BOSCH
Invented for life

Aula 4 - Leitura de sensores

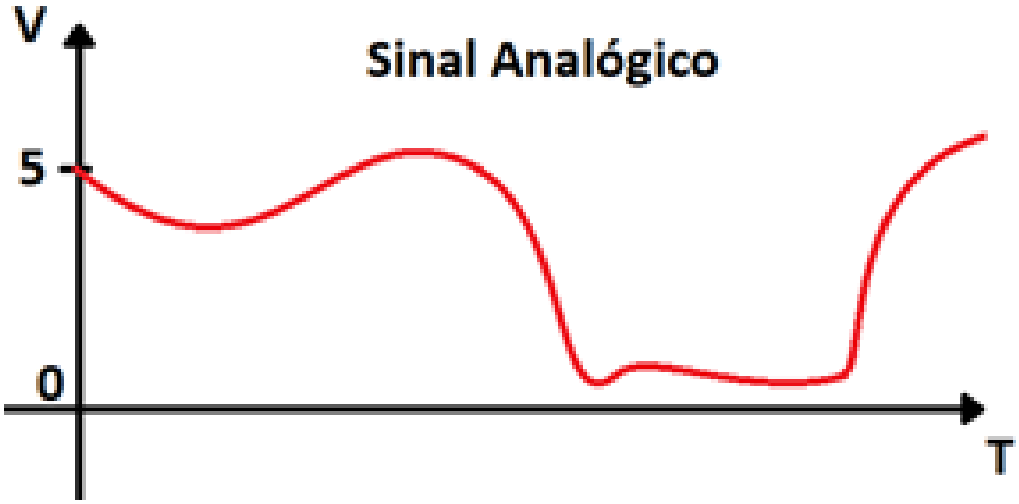
Docupedia Export

Author: Siqueira Joao (CtP/ETS)
Date: 02-Mar-2023 13:48

Table of Contents

1 Entradas Analógicas

São aquelas que, ao contrário das grandezas digitais, variam continuamente dentro de uma faixa de valores. O velocímetro de um carro, por exemplo, pode ser considerado analógico, pois o ponteiro gira continuamente conforme o automóvel acelera ou trava. Se o ponteiro girasse em saltos, o velocímetro seria considerado digital.



Como tudo na ESP é processado de forma digital, é necessário converter as grandezas analógicas em digitais e vice-versa. Esses conversores já estão embutidos na ESP, de forma que é necessário apenas que compreenda o básico do processo de conversão para poder utilizar essas portas analógicas.

ADC – Portas Analógicas ESP32

A ESP possui um conversor analógico-digital de 12 bits para 18 pinos, em que as tensões de entrada entre 0 e 5 volts aplicadas ao pino serão mapeadas em valores inteiros entre 0 e 4096 (2^{12}). Essa resolução é devida ao conversor analógico-digital (ADC) utilizado na placa da ESP.

Portanto, cada ADC pode medir 4096 níveis de tensão e cada magnitude de nível de tensão pode ser determinado com a seguinte fórmula:

$$Var = Vref / total$$
$$Var = 5 / 4096 = 0.0012207 = 1,22mV$$

Resumindo, para cada 1,22mV a ESP irá detectar uma variação, sendo assim irá aumentar ou diminuir o valor convertido.

Exemplo:

A cada aumento de 1,22mV a ESP com um sensor de temperatura irá aumentar 1°C.

2 Sensores Analógicos

São sensores cuja saída pode assumir qualquer valor entre um máximo e um mínimo (entre 0V e 5V, no caso de sensores para o Arduíno e Esp).

Exemplo: potenciômetros, sensores de temperatura, de gás, de luz e de distância.

Trabalhar com qualquer pino ADC será exatamente igual ao procedimento utilizado em placas Arduino, realizando leituras e controle por meio de funções como **analogRead** e **analogWrite**. Porém, ao conectar-se ao Wi-Fi, por exemplo, os pinos ADC2 não irão fornecer leituras, pois, até o presente momento, tais pinos serão utilizados como alimentação para o módulo, portanto, se desejar atribuir uma leitura analógica a qualquer programação conectada a internet, utilize os pinos ADC1.

