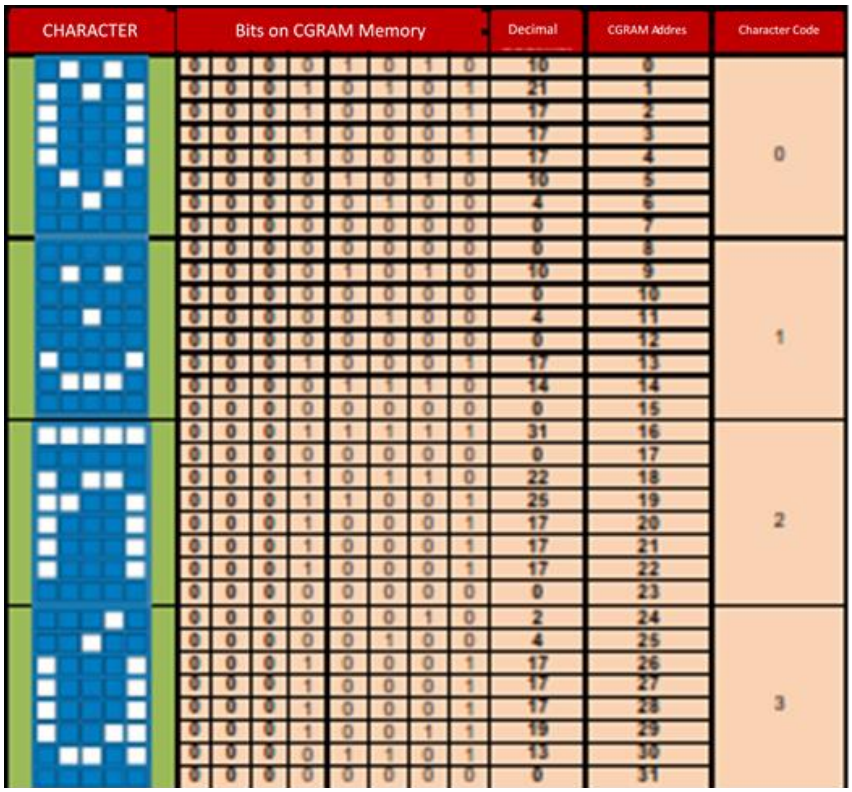

Figura 4

4. LA LIBRERIA LIQUIDCRYSTAL.H

Avrai familiarità con il concetto di "libreria" e con le funzioni che contiene.

Arduino ha creato un gran numero di biblioteche, ma parliamo di "**LiquidCrystal.h**". Contiene un gran numero di funzioni progettate per controllare e utilizzare schermi LCD basati sul controller Hitachi HD44780 o compatibili con esso.

Stiamo studiando le funzioni più rappresentative e importanti. In ogni caso, ti consigliamo di visitare www.arduino.cc dove troverai informazioni e esempi.

- **Funzione: LiquidCrystal()**

Questa funzione crea una variabile di tipo "LiquidCrystal" e stabilisce le connessioni tra lo schermo LCD e il controller Arduino; è possibile utilizzare un'interfaccia a 8 o 4 bit. In questo caso, non utilizzare le linee D0-D3. È anche possibile decidere se utilizzare il segnale di schermo R / W. Se non lo usi, come in questo caso, collegare il segnale a GND.

Sintassi:

LiquidCrystal var(RS,E,D4,D5,D6,D7); // Per un'interfaccia a 4 bit senza segnale R / W *LiquidCrystal var(RS,RW,E,D4,D5,D6,D7);* // Per un'interfaccia a 4 bit con segnale R / W *LiquidCrystal var(RS,E,D0,D1,D2,D3,D4,D5,D6,D7);* // Per un'interfaccia a 8 bit senza segnale R / W

LiquidCrystal var(RS,RW,E,D0,D1,D2,D3,D4,D5,D6,D7); // Per un'interfaccia a 8 bit con segnale R / W

var: il nome della variabile assegnata allo schermo LCD che stai per controllare.

RS: il pin Arduino collegato al RS segnale dello schermo.

RW: il pin Arduino collegato al segnale R / W dello schermo (se verrà utilizzato).

E: il perno Arduino collegato al segnale dello schermo E.

D0-D7: pin Arduino collegati alle linee di dati DB0-DB7 dello schermo. Se per DB0-DB3 non sono indicati pin, si assume un'interfaccia a 4 bit e si utilizzano solo i segnali DB4-DB7.

Esempio:

LiquidCrystal lcd(7,8,9,10,11,12); // stabilisce la connessione di uno schermo chiamato

// “LCD”. I pin Arduino da D7 a D12 sono collegati allo

// schermo RS, E, DB4, DB5, DB6 e DB7.

// Il segnale R / W deve essere collegato a GND.

- **Funzione: Begin()**

Questa funzione avvia lo schermo LCD e assegna il numero di righe e il numero di caratteri per riga secondo il modello in questione. In questo caso si utilizza uno schermo di 16 x 2 caratteri.

Sintassi:
var.begin(c,f);

var: questo è il nome che definisce lo schermo in questione (stabilito in LyquidCrystal()). c: il numero di colonne.

f: il numero di righe.

