UFAC FUNDAPE CITS

Aplicações IoT
Internet das Coisas

Prof. André Nasserala
andre.nasserala@ufac.br

SHIELD W5100

- Controlar sensores ou enviar informações remotamente é um dos grandes objetivos de quem mexe com Arduino.
- O Arduino Ethernet Shield W5100 é outro dispositivo dessa família, que além de possibilitar o acesso às informações na sua rede local, ainda pode ser conectado à internet e permitir o seu monitoramento de qualquer lugar do mundo.

SHIELD W5100

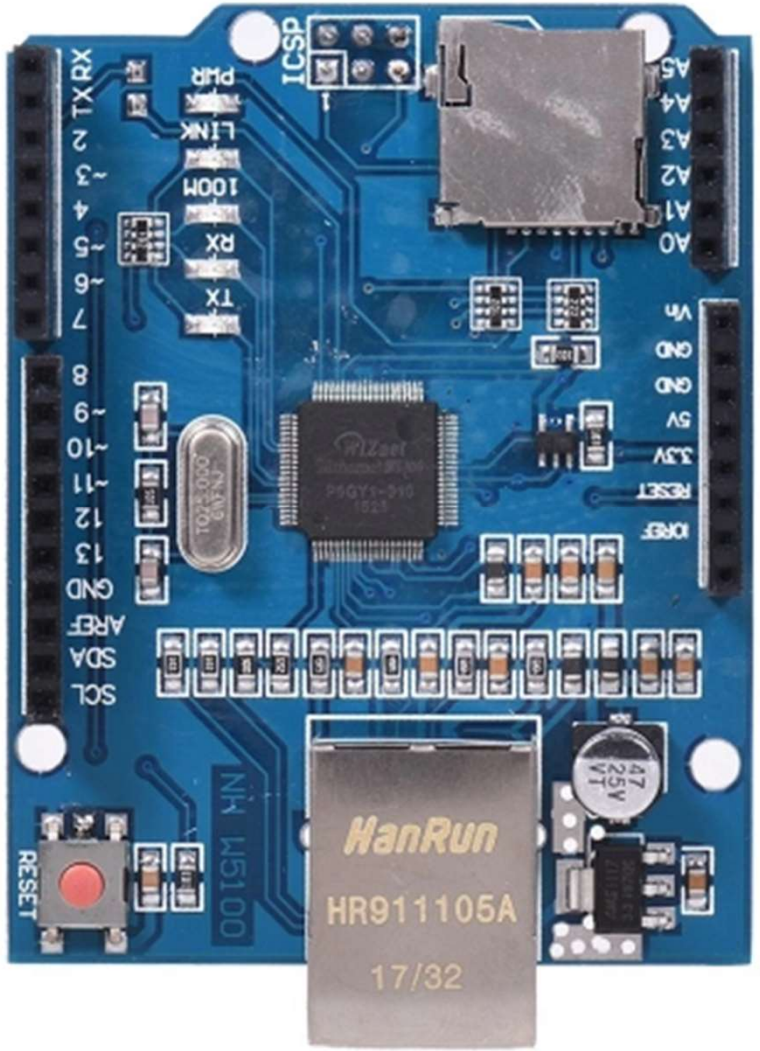
- Acoplando o Arduino Ethernet Shield W5100 ao seu Arduino, basta um simples cabo de rede para que, em poucos minutos, você passe a monitorar o estado de sensores, chaves e outros dispositivos à partir do browser do seu computador ou celular.
- Este Shield é baseado no ethernet chip Wiznet W5100 (olhem o datasheet) e fornece um endereço IP compatível com os protocolos TCP e UDP.
- O Ethernet Shield W5100 é compatível tanto com o Arduino Uno e Arduino Mega e possui um slot para cartão micro-SD, o que facilita a criação de dataloggers.

SHIELD W5100

- Usar o Ethernet Shield é uma forma simples de disponibilizar online dados referentes a sensores ou sistemas que você controle usando um Arduino.
- A biblioteca Ethernet da IDE oferece todos os recursos necessários para que dados lidos com o Arduino possam ser acessados online.
- Do ponto de vista de hardware, trata-se de uma conexão simples.
- Basta encaixar o shield no seu Arduino UNO ou Mega e ter em mãos um cabo ethernet com conetores RJ45 para ligar no seu modem/roteador.
- Se seu Arduino estiver ligado em algum sistema em específico, é necessário refazer as ligações diretamente no Shield.

Características do W5100

- Suporta 4 conexões independentes simultaneamente;
- Não suporta fragmentação de IP;
- Memória Interna 16Kbytes para Tx / Rx Buffers;
- Suporte Interface Serial (SPI modo 0, 3)
- Saídas função Multi-LED (TX, RX, Full / Half duplex, Colisão, Speed Link)
- O W5100 Wiznet fornece uma rede (IP) capaz de TCP e UDP.



