
minicurso
ARDUINO

SEMANA DE ATUALIZACAO E TREINAMENTO DE ENGENHARIA

PET MECANICA

	segunda-feira	terça-feira	quarta-feira	quinta-feira		sexta-feira
8h às 12h	**ARDUINO** (Básico) *LabZorro*	**MATLAB** (Básico) *LabZorro*	**ARDUINO** (Básico) *LabZorro*	**VISITA TÉCNICA** Lingotamento contínuo *Arcelor Mittal*	**MATLAB** (Básico) *LabZorro*	**OFICINA DE JULIA** (Enfoque em EDO) *LabZorro*
12h às 13h	**ALMOÇO**	**ALMOÇO**	**ALMOÇO**	**ALMOÇO**		**ALMOÇO**
13h às 15h	**RODA DE CONVERSA** Tecnologia em materiais e fabricação *Auditório*	**RODA DE CONVERSA** Tecnologia em fluidos e térmica *Auditório*	**VISITA TÉCNICA** Laminação de tiras à quente *Arcelor Mittal*	**RODA DE CONVERSA** Inteligência Artificial e aplicações *Auditório*		**RODA DE CONVERSA** Mercado de trabalho e indústria *Auditório*
15h às 17h	**ANSYS** (Introdução) *LabZorro*	**EXCEL** (Básico + Intermediário) *LabZorro*		**ANSYS** (Introdução) *LabZorro*		**EXCEL** (Básico + Intermediário) *LabZorro*
17h às 19h				**PHOTOSHOP** (Introdução) *Sala do PET*		