Esempio:

LiquidCrystal lcd(7,8,9,10,11,12); // Collegamenti allo schermo LCD

// si tratta di uno schermo "lcd" da 16 x 2

- **Funzione: setCursor()**

Questa funzione posiziona il cursore dello schermo LCD come desiderato. Da allora vengono visualizzati i caratteri precedenti.

Sintassi:
var.setCursor(c, f);

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal ()). c: il numero di colonne (a partire da 0).

f: il numero di righe (a partire da 0).

Esempio:

LiquidCrystal lcd(7,8,9,10,11,12); // Collegamenti allo schermo LCD

lcd.setCursor(3,0); //Posiziona il cursore nella posizione 3 (4° //carattere) di riga //0 (il 1°)

- **Funzione: home()**

Questa funzione individua il cursore nell'angolo superiore sinistro (posizione 0 della riga 0) della prima posizione dello schermo. Non elimina ciò che è stato precedentemente visualizzato.

Sintassi:
var.home();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

- **Funzione: clear()**

Questa funzione cancella lo schermo LCD e individua il cursore nell'angolo superiore sinistro (posizione 0 della riga 0).

Sintassi:
var.clear();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

Esempio:

LiquidCrystal lcd(7,8,9,10,11,12); *//Collegamenti allo schermo LCD*

lcd.clear(); *//Pulisce lo schermo*

- **Funzione: write()**

Questa funzione scrive un carattere nella posizione corrente del cursore.

Sintassi:

var.write(char);

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()). *char*: il carattere da visualizzare

Esempio:

lcd.write('A'); *//It scrive 'A'*

- **Funzione: print()**

Questa funzione stampa sullo schermo LCD a partire dalla posizione corrente del cursore.

Sintassi:

var.print(data);

var.print(data,base);

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

data: sono i dati da stampare. Questo potrebbe essere char, int, long, float o string.

base: è opzionale e mostra la base numerica desiderata: BIN = binario; DEC = Decimale (per impostazione predefinita); OCT = ottale; HEX = esadecimale; o N = nº in decimali per numeri a virgola mobile (2 per impostazione predefinita).

Esempi:

int A=19;

float PI=3.1416;

lcd.print(A,HEX); // stampa A in esadecimale (1910 = 1316)

lcd.print("Hello"); *//Stampa "Hello"*

lcd.print(PI*2,4); // Stampa 6.2832 (quattro decimali)

- **Funzione: cursor()**

Questa funzione visualizza il cursore sullo schermo LCD nella sua posizione attuale come sottolineatura (_). Questo è dove comincerà a scrivere il carattere successivo.

Sintassi:

var.cursor();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

- **Funzione: noCursor()**

Questa funzione nasconde il cursore LCD.

Sintassi:

var.noCursor();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

- **Funzione: blink()**

Questa funzione visualizza il cursore LCD sullo schermo nella sua posizione corrente come un simbolo intermittente solido (▓). Questo è dove comincerà a scrivere il carattere successivo.

Syntax:

var.blink();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

- **Funzione: noBlink()**

Questa funzione nasconde il cursore intermittente solido (▓).

Sintassi:

var.noBlink();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

- **Funzione: noDisplay()**

This Questa funzione consente di spegnere lo schermo LCD senza perdere qualunque contenuto possa essere presente o la posizione del cursore.

Sintassi:

var.noDisplay();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

- **Funzione: display()**

Questa funzione collega lo schermo LCD e recupera il contenuto visualizzato su di esso prima di eseguire noDisplay().

Sintassi:

var.display ();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

- **Funzione: scrollDisplayLeft()**

Questa funzione sposta il contenuto (il testo e la posizione del cursore) visualizzati sullo schermo in un dato momento in un punto a sinistra.

Sintassi:

var.scrollDisplayLeft();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

- **Funzione: scrollDisplayRight()**

Questa funzione sposta il contenuto (il testo e la posizione del cursore) visualizzati sullo schermo in un dato momento in un punto a destra.

Sintassi:

var.scrollDisplayRight();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

- **Funzione: leftToRight()**

Questa funzione stabilisce automaticamente in quale direzione il cursore scrive sullo schermo: da sinistra a destra. Ciò significa che i caratteri sono scritti da sinistra a destra senza influenzare quelli già scritti.

Sintassi:

var.LeftToRight();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

- **Funzione: rightToLeft()**

Questa funzione inverte la direzione che il cursore scrive sullo schermo: il diritto a sinistra. Ciò significa che i caratteri sono scritti da destra a sinistra senza influenzare quelli già scritti..

Sintassi:

var.RightToLeft();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

- **Funzione: autoscroll()**

Questa funzione attiva il movimento di scorrimento o di visualizzazione automatica. Ogni volta che un carattere viene inviato allo schermo questa funzione lo visualizza e poi sposta il resto del contenuto di un posto. Se la direzione è da sinistra a destra (LeftToRight()), il contenuto si sposta verso sinistra. Se la direzione al momento è da destra a sinistra (RightToLeft()), il contenuto si sposta verso destra.

Sintassi:

var.autoscroll();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

- **Funzione: noAutoscroll()**

Questa funzione disattiva il movimento di scorrimento o di visualizzazione automatica.