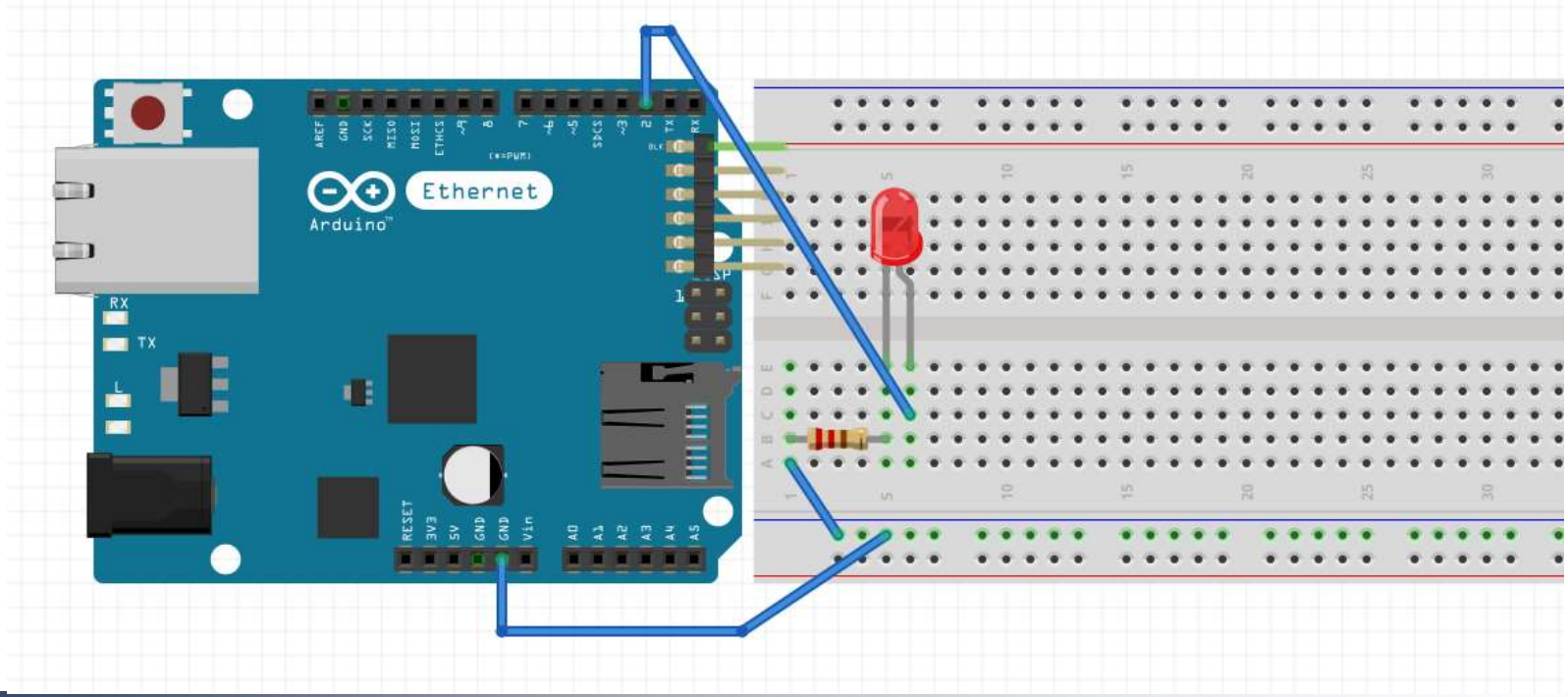
Características do W5100

- Uma série de LEDs indicativos também ajuda a saber o que está acontecendo com o Shield durante sua operação:

- Led PWR: Indica que a placa e Shield está alimentada;
- Led LINK: Indica a presença de uma ligação de rede e pisca quando o shield transmite ou recebe dados;
- Led FULLD: Indica que a conexão de rede é full duplex;
- Led 100M: Indica a presença de uma conexão de rede 100 Mb/s (em oposição a 10 Mb/s)
- Led RX: Pisca quando o shield recebe dados;
- Led TX: Pisca quando o shield envia os dados;
- Led COLL: Pisca quando são detectadas colisões de rede

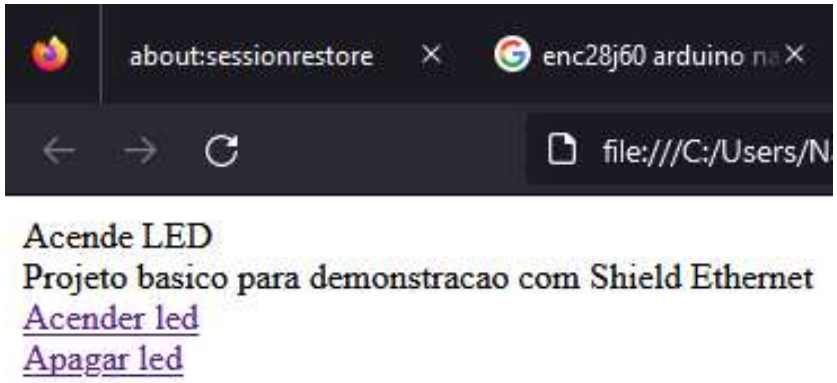
Programando o Arduino W5100

- Altere esses parâmetros do código de acordo com a sua configuração de rede, salve o programa e carregue-o no seu Arduino.
- O objetivo desse exemplo é exibir em uma página html, a opção de acender e apagar um LED.



Enviando Informações pela Rede

- Encaixe o Arduino Ethernet Shield W5100 ao seu Arduino e ligue-o à um roteador ou hub usando um cabo de rede comum.
- Vamos usar o webserver embutido na placa para enviar ao browser duas informações sobre a porta do Arduino, informando sobre o estado (ligado/desligado) de um led ligado à porta 2.
- Para testar o funcionamento, abra o browser no seu computador e digite na barra de endereços o IP que você configurou no programa, no nosso caso 192.168.99.2.



Programando o Arduino W5100

```
#include <SPI.h> // Biblioteca utilizada para comunicação com o Arduino
#include <Ethernet.h>

// A linha abaixo permite definir o endereço físico (MAC ADDRESS) da placa...
//de rede.
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

byte ip[] = { 192, 168, 99, 2 }; // Define o endereço IP.

// Porta onde estará aberta para comunicação Internet e Arduino.
EthernetServer server(80);

String readString;
int Pin = 2; // Pino digital onde será ligado e desligado o LED.

void setup(){
        pinMode(Pin, OUTPUT); // Define o Pino 9 como saída.
        Ethernet.begin(mac, ip); // Chama o MAC e o endereço IP da placa Ethernet.
        // Inicia o servidor que esta inserido junto a placa Ethernet.
        server.begin();
}
```

Programando o Arduino W5100

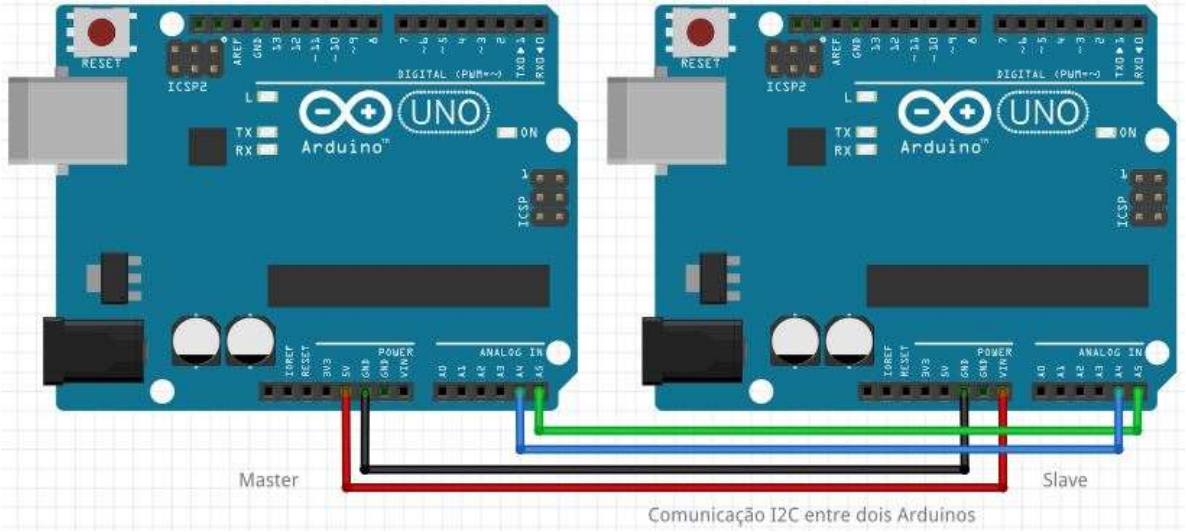
```
void loop() {
  EthernetClient client = server.available();
  if (client) {
    while (client.connected()) {
    if (client.available()) {
      char c = client.read();
      if (readString.length() < 100) {
        readString += c;
      }
      if (c == '\n') {
        client.println("HTTP/1.1 200 OK");
        client.println("Content-Type: text/html");
        client.println();
        client.println("Acende LED <br />");
        client.println("Projeto basico para demonstracao com Shield Ethernet <br />");
        client.println("<a href=""LedOn"">Acender led</a><br />");
        client.println("<a href=""LedOff"">Apagar led</a><br />");
        delay(1);
        client.stop();
```