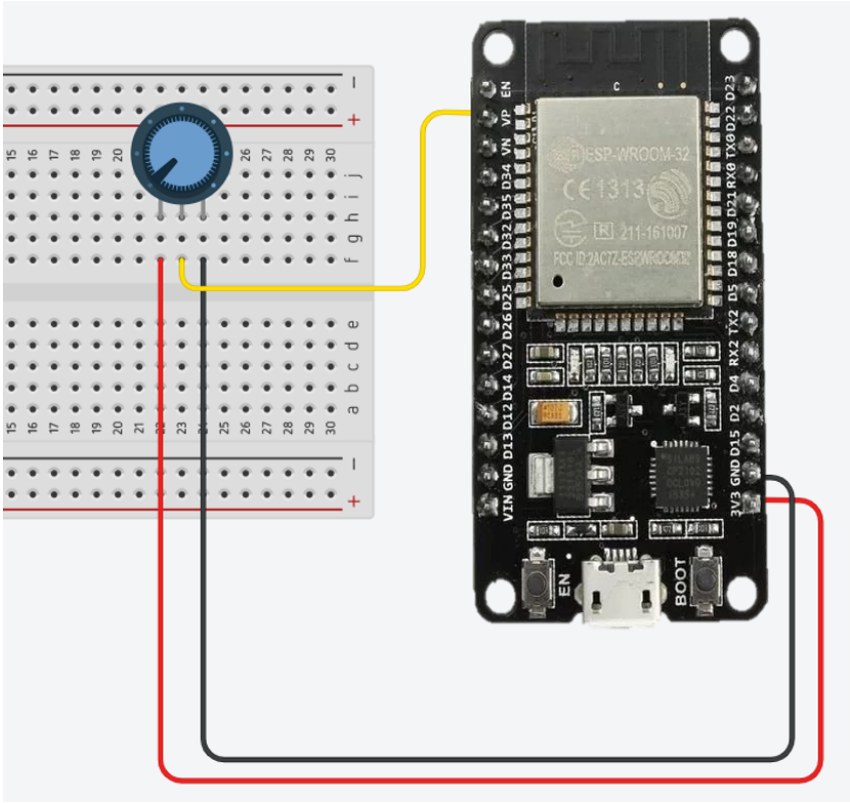
2.1 AnalogRead()

A função AnalogRead() é usada para medir a tensão entre 0 a 5 Volts e converte em um valor digital entre 0 a 4096. Por exemplo, se aplicarmos 0 Votlts no pina ADC o AnalogRead irá retornar para nós o valor digital Zero. Da mesma forma se aplicarmos 5 Volts no pino ADC, teremos uma saída no AnalogRead() de 4096 valores digitais.

2.1.1 Exemplo: Um potenciômetro tem seu terminal central como analógico, em que irá variar de 0 a 4096. Para visualizar isso vamos utilizar o AnalogRead().

AnalogRead

```
#define potPin 36 //Pino analógico no GPIO 36
int valPot=0;
void setup() {
}

void loop() {
  valPot=analogRead(potPin);
}
```



