Nota de Aula LCD

Os Displays LCD são muito úteis para quem pretende usar um Micro-controlador para desenvolver uma aplicação. Eles permitem uma interface visual entre homem e máquina (HMI em inglês), barata e simples de usar. No LCD você pode enviar textos, números, símbolos e até imagens que podem dar uma indicação do que o Micro-controlador esta fazendo, dos dados que podem estar sendo coletados ou transmitidos, etc. Para quem não sabe, LCD significa em inglês – Liquid Crystal Display – ou mostrador de cristal liquido. Uma grande vantagem dos LCDs é que não precisam de muita energia para funcionar. Lembram-se dos relógios digitais de LCD, há décadas atrás ? As pilhas podiam durar meses e até um ano. Mas os LCDs que estamos falando, tem LEDs atrás para que possam ser usados em ambientes com pouca ou nenhuma luz.

Para o uso de Micro-controladores como o Arduino, os displays LCD mais comuns são o 16×2 (16 caracteres x 2 linhas) ou 20×4 (20 caracteres x 4 linhas).

LCD 16×2 azul

LCD 16×2 verde

LCD 20×4 azul com I2C

LCD 20×4 verde

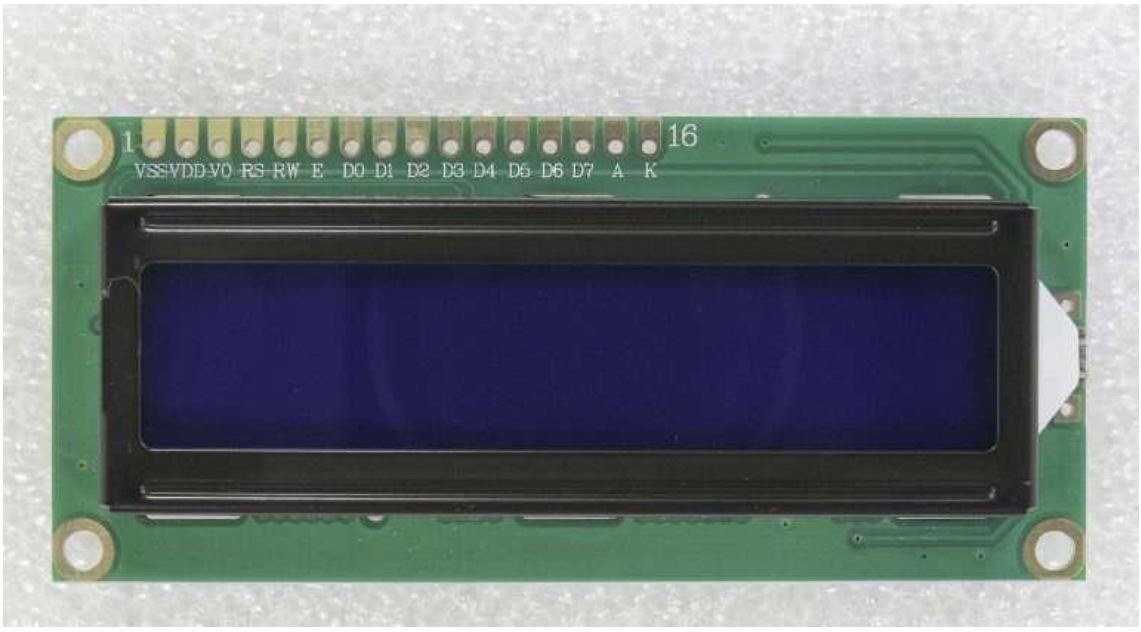


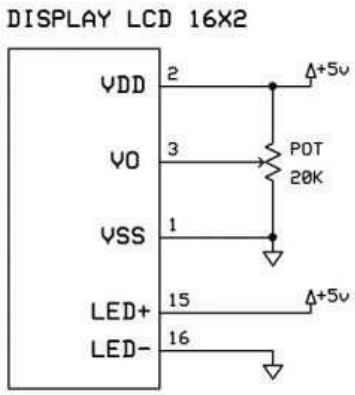
Figura 1 - LCD 16×2 azul

Na parte superior do Display, existem 16 furos onde podem ser soldados pinos ou cabos de comunicação e alimentação. Atenção : a ligação incorreta ou uso de tensões acima de 5V poderá danificar o display.