Sintassi:

var.noAutoscroll();

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

- **Funzione: createChar()**

Questa funzione crea un carattere definito dall'utente. È in grado di creare un totale di otto caratteri da 5 a 8 pixel numerati da 0 a 7. Una matrice di 8 byte, una per riga, determina l'aspetto o il design di un personaggio. I cinque bit più leggeri di ogni byte corrispondono ai 5 pixel che compongono ogni riga del carattere. Una volta che questa funzione ha creato un carattere, l'utente può visualizzarlo sullo schermo semplicemente indicando il suo numero.

Sintassi:

var.createChar(n,dato);

var: Questo è il nome che definisce lo schermo in questione (stabilito come LyquidCrystal()).

n: rappresenta il numero del carattere da definire (da 0 a 7).

data: questo è il nome della matrice che contiene i byte che definiscono il carattere personalizzato.

Esempio:

bytearrowhead[8]={B00100,B01110,B10101,B00100,B00100,B00100, B00100, B00100}; //Crea la matrice "punta di freccia" di 8 byte che definisce il nuovo carattere grafico

lcd.createChar(0, arrowhead); //Crea il carattere nº 0 dal contenuto definito nello schema "punta di freccia"

lcd.write(byte(0)); //visualizza il carattere grafico nº 0 (punta di freccia)

5. LA MEMORIA DATI DI EPROM

I controller che compongono le varie tavole Arduino fanno parte di una memoria speciale: la memoria EEPROM. Essa consente di leggere o scrivere dati a 8 bit tra 0 e 255. Questa memoria ha una caratteristica speciale: conserva tutte le informazioni che hai salvato anche se l'alimentazione si spegne; non le cancella.

Ecco perché è l'ideale per memorizzare informazioni che possono essere modificate, come codici di accesso, parametri di configurazione, numeri di telefono, password, ecc. Come già sappiamo, la scheda NANO Arduino include un controller di modello ATmega328 che dispone di un 1KB (1024 byte) Memoria EEPROM.

Esiste anche un accesso a una libreria: "EEPROM.h". Ha solo due funzioni e li puoi usare per leggere e scrivere dati su questa memoria. Si può includere questa libreria nei programmi usando **#include<EEPROM.h>.**

- **Funzione:read()**

Questa funzione legge i dati a 8 bit che si trovano nella memoria EEPROM.

Sintassi:

EEPROM.read(dir);

dir: Questo determina la direzione in cui si legge EEPROM. Ha 1024 posizioni (KB) quindi la gamma possibile è da 0 a 1023.

Esempio:

```
#include EEPROM.h          //Include la libreria
byte value;                // variabile a 8 bit
value = EEPROM.read(279);  //Legge il byte in posizione 279
```

- **Funzione:write()**

Questa funzione scrive un valore a 8 bit (tra 0 e 255) in qualsiasi delle posizioni disponibili nella EEPROM.

Sintassi:

EEPROM.write(dir, value);

dir: Ciò determina in quale direzione si scriva EEPROM. Ha 1024 posizioni (KB) e la gamma possibile è da 0 a 1023

value: Questo è il valore a 8 bit (tra 0 e 255) che si desidera scrivere nella direzione indicata.

Esempio:

```
EEPROM.write(982, 33);          //Scrive il valore 33 in posizione 982
```

IMPORTANTE: La memoria EEPROM ha una capacità stimata di 100.000 cicli di scrittura/cancellazione. Una scrittura EEPROM può richiedere fino a 3,3 mS per essere completa, quindi è necessario essere attenti a quanto si scrive; se si supera il limite potrebbe essere disabilitato.

SEZIONE PRATICA

6. COLLEGARE LO SCHERMO LCD

State per iniziare la sezione pratica per collegare lo schermo LCD al controller NANO Arduino sulla scheda del modulo. Guardate attentamente le connessioni e assicuratevi di assemblare tutto correttamente. Utilizzerete lo schermo LCD in quasi tutti gli esempi delle unità rimanenti di questo corso. Ecco lo schema elettrico:

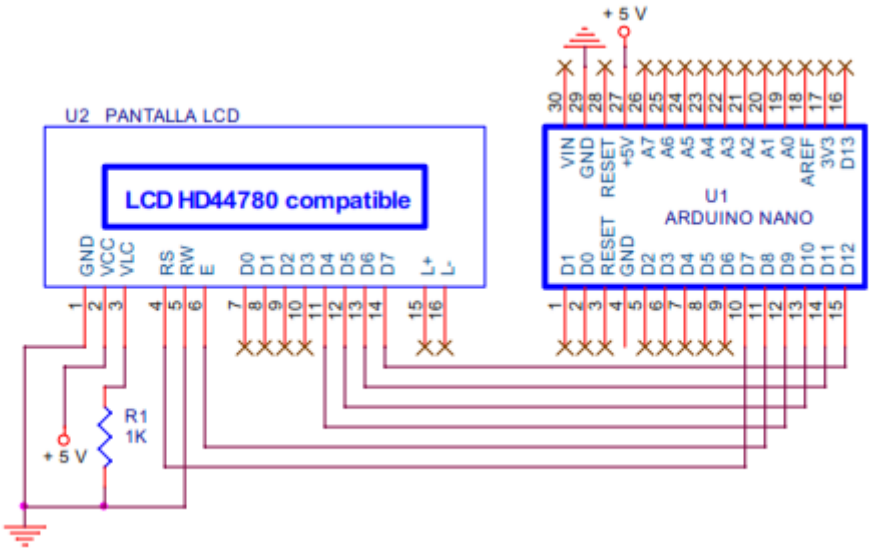

Figura 5

I pin D9:D12 Arduino sono collegati ai segnali DB4:DB7 dallo schermo. Monitorato dal vostro programma Arduino invierà le istruzioni e i caratteri da visualizzare. D7 e D8 controllano rispettivamente i segnali RS e E. RW deve essere collegato a GND.