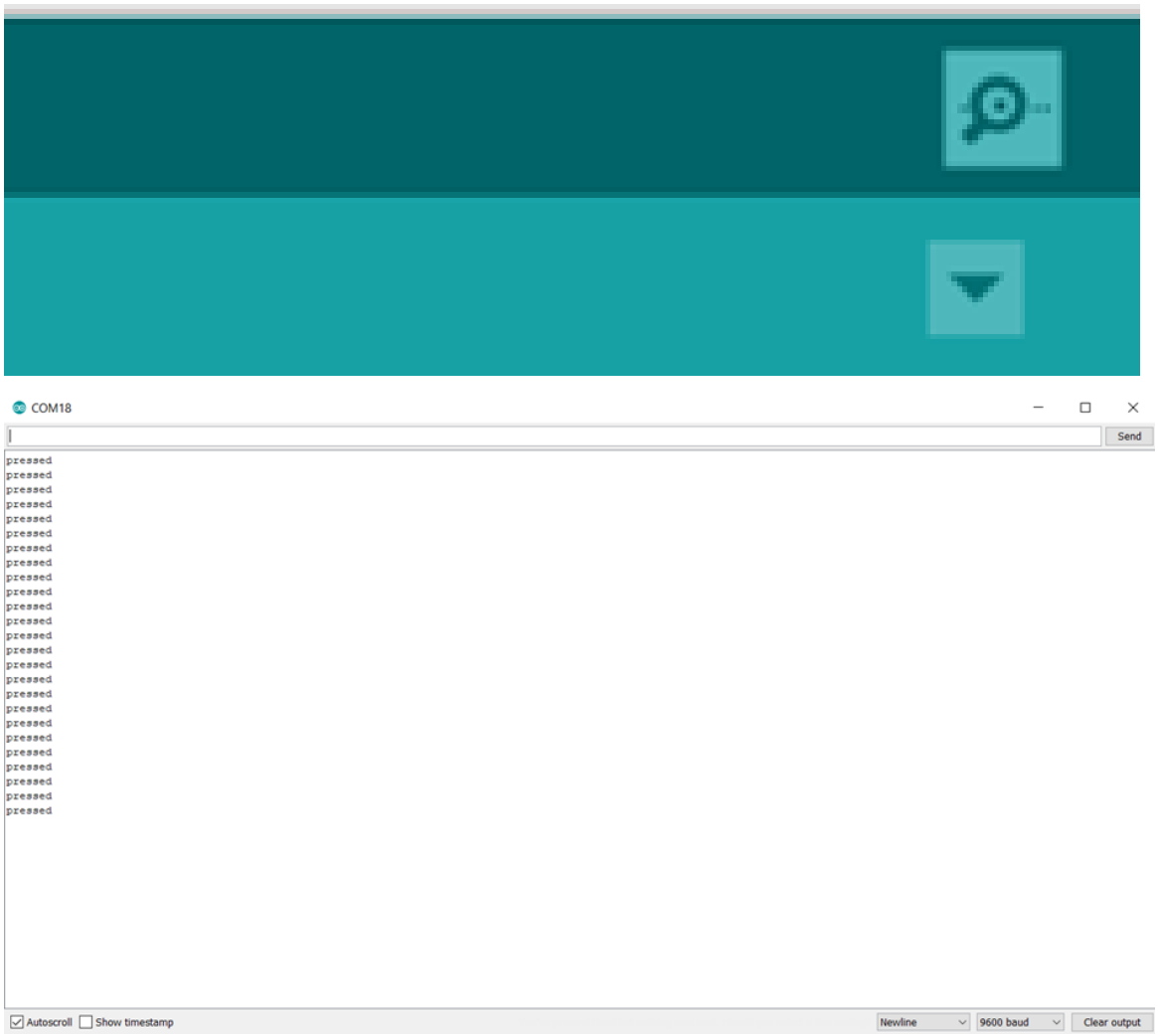
3 Como visualizar os dados?

Vamos visualizar os dados pelo terminal Serial na interface do ARDUINO.

3.1 Serial Monitor

Faz parte do Software Arduino IDE. Seu trabalho é permitir que você envie mensagens de seu computador para uma placa via USB e também receba mensagens da placa.

O Monitor Serial fica no canto superior direito da IDE. Dentro do monitor serial, no canto inferior direito é possível verificar o **Baud Rate**, em que representa a velocidade de comunicação serial.



Para apresentar os dados no serial monitor é necessário declararmos o inicio dele na programação, em que o parâmetro será o Baud Rate para definirmos a taxa de dados por segundo para transmissão de dados seriais. Para fazer essa comunicação com o seu monitor serial certifique-se que você está declarando uma velocidade válida. O valor Baud Rate que você colocar no comando Serial.Begin() deve ser o mesmo que está selecionado no Monitor Serial.