OBS: A pinagem do LCD 16×2 é idêntica aos outros LCDs

Pinagem e função de cada pino :
pino 1 – VSS – Pino de alimentação (zero volts – GND)
pino 2 – VDD – Pino de alimentação de +5V
pino 3 – VO – Pino de ajuste do contraste do LCD –
pino 4 – RS – Seleção de Comandos (nível 0) ou Dados (nível 1)
pino 5 – R/W – Read(leitura – nível 1) / Write (escrita – nível 0) aterrar!
pino 6 – E – Enable (Ativa o display com nível 1 ou Desativa com nível 0)
pino 7 – D0 – data bit 0 (usado na interface de 8 bits) em aberto
pino 8 – D1 – data bit 1 (usado na interface de 8 bits) em aberto
pino 9 – D2 – data bit 2 (usado na interface de 8 bits) em aberto
pino 10 – D3 – data bit 3 (usado na interface de 8 bits) em aberto
pino 11 – D4 – data bit 4 (usado na interface de 4 e 8 bits)
pino 12 – D5 – data bit 5 (usado na interface de 4 e 8 bits)
pino 13 – D6 – data bit 6 (usado na interface de 4 e 8 bits)
pino 14 – D7 – data bit 7 (usado na interface de 4 e 8 bits)
pino 15 – A – Anodo do LED de iluminação (+4,5V CC)
pino 16 – K – Catodo do LED de iluminação (GND)

Para ajuste do contraste do LCD, a tensão no pino 3 (VO) deve ser ajustada. Use um potenciômetro de 10K ohms. Nas extremidades do POT conecte o +5V e o GND. O pino central conecte no pino 3 do LCD.