Programando o Arduino W5100

```
if(readString.indexOf("LedOn") > 0) {
            digitalWrite(Pin, HIGH); // Liga LED.
          }
          else {
            if(readString.indexOf("LedOff") > 0) {
              digitalWrite(Pin, LOW); // Desliga LED.
             }
           }
           readString="";
        }
      }
     }
   }
}
```

Protocolo i2C

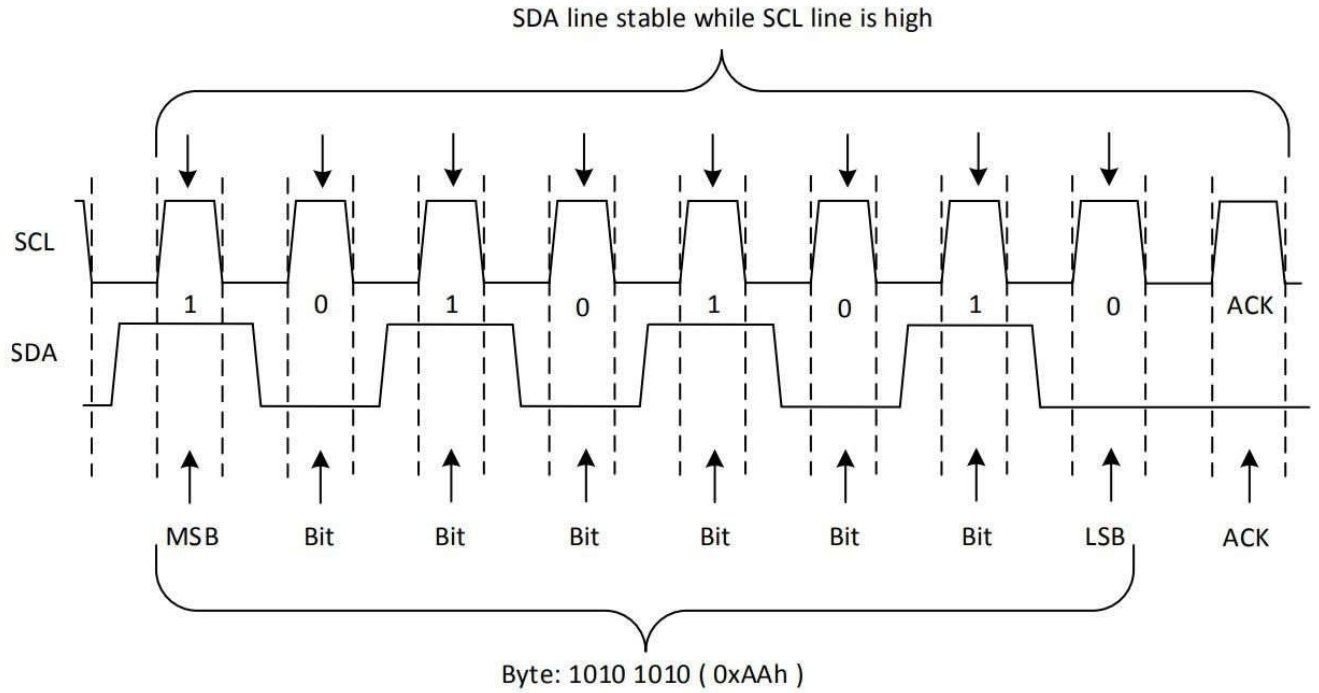
- O protocolo I2C (ou Inter-IC), às vezes chamado TWI, ou Two Wire Interface, foi desenvolvido pela Philips Semiconductors (também conhecida como NXP) para criar um barramento bidirecional simples, utilizando apenas duas linhas de comunicação para controle entre os CIs.



Comunicação I2C entre dois Arduinos

Protocolo i2C

- O protocolo utiliza apenas duas linhas:
- A linha serial de dados (serial data line, ou SDA) e
- A linha serial do clock (serial clock line, ou SCL).
- No Arduino UNO, elas correspondem ao pino analógico 4 (SDA) e ao pino analógico 5 (SCL).



Protocolo i2C

- Você pode ter até 128 dispositivos I2C (também chamados de nós) conectados às mesmas duas linhas de comunicação.
- Alguns dispositivos I2C utilizam +5 V, outros +3,3 V, por isso você deve estar atento a essa característica quando for utilizá-los.
- Certifique-se de ler o datasheet e de utilizar a voltagem correta, antes de conectar os componentes de um dispositivo I2C.
- Se você deseja ter dois Arduinos se comunicando, o protocolo I2C será uma boa escolha para seu projeto.

Protocolo i2C

- Para iniciar uma comunicação I2C em um Arduino, você utiliza o comando *Wire.begin().*
- Isso inicializará a biblioteca Wire e definirá o Arduino como o dispositivo I2C mestre.
- Também configurará os pinos analógicos 4 e 5 como pinos I2C.
- Para iniciar a comunicação como um escravo (por exemplo, dois Arduinos conectados utilizando I2C), o endereço do dispositivo escravo tem de ser incluído nos parênteses.
- *Comando:*
- *Wire.begin(5);*

Protocolo i2C

- Em outras palavras, o comando: *Wire.begin(5);* Fará com que o segundo Arduino se junte ao barramento I2C como dispositivo escravo no endereço 5.
- Um byte pode ser recebido do dispositivo I2C, utilizando:
- *Int x = Wire. Receive();*
- Antes de fazê-lo, você deve solicitar o número de bytes, utilizando Wire.requestFrom(address, quantity).

- Assim, o comando:
- *Wire.requestFrom(5,10);*
- Solicitaria dez bytes do dispositivo 5.