Serial.begin()

```
void setup(){
    Serial.begin(9600);
}
```

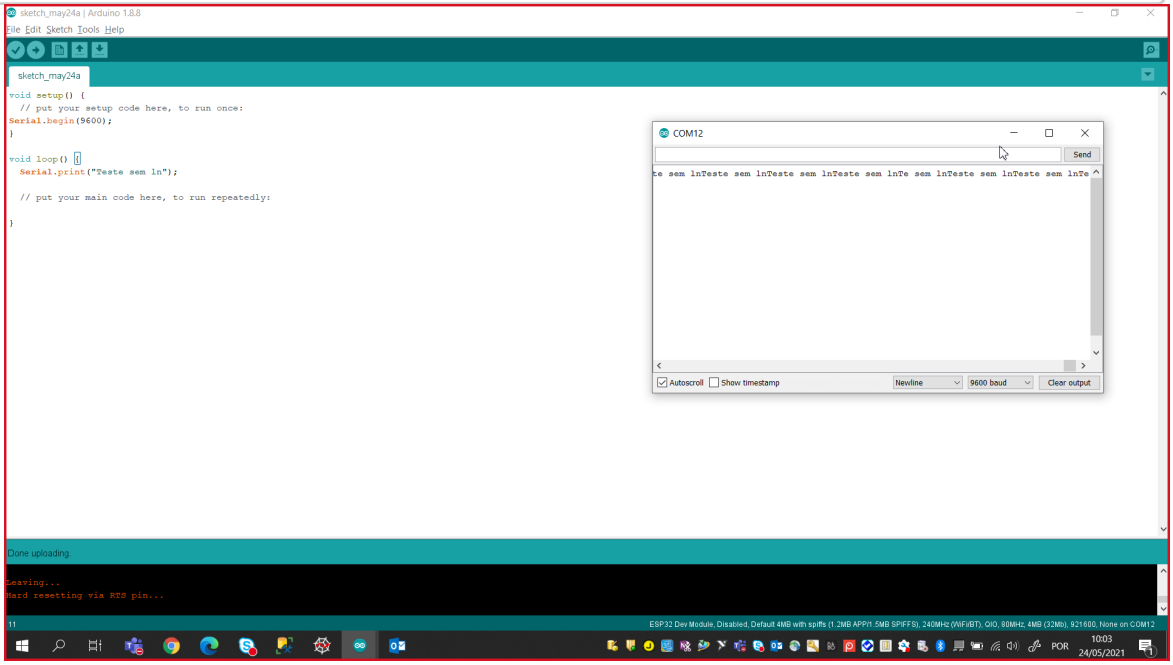
O segundo passo é a definição do que vai ser apresentado no monitor.

Temos duas opções:

1. Serial.print(frase); - as frases sempre vão estar uma ao lado da outra.
2. Serial.println(frase); - as frases sempre estarão uma embaixo da outra (pulando sempre uma linha).

Serial.print()

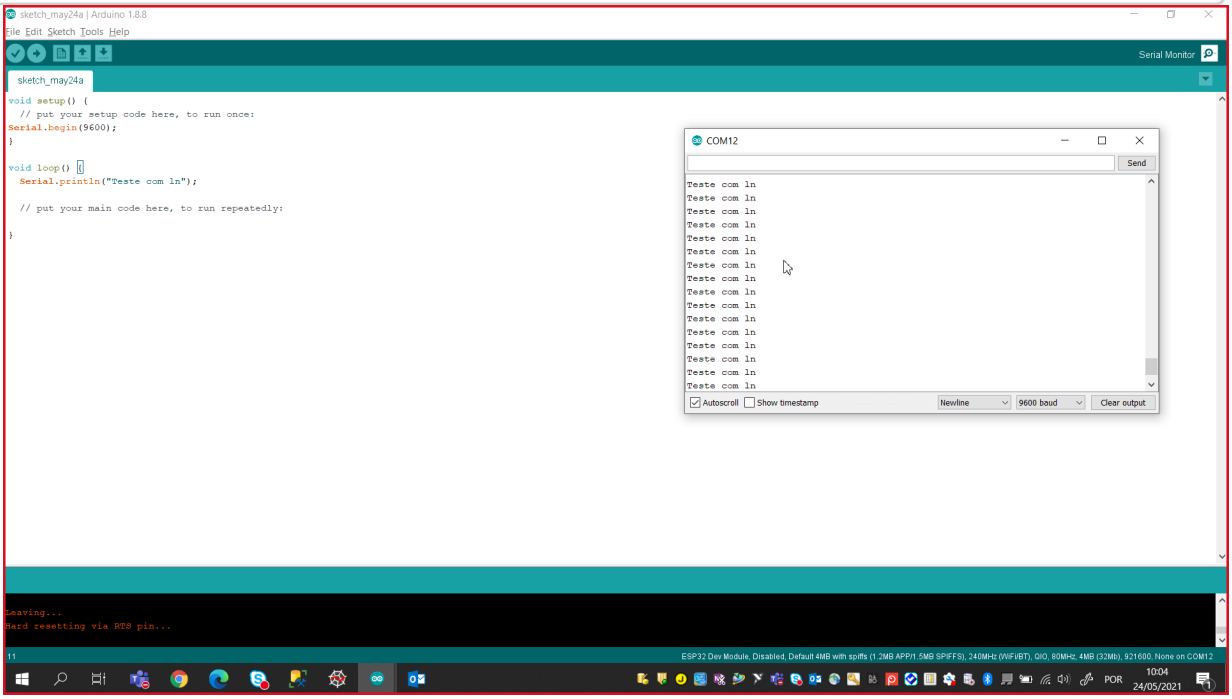
```
void setup(){
    Serial.begin(9600);
}
void loop(){
    Serial.print("Teste sem ln");
}
```