Comissão do Minicurso

Renan Marques

Vitorino Biazi

Thiago Bragança

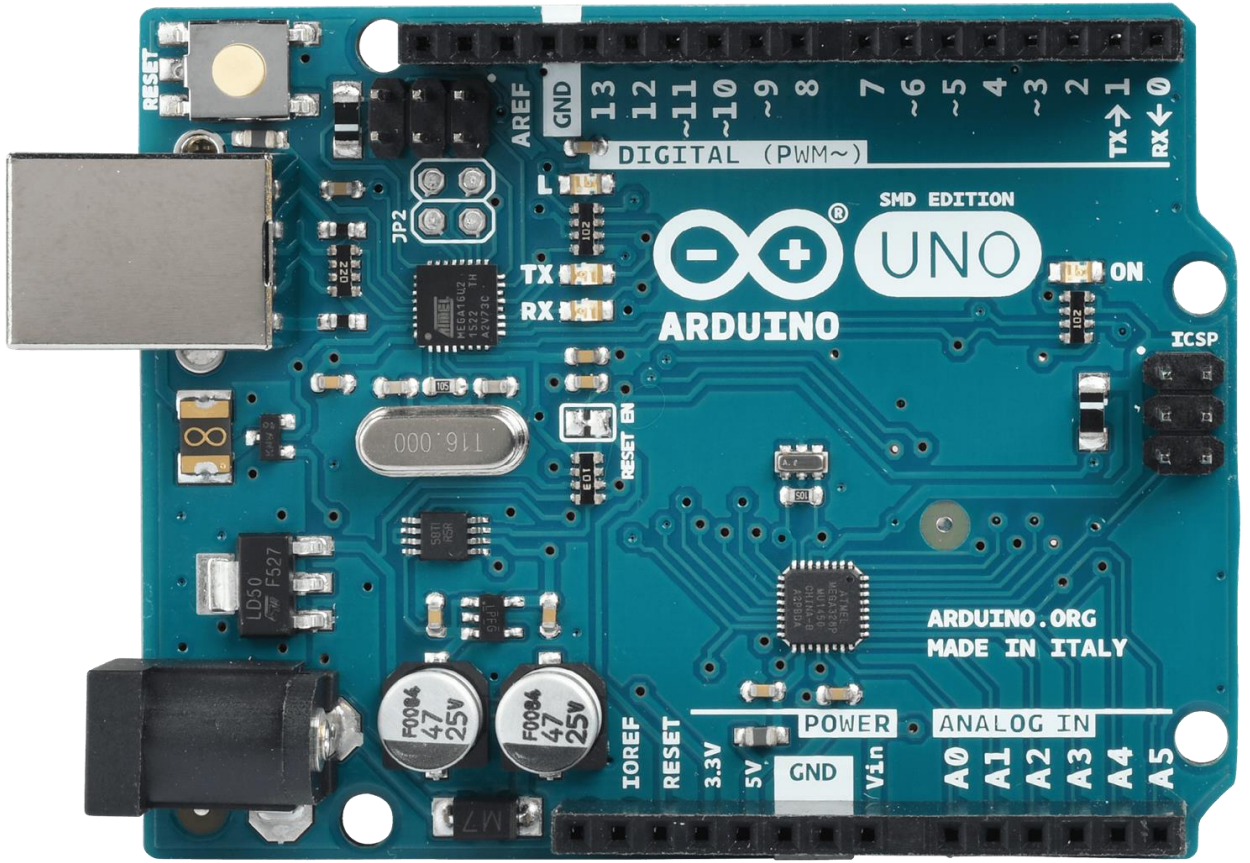
Robertson Junior

Jhonata Moraes

Gabriel Nunes

O Arduino

- Plataforma open-source
- Começou com estudantes de design
- Se baseia no processing (linguagem destinada para interfaces gráficas)
- Popularidade entre estudantes
- Diversas aplicações