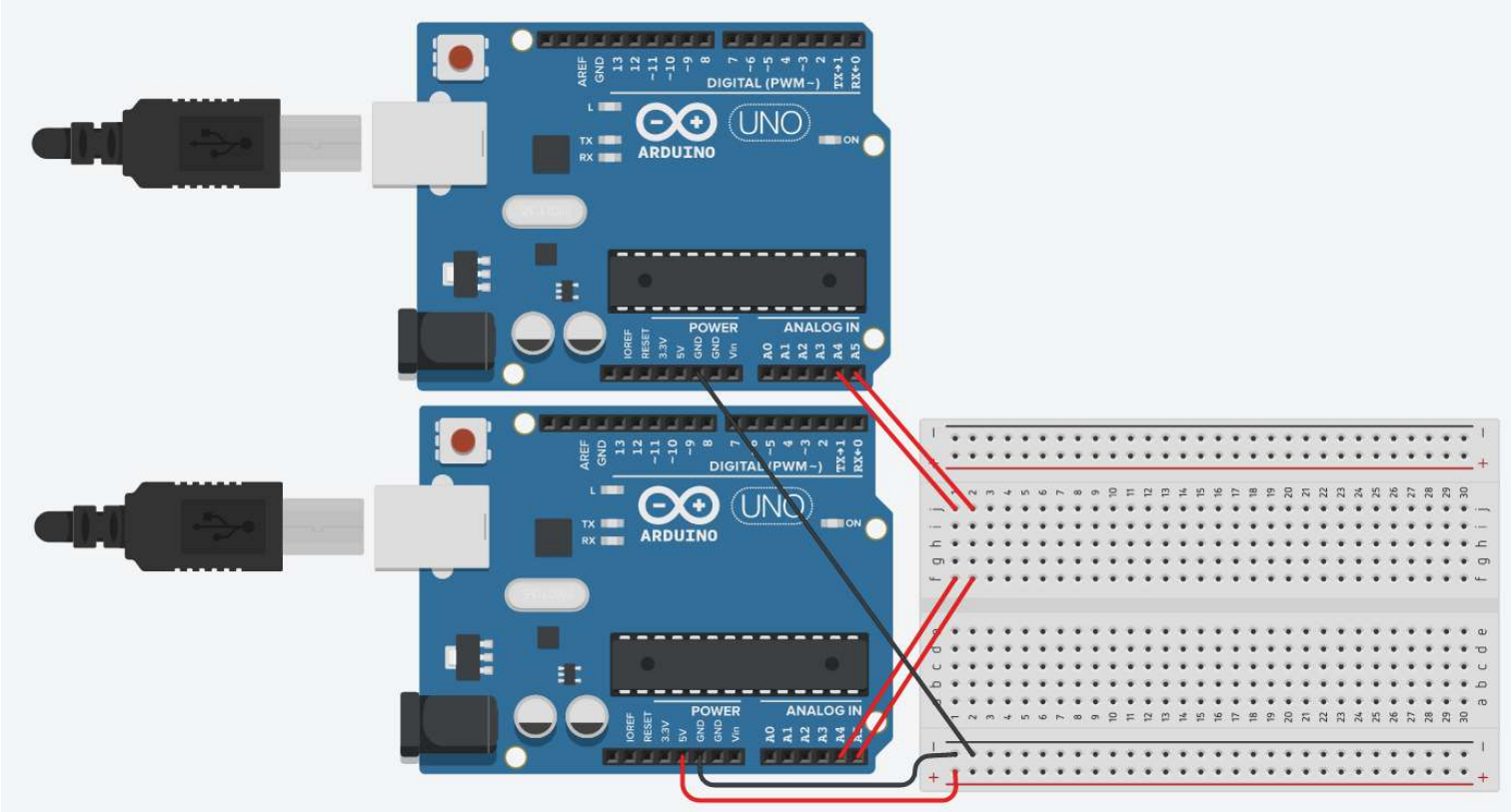
Protocolo i2C

- Enviar dados para o dispositivo é tão simples quanto utilizar o comando:
- *Wire.beginTransmission(5);*
- O qual define que a transmissão deve ser feita para o dispositivo de número 5.

- Depois, podemos utilizar:
- *Wire.send(x);*
- Para enviar um byte, ou:
- *Wire.send("Wire test.");*
- Para enviar dez bytes.

Exemplo - 01

- Vamos interligar 2 Arduinos UNO para enviar blocos de textos do Master para o Slave;



Código Master

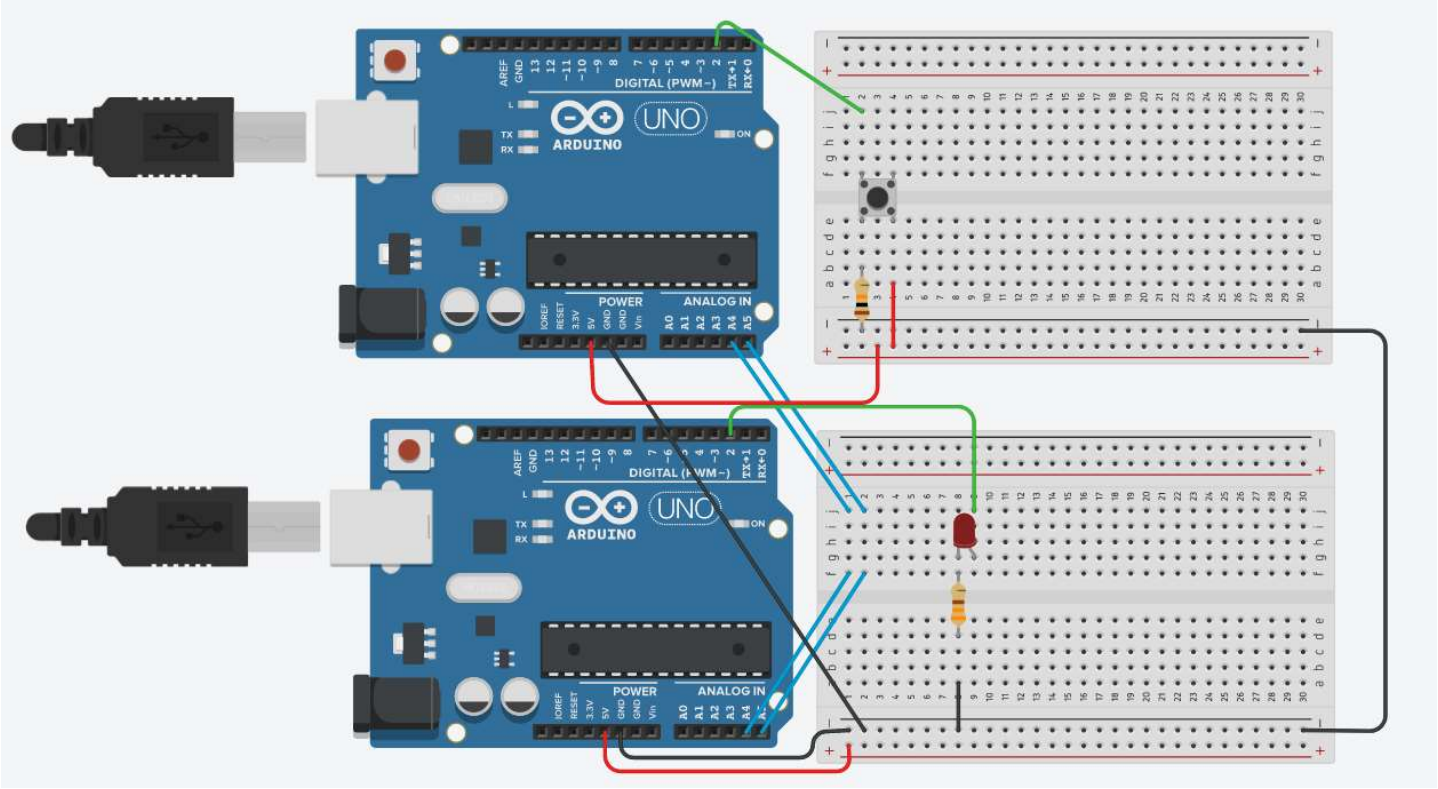
```
#include <Wire.h>
byte x = 4;                          // Variável para transmitir
void setup()
{
   Wire.begin();                     // Configura o barramento I2C
}
void loop()
{
  Wire.beginTransmission(15); // Transmite para o dispositivo número 15
  Wire.write("VASCAO ");          // Envia 7 bytes
  Wire.write(x);                     // Envia 1 byte
  Wire.endTransmission();        // Para transmissão
  x++;                                 // Incrementa x
  delay(100);
}
```