Serial.print()

```
void setup(){
    Serial.begin(9600);
}
void loop(){
```

```
    Serial.println("Teste com ln");
}
```



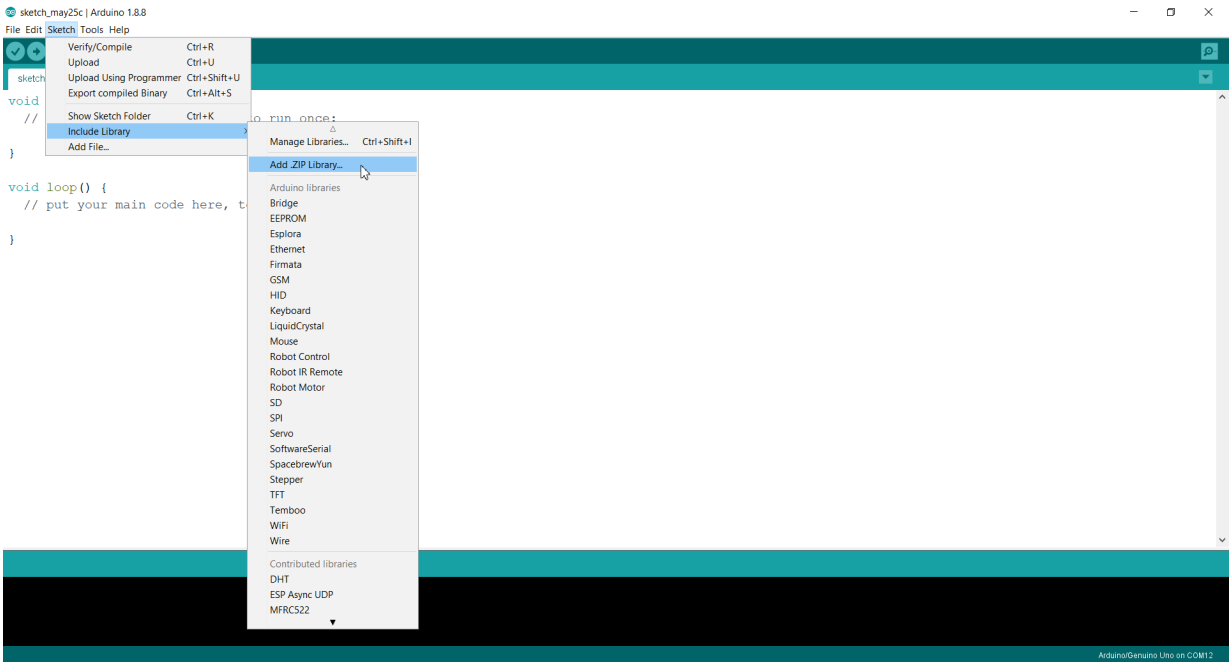
3.1.1 Voltando ao exemplo anterior...