Lo schermo è alimentato da + 5 V attraverso il GND e il pin VCC. Spediamo una tensione variabile tra 0 e 5 tramite il pin VLC, numero 3, per regolare il contrasto. Di solito lo legheremo a GND con R1, una resistenza. Il valore della resistenza può variare in base al modello dello schermo.

Poiché è impossibile prevedere esattamente che cosa accada, inizieremo con una resistenza di 1K Ω. Se si nota che i caratteri sembrano inadeguati, dovete modificare il valore del resistore con un tipo più alto, ad esempio 4K7 Ω.

Siamo sicuri di raggiungere il giusto contrasto da qualche parte tra quest due figure.

Usiamo i pin 15 e 16 dello schermo LCD per controllare la retroilluminazione, anche se non tutti i modelli hanno questa funzione. In questo caso non li utilizzeremo ed è possibile che il vostro schermo non abbia nemmeno questi due pin. Se li ha, non collegateli.

Si consiglia di avviare l'assemblaggio con i cavi di collegamento come nella figura e nella foto.

Installerai lo schermo dopo e dovrà solo coprire uno di questi collegamenti. Guarda con attenzione:

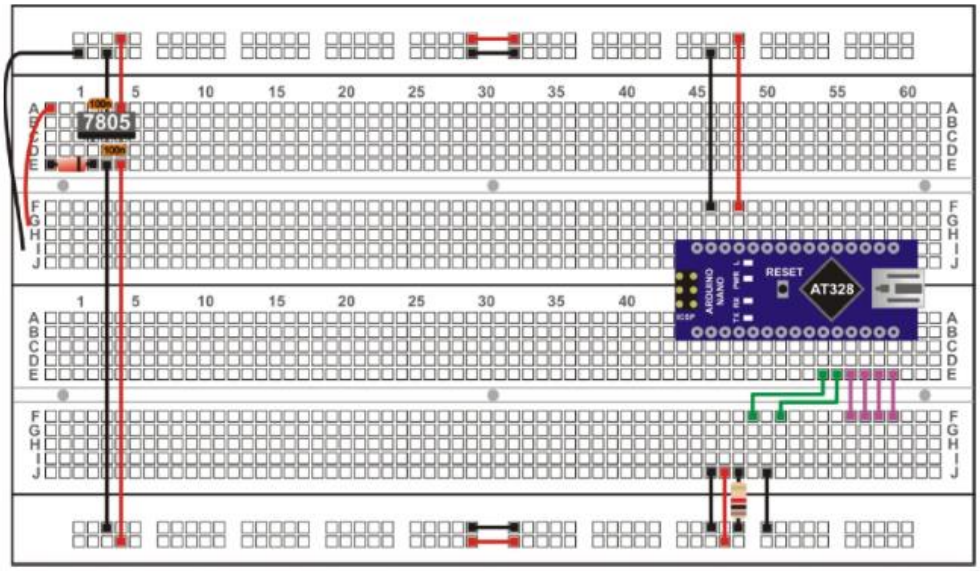

Figura 6

Ora posizionate lo schermo LCD; assicuratevi di inserire il pin 1 sullo schermo nella stessa colonna di fori del filo nero GND, il pin 2 nella stessa colonna del filo rosso +5 V, il pin 3 nella stessa colonna della resistenza e così via.