Principais aplicações

Automação e projetos residenciais

Robótica

Família Arduino

Arduino Uno

Arduino Leonardo

Arduino Due

Arduino Yún

Arduino Tre

Arduino Micro

Arduino Robot

Arduino Esplora

Arduino Mega ADK

Arduino Ethernet

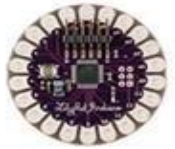
Arduino Mega 2560

Arduino Mini

LilyPad Arduino USB

LilyPad Arduino Simple

LilyPad Arduino SimpleSnap

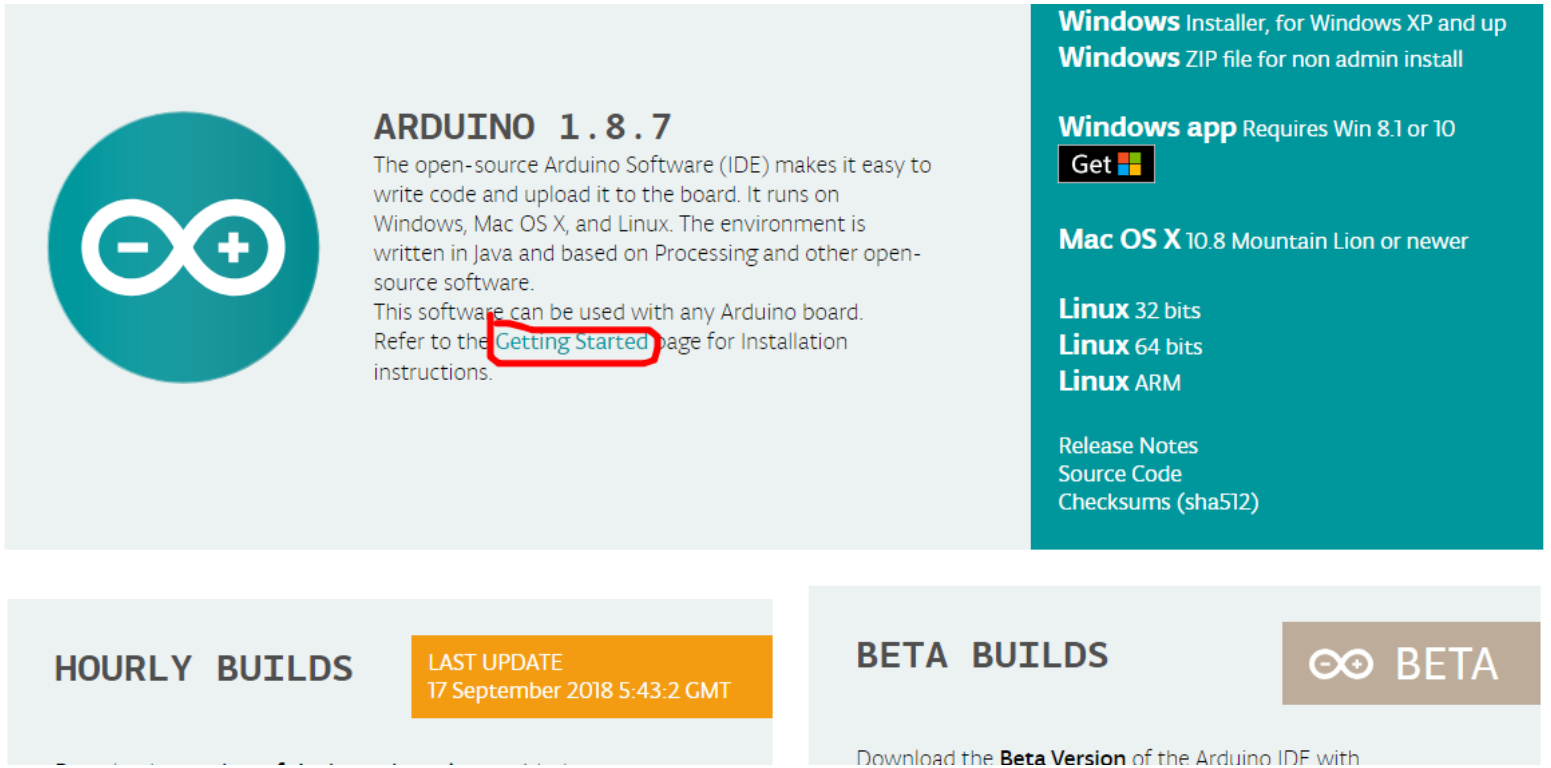


LilyPad Arduino

Arduino Nano

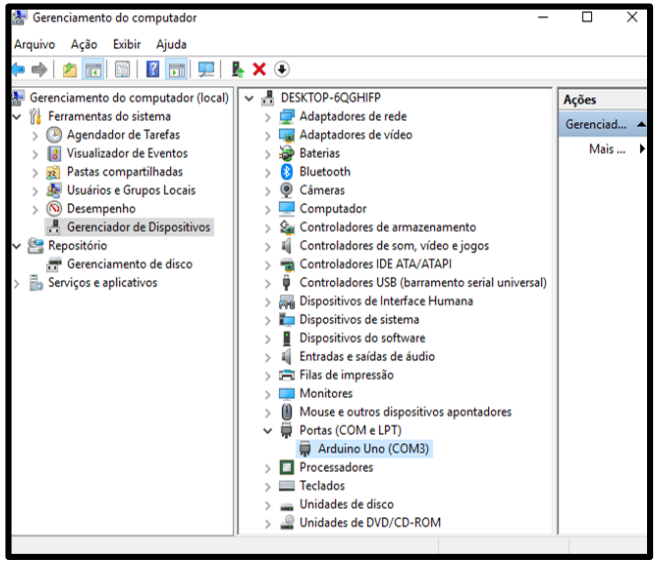
Arduino Pro Mini