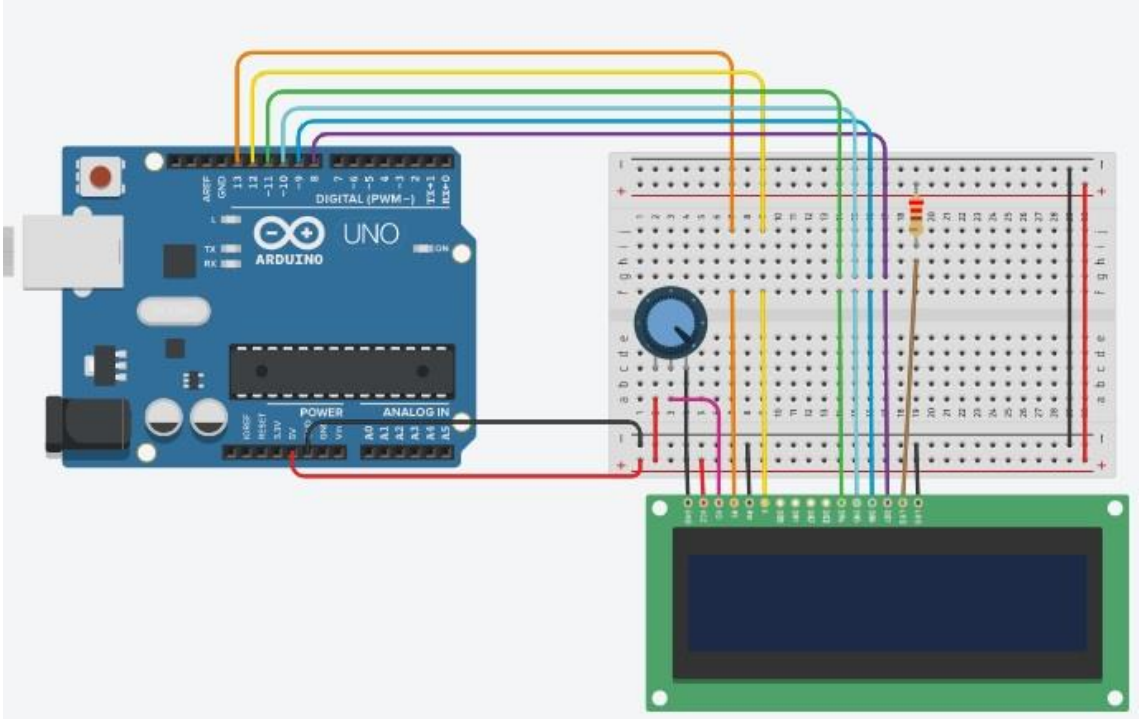


A iluminação de backlight do LCD é feita pelo LED. O pino 15 (Anodo do LED) pode ser conectado diretamente em +4,5 V e o pino 16 (catodo do LED) deve ser conectado no terra (GND). Dessa forma o LED deve ser ligado a um resistor de 220 ohms em 5V. Este resistir terá uma queda de tensão de 0,5V

Os pinos de controle 4 e 6 bem como os sinais de dados 11 a 14 devem ser ligados ao Arduino.

O pino 5 deve ser aterrado (ativado somente como escrita) e os pinos 7 a 10 devem ficar em aberto.

A figura a seguir apresenta como pode ficar a ligação do LCD no Arduino utilizando o tinkercad



Funcões da biblioteca LiquidCrystal.h

Comandos para o LCD		
Função	Sintaxe no programa	Descrição
LiquidCrystal()	LiquidCrystal(rs, enable, d4, d5, d6, d7)	Indica como os pinos do display estão interligados na placa Arduino
begin()	lcd.begin(colunas, linhas)	Configura o modelo do display em nosso caso 16×2
clear()	lcd.clear()	Limpa o display e posiciona o cursor no canto superior esquerdo do display
home()	lcd.home()	Posiciona o cursor no canto superior esquerdo do display
setCursor()	lcd.setCursor(coluna, linha)	Posiciona o cursor na linha e coluna indicada no comando
write()	lcd.write(caractere)	Escreve um caractere no display
print()	lcd.print(dado)	Escreve um dado qualquer no display (texto ou número)
cursor()	lcd.cursor()	Exibe o cursor na tela do LCD no formato de um undeline
noCursor()	lcd.noCursor()	Oculta o cursor
blink()	lcd.blink()	Faz com que o cursor fique piscante na tela do Arduino
noBlink()	noBlink()	Faz com que o display pare de piscar
noDisplay()	lcd.noDisplay()	Desliga o display sem perder os dados exibidos

display()	lcd.display()	liga o display e restaura o texto escrito após um comando “noDisplay()”
scrollDisplayLeft()	lcd.scrollDisplayLeft()	Move o conteúdo do display (texto+cursor) um espaço para a esquerda.
scrollDisplayRight()	lcd.scrollDisplayRight()	Move o conteúdo do display (texto+cursor) um espaço para a direita.
autoscroll()	lcd.autoscroll()	Ativa a rolagem automática do LCD. Isso faz com que cada caractere empurre os anteriores um espaço. Se a direção do texto atual é da esquerda para a direita (o padrão), o visor desloca para a esquerda; se a direção atual é da direita para a esquerda, o visor desloca para a direita.
noAutoscroll()	lcd.noAutoscroll()	Desliga o modo automático autoscroll.
leftToRight()	lcd.leftToRight()	Define a direção para o texto ser escrito no LCD da esquerda para a direita, o padrão. Isto significa que os caracteres subsequentes escritos para a exibição vai da esquerda para a direita, mas não afeta o texto enviado anteriormente para o display.
rightToLeft()	lcd.rightToLeft()	Define a direção para o texto ser escrito no LCD da direita para a esquerda. Isto significa que os caracteres subsequentes escritos para a exibição vai da direita para a esquerda, mas não afeta o texto enviado anteriormente para o display.
createChar()	lcd.createChar()	Cria um caractere personalizado (gylph) para uso no LCD. Até oito caracteres de 5×8 pixels são suportadas (numeradas de 0 a 7). O aparecimento de cada caracter personalizada é especificada por uma matriz de oito bytes, uma para cada linha. Os cinco bits menos significativos de cada byte de determina os pixels daquela linha. Para exibir um caractere personalizado na tela, use o comando lcd.write(byte(numero));

