UNIDADE 6: ECRÃS LCD (LIQUID-CRYSTAL DISPLAY)

OBJETIVOS

Existe uma ampla gama de periféricos digitais de input e output. Até agora temos vindo a utilizar os mais comuns – botões ou interruptores simples e económicos – como dispositivos para definir os níveis lógicos "1" e "0", e os LEDs e os diodos para os representar.

Mas agora é o momento de abordarmos outros periféricos digitais mais importantes e mais conhecidos. Nesta unidade, vamos dar uma espreitadela sobre o ecrã de LCD como um periférico de output: este ecrã permite exibir qualquer tipo de informação de output, incluindo números, letras e símbolos.

SECÇÃO TEÓRICA

- ECRÃS LCD
- O GRUPO DE CARATERES
- CARATERES GRÁFICOS
- A BIBLIOTECA LiquidCrystal.h
- A MEMÓRIA DE DADOS EEPROM

SECÇÃO PRÁTICA

- LIGAR ECRÃS LCD
- EXEMPLO 1: Olá Mundo!!
- EXEMPLO 2: O ecrã
- EXEMPLO 3: O cursor
- EXEMPLO 4: Piscar
- EXEMPLO 5: Orientação do texto
- EXEMPLO 6: *Scrolling*
- EXEMPLO 7: *AutoScroll*
- EXEMPLO 8: Carateres *standard*
- EXEMPLO 9: Exibir a informação
- EXEMPLO 10: Mostrar números inteiros
- EXEMPLO 11: Mostrar números com variações pontuais
- EXEMPLO 12: Menú
- EXEMPLO 13: Selecionar opções
- EXEMPLO 14: Uma última melhoria

MATERIAIS

-Laptop ou computador de secretária.
-Ambiente de trabalho Arduino IDE; deve incluir o material suplementar já instalado e configurado.
-Placa de controlo Arduino UNO.
-Um cabo USB.

ÍNCIDE

SECÇÃO TEÓRICA

1. ECRÃS LCD

Este é um periférico de output que não só mostra números, mas também todos os tipos de caracteres, textos, símbolos e até gráficos simples. Tem milhares de usos e já os viu várias vezes, com diferentes números de linhas e de caracteres por linha, com retroiluminação de cores e caracteres de diferentes cores e tamanhos.

Todos eles têm o seu próprio controlador que administra todas as operações internas. A maioria deles é compatível com o popular Hitachi HD44780. Por isso, é indiferente o ecrã que utilizar: 2 linhas por 16 caracteres, uma ou 4 linhas por 20 caracteres.

Vai utilizar um ecrã LCD de 2 x 16 LCD. Na Figura 1 pode observar um exemplo com uma representação simplificada do diagrama elétrico.

Figura 1

Um ecrã LCD é um periférico digital. Os seus sinais podem ser ligados diretamente aos pins de entrada e de saída do controlador e estão divididos em três grupos:

- **Fornecimento de energia**: Os pins de input e de output fornecem a energia que o ecrã necessita e situam-se normalmente entre os + 5V GND e os VCC. Há também um terceiro pin, um VLC, que fornece uma energia alternada entre 0 V e +5 V que pode ser usada para ajustar o contraste do ecrã.
- **Controlo**: Esses são os sinais que decidem se o ecrã recebe um comando ou alguns dados, se será lido ou gravado pelo controlador ou se o ecrã está ativo ou não.
- **Dados**: O controlador utiliza estes sinais para enviar comandos e dados adequados para o ecrã.

O pin nº 1 é o primeiro da esquerda, observe onde se posiciona o pin número 1, na Figura 2.

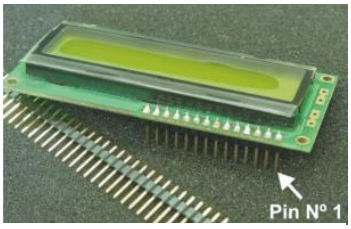

Figura 2

Observe a tabela seguinte, onde se apresenta a descrição de cada um dos pins.

Nº PIN	NOME	TIPO	DESCRIÇÃO
1	**Vss**	Fornecimento de voltagem	Fonte de alimentação terrestre (0 V)
2	**Vdd**	Fornecimento de voltagem	Fornecimento de energia de +5 Vcc
3	**VLC**	Fornecimento de voltagem	Ajuste de contraste: alternância de voltagens entre 0 e +5 Vcc.
4	**RS**	Input	Output do controlador Arduino. Seleciona entre instruções e dados. RS=0 Arduino transfere instruções RS=1 Arduino transfere dados (códigos ASCII)
5	**R/W**	Input	Output do Arduino. Controla a leitura e a escrita: R/W=0 O Arduino desempenha dados de escrita no ecrã LCD R/W=1 O Arduino desempenha dados de leitura no ecrã LCD
6	**E**	Input	Output do Arduino. Ativa o ecrã: E=0 Ecrã LCD desativado (em grande incompatibilidade) E=1 ecrã LCD ativo
7-14	**DB0:DB7**	Input/Output	Linhas de barramento de dados e instruções. O Arduino transfere instruções ou dados para o ecrã de acordo com o sinal RS. As linhas DB0:DB7 são utilizadas com uma interface de 8 bit. As linhas DB4:DB7 são utilizadas com uma interface de 4 bit.
15	**L+**	Fornecimento de energia	Voltagem positiva para a iluminação de fundo (+5 Vcc)
16	**L-**	Fornecimento de energia	Voltagem negativa para a iluminação de fundo (0 V)

Não vamos usar os números dos pins 15 e 16: são opcionais. É bem possível que nem apareçam no ecrã; somente os que têm luz de fundo é que os usam.

Os ecrãs estão entre os periféricos mais poderosos e versáteis. Como já foi referido, podem exibir todos os tipos de mensagens compostas de texto, números ou símbolos, assim como fornecer uma variedade de efeitos de visualização, como movimentos para a esquerda ou direita, cintilar, , *scrolling* etc. Um ecrã LCD tem um controlador separado.