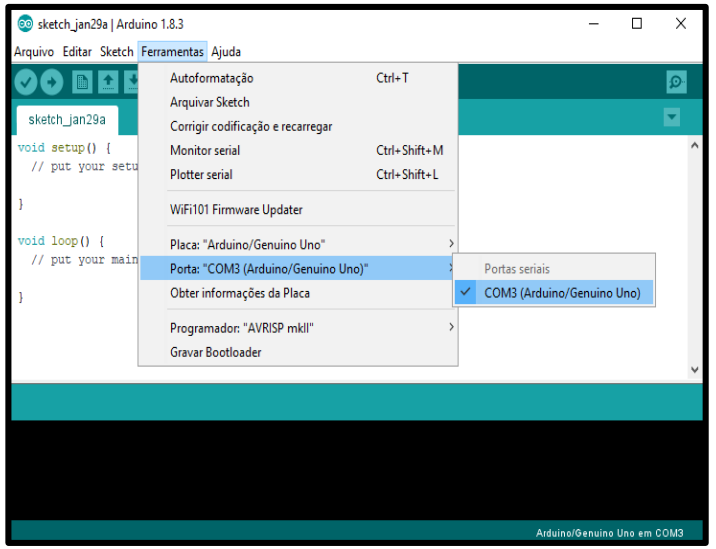
Software

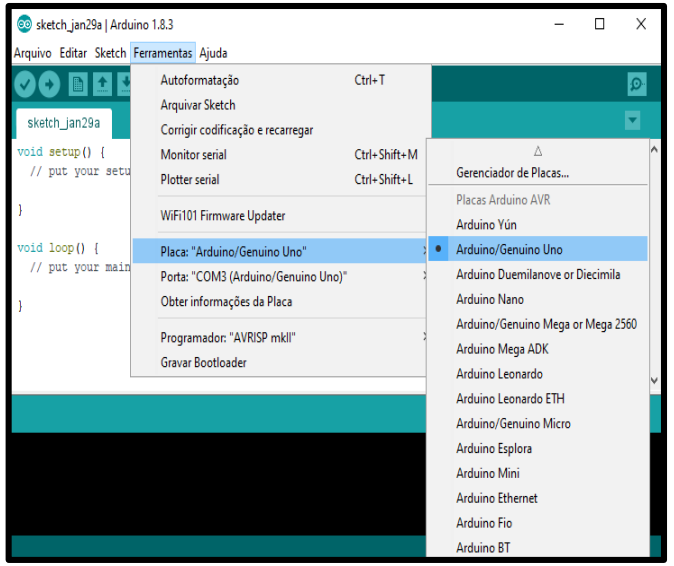

ARDUINO 1.8.7
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.
Windows Installer, for Windows XP and up
Windows ZIP file for non admin install
Windows app Requires Win 8.1 or 10
Get
Mac OS X 10.8 Mountain Lion or newer
Linux 32 bits
Linux 64 bits
Linux ARM
Release Notes
Source Code
Checksums (sha512)
HOURLY BUILDS
LAST UPDATE
17 September 2018 5:43:2 GMT
BETA BUILDS
BETA
Download the Beta Version of the Arduino IDE with