Software de Teste do LCD

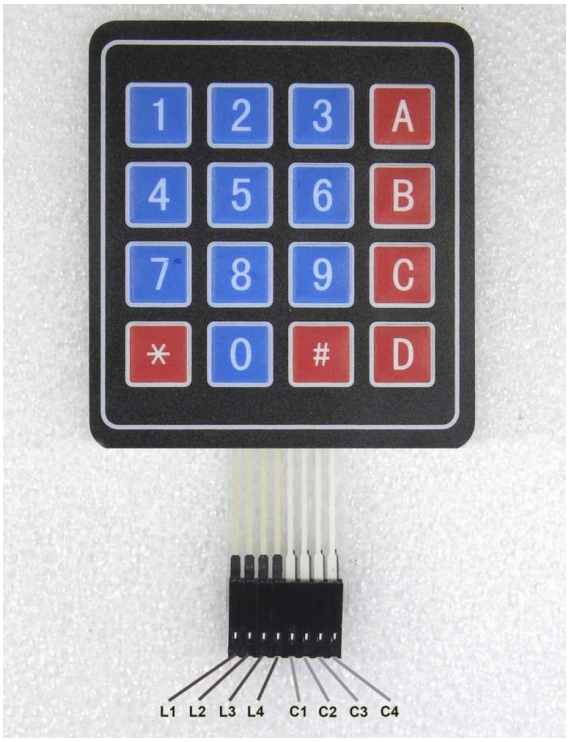
```
/*
  Teste LCD
* LCD RS pin to digital pin 13
 * LCD Enable pin to digital pin 12
 * LCD D4 pin to digital pin 11
 * LCD D5 pin to digital pin 10
 * LCD D6 pin to digital pin 9
 * LCD D7 pin to digital pin 8
```

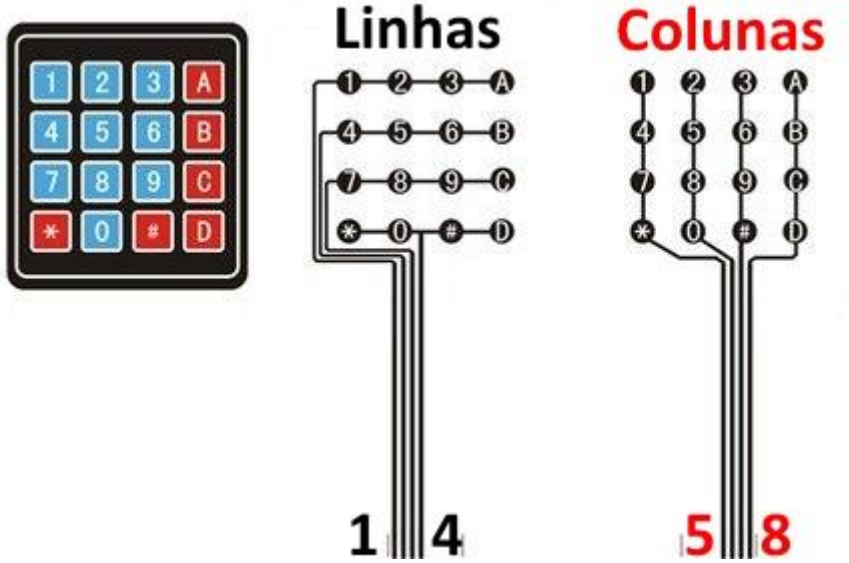
```
 * LCD R/W pin to ground
 * LCD VSS pin to ground
 * LCD VCC pin to 5V
 * 10K POT: +5V and ground
 * Central Pin LCD VO pin (pin 3)

 http://www.arduino.cc/en/Tutorial/LiquidCrystal
 */

// Inclusao bibliotecas
#include <LiquidCrystal.h>

// Configura LCD com os pinos
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

void setup() {
  // Inicializa LCD com numero de colunas e linhas do LCD:
  lcd.begin(16, 2);
//Posicionar cursor para centralizar mensagem
  lcd.setCursor(2, 0);

  // Enviar msg no LCD
  lcd.print("Jorge Street");
}

void loop() {
  // Colocar cursor na segunda linha
  // 1ª linha – linha 0, 2ª linha – linha 1
  lcd.setCursor(0, 1);
  // Coloca o numero de segundos após o Reset
  lcd.print(millis() / 1000);
}
```