O ecrã LCD que vai utilizar e que está incluído no kit de materiais sugerido pode ver-se na Figura 2, acima. Este ecrã tem duas linhas de 16 caracteres cada (2 x16). Também vai aperceber-se como terá de soldar uma fita de 14/16 pins macho; isto vai facilitar inseri-los no quadro prático do

módulo prático mais tarde. Vai ter ppossibilidade de fazer as ligações com o controlador muito mais rapidamente.

2. O GRUPO DE CARATERES

A comunicação entre o Arduino e os ecrãs LCD obtém-se basicamente através de pins digitais DB0-DB7. Pode usar oito pins ou, como no nosso caso, apenas quatro. A isto chama-se trabalhar com uma “interface de 4 bits.”

O Arduino envia comandos ou instruções para o ecrã através dos pins. O ecrã pode então realizar uma série de efeitos de exibição: scrolling, cintilar, excluir, posicionar o cursor, etc... O Arduino também envia os códigos de carateres ASCII que desejar exibir. Estes são códigos de 8 bits. Se utilizar uma “interface de 8 bits”, o Arduino precisa apenas de uma única transferência para exibir cada carater; uma “interface de 4 bits” precisa de duas transferências para exibir cada carater. É um pouco mais lento, mas utiliza menos cabos de ligação. De qualquer forma, não se preocupe muito com isso: as funções que vai estudar facilitarão bastante esta tarefa.

A Figura 3 apresenta uma amostra de um grupo de carateres que os ecrãs LCD exibem; este conjunto é estabelecido pelo fabricante. A memória ROM interna contem a definição de cada um dos carateres e é capaz de alternar entre as diferentes modelos e versões.

Figura 3

Os quatro bits mais leves, B3:B0, são representados nas linhas à esquerda. Os quatro mais pesados, B7:B4, são representados no topo da tabela nas colunas em código binário. Tudo o que é necessário fazer-se para codificar um carater é selecioná-lo e, em seguida, encontrar em qual a coluna ou linha em que se encontra. Veja o exemplo seguinte: o carater "F" está na coluna 4 (0100) e na linha 6 (0110). O seu código binário é, portanto, 0100 0110 (0x46), o que corresponde exatamente ao código ASCII para o carater “F”. Anteriormente referiu-se que nem todos os ecrãs têm o mesmo conjunto de carateres. Depende do fabricante, do modelo, da versão, etc. No entanto, os códigos correspondentes aos carateres ASCII padrão são comuns a todos os ecrãs. Estes são os que se encontram nas colunas 2 (0010) a 7 (0111).

3. OS CARATERES GRÁFICOS

Pode criar um total de até oito carateres gráficos de 5x8 pontos ou “pixels”. Cada carater é numerado de 0 a 7 e precisa de um total de 8 bytes para serem definidos. O ecrã LCD possui uma memória RAM interna chamada CGRAM para executar essa tarefa. Depois de definidos, pode exibi-los enviando o número que corresponde ao carater (entre 0 e 7). Não se esqueça: os carateres gráficos são armazenados na memória RAM. Se desligar o sistema, perde-os e o Arduino precisará de os definir novamente.

Os caracteres gráficos são definidos inserindo bytes em posições sucessivas da memória CGRAM; os padrões binários dos bytes definem cada carater. O CGRAM é uma memória volátil capaz de armazenar um total de 64 bytes. Um carater de 5x8 pontos precisa de 8 bytes para o definir. Então pode definir até oito caracteres diferentes (8x8) da forma que quiser.

A tabela apresentada na Figura 4 contem quatro carateres 5 x 8 definidos como no exemplo. Estão inseridos nas primeiras 32 posições do CGRAM. O primeiro encontra-se nas posições 0 a 7, o segundo nas posições 8 a 15 e assim por diante.

Cada bit de cada um dos bytes que tem um valor de “1” ativa o seu ponto ou pixel correspondente no ecrã LCD. O primeiro carater gráfico do CGRAM é exibido enviando o número 0 como se fosse um código ASCII. O segundo carater é visualizado enviando o número 1 e assim por diante com todos os que tiver definido.