Arduino IDE

- Plug and play
- Verificação do driver
- Escolha das portas
- Escolha da placa

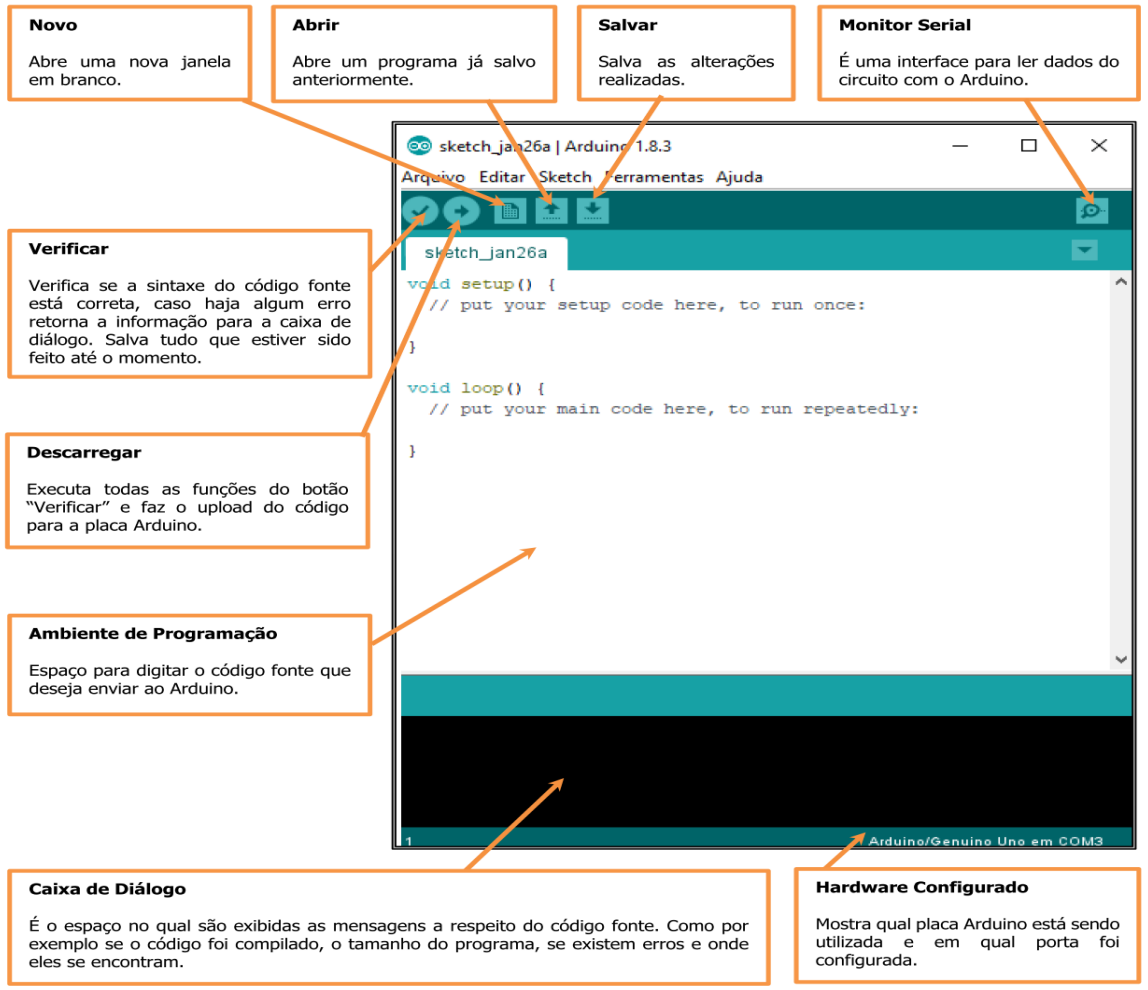






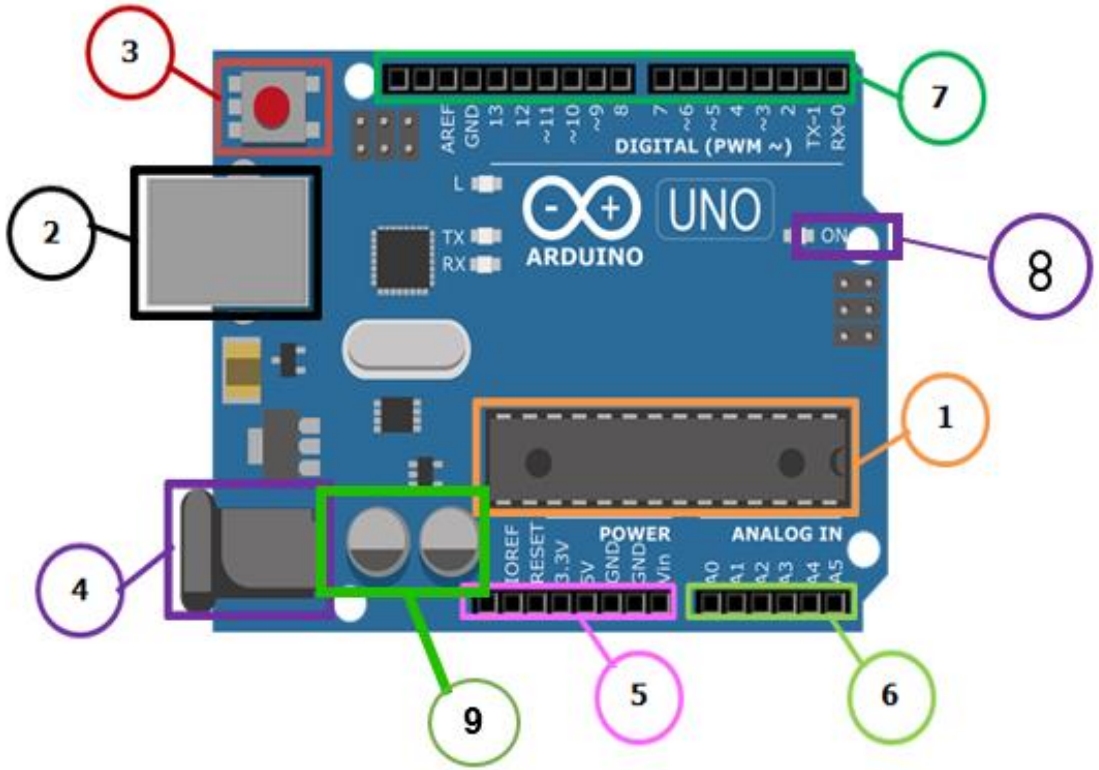
Arduino IDE

- Conectar a placa a uma porta USB do computador
- Desenvolver um sketch com comandos para a placa
- Upload do sketch para a placa utilizando a comunicação USB
- Aguardar a reinicialização da placa. Após a reinicialização, o sketch passa a ser executado pela placa.



Hardware

1. Microcontrolador: Atmel ATMEGA328P
2. Porta USB
3. Botão Reset
4. Conector P4
5. Pinos de alimentação e referência
6. Pinos analógicos
7. Pinos digitais:
8. Led ON
9. Capacitores