AnalogRead+SerialMonitor

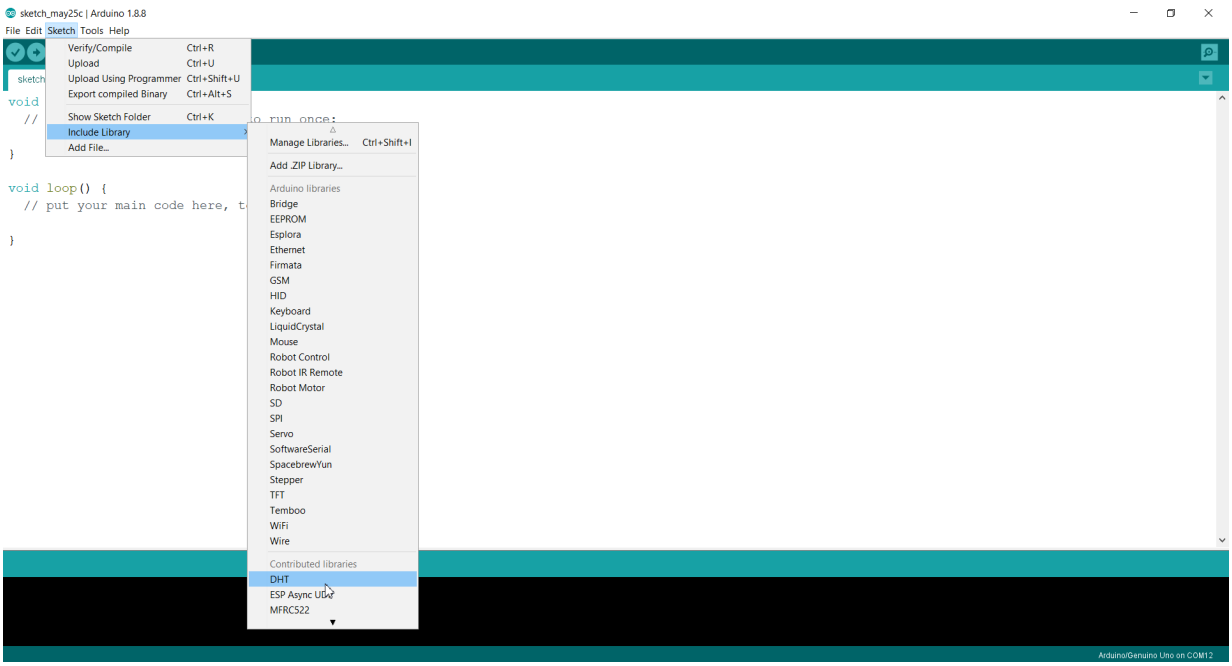
```
#define potPin 36
int valPot=0;
void setup() {
Serial.begin(9600);
}

void loop() {
  valPot=analogRead(potPin);
  Serial.print("Valor original: ");
  Serial.println(valPot);
  delay(1000); //Espera um tempo para continuar o Loop
}
```

3.2 Sensor de temperatura e umidade

O DHT11 é um sensor de temperatura e umidade que permite fazer leituras de tempraturas entre 0 e 50 Celsius e umidade entre 20 a 90%, muito usado em projetos de Arduino e ESP. Para leitura da conversão dele iremos usar a biblioteca 'DHT.zip' ou 'DHT-sensor-library-master.zip', para utilizar esse zip siga os passos abaixo:



Selecione 'Add .ZIP Library' e escolha o arquivo DHT.zip. Assim ele estará incluso entre as suas bibliotecas.




```
#include <dht.h>
```

Para a leitura do sensor vamos utilizar o seguinte programa.

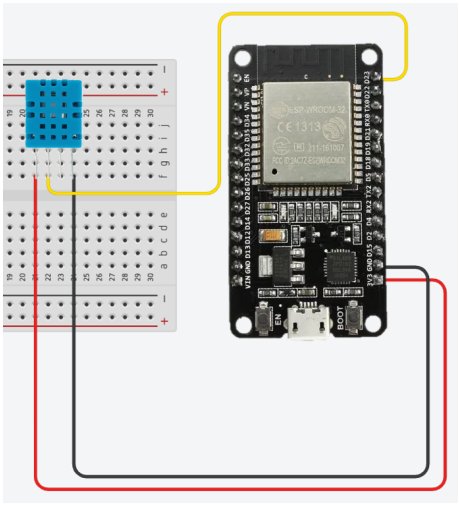
Temperatura e umidade

```
#include "dht.h"
const int pino=23;
dht DHT;
```

```
void setup() {
  Serial.begin(9600);
  delay(200);

}

void loop() {
  DHT.read11(pino);
  Serial.print("Umidade: ");
  Serial.println(DHT.humidity);
  Serial.print("Temperatura:");
  Serial.println(DHT.temperature, 0);
  delay(1000);

}
```

ou

```
#include "DHT.h"
#define DHTPIN 4
//our sensor is DHT
//#define DHTPIN 4
//our sensor is DHT11 type
#define DHTTYPE DHT11
//create an instance of DHT sensor
DHT dht(DHTPIN, DHTTYPE);11 type
#define DHTTYPE DHT11
//create an instance of DHT sensor
DHT dht(DHTPIN, DHTTYPE);
void setup()
{
  Serial.begin(115200);
  dht.begin();
}
void loop()
{
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
  else {
    Serial.print("Umidade: ");
    Serial.println(h);
    Serial.print("Temperatura: ");
    Serial.println(t);
  }
  delay(100);
}
```