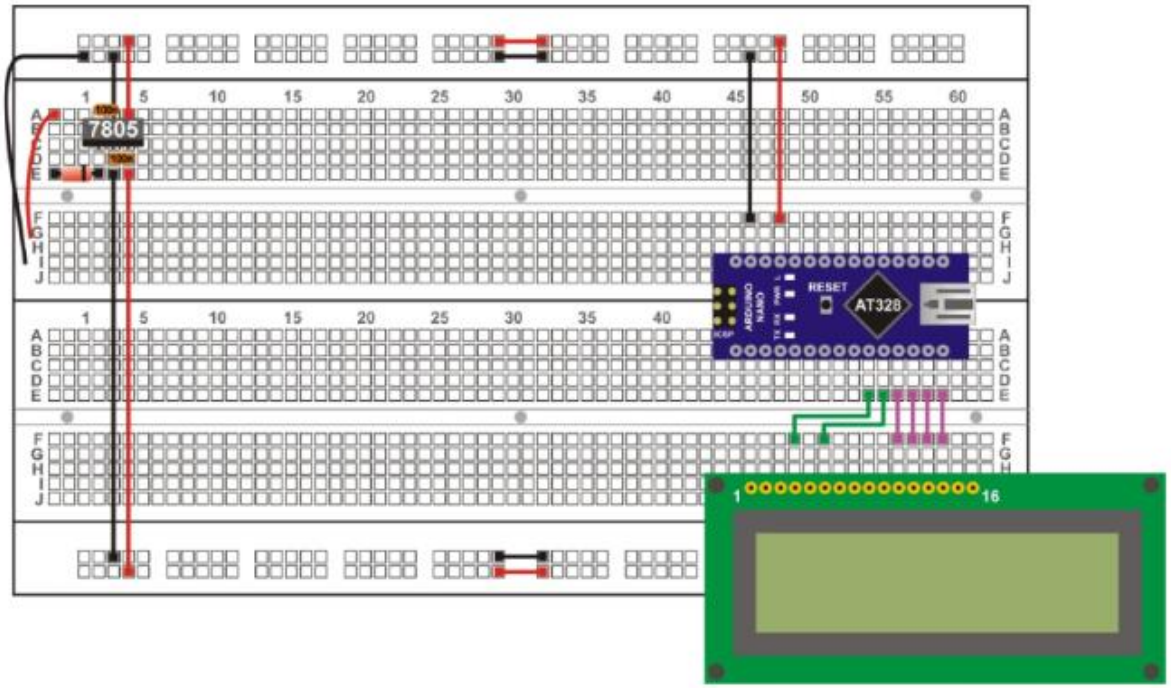

Figura 7

Pensate anche ad altri modi di connettere I componenti, e potebbe essere anche meglio.Ricordate che lo schermo LCD rimarrà connesso per quasi tutto il resto del corso. Per questo motivo dovreste lasciare spazio sulla scheda del modulo per i componenti e le periferiche che utilizzerete in futuro.

7. ESEMPIO 3: Hello world!

Ecco il primo programma che state per usare per visualizzare il famoso messaggio sullo schermo LCD.

Potete includere la libreria delle funzioni LCD utilizzando **#include**. È possibile stabilire la variabile LCD e configurare le connessioni utilizzando **LiquidCrystal()**. Assicuratevi che le connessioni corrispondano a quelle del diagramma di circuito per l'assemblaggio che avete appena fatto.

Selezionate il modello dello schermo da 16 x 2 usando la funzione **lcd.begin(16,2)** nel **setup()**.

Infine, il programma principale **loop()** trasmette il messaggio da visualizzare utilizzando la funzione **lcd.print()**. Successivamente il messaggio viene posto su un ciclo continuo chiamato **while(1)**, che non fa nulla di utile, potresti chiamarlo la fine dell'esecuzione.

8. ESEMPIO 2: Lo schermo

Vi consiglio i seguenti esempi per familiarizzare con alcune delle funzioni incluse nella libreria "**LiquidCrystal.h**" e con i vari effetti che producono.

Useremo le funzioni **noDisplay()** e **display()** per abilitare o disattivare lo schermo: lo schermo diventerà vuoto o visualizzerà qualunque contenuto abbia in quel momento.

9. ESEMPIO 3: Il cursore

Il cursore indica dove sarà scritto sullo schermo il carattere successivo. Questo esempio utilizza le funzioni **cursor()** e **noCursor()** per attivare il carattere da visualizzare o meno.

Il cursore viene visto come sottolineatura (_).

10. ESEMPIO 4: Lampeggio

Il cursore può anche essere visto come un simbolo solido intermittente (▓) che indica dove il carattere successivo sarà scritto sullo schermo. Questo è ciò che avviene per la funzione **blink()** e **noBlink()**.

Il cursore lampeggia e si spegne spontaneamente; lo schermo LCD lo controlla automaticamente. Il cursore di esempio è abilitato e mantenuto lampeggiante per tre secondi e disattivato per altri tre in modo da poter vedere quali funzioni agiscano.

11. ESEMPIO 5: Direzione del testo

Utilizzando le funzioni **leftToRight()** e **rightToLeft()** è possibile determinare come scrivere e visualizzare il testo sullo schermo. È possibile scrivere e visualizzare da sinistra a destra o da destra a sinistra. Il cursore scrive da sinistra a destra per impostazione predefinita. Questo esempio mostra un altro modo per visualizzare un messaggio. Questa volta il testo viene visualizzato in una matrice:

char Mens1[]={"ILove Arduino"}; //Messaggio da visualizzare

Due **for()** inviano ogni carattere dall'array alla schermata a intervalli di 0.150 secondi utilizzando la funzione **lcd.write()**. Il cursore va alla posizione 0 della riga 0 (la riga 0 è la prima sul display LCD) nel primo **per()** e inizia a scrivere i caratteri da sinistra a destra come di consueto. Il cursore passa alla posizione 1 della riga 15 (l'ultima nella seconda riga) nel secondo **per()** e inizia a scrivere i caratteri da destra a sinistra.

Successivamente cambiamo la sequenza di tempo a tre secondi e la funzione **lcd.clear()** cancella lo schermo. Poi c'è una sequenza finale di un secondo e poi il ciclo ricomincia.

12. ESEMPIO 6: Scorrimento

Le funzioni **scrollDisplayLeft()** e **scrollDisplayRight()** consentono anche di ottenere alcuni effetti di visualizzazione piacevoli.