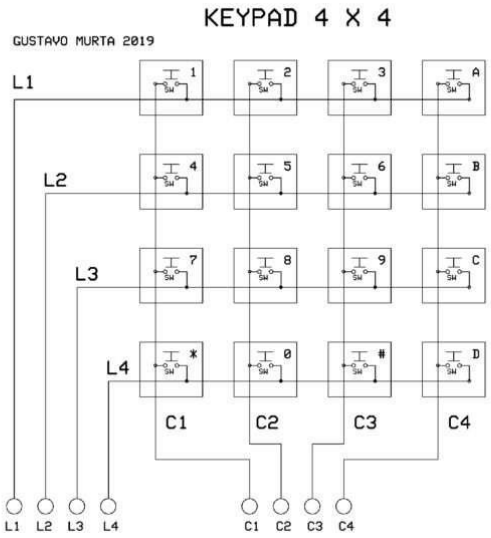
Nota de Aula – Teclado

Abaixo, segue a pinagem do Teclado Matricial. O teclado possui 16 teclas push button e suas vias são separadas por linhas (1 a 4) e colunas (5 a 8). O teclado funciona da seguinte forma:

Ao pressionar por exemplo a tecla 1, serão interligadas as vias 1 (linha1) e 5 (coluna1) e ao pressionar a tecla C, conectaremos a via 3 (linha 3) com a via 8 (coluna 4).