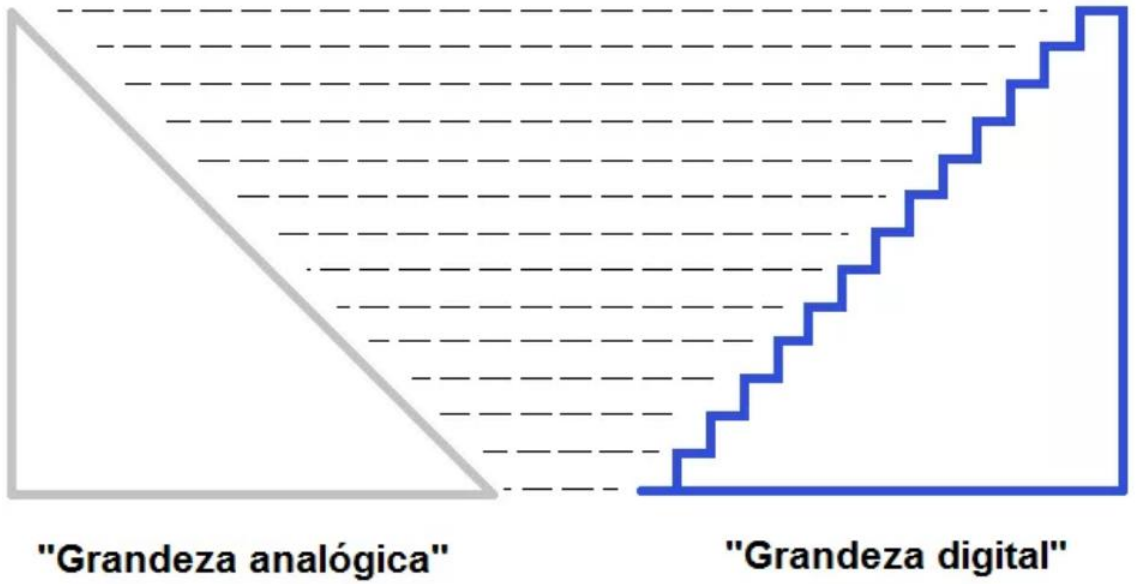
Relembrando

- O termo Bit vem de Binário
- Menor unidade de medida de transmissão de dados
- 1 Byte = octeto
- Potências de 2

Nome	Abrev	Tamanho em Bytes	Tamanho no SI
quilo	K	2^{10} ="1024"	10^{3} ="1000"
mega	M	2^{20} ="1 048 576"	10^{6} ="1 000 000"
giga	G	2^{30} ="1 073 741 824"	10^{9} ="1 000 000 000"
tera	T	2^{40} ="1 099 511 627 776"	10^{12} ="1 000 000 000 000"
peta	P	2^{50} ="1 125 899 906 842 624"	10^{15} ="1 000 000 000 000 000"
exa	E	2^{60} ="1 152 921 504 606 846 976"	10^{18} ="1 000 000 000 000 000 000"
zetta	Z	2^{70} ="1 180 591 620 717 411 303 424"	10^{21} ="1 000 000 000 000 000 000 000"
yotta	Y	2^{80} ="1 208 925 819 614 629 174 706 176"	10^{24} ="1 000 000 000 000 000 000 000 000"

Grandezas analógicas e digitais

- Pinos digitais
- Pinos analógicos
- Conversor ADC
- Pino de saída analógico ? PWM



Linguagem

- **C++**
- Chaves { }
- Ponto e virgula ;
- Parênteses ( )
- Blocos de Comentários /* ... */
- Comentários em linhas //

Variáveis

- Cada variável tem a função de armazenar um respectivo dado, podendo usa-lo novamente depois
- Operador de atribuição “=“

```
int var1;     // ambas as declarações estão corretas.
int var2 = 0; // foi atribuído ou armazenado o valor 0 em
              var2.
```

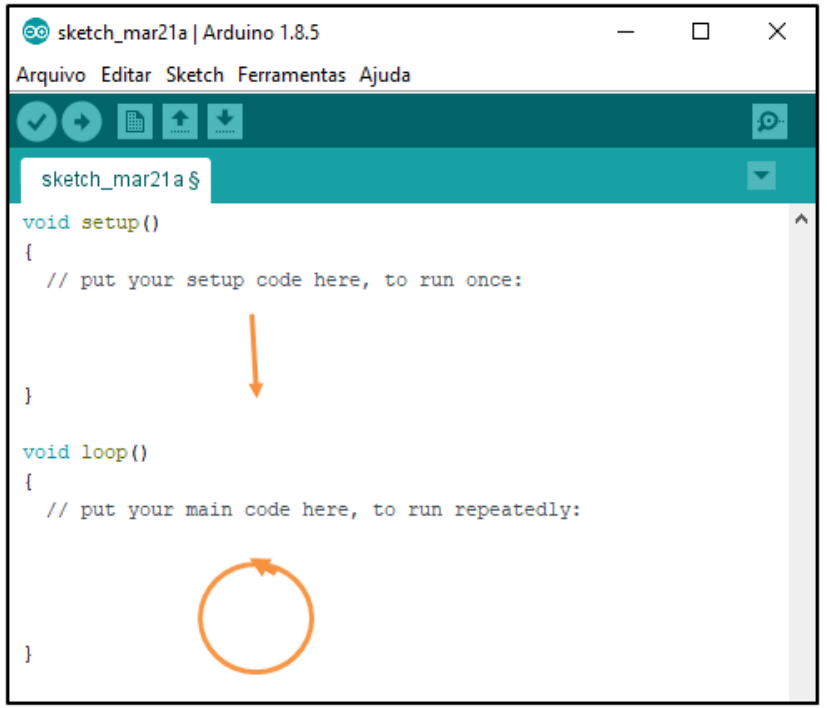
- Variaveis mais usadas
  - Char – um caractere de acordo com a tabela ASCII (Aritmética)
  - Byte – possui 8 bits
  - Int – possui 2 bytes
  - Float – possui 4 bytes, casas decimais
  - String – Matriz de char
- Variaveis constantes (Pré-Definidas no Arduino)
  - True/False
  - INPUT/OUTPUT
  - HIGH/LOW