Queste funzioni spostano tutto ciò che è sullo schermo in un dato momento a sinistra o a destra. Questo esempio mostra un testo a due righe sullo schermo. Successivamente l'intero testo inizia a spostarsi a sinistra rispetto alla sua posizione, a intervalli di 3 secondi.

Quando l'ultimo carattere scompare sulla sinistra, il testo inizia a muoversi di nuovo finché non scompare sulla destra. Poi comincia a spostarsi di nuovo verso sinistra finché non ritorna alla sua posizione originale. Dopo di che lo schermo si accende e spegne per un po'.

13. ESEMPIO 7: Scorrimento automatico

Questo è un altro esempio che ti permetterà di creare effetti di visualizzazione. La funzione **autoscroll()** sposta tutto il testo in uno spazio a sinistra ogni volta che viene aggiunta una lettera e **noAutoscroll()** disabilita lo scorrimento.

Quando la funzione di scorrimento automatico è attivata, la funzione sposta automaticamente ogni carattere scritto sullo schermo. Nell'esempio precedente era necessario utilizzare le funzioni **scrollDisplayLeft()** o **scrollDisplayRight()** per ottenere questo effetto.

La funzione sposta automaticamente i caratteri a sinistra per impostazione predefinita o a destra. Tutto dipende dall'ultima funzione utilizzata: **leftToRight()** o **RightToLeft().**

L'esempio seguente mostra due messaggi precedentemente definiti in due array distinti:

```
char Mens1[]={"ARDUINO"};          //Messaggio 1
char Mens2[]={"I'Love you"};       //Messaggio 2
```

Il primo **for()** scrive il contenuto dell'array "Mess1" sullo schermo, carattere per carattere. Scrive automaticamente i caratteri da sinistra a destra a intervalli di tre secondi. Il secondo **for()** lo scrive sullo schermo carattere per il carattere del contenuto dell'array "Mess2". Tuttavia, prima di eseguire questo ciclo, il cursore si posiziona nell'ultima posizione della seconda riga dello schermo e esegue la funzione **lcd.autoscroll()**. Scrive i caratteri da sinistra a destra ma ogni volta che si scrive sposta automaticamente il contenuto della schermata a sinistra per impostazione predefinita. Infine, viene eseguito **lcd.noAutoscroll()** e dopo che **lcd.clear()** elimina lo schermo il ciclo inizia nuovamente.

14. ESEMPIO 8: Caratteri personalizzati

Per concludere, questi esempi ci mostrano come utilizzare le funzioni più utili nella libreria "**LiquidCrystal.h**" e creerete e userete i personaggi grafici.

Per essere più precisi, creerete cinque caratteri: un cuore, un volto felice, un volto serio, una figura con le braccia in alto e un'altra con le braccia verso il basso.

Ognuno di esse viene definita usando una matrice. Queste sono le cinque figure: heart[8]; happy[8]; serious[8]; down[8] e up[8]. Ciascuno dei caratteri contiene una descrizione binaria di quali punti o pixel devono essere abilitati in ogni caso.

Le funzioni **lcd.createChar()** trasferiscono il contenuto degli array alla memoria CGRAM dello schermo durante **setup()**. Le funzioni trasferiscono un totale di 40 byte, otto per carattere. In aggiunta a ciascuno dei caratteri viene assegnato un numero compreso tra 0 e 5:

```
lcd.createChar(0, heart);      //Crea un nuovo carattere nº 0
lcd.createChar(1, happy);      // Crea un nuovo carattere nº 1
lcd.createChar(2, serious);    // Crea un nuovo carattere nº 2
lcd.createChar(3, down);       // Crea un nuovo carattere nº 3
lcd.createChar(4, up);         // Crea un nuovo carattere nº 4
```

Infine, è possibile visualizzare ogni carattere grafico sullo schermo ovunque il cursore sia come se fosse un normale carattere ASCII. Tutto quello che devi fare è usare la funzione **lcd.write(n)**, "n" essendo il numero assegnato a ciascun carattere come nell'esempio da 0 a 5.

15. ESEMPIO 9: Display

Ecco un esempio che è veramente pratico. State per creare un collegamento tra il canale di comunicazione seriale del vostro PC e dello schermo LCD. Naturalmente il controllore NANO Arduino sarà al centro.

Il controller funziona come un ponte tra i due. Da un lato è collegato al PC tramite la porta USB; dall'altro è collegato allo schermo nello stesso modo in cui lo abbiamo collegato fino ad ora. Utilizzando un

monitor seriale, invierete un numero di caratteri dal PC utilizzando il canale di comunicazione seriale; quando saranno arrivati, saranno ricevuti dal controllore Arduino e visualizzati sullo schermo LCD.

16. ESEMPIO 10: Visualizzazione di numeri interi

Hai già visto come è possibile visualizzare tutti i tipi di testi e caratteri sullo schermo LCD e anche ottenere diversi tipi di effetti di visualizzazione. È anche possibile visualizzare numeri.

In questo esempio vengono visualizzati i numeri da 1 a 15 in sequenza ogni volta che si preme un pulsante collegato al pin D2. Inoltre, il numero in questione verrà visualizzato come un numero decimale, esadecimale e binario.