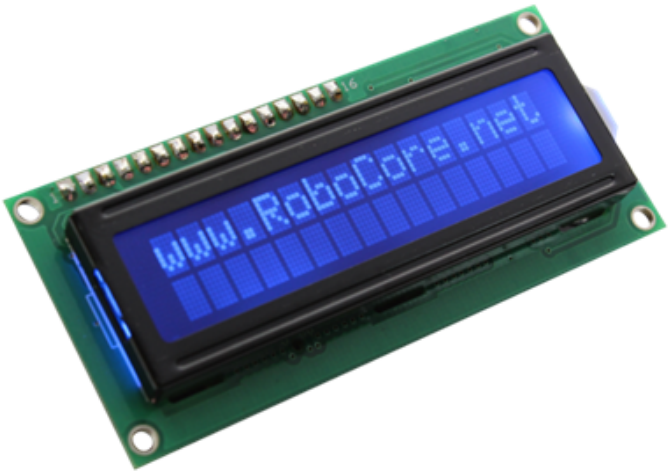


3.3 Liquid Crystal Display

Displays LCD (*Liquid Crystal Display)* são componentes que, assim como o display sete segmentos, possibilitam uma interação visual entre o homem e a máquina. Eles são muito utilizados em diversas áreas, inclusive usados para a fabricação de computadores, laptops, televisões, etc.

3.3.1 Conceito e estrutura

Os LCD's para Arduino são pouco mais limitados que os utilizados em grandes tecnologias, porém, permitem a utilização de números, letras, e até símbolos customizáveis, o que permite a impressão fácil de informações vindas de sensores, por exemplo. Os mais comuns são os de 16x2 e 20x4.



3.3.2 Conexão do LCD

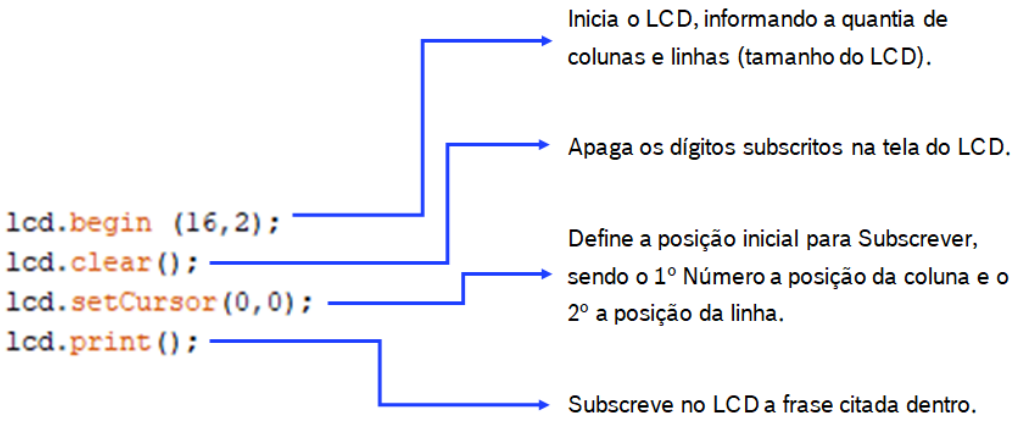
3.3.3 Programação do LCD no Arduino IDE

Para controle e configuração do LCD através do Arduino, basta importar a biblioteca "*LiquidCrystal*" e declarar os pinos usados para a conexão do mesmo antes do "*void setup*".