EXEMPLO STRING:

```
char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};

char myString[] = "Esta eh a primeira linha"
" esta eh a segunda"
" etecetera";
```

Funções Principais

- Função Setup
  - Tipo void
  - Executa uma única vez
  - Declaração de pinos e incializações
- Função Loop
  - Tipo void
  - Repete o código programado como num ciclo
  - Comando a serem executados



Declarações dos pinos e Leitura de Dados

Pinos Digitais

- São pinos de saída e entrada, portanto precisam de declaração (8bits)
- pinMode(pinousado1,INPUT)
  - digitalRead(pinousado1)
- pinMode(pinousado2,OUTPUT)
  - digitalWrite(pinousado2,HIGH)
  - digitalWrite(pinousado2,LOW)
- pinMode(pinousado3,INPUT_PULLUP)

Pinos Analógicos

- São pinos de entrada, portanto não necessitam de declaração (10bits)
- analogRead(pinousado)
- analogWrite(pinousado,valor)
  - PWM (pinos com ~)

Comunicação Serial e outras funções

- Funções utilizadas para comunicação entre o usuário e o monitor serial do Arduino
- Serial.begin(9600)
- Serial.Write(“Mensagem”)
- Serial.println(“Mensagem”)
- Serial.avaliable( )

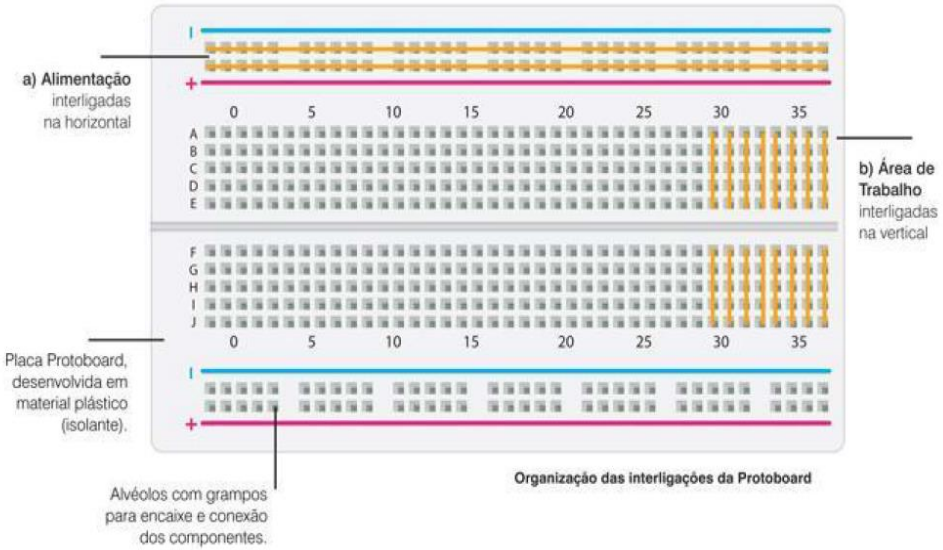
- millis()
  - 50 dias
- micros()
  - 70 minutos
- delay(ms)
- delayMicroseconds(μs)
- tone(pino,frequência,duração)
  - noTone() [caso não seja fornecida duração]

Referencia de tensão

- Alterar tensão de limite superior do arduino (aumentar exatidão)
  - analogreference(tipo)
- Tipos
  - **EXTERNAL**: A voltagem aplicada ao pino **AREF** (0 à 5V somente) é usada como referência.
  - **AR_DEFAULT**: A referência padrão analógica de 3.3V.
  - **AR_INTERNAL**: Uma referência embutida de 2.23V.
  - **AR_INTERNAL1V0**: Uma referência embutida de 1.0V.
  - **AR_INTERNAL1V65**: Uma referência embutida de 1.65V.
  - **AR_INTERNAL2V23**: Uma referência embutida de 2.23V.