Tudo o que tem a fazer é define-los e criar uma matriz para cada carater. De seguida, os conteúdos de cada matriz são copiados para o ecrã CGRAM através da função **crateChar()**. Tem de criar o mesmo número de matrizes e de carateres; existem quatro no exemplo seguinte:

```
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};

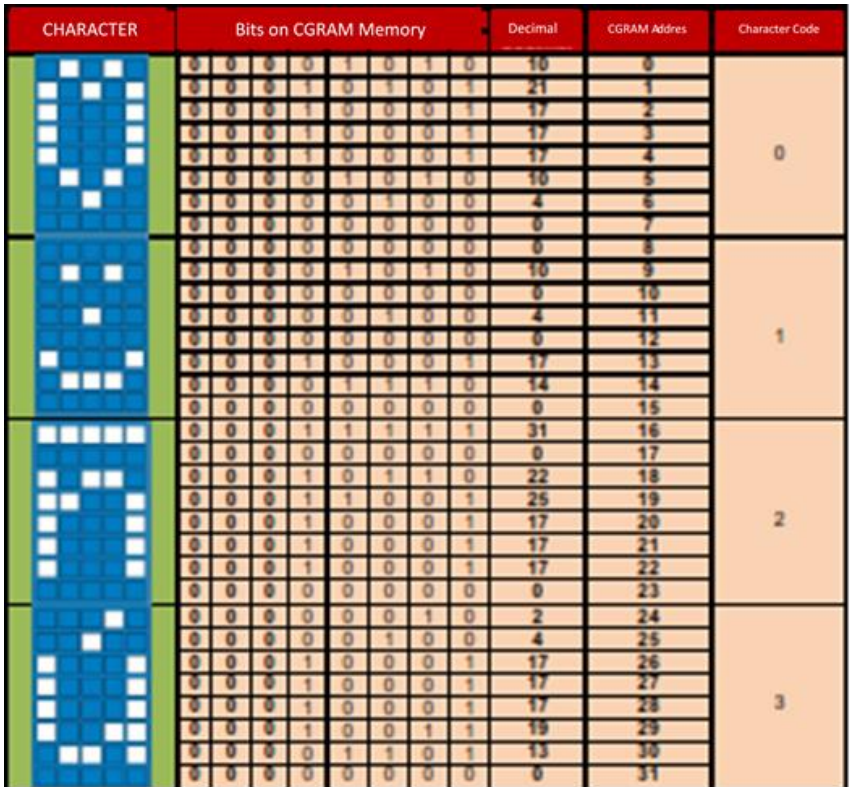

Figura 4

4. A BIBLIOTECA LIQUIDCRYSTAL.H

Você já estará familiarizado com o conceito de uma "biblioteca" e como lidar com a variedade de funções que ela contém. O Arduino criou um grande número de bibliotecas, mas vamos falar sobre uma delas: "LiquidCrystal.h". Ele contém um grande número de funções projetadas para controlar e operar telas LCD baseadas no controlador Hitachi HD44780 ou compatíveis com ele.

Vamos estudar as funções mais representativas e importantes. Em qualquer caso, eu recomendo que você visite www.arduino.cc onde você encontrará muitas informações e exemplos de todos eles.

- **Função: LiquidCrystal()**

Esta função cria uma variável do tipo “LiquidCrystal” e estabelece as conexões entre a tela LCD e o controlador Arduino; você pode usar uma interface de 8 ou 4 bits entre eles. Nesse caso, não use as

linhas D0-D3. Você também pode decidir se deseja ou não usar o sinal de tela de R / W. Se você não usá-lo, como neste caso, conecte o sinal ao GND. Você fará isso na seção de prática.

Sintaxe:

LiquidCrystal var(RS,E,D4,D5,D6,D7); //For a 4 bit interface without R/W signal
LiquidCrystal var(RS,RW,E,D4,D5,D6,D7); //For a 4 bit interface with R/W signal
LiquidCrystal var(RS,E,D0,D1,D2,D3,D4,D5,D6,D7); //For an 8 bit interface without R/W signal

LiquidCrystal var(RS,RW,E,D0,D1,D2,D3,D4,D5,D6,D7); //For an 8 bit interface with R/W signal

var: the name for the variable assigned to the LCD screen that you're going to control.

RS: the Arduino pin connected to the screen RS signal.

RW: the Arduino pin connected to the screen R/W signal (if it's going to be used).

E: the Arduino pin connected to the screen E signal.

D0-D7: Arduino pins that are connected to the screen DB0-DB7 data lines. If there are no pins indicated for DB0-DB3, we assume a 4 bit interface and only employ DB4-DB7 signals.

Exemplo:

LiquidCrystal lcd(7,8,9,10,11,12); *//Establishes the connection of a screen called "lcd". Arduino //pins D7 to D12 are connected to the screen RS, E, DB4, DB5, //DB6 and DB7 signals. The R/W signal must be connected to //GND.*

- **Função: Begin()**

Esta função inicia a tela LCD e atribui a ela o número de linhas e o número de caracteres por linha, de acordo com o modelo em questão. Nesse caso, você usará uma tela de 16 x 2 caracteres.

Sintaxe:

var.begin(c,f);

var: this is the name that defines the screen in question (established in LyquidCrystal()).

c: the number of columns.

f: the number of rows.

Exemplo:

LiquidCrystal lcd(7,8,9,10,11,12); //Connections to the LCD screen *lcd.begin(16,2);* this // is a 16 x 2 “lcd” screen

- **Função: setCursor()**

Esta função posiciona o cursor da tela LCD como desejado. A partir de então, os caracteres precedentes serão exibidos.

Sintaxe:

var.setCursor(c,f);

var: This is the name that defines the screen in question (established as LyquidCrystal()).

c: the number of columns (starting from 0).

f: the number of rows (starting from 0).

Exemplo:

LiquidCrystal lcd(7,8,9,10,11,12); //Connections to the LCD screen

lcd.setCursor(3,0); //Places the cursor in position 3 (4th //character) of row //0 (the 1st)

- **Função: home()**

Essa função localiza o cursor no canto superior esquerdo (posição 0 da linha 0) da primeira posição da tela. Não exclui o que estava sendo exibido anteriormente.

Sintaxe:

var.home();

var: This is the name that defines the screen in question (established as LyquidCrystal()).

- **Função: clear()**

Esta função limpa a tela LCD e localiza o cursor no canto superior esquerdo (linha 0 posição 0).

Sintaxe:

var.clear();

var: This is the name that defines the screen in question (established as LyquidCrystal()).

Exemplo:

LiquidCrystal lcd(7,8,9,10,11,12); *//Connections to the LCD screen*

lcd.clear(); *//Clears the screen*

- **Função: write()**

Esta função escreve um caractere na posição atual do cursor.

Sintaxe:

var.write(char);

var: This is the name that defines the screen in question (established as LyquidCrystal()).

char: The character to be displayed

Exemplo:

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

- **Função: print()**

Esta função imprime na tela LCD a partir da posição atual do cursor.

Sintaxe:

var.print(data);

var.print(data,base);

var: This is the name that defines the screen in question (established as LyquidCrystal()).

data: This is the data to be printed. This might be char, int, long, float or string.

base: This is optional and shows the desired numerical base: BIN=Binary; DEC=Decimal (by default); OCT=Octal; HEX=Hexadecimal; or N=nº in decimals for floating-point numbers (2 by default).

Exemplos:

int A=19;

float PI=3.1416;

lcd.print(A,HEX); *//Prints A in hexadecimal (1910 = 1316)*

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

lcd.print(PI*2,4); *//Prints 6.2832 (four decimals)*

- **Função: cursor()**

Esta função exibe o cursor na tela LCD em sua posição atual como um sublinhado (_). É aqui que ele começará a escrever o próximo caracter.

Sintaxe:

var.cursor();

var: This is the name that defines the screen in question (established as LyquidCrystal()).

- **Função: noCursor()**

Esta função oculta o cursor do LCD.

Sintaxe:

var.noCursor();

var: This is the name that defines the screen in question (established as LyquidCrystal()).

- **Função: blink()**

