

A ligação dos módulos é bem simples. Cabem algumas observações:

Este é o original:

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int umidade;

void setup()
{
  lcd.begin(16, 2);
  pinMode(13, OUTPUT);
}

void loop()
{
  digitalWrite(13, LOW);
  lcd.setCursor(0, 0);
  umidade = analogRead(A0);
  umidade = map(umidade, 1023, 0, 0, 200);
  lcd.print("Umidade em:");
  lcd.print(umidade);
  lcd.print(" %");
  if(umidade < 20)
  {
    digitalWrite(13, HIGH);
    delay(10000);
    //delay(300000);
  }
  delay(500);
  lcd.clear();
}
```

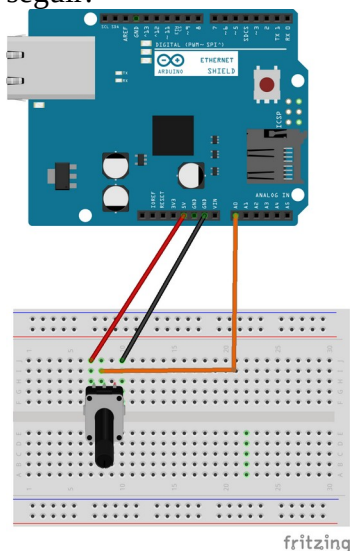
Com o *display* nós já trabalhamos. Para entender a leitura analógica e o comando ‘map’, vejam o que segue.

```

1 //original - projeto OPET água 2
2
3 #include <LiquidCrystal.h> //inclui biblioteca de funções de controle do display LCD
4
5 /*
6 LiquidCrystal, a seguir, é o nome da biblioteca; lcd é o nome da variável
7 Ao criarmos uma variável correspondente à biblioteca LiquidCrystal, em função do tipo de linguagem que está sendo utilizado,
8 teremos um OBJETO na memória do Arduino: e este objeto é do tipo display de cristal líquido.
9 Ao trabalharmos com um objeto, ele terá propriedades e funções associadas. As propriedades geralmente são identificadas como
10 atributos, e as funções como métodos. Métodos (ou funções) são os que realizam ações, por exemplo, limpar o display (apagar o
11 que está escrito nele).
12 Na linha que segue, então, temos a declaração de que 'lcd' será um objeto do tipo capaz de controlar o display, mas, há mais um
13 detalhe. Entre parênteses estão sendo passados PARÂMETROS para o objeto. POR meio deles, ele será capaz de interagir com a
14 plaquinha do display. Os parâmetros são: 12, 11, 5, 4, 3, 2. Estes números indicam quais são os pinos que estão sendo utilizados
15 para controle do display. Eles tem um significado para a biblioteca de controle (e variam de biblioteca para biblioteca e de
16 controlador para controlador), que é relativo à sequência de sinais de comando e controle. São eles: RS, ENABLE, D4, D5, D6, D7
17 Seu significado:
18 RS = é utilizado para seleção de comandos ao display (quando colocado em nível 0), ou para indicar os dados, ou seja,
19 os caracteres que devem ser escritos (quando colocado em nível 1); a biblioteca faz isso automaticamente.
20 ENABLE = habilita o display quando em nível 1 ou desabilita quando em nível 0
21 D4-D7 = sinais de dados, sendo D4 o bit menos significativo em relação a D7. Aqui cabem duas observações: a primeira é a de que
22 são usados 8 bits para formar um caractere ou um sinal de controle (D0, D1, D2, D3, D4, D5, D6, D7). Mas, para usá-los teríamos
23 que utilizar 8 pinos de comunicação (mais 3 para os sinais anteriores - além dos dois para +5V e GND). Então, para
24 'economizar conexões' a biblioteca faz um pequeno ajuste na transmissão de dados, enviando 4 bits de cada vez - envia D0 a D3
25 usando as linhas D4 a D7, e, depois, envia D4 a D7 enviando as linhas D4-D7. Portanto, são dois envios para formar um caractere
26 de exibição ou de controle: gasta-se tempo e economiza-se conexões. O que nos leva à segunda observação: em nosso projeto do
27 CyberPower o display utilizado possui uma plaquinha a mais soldada em seus pinos, que converte as linhas de controle e de
28 comunicação de dados para uma sequência; os dados que trafegam em paralelo, todos ao mesmo tempo, passam a trafegar de forma
29 serializada, um após o outro. Com isso, no CyberPower usamos somente 4 conexões com o Arduino: o +5V, o GND e um sinal de relógio
30 (ou sincronismo) e um de dados. Depois podem olhar o CyberPower para ver a diferença...
31 */
32 LiquidCrystal lcd(12, 11, 5, 4, 3, 2); //cria uma instância de controle do display
33
34 int umidade; //variável inteira para armazenar o valor da umidade
35
36 void setup()
37 {
38   lcd.begin(16, 2); //inicia o display como sendo de 16 colunas e 2 linhas
39   pinMode(13, OUTPUT); //inicia o pino 13 como sendo de saída
40 }
41
42 void loop()
43 {
44   digitalWrite(13, LOW); //escreve no pino 13 um sinal baixo (LOW, 0) o que foi feito para desligar o relé conectado a ele
45   lcd.setCursor(0, 0); //posiciona o cursor na posição coluna 0, linha 0 (canto superior esquerdo do display)
46   umidade = analogRead(A0); //faz a leitura de um sinal analógico na porta A0
47   //notem que não foi declarado um pinMode(A0, INPUT), que seria o mais correto
48   //mas, o Arduino consegue efetuar a leitura e armazenar o valor na variável 'umidade'
49   umidade = map(umidade, 1023, 0, 0, 200); //este comando é interessante, ele transforma um conjunto de valores em outro
50   //Ou seja, map(umidade, 1023, 0, 0, 200), vai usar o valor lido e armazenado na
51   //variável umidade, com valores entre 1023 e 0, para valores entre 0 e 200 e
52   //o resultado será colocado na própria variável umidade
53   lcd.print("Umidade em:"); //este comando imprime na posição atual do cursor a mensagem 'Umidade em:'
54   lcd.print(umidade); //este comando imprime na posição atual do cursor o valor da variável umidade
55   lcd.print(" %"); //este comando imprime na posição atual do cursor a mensagem ' %'
56   if(umidade < 20) //testa para saber se o valor da variável umidade é menor do que 20, se for executa o bloco seguinte
57   {
58     digitalWrite(13, HIGH); //escreve um valor alto (HIGH, 1) na porta 13 - o que deverá ligar o relé associado
59     delay(10000); // aguarda 10000 milissegundos, ou 10 segundos
60     //delay(300000);
61   }
62   delay(500); //aguarda 500 milissegundos ou 0,5s
63   lcd.clear(); //limpa o display
64 }

```

Para compreenderem melhor a leitura analógica e o comando ‘map’, podem montar o circuito a seguir:



E, após, testar os códigos que seguem.

1 – Somente o potenciômetro e a leitura dos valores:

```
1 void setup() {  
2   Serial.begin(9600);  
3   pinMode(A0, INPUT);  
4 }  
5  
6 void loop() {  
7   int valor = analogRead(A0);  
8  
9   Serial.println(valor);  
10  delay(1500);  
11 }
```

Explicação:

- Na função 'setup()', a linha 2 cria uma instância da interface serial com 9600bps de velocidade; a linha 3 informa que a porta 'A0' será uma entrada ('INPUT').
- Na função 'loop()', na linha 7 é declarada uma variável inteira ('int') chamada de 'valor', a qual recebe o valor que for lido na porta 'A0', o que é realizado pelo comando 'analogRead(A0)'. A linha 9 solicita que seja escrito o conteúdo da variável 'valor', na interface serial, pulando uma linha após ('Serial.println'). Os valores estarão entre 0 (mínimo) e 1023 (máximo), dependendo da posição do potenciômetro.

Compile o programa, ligue o monitor serial (Ferramentas/ Monitor Serial) e confira.

2 – O código de leitura e o 'mapeamento' para um intervalo de valores:

```
1 void setup() {  
2   Serial.begin(9600);  
3   pinMode(A0, INPUT);  
4 }  
5  
6 void loop() {  
7   int valor = analogRead(A0);  
8   valor = map(valor, 0, 1023, 0, 100);  
9   Serial.println(valor);  
10  delay(1500);  
11 }
```

Explicação:

- Na função 'setup()', a linha 2 cria uma instância da interface serial com 9600bps de velocidade; a linha 3 informa que a porta 'A0' será uma entrada ('INPUT').
- Na função 'loop()', na linha 7 é declarada uma variável inteira ('int') chamada de 'valor', a qual recebe o valor que for lido na porta 'A0', o que é realizado pelo comando 'analogRead(A0)'. A linha 8 faz com que a variável 'valor' tenha seu conteúdo ajustado: a função 'map' funciona lendo um valor e, sabendo seu mínimo e seu máximo, ajustando-o para um novo intervalo mínimo e máximo; em nosso exemplo, o conteúdo da variável 'valor' será ajustado de 0-1023 para 0-100 (no projeto de vocês, pelo código, o ajuste é para 0-200, vocês podem testar outros valores, por exemplo entre 30 e 90,..., incluindo negativos). A linha 9 solicita que seja escrito o conteúdo da variável 'valor', na interface serial, pulando uma linha após ('Serial.println'). Os valores, agora, estarão entre 0 (mínimo) e 100 (máximo), dependendo da posição do potenciômetro.

Compile o programa, ligue o monitor serial (Ferramentas/ Monitor Serial) e confira.