Código Slave

```
#include <Wire.h>                    //Inclui a biblioteca I2C
void setup()
{
  Wire.begin(15);                    //Barramento I2C do endereço 15
  Wire.onReceive(receiveEvent);      //Recepção de dados (chama função auxiliar)
  Serial.begin(9600);                //Inicia comunicação Serial com 9600 de baud rate
}
void loop()
{
  delay(100);
}
void receiveEvent(int howMany)  //Função auxiliar para processar os dados recebidos do Master
{
  while (1 < Wire.available())       //Loop para receber toda String de dados
  {
    char c = Wire.read();            //Recebe um byte caractere
    Serial.print(c);                 //Imprime na Serial
  }
  int x = Wire.read();               //recebe um byte do tipo inteiro
  Serial.println(x);                 //Imprime na Serial
}
```

Exemplo - 02

- Vamos interligar 2 Arduinos, um com botão e outro com um led, para o botão do Master acender o led do Slave;



Código Master

- #include <Wire.h>
- byte x = 0; // Variável para transmitir
- int estado = 0; // variável para leitura do pushbutton
- int guarda_estado = LOW; // variável para armazenar valores do pushbutton

- void setup()
- {
- Wire.begin(); // Configura o barramento I2C
- pinMode(2, OUTPUT); // Pino do Botão
- }

Código Master

```
void loop()
{
  estado = digitalRead(2);
  if (estado == HIGH) {
    guarda_estado = !guarda_estado;
    delay(500);
  }

  if (guarda_estado == HIGH) {
    x = 1;
    Wire.beginTransmission(15);
    Wire.write(x);              // Envia 1 byte
    Wire.endTransmission();     // Para transmissão
  }
```

Código Master

```
else {
      x = 0;
      Wire.beginTransmission(15);
      Wire.write(x);              // Envia 1 byte
      Wire.endTransmission();  // desliga o led
   }
  delay(100);
}
```

Código Slave

```
#include <Wire.h>                 //Inclui a biblioteca I2C
void setup()
{
  Wire.begin(15);                 //Barramento I2C do endereço 15
  Wire.onReceive(receiveEvent);  //Recepção de dados
  pinMode(2, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
  delay(100);
}
```

Código Slave

```
//Função auxiliar para processar os dados recebidos do Master
void receiveEvent(int howMany)
{
  int x = Wire.read();        //recebe um byte do tipo inteiro
  Serial.println(x);          //Imprime na Serial
  if ( x == 1 ) {
    digitalWrite(2, HIGH);
  }
  else {
    digitalWrite(2, LOW);
  }
}
```

Buzzer

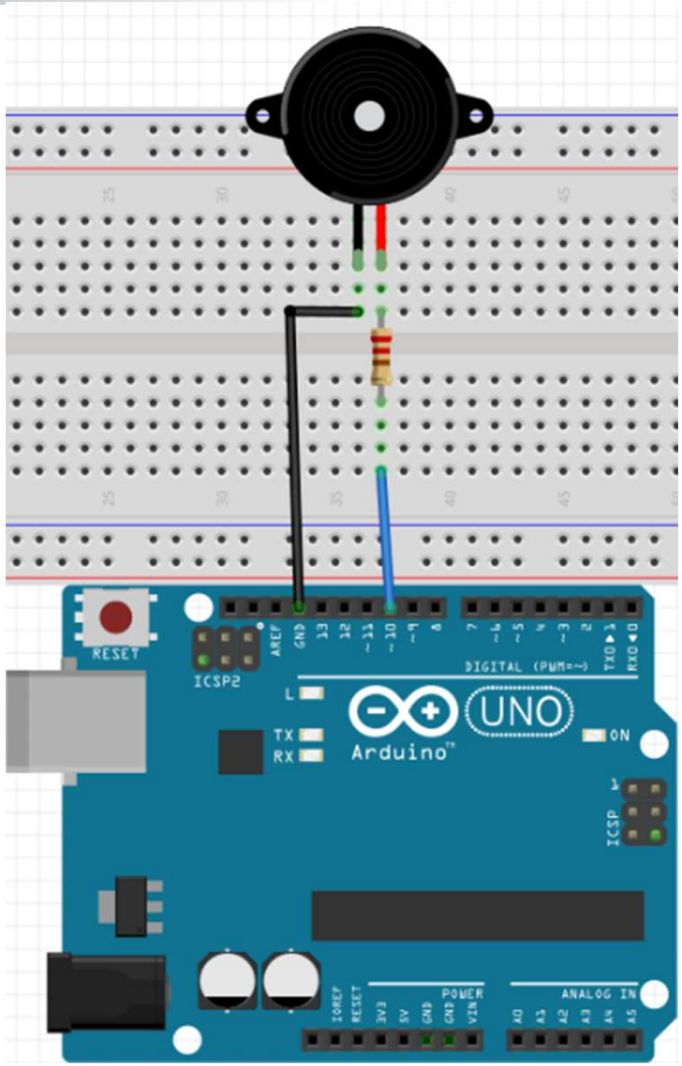
- O buzzer é um componente muito simples e barato, e chega a ser um auto falante alternativo, porém limitado.
- Por funcionar semelhante a um alto falante, ele pode ser usado para aplicações bem interessantes.
- Existem dois tipos de buzzer, o passivo e o ativo;
- A diferença entre os dois é que o som reproduzido pelo passivo acompanha a forma do sinal elétrico que aciona ele.
- Ou seja, se eu acionar o buzzer com um sinal elétrico que imita o som de uma flauta, o buzzer consegue reproduzir o som da flauta.

Buzzer

- Por outro lado, o buzzer ativo apenas emite um apito quando a tensão em seu pino passa de um determinado valor.
- Portanto, o buzzer ativo possui um timbre próprio e age como se fosse um "instrumento musical" à parte.
- Se utilizarmos um pino digital do Arduino para acionar um buzzer passivo, ele irá se comportar como um buzzer ativo, pois o sinal elétrico irá oscilar entre ativo e não ativo.
- Agora, se utilizarmos um módulo ou uma shield que reproduza arquivos de áudio, o som do buzzer passivo conseguirá reproduzir o som real.

Exemplo

- O circuito do buzzer é bem simples, pois basta ligar o negativo do buzzer no GND e o positivo em um resistor que pode ser de 100Ω (não tem valor exato).
- A outra ponta do resistor liga em um pino digital do Arduino.
- A imagem ao lado ilustra a ligação.



Como usar

- Cada nota musical possui uma certa frequência, então, para tocar uma nota, nós vamos mandar um sinal de onda quadrada pro buzzer em uma determinada frequência.