Esta função exibe o cursor do LCD na tela em sua posição atual como um símbolo sólido intermitente (▓). É aqui que ele começará a escrever o próximo caractere.

Sintaxe:

var.blink();

var: This is the name that defines the screen in question (established as LyquidCrystal()).

- **Função: noBlink()**

Esta função oculta o cursor intermitente sólido (▓).

Sintaxe:

var.noBlink();

var: This is the name that defines the screen in question (established as LyquidCrystal()).

- **Função: noDisplay()**

Esta função desliga a tela LCD sem perder o conteúdo que pode estar nela ou a posição do cursor.

Sintaxe:

var.noDisplay();

var: This is the name that defines the screen in question (established as LyquidCrystal()).

- **Função: display()**

Esta função conecta a tela LCD e recupera o conteúdo exibido antes que noDisplay () seja executado.

Sintaxe:

var.display();

var: This is the name that defines the screen in question (established as LyquidCrystal()).

- **Função: scrollDisplayLeft()**

Esta função desloca o conteúdo (o texto e a posição do cursor) exibido na tela a qualquer momento, um lugar à esquerda.

Sintaxe:

var.scrollDisplayLeft();

var: This is the name that defines the screen in question (established as LyquidCrystal()).

- **Função: scrollDisplayRight()**

Esta função desloca o conteúdo (o texto e a posição do cursor) exibido na tela a qualquer momento, um lugar à direita.

Sintaxe:

var.scrollDisplayRight();

var: Este é o nome que define a tela em questão (estabelecido como LyquidCrystal ()).

- **Função: leftToRight()**

Esta função estabelece automaticamente qual direção o cursor escreve na tela: da esquerda para a direita. Isso significa que os caracteres são escritos da esquerda para a direita sem afetar os que já foram escritos.

Sintaxe:

var.LeftToRight();

var: Este é o nome que define a tela em questão (estabelecido como LyquidCrystal ()).

- **Função:rightToLeft()**

Esta função inverte a direção que o cursor grava na tela: da direita para a esquerda. Isso significa que os caracteres são escritos da direita para a esquerda sem afetar os que já foram escritos.

Sintaxe:

var.RightToLeft();

var: Este é o nome que define a tela em questão (estabelecido como LyquidCrystal ()).

- **Função: autoscroll()**

Esta função ativa o movimento de rolagem ou exibição automática. Cada vez que um caractere é enviado para a tela, essa função o exibe e, em seguida, move o restante do conteúdo em um único lugar. Se a direção no momento for da esquerda para a direita (leftToRight ()), o conteúdo será movido para a esquerda. Se a direção no momento é da direita para a esquerda (rightToLeft ()), o conteúdo se move para a direita.

Sintaxe:

var.autoscroll();

var: This is the name that defines the screen in question (established as LyquidCrystal()).

- **Função: noAutoscroll()**

This function deactivates the scrolling or automatic display movement.

Sintaxe:

var.noAutoscroll();

var: This is the name that defines the screen in question (established as LyquidCrystal()).

- **Função: createChar()**

Essa função cria um caractere definido pelo usuário. É capaz de criar um total de oito caracteres de 5 x 8 pixels numerados de 0 a 7. Uma matriz de 8 bytes, um por linha, determina a aparência ou o design de um personagem. Os cinco bits mais claros de cada byte correspondem aos 5 pixels que compõem cada linha do caractere. Uma vez que esta função tenha criado um caractere, o usuário pode exibi-lo na tela simplesmente indicando seu número.

Sintaxe:

var.createChar(n,dato);

var: This is the name that defines the screen in question (established as LyquidCrystal()).

n: this represents the number of the character to be defined (from 0 to 7).

data: This is the name of the matrix that contains the bytes that define the custom character.

Exemplo:

bytearrowhead[8]={B00100,B01110,B10101,B00100,B00100,B00100, B00100, B00100};
//Creates the 8 byte "arrowhead" matrix which defines the new graphic character

lcd.createChar(0, arrowhead);
//Creates the new n° 0 character from the defined content in the "arrowhead" array….
lcd.write(byte(0));
//displays the nº 0 graphic character (the arrowhead)

5. A MEMÓRIA DE DADOS EEPROM

Se você acha que o assunto a seguir não tem nada a ver com o assunto, você estaria absolutamente certo: não é. Mas acontece que é um bom momento para falar sobre a memória de dados EEPROM do controlador Arduino UNO. Isso ajudará você a melhorar muitos dos seus projetos atuais e futuros.

Os controladores que compõem as diversas placas Arduino fazem parte de uma memória especial: a memória EEPROM. Ele permite que você leia ou grave dados de 8 bits entre 0 e 255. Essa memória tem um recurso especial: ela retém todas as informações que você salvou, mesmo que a energia seja desligada; não elimina.

É por isso que é ideal para armazenar informações não voláteis que podem ser modificadas, como códigos de acesso, parâmetros de configuração, números de telefone, senhas, etc ... Como você já sabe, a placa NANO Arduino inclui um controlador de modelo ATmega328 que tem 1KB (1024 bytes) Memória EEPROM.

A essa altura, você sabe mais do que o suficiente para usá-lo. Você também tem acesso a uma biblioteca: "EEPROM.h". Ele tem apenas duas funções e você pode usá-las para ler e gravar dados nessa memória. Como de costume, você deve incluir essa biblioteca em seus programas usando #include <EEPROM.h>.

- **Função: read()**

Esta função lê os dados de 8 bits que estão na memória EEPROM.

Sintaxe:

EEPROM.read(dir);

dir: This determines which direction you read in from the EEPROM. It has 1024 positions (KB) so the possible range is from 0 to 1023.

Exemplo:

```
#include EEPROM.h          //Includes the library
byte value;                //8 bit variable
value = EEPROM.read(279);  //Reads the byte in position 279
```

- **Função: write()**

Esta função grava um valor de 8 bits (entre 0 e 255) em qualquer uma das posições disponíveis na EEPROM.

Sintaxe:

EEPROM.write(dir, value);

dir: This determines which direction you write in from the EEPROM. It has 1024 positions (KB) so the possible range is from 0 to 1023.

value: This is the 8 bit value (between 0 and 255) which you want to write in the direction indicated.

Exemplo:

```
EEPROM.write(982, 33);          //Write the value 33 in position 982
```

IMPORTANTE: A memória EEPROM tem uma capacidade estimada de 100.000 ciclos de gravação / exclusão. Uma gravação de EEPROM pode levar até 3,3 mS para ser concluída, portanto, talvez seja necessário ter cuidado com a frequência com que você escreve nela; se você ultrapassar o limite, ele poderá ser desabilitado. Tenha isso em mente.

SECÇÃO PRÁTICA

6. LIGAR O ECRÃ LCD

Vai começar a secção prática a ligar o ecrã LCD ao controlador Arduino na placa do módulo. Observe atentamente as ligações e certifique-se de que monta tudo corretamente. Aqui está o diagrama do circuito (Figura 5):

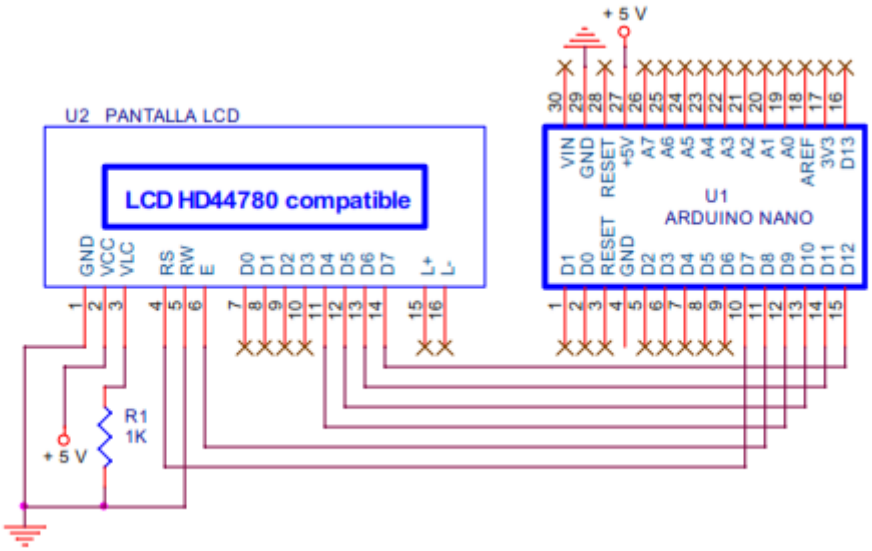

Figura 5

Os pins Arduino D9:D12 são conectados aos sinais DB4:DB7 do ecrã. Monitorizado pelo seu programa, o Arduino envia as instruções e carateres a serem exibidos. D7 e D8 controlam os sinais RS e E respetivamente. RW tem de estar ligado ao GND.

O ecrã é alimentado por +5 V através do GND e do pin VCC. Enviamos uma voltage variável entre 0 e 5 através do pin VLC, número 3, para ajustar o contraste. Normalmente ligamo-lo ao GND com R1, uma resistência. O valor da resistência pode variar de acordo com o modelo do ecrã.

Como é impossível prever exatamente que tipo de modelo encontrará, vamos começar com uma resistência de 1K Ω. Se se aperceber que os carateres parecem muito austeros e que dificilmente os pode obter, mude o valor da resistência para um valor maior, como 4K7 Ω, por exemplo. Vai acertar no nível de contraste adequado entre esses dois valores.

Utilizamos os pins 15 e 16 do ecrã LCD para controlar a luz preta, embora nem todos os modelos tenham esta característica. Vamos utilize-los e é possível que o seu ecrã nem sequer tenha estes dois pins. Caso não os tenha, não os ligue.

Recomendamos que comece a montagem com a ligação dos fios tal como se apresenta na Figura 6. Vai instalar o ecrã mais tarde e pode tapar algum destes fios. Observe atentamente.

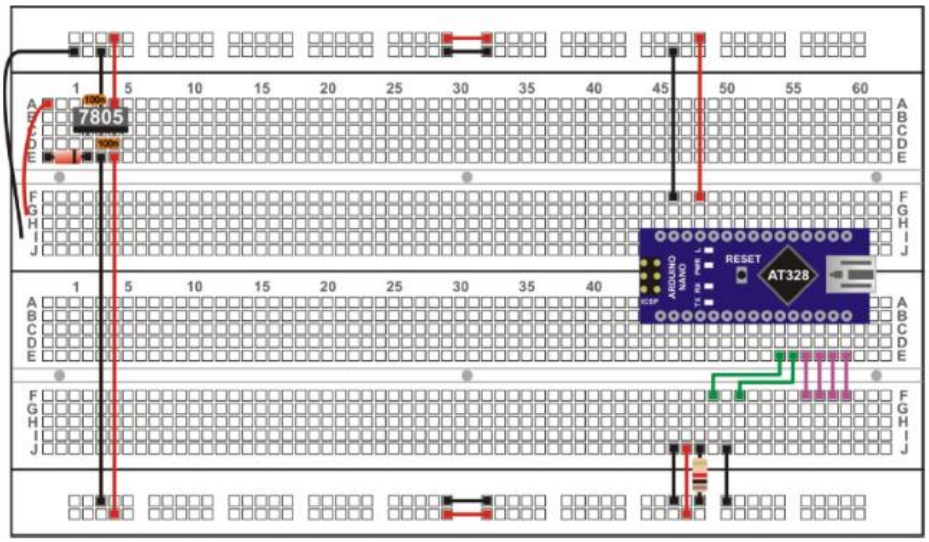

Figura 6

Posicione o ecrã LCD; certifique-se que insere o pin 1 no ecrã na mesma coluna de furos como o fio preto GND, o pin 2 na mesma coluna do fio +5 V vermelho, o pin 3 na mesma resistência e assim por diante (Figura 7).

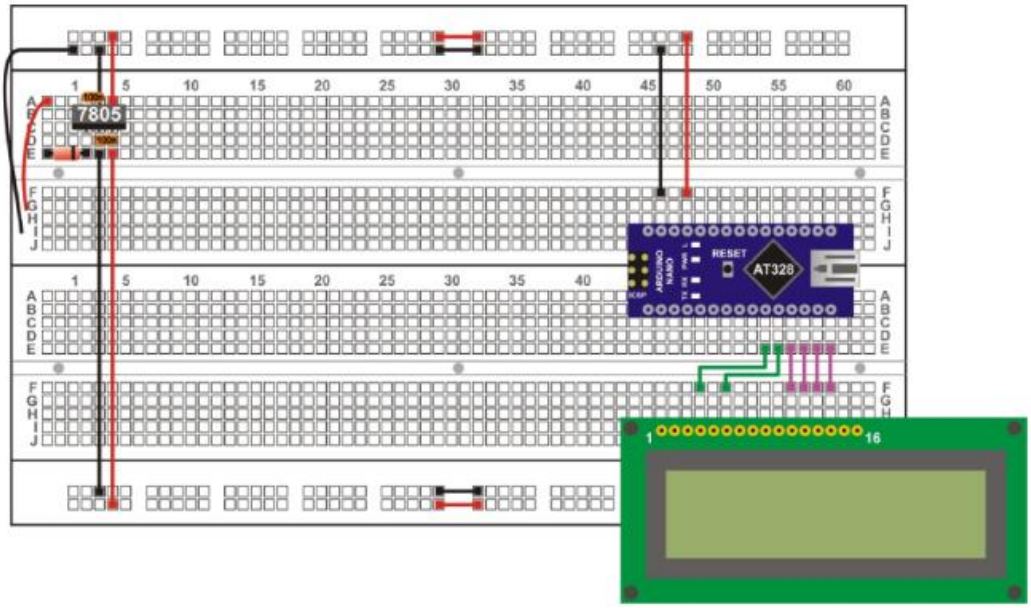

Figura 7

Tem razão em pensar noutra forma de conectar os componentes e pode até ser melhor. Deve lembrar-se, no entanto, que o ecrã LCD ficará conectado durante quase todo o restante do curso. Por

esse motivo, deve deixar espaço na placa do módulo para os componentes e periféricos que vai usar no futuro.

7. EXEMPLO 1: Olá mundo!

Aqui está o primeiro programa que vai usar para exibir a famosa mensagem no ecrã LCD.

Pode incluir a biblioteca de funções do LCD usando #include. Pode estabelecer a variável lcd e configurar as conexões usando **LiquidCrystal()**. Verifique se as conexões correspondem às do diagrama de circuito da montagem que acabou de fazer.

Selecione o ecrã modelo 16 x 2 utilizando a função **lcd.begin(16,2)** no **setup().**

Por último, o programa **loop()** principal transmite a mensagem a ser exibida, utilizando a função **lcd.print()**. De seguida, a mensagem é colocada num loop contínuo denominado **while(1)** que na verdade não faz nada de útil. Pode mesmo chamar-lhe o final da execução.