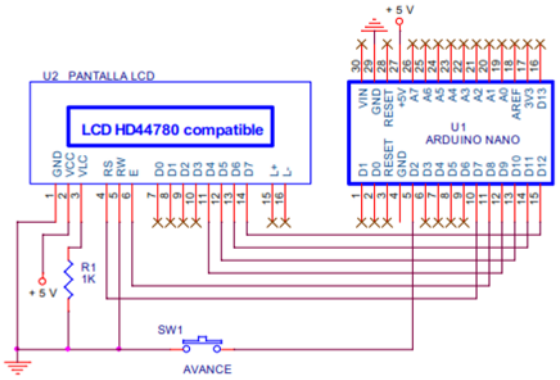

Figura 8

Il programma non presenterà difficoltà sulla base di ciò che conoscete ora. Ecco un piccolo dettaglio che è sempre utile: poichè i numeri che visualizzerai hanno diverse lunghezze, è una buona idea cancellare la prima riga dello schermo LCD in cui li visualizzerai. Si può fare questo riempiendo la linea in questione con spazi vuoti (“ “); in questo modo non ci sono residui.

Date un'occhiata al programma. Provate a togliere le prime due funzioni di loop() e vedere cosa succede: penso che capirai cosa si intenda per "residui".

17. ESEMPIO 11: Visualizzazione di numeri a virgola mobile

A seguito dell'esempio precedente, parleremo della visualizzazione di numeri a virgola mobile. In questo caso, mostreremo la radice quadrata del numero N a due cifre decimali e N moltiplicata dalla PI (3.1416)

costante a quattro cifre decimali. Il numero N avanza in sequenza ogni volta che si preme il pulsante D2; va da 0 a 9.

Sia lo schema elettrico che l'assieme sono gli stessi utilizzati per l'esempio precedente 10.

18. ESEMPIO 12: Menu

Uno schermo LCD in una periferica ideale per creare un'interfaccia utente intuitiva. E infatti ci sono menu di opzioni che consentono di eseguire una selezione delle diverse attività da eseguire.

In questo esempio l'idea è di creare un menu di sei opzioni che appaiono sullo schermo una dopo l'altra. Utilizzando un pulsante per andare avanti e un altro per tornare indietro, potrai sfogliare il menu e dare un'occhiata a tutte le opzioni disponibili. Ecco come si presenta lo schema elettrico:

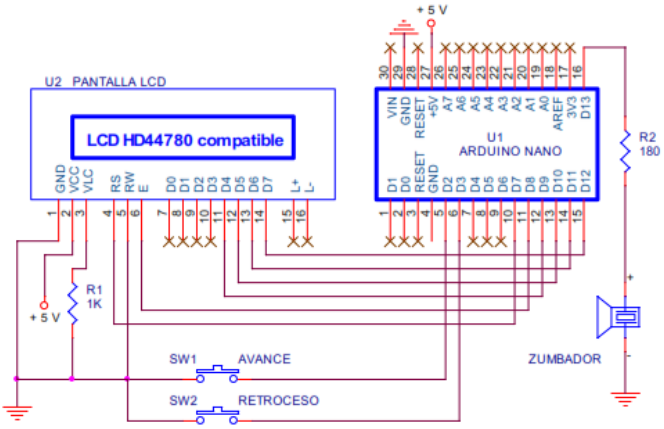

Figura 9

Le connessioni tra il controller Arduino e lo schermo sono le stesse di quelle utilizzate fino ad ora. In questo caso c'è un secondo pulsante denominato SW2. Il pulsante SW1 è collegato alla voce D2 e le opzioni disponibili sono mostrate sullo schermo da primo all'ultimo. Quando colleghiamo il pulsante SW2 al perno D3 possiamo vedere le opzioni in ordine inverso: dall'ultimo al primo. Entrambi i perni sono configurati come resistenze di sollevamento. In questo modo non è necessario mettere due resistenze esterne.

Lo schermo esegue un movimento verticale per visualizzare tutte le opzioni disponibili nel menu uno dopo l'altro. Utilizzeremo anche un buzzer elettrico collegato al pin di uscita D13. Quando si preme il pulsante SW1 o SW2, emette un rumore.

Ora diamo un'occhiata al programma. Sarà a vostra cura studiarlo, esaminarlo e cercare di capire come funziona. Tutto ciò che posso dire è che abbiamo cercato di trovare la soluzione più facile e più istruttiva; siamo sicuri che questa non è l'unica soluzione e nemmeno la migliore. Spero che voi riusciate a trovare soluzioni migliori attraverso la vostra iniziativa.

Avvio:

Prima di tutto includiamo la libreria "LiquidCrystal.h" e colleghiamo lo schermo. Successivamente si dichiara l'opzione e N variabili. La prima è l'opzione in uso e le altre il numero di opzioni disponibili nel menu. Devi anche dichiarare una matrice che definisca i caratteri grafici.

Funzioni:

Sono state create due funzioni per questo progetto: **visualize()** e **beep()**. La prima visualizza il messaggio sullo schermo che corrisponde all'opzione in uso in base al valore della variabile "Option". La funzione **beep()** fa un suono acustico.