- As frequências das notas são aproximadamente:
- Dó – 262 Hz
- Ré – 294 Hz
- Mi – 330 Hz
- Fá – 349 Hz
- Sol – 392 Hz
- Lá – 440 Hz
- Si – 494 Hz

Usando a função do Arduino

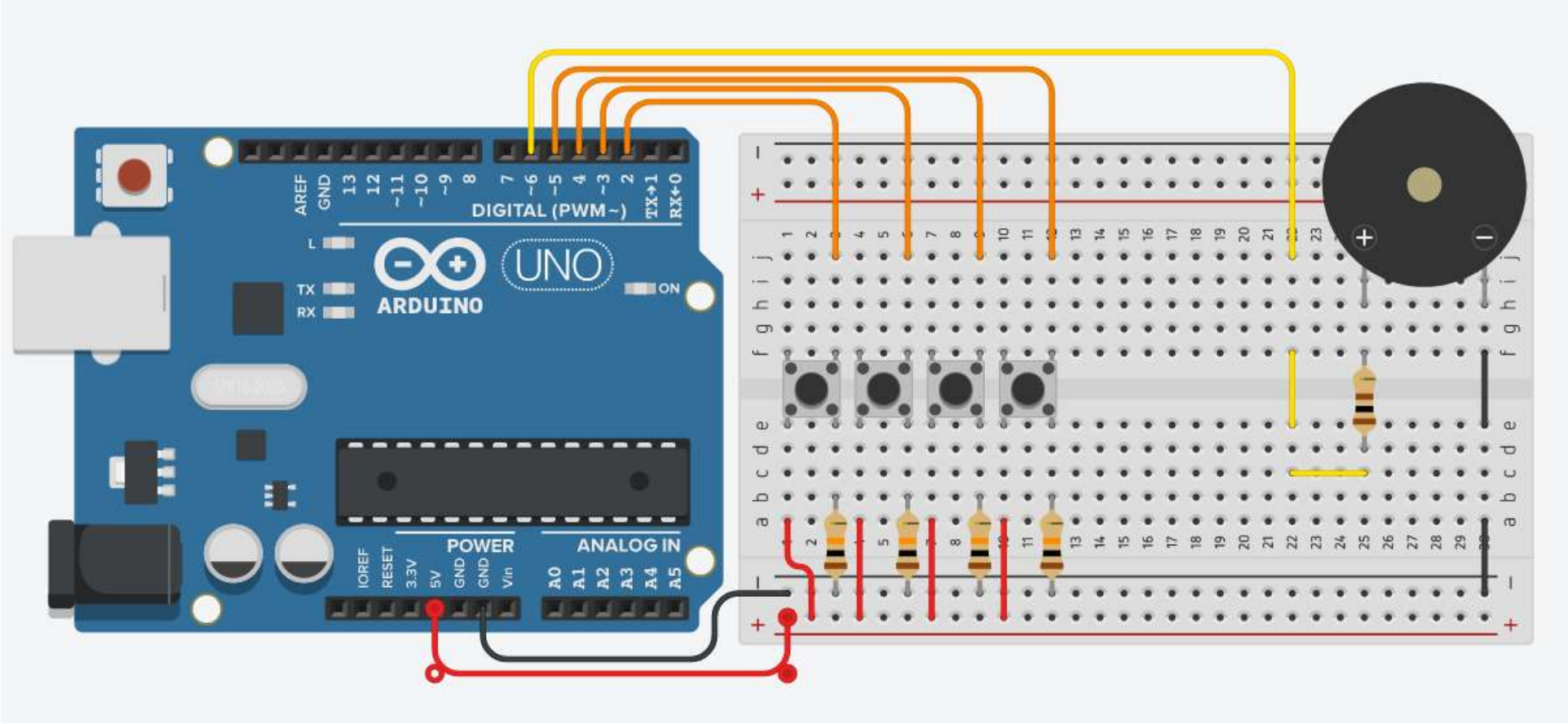
- Existe um comando que facilita muito nossa vida, que é o tone().

- Esse comando gera, automaticamente, um sinal de onda quadrada no pino e na frequência desejada.

- E ele possui a seguinte configuração:
- tone(pino, frequência em Hz, duração em ms)

- O parâmetro da duração define o tempo em ms da emissão sonora.

Exemplo Piano de 4 notas



PROJETO IOT

Código

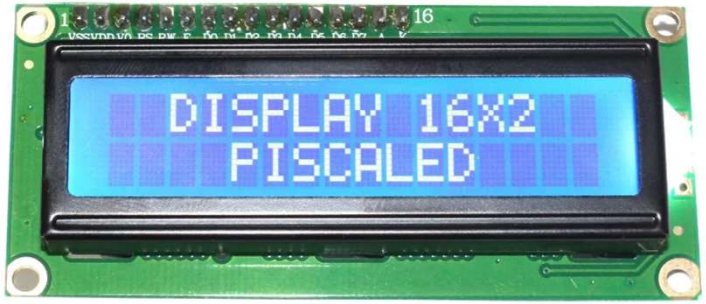
```
void setup(){
  pinMode(2,INPUT);
  pinMode(3,INPUT);
  pinMode(4,INPUT);
  pinMode(5,INPUT);
  pinMode(6,OUTPUT);
}
```

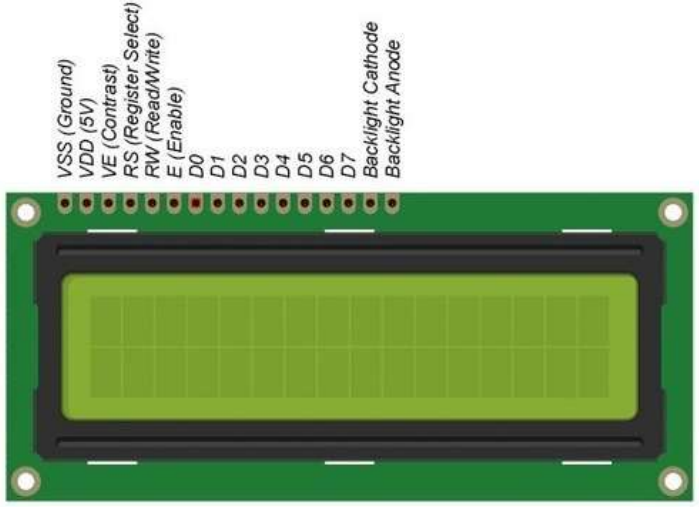
Código

```
void loop(){
  int leitura, tempo = 300;
  leitura = digitalRead(2);
  if ( leitura == HIGH ) {
        tone(6,262,tempo);
  }
  leitura = digitalRead(3);
  if ( leitura == HIGH ) {
        tone(6,294,tempo);
  }
  leitura = digitalRead(4);
  if ( leitura == HIGH ) {
        tone(6,330,tempo);
  }
  leitura = digitalRead(5);
  if ( leitura == HIGH ) {
        tone(6,349,tempo);
  }
}
```

LCD 16x2

- Quando esse tipo de Display LCD foi desenvolvido, os barramentos de dados dos Micro-controladores usavam 8 bits.
- Na era Arduino, as portas digitais disponíveis são reduzidas.
- Para resolver essa limitação, foi desenvolvida uma interface I2C especialmente dedicada para os LCDs.
- O chip usado nesse módulo é o PCF8574.
- Ele é um expansor de portas paralelas, tem uma interface I2C e pode controlar até 8 bits tanto como entrada ou como saída (dependendo da configuração).