8. EXEMPLO 2: O Ecrã

Sugerimos que você faça os seguintes exemplos para se familiarizar com algumas das funções incluídas na biblioteca “LiquidCrystal.h” e os vários efeitos que elas produzem.

Você usará as funções no Display () e display () para ativar ou desativar a tela: a tela ficará em branco ou exibirá o conteúdo que tiver no momento.

9. EXEMPLO 3: O Cursor

O cursor indica onde o próximo caractere recebido será escrito na tela. Este exemplo usa as funções cursor () e noCursor () para permitir que o caractere seja exibido ou não.

Em teoria, o cursor é visto como um sublinhado (_).

10. EXEMPLO 4: Piscar

O cursor também pode ser visto como um símbolo sólido intermitente (▓) que indica onde o próximo caractere recebido será escrito na tela. É para isso que as funções blink () e noBlink () são para.

O cursor pisca e apaga espontaneamente; a tela LCD controla automaticamente. No exemplo, o cursor está habilitado e permanece piscando por três segundos e, em seguida, desabilitado para outros três, para que você possa ver o que essas funções fazem.

11. EXEMPLO 5: Orientação do Texto

Usando as funções leftToRight () e rightToLeft (), você pode determinar como você escreve e exibe o texto na tela. Você pode escrever e exibir da esquerda para a direita ou da direita para a esquerda. O cursor escreve da esquerda para a direita por padrão, da mesma forma que fazemos quando escrevemos algo.

Este exemplo mostra outra maneira de exibir uma mensagem. Desta vez, o texto é exibido em uma matriz:

char Mens1[]={"ILove Arduino"}; //Message to be displayed

Dois loops for () enviam cada caractere da matriz para a tela em intervalos de 0,150 segundos usando a função lcd.write (). O cursor vai para a posição 0 da linha 0 (a linha 0 é a primeira no LCD) no primeiro loop for () e começa a escrever os caracteres da esquerda para a direita como de costume. O cursor vai para a posição 1 da linha 1 (a última na segunda linha) no segundo loop for () e começa a escrever os caracteres da direita para a esquerda.

Em seguida, alteramos a sequência de tempo para três segundos e a função lcd.clear () limpa a tela. Então há uma seqüência final de um segundo e então o ciclo começa novamente.

12. EXEMPLO 6: Scrolling

As funções scrollDisplayLeft () e scrollDisplayRight () também permitem que você obtenha alguns efeitos de exibição agradáveis - você deve tê-los visto em outdoors.

Essas funções movem o que estiver na tela a qualquer momento para a esquerda ou para a direita. Este exemplo mostra um texto de duas linhas na tela. Subsequentemente todo o texto começa a se mover para a esquerda de uma posição para a próxima em intervalos de 3 segundos.

Quando o último caractere desaparece à esquerda, o texto começa a se mover novamente até desaparecer à direita. Então ele começa a se mover para a esquerda novamente até que finalmente retorne à sua posição original. Depois disso, a tela entra e sai por um tempo.

13. EXEMPLO 7: AutoScroll

Este é outro exemplo que lhe permitirá criar efeitos de exibição impressionantes. A função autoscroll () move todo o texto um espaço para a esquerda cada vez que uma letra é adicionada e noAutoscroll () desativa o scrolling.

Quando a rolagem automática é ativada, a função move automaticamente cada caractere escrito na tela. No exemplo anterior, você tinha que usar as funções scrollDisplayLeft () ou scrollDisplayRight () para obter esse efeito.

A função move os caracteres automaticamente para a esquerda, por padrão ou para a direita. Tudo depende da última função que você usou: leftToRight () ou RightToLeft **().**

O exemplo a seguir mostra duas mensagens que foram definidas anteriormente em duas matrizes separadas:

```
char Mens1[]={"ARDUINO"};          //Message 1

char Mens2[]={"I'Love you"};       //Message 2
```

O primeiro loop for () grava o conteúdo do array “Mess1” no caractere de tela por caractere. Ele grava automaticamente os caracteres da esquerda para a direita em intervalos de três segundos.

O segundo loop for () grava o conteúdo da matriz “Mess2” no caractere de tela por caractere. No entanto, antes de executar este loop, o cursor se localiza na última posição da segunda linha da tela e executa a função lcd.autoscroll (). Escreve os caracteres da esquerda para a direita, mas cada vez que escreve um, move automaticamente o conteúdo da tela para a esquerda por padrão.

O último de todos os lcd.noAutoscroll () é executado e depois de lcd.clear () limpa a tela, o ciclo começa novamente.

14. EXEMPLO 8: Custom characters

Para finalizar esta seção de exemplos que mostram como usar as funções mais úteis da biblioteca "LiquidCrystal.h", você criará e usará seus próprios personagens gráficos.

Para ser mais específico, você vai criar cinco personagens: um coração, um rosto feliz, um rosto sério, uma figura com os braços para cima e outra com os braços para baixo.

Cada um deles é definido usando uma matriz. Estas são as cinco figuras: coração [8]; feliz [8]; sério [8]; para baixo [8] e para cima [8]. Cada um dos caracteres contém uma descrição binária de quais pontos ou pixels devem ser ativados em cada caso.

As funções lcd.createChar () transferem o conteúdo das matrizes para a memória CGRAM da tela durante a configuração (). As funções transferem um total de 40 bytes, oito por caractere. Além disso, cada um dos personagens recebe um número entre 0 e 5:

```
lcd.createChar(0, heart);      //Create new character nº 0
lcd.createChar(1, happy);      //Create new character nº 1
lcd.createChar(2, serious);    //Create new character nº 2
lcd.createChar(3, down);       //Create new character nº 3
lcd.createChar(4, up);         //Create new character nº 4
```

Por último, você pode exibir cada caractere gráfico na tela, onde quer que o cursor esteja, como se fosse um caractere ASCII normal. Tudo o que você precisa fazer é usar a função lcd.write (n), sendo "n" o número atribuído a cada caractere, como no exemplo 0 a 5.

15. EXEMPLO 9: Exibir a Informação

Veja um exemplo que é realmente prático. Você criará um link entre o canal de comunicação serial em seu PC e a tela LCD. Naturalmente, o controlador NANO Arduino estará no meio.

O controlador funciona como uma ponte entre os dois. De um lado, ele está conectado ao PC através da porta USB; por outro lado, ele está conectado à tela da mesma maneira que a conectamos até agora. Usando um monitor serial, você enviará vários caracteres do PC usando o canal de comunicação serial. eles serão recebidos pelo controlador do Arduino e exibidos na tela LCD assim que chegarem.

16. EXEMPLO 10: Mostrar Números Inteiros

Você já viu como pode exibir todos os tipos de textos e caracteres na tela LCD e também obter diferentes tipos de efeitos de exibição. Você também pode exibir números bem claro.

Este exemplo vai exibir os números de 1 a 15 em seqüência toda vez que você apertar um botão conectado ao pino D2. Além disso, o número em questão será exibido como um número decimal, hexadecimal e binário.

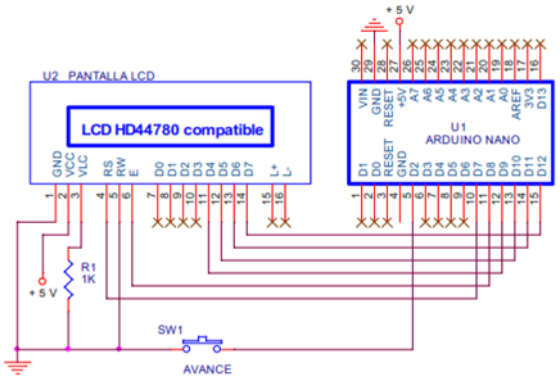

Figure 8

O programa não apresentará nenhuma dificuldade, dado o que você sabe agora. Veja um pequeno detalhe que será útil todo o tempo: como os números que você exibirá têm comprimentos diferentes, é uma boa ideia excluir a primeira linha da tela de cristal líquido em que você deseja exibi-los. Você pode fazer isso preenchendo a linha em questão com espaços em branco (“”); Dessa forma, não há "sobras”.

Dê uma olhada no programa. Experimente e tire as duas primeiras funções loop () e veja o que acontece: acho que você vai entender o que quero dizer com "sobras" agora.

17. EXEMPLO 11: Mostrar Números com Variações Pontuais

Na sequência do exemplo acima, vamos falar sobre a exibição de números de ponto flutuante. Nesse caso, exibiremos a raiz quadrada do número N com duas casas decimais e N multiplicado pela constante PI (3,1416) com quatro casas decimais. O número N avança seqüencialmente toda vez que você aperta o botão D2; vai de 0 a 9.

O diagrama de circuito e o conjunto são os mesmos que você usou para o exemplo anterior 10.

18. EXEMPLO 12: Menu

Uma tela de LCD em um periférico ideal para fazer uma interface de usuário intuitiva. E, de fato, existem menus de opções que permitem fazer uma seleção das diferentes tarefas a serem executadas.

Neste exemplo, a ideia é fazer um menu de seis opções que aparecem na tela uma após a outra. Ao usar um botão para avançar e outro para voltar, você poderá navegar pelo menu e observar todas as opções disponíveis. Veja como é o diagrama de circuito:

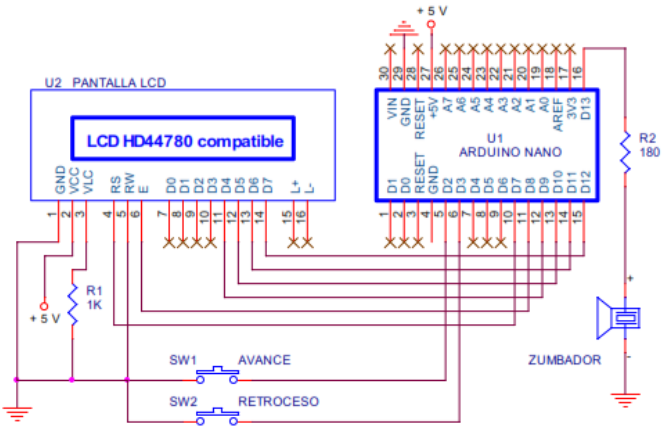

Figura 9

As conexões entre o controlador Arduino e a tela são as mesmas que você utilizou até agora. Nesse caso, há um segundo botão chamado SW2. O botão SW1 é conectado à entrada D2 e as opções disponíveis são mostradas na tela do primeiro ao último. Quando ligamos o botão SW2 ao pino D3, podemos ver as opções na ordem inversa: da última para a primeira. Ambos os pinos são configurados como resistores pull-up. Desta forma você economiza para ter que encaixar os dois resistores externos correspondentes.

A tela faz uma rolagem vertical para exibir todas as opções disponíveis no menu, uma após a outra. Também vamos usar uma sirene piezoelétrica conectada ao pino de saída D13. Quando o botão SW1 ou SW2 é pressionado, eles fazem um ruído.

Agora você vai dar uma olhada no programa. Vou deixar que você estude e examine e tente entender como funciona. Tudo o que vou dizer é que tentei encontrar a solução mais fácil e instrutiva possível; Tenho certeza de que não é o único nem o melhor; Espero que você possa melhorar muito usando sua própria iniciativa.

Comece:

Primeiro de tudo você inclui a biblioteca “LiquidCrystal.h” e conecta a tela; nada de novo lá. Em seguida, você declara as variáveis Option e N option. O primeiro contém a opção que está em uso no momento e o segundo, o número de opções disponíveis no menu. Você também tem que declarar uma matriz de ponta de flecha que define os caracteres gráficos.

Funções:

Duas funções foram criadas para este projeto: visualize () e beep (). O primeiro exibe a mensagem na tela que corresponde à opção em uso no momento com base no valor da variável “Opção”. E você já viu a função beep (): faz um sinal sonoro.

Setup():

Configure os pinos D2 e D3 como entradas com resistores pull-up internos, o pino D13 como saída, a tela com duas linhas de 16 caracteres e, finalmente, gere o caracter gráfico nº 0 (uma ponta de seta). Não há realmente nada importante aqui.

loop():

- começa limpando a tela e exibindo o caractere gráfico nº 0 (a ponta da flecha).
- localiza o cursor na primeira linha e exibe a opção em uso no momento com base na variável “option”. Ele usa a função visualise ().

- localiza o cursor na segunda linha e exibe a seguinte opção (Opção + 1). Mais uma vez, ele usa a função visualise ().
- aguarda até que alguém aperte os botões SW1 ou SW2 conectados a D2 e D3.
- se D2 for pressionado, elimina o debounce e aguarda até que o botão seja liberado. Ele faz um som de assobio e aumenta a variável "Option". Vai para a próxima opção no menu.
- se D3 for pressionado, elimina o debounce e aguarda até que o botão seja liberado. Ele faz um som de assobio e aumenta a variável "Option". Volta para a opção anterior no menu.

19. EXEMPLO 13: Selecionar opções

Considere este exemplo como uma extensão do anterior. Observe atentamente o circuito no diagrama representado na Figura 10.

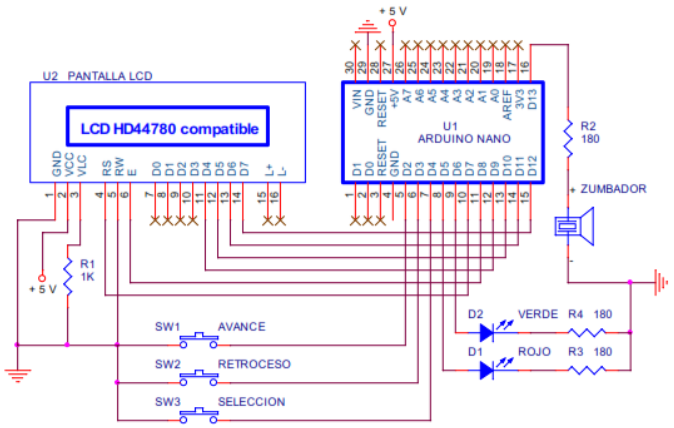

Figura 10

Adicionamos outro botão SW3 conectado ao D4 que executa a opção selecionada no momento. Adicionamos mais dois ecrãs LED: um vermelho e outro verde ligados a D5 e D6. Estes ecrãs mostram o resultado da execução da opção.

Este programa é muito semelhante ao do exemplo anterior. A única diferença consiste na função Execute() que corre cada vez que o botão SW3 é pressionado. Esta nova funçãoavalia o valor da variável “Option”. Dependendo da opção que tiver sido selecionada, a tarefa devida é executada e ativa ou desativa os ecrãs LED vermelho e verde ligados aos outputs D5 e D6.

20. EXEMPLO 14: Uma última melhoria

Imagine uma aplicação que não tenha apenas seis opções no menú, como tem sido hábito até ao momento, mas sim dezenas de opções no menú. Sempre que ligar o sistema ou o reiniciar ao pressionar RESET, o sistema inicia sempre na primeira opção; terá que fazer scroll down até encontrar a opção que pretende.

Talvez seja boa ideia dar uma olhadela ao ecrã LCD que apareceu na última opção que selecionou para que não tenha que o procurar sempre que reiniciar o sistema: o que reconhece? Mais uma vez, poderia utilizar a memória de dados EEPROM. É isso mesmo: cada vez que selecionar uma nova opção, ela fica registada na memória EEPROM. No final do dia, cada opção é representada apenas com um número guardado na variável “Option”.

Sempre que se reinicia o sistema, é atribuído a esta variável o último valor gravado na memória EEPROM. Vai utilizar a posição 1 das 1024 disponíveis.

Grave o programa e certifique-se que funciona. Selecione e execute as diversas opções. De seguida, desligue o sistema e ligue-o de novo. A última opção que selecionou aparece sempre.

REFERÊNCIAS

LIVROS

[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

SÍTIOS 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