Setup():

Configurare i pin D2 e D3 come input con resistenze pull-up interne, il pin D13 come output, lo schermo con due righe di 16 caratteri e infine generare il carattere grafico nº 0 (una freccia). Non c'è niente di veramente importante qui.

loop():

- iniziare cancellando lo schermo e visualizzando il carattere grafico nº 0 (la freccia).
- individuare il cursore sulla prima riga e visualizza l'opzione in uso in base alla variabile "opzione". Utilizza la funzione **visualize().**
- individuare il cursore nella seconda riga e visualizza la seguente opzione (Option + 1). Ancora una volta utilizza la funzione **visualize().**
- attendere fino a quando sono premuti i pulsanti SW1 o SW2 collegati a D2 e D3.

• se viene premuto D2 elimina il debug e attende fino a quando il pulsante non viene rilasciato. Emette un fischio e aumenta la variabile "Option". Si passa alla prossima opzione del menu.

• se viene premuto D3, elimina il debug e attende fino a quando il pulsante non viene rilasciato. Emette un fischio e aumenta la variabile "Option". Tornerà all'opzione precedente nel menu.

19. ESEMPIO 13: Selezione dell'opzione

Pensate a questo come estensione dell'esempio precedente. Date un'occhiata allo schema elettrico.

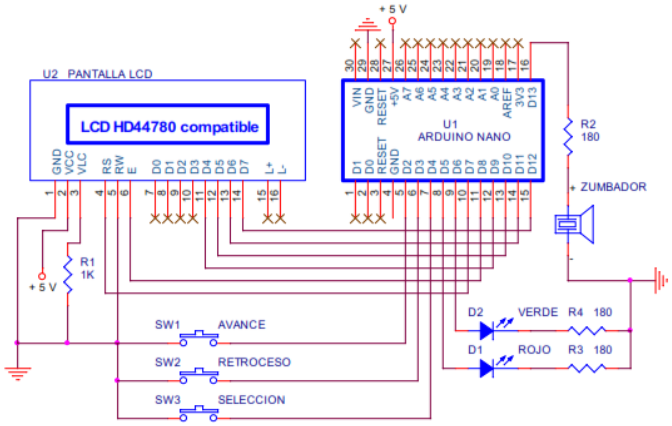

Figura 10

Aggiungiamo un altro pulsante SW3 collegato a D4; esegue l'opzione selezionata. Aggiungiamo anche due schermi LED: uno rosso e uno verde collegato a D5 e D6. Essi mostrano il risultato dell'esecuzione dell'opzione.

Questo programma è molto simile all'esempio precedente. L'unica differenza è la funzione **Execute()** che viene eseguita ogni volta che viene premuto il pulsante SW3. Questa nuova funzione valuta il valore della variabile "Opzione". A seconda di quale opzione è stata selezionata, viene eseguita l'attività appropriata e vengono abilitate o disabilitate le schermate a LED rosse e verdi collegate agli output D5 e D6.

20. ESEMPIO 14: Un ultimo miglioramento

Immagina un'applicazione che non abbia solo sei opzioni nel suo menu come quelle finora, bensì dieci. Ogni volta che si accende il sistema o si riavvia premendo RESET, esso inizia sempre dalla prima opzione; dovrai scorrere fino a trovare quello che vuoi.

Forse sarebbe una buona idea se lo schermo LCD mostrasse l'ultima opzione selezionata in modo da non doverla cercare ogni volta che riavviate il sistema. Ancora una volta è possibile utilizzare la memoria dati EEPROM. Alla fine della giornata, ogni opzione viene rappresentata con un numero salvato nella variabile "Opzione".

Ogni volta che il sistema viene riavviato, a questa variabile viene assegnato l'ultimo valore registrato nella memoria EEPROM. Utilizzerà la posizione 1 di 1024 disponibili.

Registrare il programma e assicurarsi che funzioni. Selezionare ed eseguire le varie opzioni. Spegnere quindi il sistema e riaccenderlo. Prestare attenzione a come appare l'ultima opzione selezionata.

RIFERIMENTI

PUBBLICAZIONI

[1]. **EXPLORING ARDUINO**, Jeremy Blum, Ed.Willey
[2]. Practical ARDUINO, Jonathan Oxer & Hugh Blemings, Ed.Technology in action
[3]. Programing Arduino, Next Steps, Simon Monk
[4]. Sensores y actuadores, Aplicaciones con ARDUINO, Leonel G.Corona Ramirez, Griselda S. Abarca Jiménez, Jesús Mares Carreño, Ed.Patria
[5]. ARDUINO: Curso práctico de formación, Oscar Torrente Artero, Ed.RC libros
[6]. 30 ARDUINO PROJECTS FOR THE EVIL GENIUS, Simon Monk, Ed. TAB
[7]. Beginning C for ARDUINO, Jack Purdum, Ph.D., Ed.Technology in action
[8]. ARDUINO programing notebook, Brian W.Evans

SITI WEB

[1]. https://www.arduino.cc/
[2]. https://www.prometec.net/
[3]. http://blog.bricogeek.com/
[4]. https://aprendiendoarduino.wordpress.com/
[5]. https://www.sparkfun.com