Interface I2C para Displays LCD

- A velocidade da interface I2C esta limitada a 100 KHz.
- A tensão de alimentação pode ser 3,3V ou 5V, o que o habilita para todos os Micro-controladores mais comuns.
- Essa foto é a parte de trás do Display LCD com a Interface I2C já conectada.

Interface I2C para Displays LCD

- Para conectar com o Arduino ou outro Micro-contolador , somente quatro pinos são necessários :
- GND – conecte no terra do Arduino
- VCC – conecte na alimentação de 5V
- SDA – serial Data – interface I2C
- SCL – serial Clock – interface I2C
- O potenciômetro Azul nessa interface é usado para ajuste do contraste.
- Após o Display energizado e programado, ajuste-o para tornar a imagem visível.
- O jumper LED é usado para ativar o LED Backlight. Se não quiser usar o LED para economizar energia, retire esse jumper.
- O led vermelho na placa serve como indicação que ela esta energizada.

Exemplo

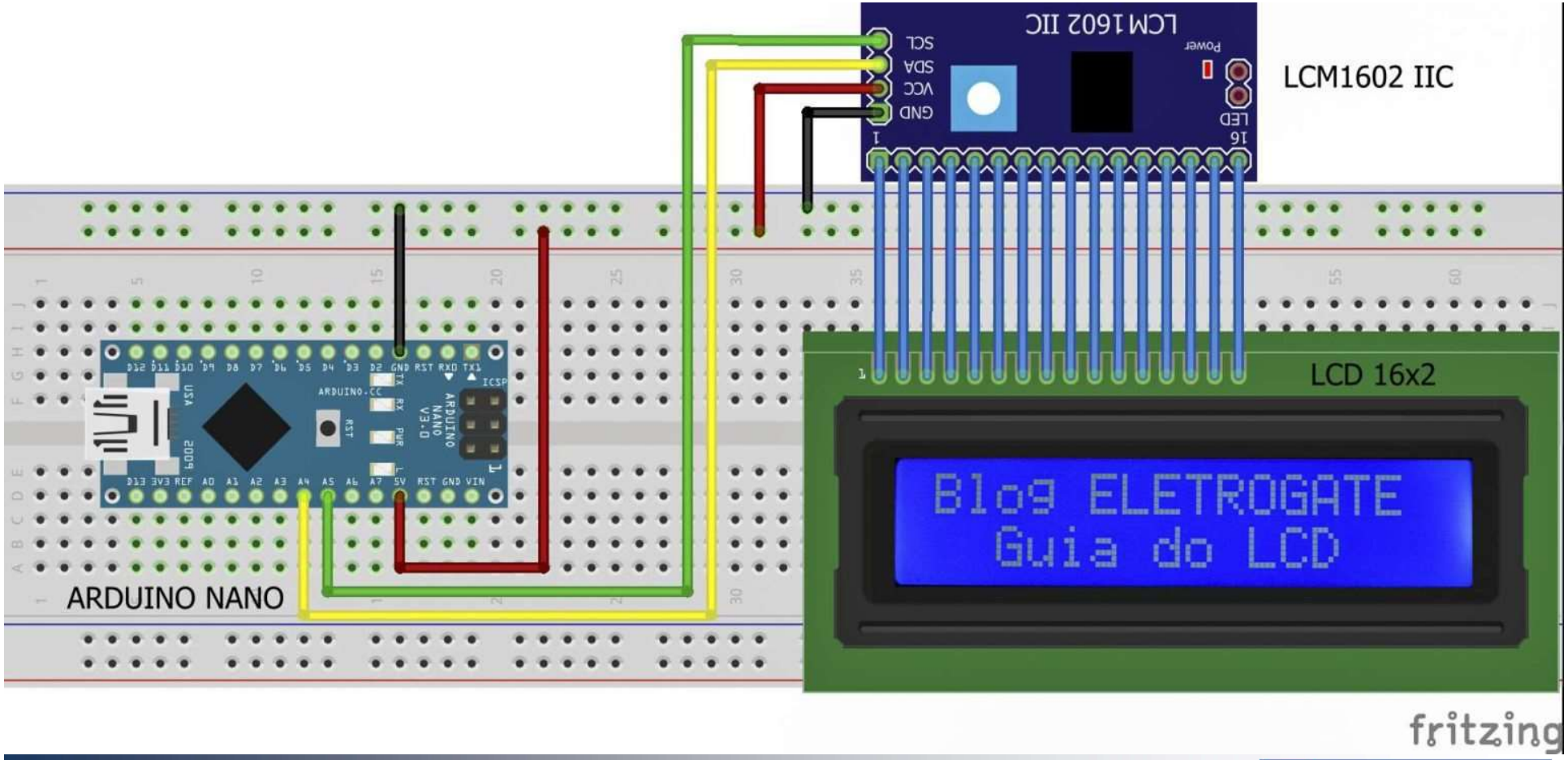
- Ligações das Portas I2C (para Arduino) :

- Porta SCL = pino A5
- Porta SDA = pino A4

- Não se esqueça de conectar o GND da Interface I2C no GND do Arduino. O mesmo para 5V.
- O consumo de corrente é de cerca de 27,5 mA. Faça o ajuste do contraste usando o potenciômetro da Interface I2C.

Exemplo

- Esse é o diagrama Fritzing do circuito para teste do Display LCD 16×2 (Interface I2C) com Arduino Nano :



I2C Scanner

- O primeiro teste a ser feito, é a identificação do endereço I2C da interface. Rode o Sketch I2C Scanner.

- Como exemplo, o endereço encontrado foi:

- I2C scanner. Procurando …
- Endereco I2C encontrado: 63 (0x3F)
- Encontrado 1 dispositivo(s).

I2C Scanner

```
// I2C Scanner
// Written by Nick Gammon
// Date: 20th April 2011
#include <Wire.h>
void setup()
{
  Serial.begin (9600);
  Serial.println ();
  Serial.println ("I2C scanner. Procurando ...");
  byte count = 0;

  Wire.begin();
  for (byte i = 8; i < 120; i++)
  {
    Wire.beginTransmission (i);
    if (Wire.endTransmission () == 0)
    {
      Serial.print ("Endereco I2C encontrado: ");
      Serial.print (i, DEC);
      Serial.print (" (0x");
      Serial.print (i, HEX);
      Serial.println (")");
      count++;
      delay (1);
    }
  }
  Serial.print ("Encontrado ");
  Serial.print (count, DEC);
  Serial.println (" dispositivo(s).");
}
void loop() {}
```

Biblioteca Liquid Crystal I2C

- Com a Biblioteca Liquid Crystal I2C, é possível criar programas para diversas funções, tais como:

- Imprimir mensagem;
- Piscar a mensagem;
- Teste do cursor;
- Teste de rolagem da mensagem;
- Teste de direção da mensagem.