include

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(19,23,18,17,16,15);
void setup(){
    lcd.begin(16,2);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print();
}
```

No *"Void setup"* precisamos declarar algumas configurações:



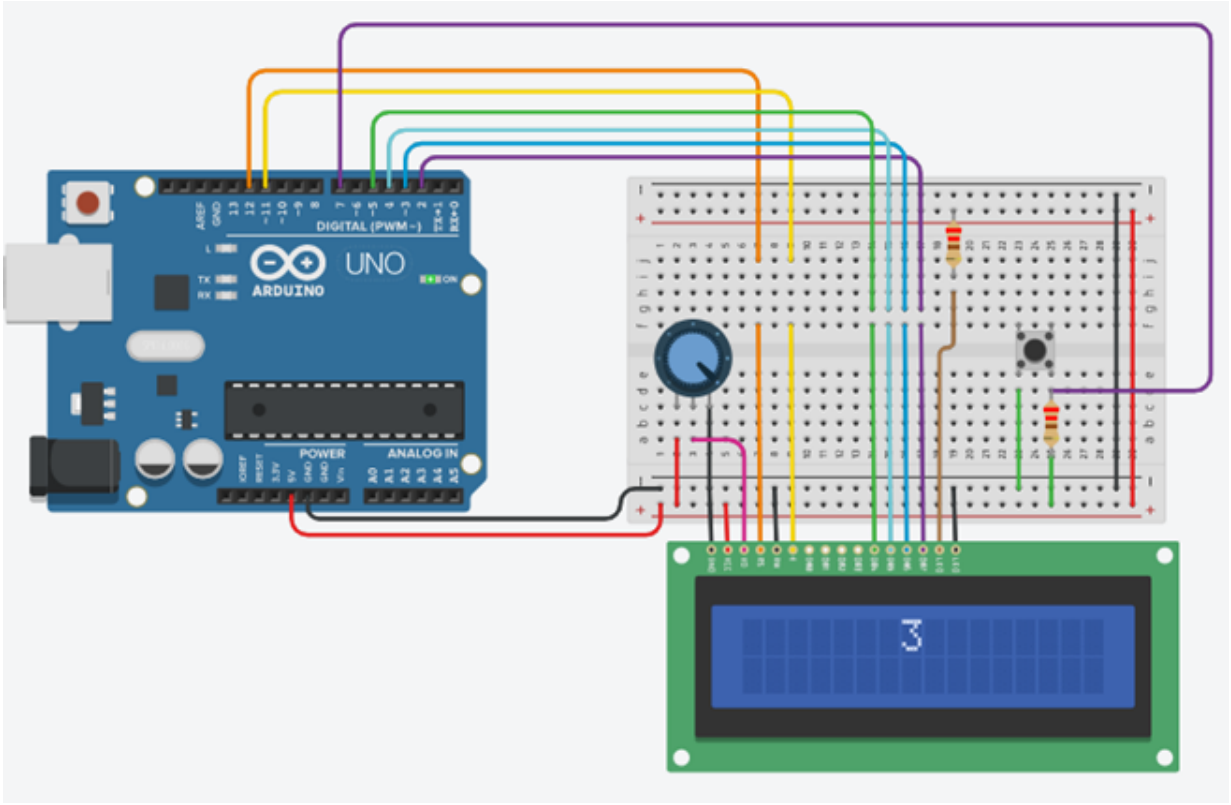
3.3.4 Prática - Escrever "Hello Word" no LCD:

include

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(19,23,18,17,16,15);
void setup(){
    lcd.begin(16,2);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Hello");
    lcd.setCursor(0,1);
    lcd.print("World");
}
void loop{}
```

3.3.5 Exercício 1:

- Tarefa: Faça um circuito contador de 0 a 10 com um botão, a cada clique do botão acrescenta 1 ao contador, caso o contador chegue em 10 ele zera. Imprima os números no LCD e substitua-os a cada soma.
- Tempo estimado: 30 minutos.



3.3.6 Resposta:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

int x = 0;

void setup(){
  lcd.begin(16,2);
  pinMode(7,INPUT_PULLUP);
  Serial.begin(9600);
  lcd.clear();
  lcd.setCursor(0,0);
}
```

```
void loop()
{
  if (digitalRead(7) == LOW){
  x = x + 1;
  lcd.clear();
  delay(200);

  }
lcd.setCursor(7,0);
lcd.print(x);
if (x> 10){
  x = 0;
  lcd.clear();
  }
}
```

3.3.7 Exercício 2:

- Tarefa: Faça uma programação para mostrar a palavra “Subindo” na tela do LCD, onde o display comece “em branco”, e a palavra vá subindo pelo display até sumir.
- Tempo estimado:10 minutos.

3.3.8 Exercício 3:

- Tarefa: utilizando a mesma programação da aula anterior sobre o efeito fading do LED com o Potenciômetro, faça com que o valor lido na entrada Analógica do LED seja mostrado no LCD.
- Tempo estimado: 20 minutos.