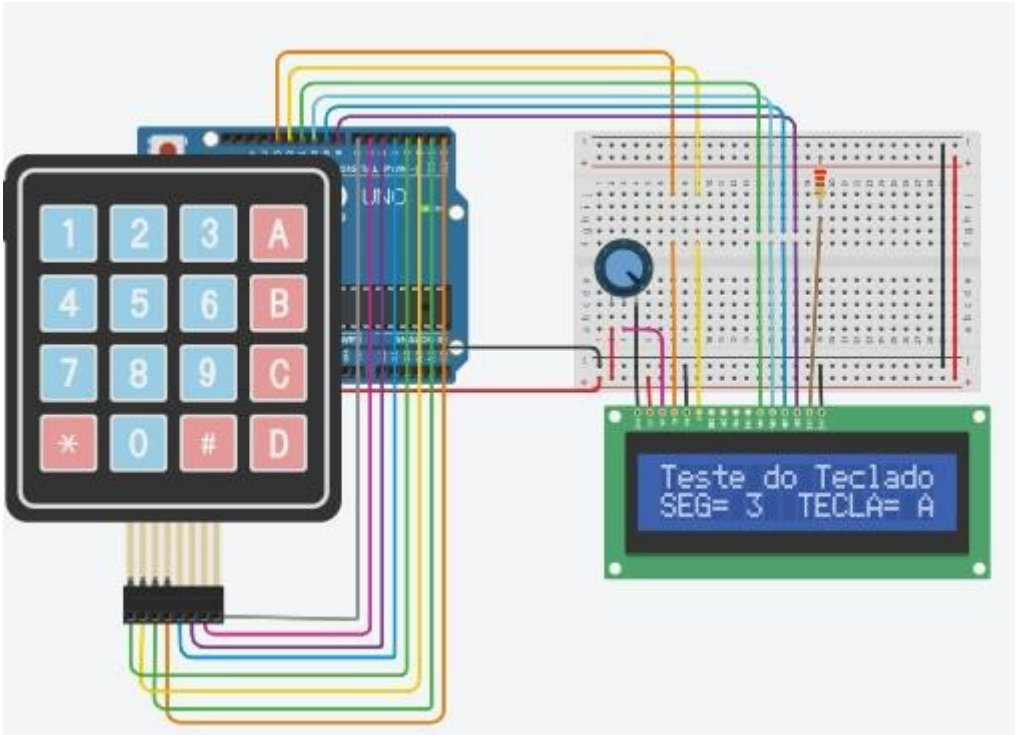






Abaixo segue um exemplo do ligação do teclado ao Arduino

LCD e Teclado no TinkerCad



O programa a seguir faz o teste de ligação do teclado e mostra como configurar o teclado no Arduino utilizando a bliblioteca Keypad.h

Software de Teste

```
/*
  LCD
* LCD RS pin to digital pin 13
 * LCD Enable pin to digital pin 12
 * LCD D4 pin to digital pin 11
 * LCD D5 pin to digital pin 10
 * LCD D6 pin to digital pin 9
 * LCD D7 pin to digital pin 8
 * LCD R/W pin to ground
 * LCD VSS pin to ground
 * LCD VCC pin to 5V
 * 10K POT: +5V and ground
 * Central Pin LCD VO pin (pin 3)

 http://www.arduino.cc/en/Tutorial/LiquidCrystal

Teclado
   * Linha1 pin  to digital pin 3
   * Linha2 pin  to digital pin 2
   * Linha3 pin  to digital pin 1
   * Linha4 pin  to digital pin 0
   * Coluna1  pin to digital pin 4
   * Coluna2  pin to digital pin 5
   * Coluna3  pin to digital pin 6
   * Coluna4  pin to digital pin 7

 */


// Inclusao bibliotecas
#include <LiquidCrystal.h>
#include <Keypad.h>

// Configura LCD com os pinos
LiquidCrystal lcd(13, 12, 11, 10, 9, 8);

  // teclado 4x4
const byte ROWS = 4; // 4 linhas
const byte COLS = 4; // 4 colunas
  // Define mapeamento do teclado

  char keys[ROWS][COLS] =
{
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};
```

```
  // Conectar linhas aos pinos do Arduino
 byte rowPins[ROWS] = { 3, 2, 1, 0 };

  // Conectar colunas aos pinos do Arduino
byte colPins[COLS] = { 4, 5, 6, 7 };

  // Criar o teclado
Keypad kpd = Keypad( makeKeymap(keys), rowPins,  colPins, ROWS, COLS );

void setup() {
  // Inicializa LCD com numero de colunas e linhas:
  lcd.begin(16, 2);
  // Enviar msg no LCD
  lcd.print("Teste do Teclado"); }

void loop()
  {
 // le teclado
    char tecla   = kpd.getKey();
  // posiciona cursor
  lcd.setCursor(0, 1);
    // coloca segundos desde o reset:
  lcd.print("SEG= ");
  lcd.print(millis() / 1000);
    // Verifica se há tecla válida
  if(tecla)  // if (tecla != NO_KEY)
   {
      //Coloca cursor na coluna 8 linha 1
       lcd.setCursor(8, 1);
     //coloca tecla no LCD
       lcd.print("TECLA= ");
      lcd.print(tecla);
    }
  }
```