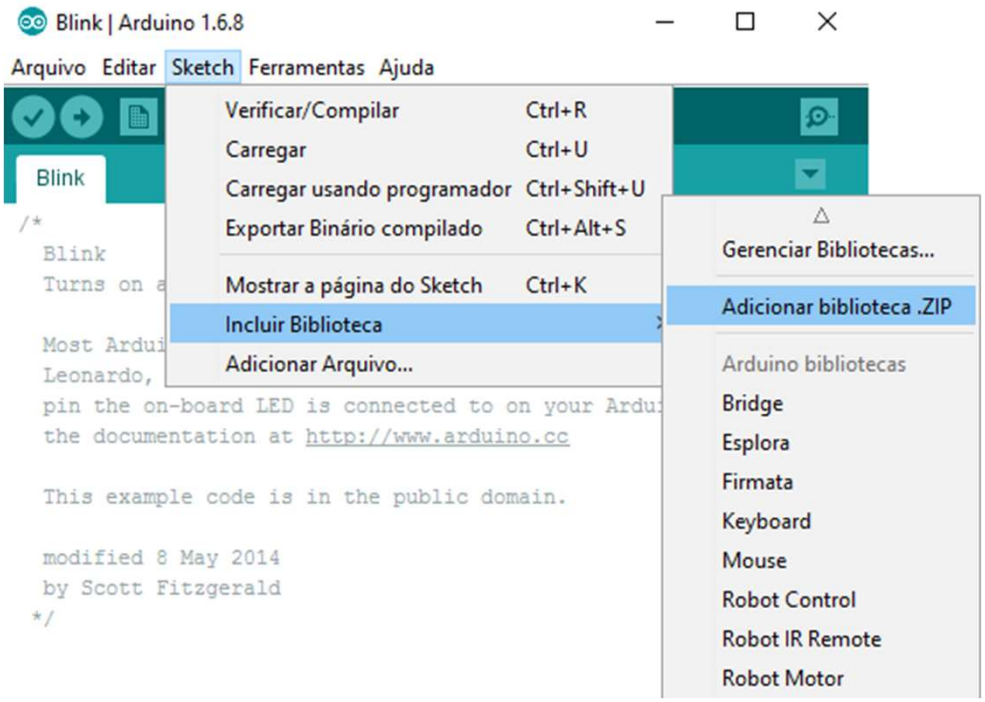
Biblioteca Liquid Crystal I2C

- Com a Biblioteca Liquid Crystal I2C, é possível criar programas para diversas funções, tais como:

- Imprimir mensagem;
- Piscar a mensagem;
- Teste do cursor;
- Teste de rolagem da mensagem;
- Teste de direção da mensagem.

Biblioteca Liquid Crystal I2C

- Disponível na IDE do Arduino existe um Gerenciador de Bibliotecas onde você pode adicionar rapidamente uma biblioteca que está compactada no formato .ZIP ou até mesmo a pasta.
- Para isso deve-se na IDE abrir o menu: Sketch > Incluir Biblioteca (Include Library) > Adicionar Biblioteca .ZIP (Add .ZIP Library...) conforme a imagem ao lado.



Código

```
#include <Wire.h>                 // usando a biblioteca Wire
#include <LiquidCrystal_I2C.h>    // usando a biblioteca LiquidCrystal I2C

LiquidCrystal_I2C lcd(0x3F, 16, 2);  // Configura endereço I2C e display com 16 caracteres e 2 linhas
int thisChar = 0 ;

void setup()
{
  lcd.init();                  // inicializa LCD
  lcd.backlight();             // ativa led de backlight
}

void Hello ()                  // imprimindo mensagem
{
  lcd.setCursor(0, 0);         // selecionando coluna 0 e linha 0
  lcd.print("Blog ELETROGATE");   // print da mensagem
  lcd.setCursor(2, 1);         // selecionando coluna 2 e linha 1
  lcd.print("Guia do LCD");    // Print da mensagem
  delay(1000);                 // atraso de 1 segundo
}
```

Código

```
void Flash ()
{
  lcd.noDisplay();            // desliga display
  delay(1000);                // atraso de meio segundo
  lcd.display();              // liga display
  delay(1000);                // atraso de meio segundo
  lcd.clear();               // limpa a tela
  delay(1000);                // atraso de 1 segundo
}

void Blink ()                 // teste do cursor
{
  lcd.noBlink();              // apaga cursor
  delay(1000);                // atraso de 1 segundo
  lcd.blink();               // acende cursor
  delay(1000);                // atraso de 1 segundo
  lcd.clear();               // limpa a tela
  delay(1000);                // atraso de 1 segundo
}

void AutoScroll ()            // teste de rolagem de mensagem
{
  lcd.setCursor(16, 1);       // selecionando coluna 16 e linha 1
  lcd.autoscroll();           // configura rolagem automatica de mensagem
  for (thisChar = 0; thisChar < 10; thisChar++)    // imprime de 0 a 9
  {
    lcd.print(thisChar);      // imprime o numero
    delay(350);               // atraso de 350 ms
  }
  lcd.noAutoscroll();         // desliga rolagem autoamtica
  lcd.clear();               // limpa a tela
  delay(1000);                // atraso de 1 segundo
}
```

Código

```
void dirText ()                // teste de direcao de mensagem
{
  lcd.clear();                 // limpa a tela
  lcd.cursor();                // liga o cursor
  lcd.setCursor(10, 0);        // selecionando coluna 10 e linha 1
  for (thisChar = 1; thisChar < 10; thisChar++)    // imprime de 1 a 9
  {
    lcd.rightToLeft();         // imprime da direita para a esquerda
    lcd.print(thisChar);       // imprime o numero
    delay(350);                // atraso de 350 ms
  }
  for (thisChar = 1; thisChar < 10; thisChar++)    // imprime de 1 a 9
  {
    lcd.leftToRight();         // imprime da esquerda para a direita
    lcd.print(thisChar);       // imprime o numero
    delay(350);                // atraso de 350 ms
  }
  lcd.noCursor();              // desliga o cursor
}

void loop()
{
  Hello ();       // imprimindo mensagem
  Flash ();       // piscando a mensagem
  Blink ();       // teste do cursor
  AutoScroll ();  // teste de rolagem de mensagem
  dirText ();     // teste de direcao de mensagem
  delay(1000);    // atraso de 1 segundo
}
```

Bibliografia

- MONK, Simon. Programação com Arduino. Porto Alegre – RS. Editora: Bookman – 2017. ISBN: 9788582604465
- VIDAL, Vitor, Gustavo Murta. Arduino Start. Eletrogate – 2018. Belo Horizonte – MG. Disponível em: https://conteudo.eletrogate.com/apostila-arduino-start.
- MALVINO, Albert Paul. Eletrônica: Volume 1. 4.ed. São Paulo – SP: Makron Books, 1997. ISBN: 8534603782.
- SENAI, Senai SP. FUNDAMENTOS DE ELETRÔNICA - 1ªED. Editora: Senai SP – São Paulo 2015. ISBN: 9788583932086
- ELETROGATE, https://blog.eletrogate.com/ethernet-shield-w5100-com-arduino/. Em outubro de 2022.
- FELIPE-FLOP, https://www.filipeflop.com/blog/tutorial-ethernet-shield-w5100/. Em outubro de 2022.