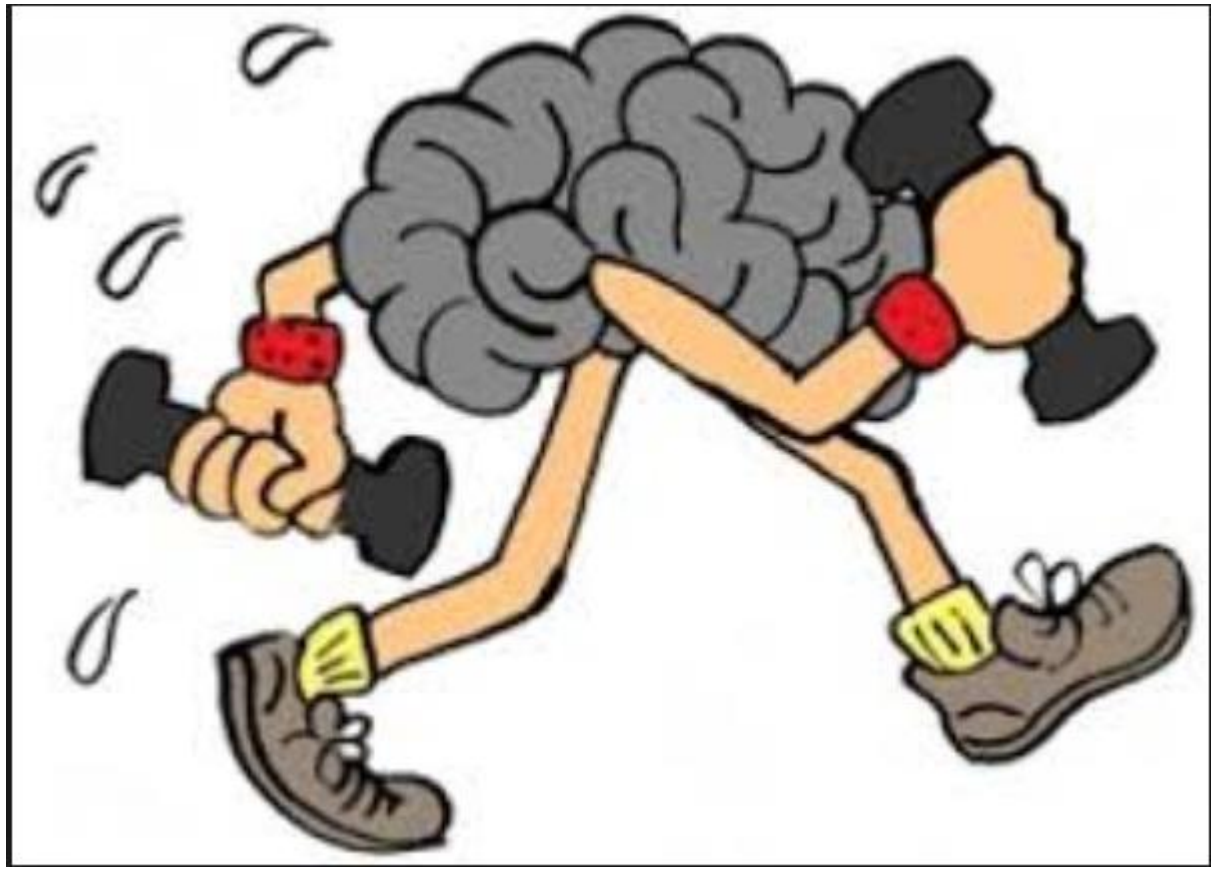
Protoboard

É uma placa para a montagem dos circuitos eletrônicos. Ela possui diversos furos que estão interligados de forma a facilitar as conexões dos componentes, como mostra a figura a seguir.

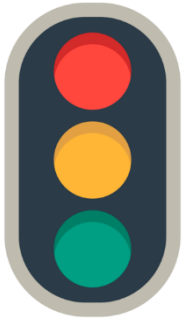


Organização das interligações da Protoboard

Agora vamos fazer alguns projetos



Projeto Semáforo



Parabéns! Você acaba de ser contratado para fazer o projeto de um semáforo de trânsito em um ponto da avenida Fernando Ferrari, para a travessia de pedestres, mas antes, é claro que você vai aplicar os seus conhecimentos em Arduino para executar a construção de um protótipo.

Projeto Semáforo

Para construir esse protótipo você vai precisar dos seguintes componentes:

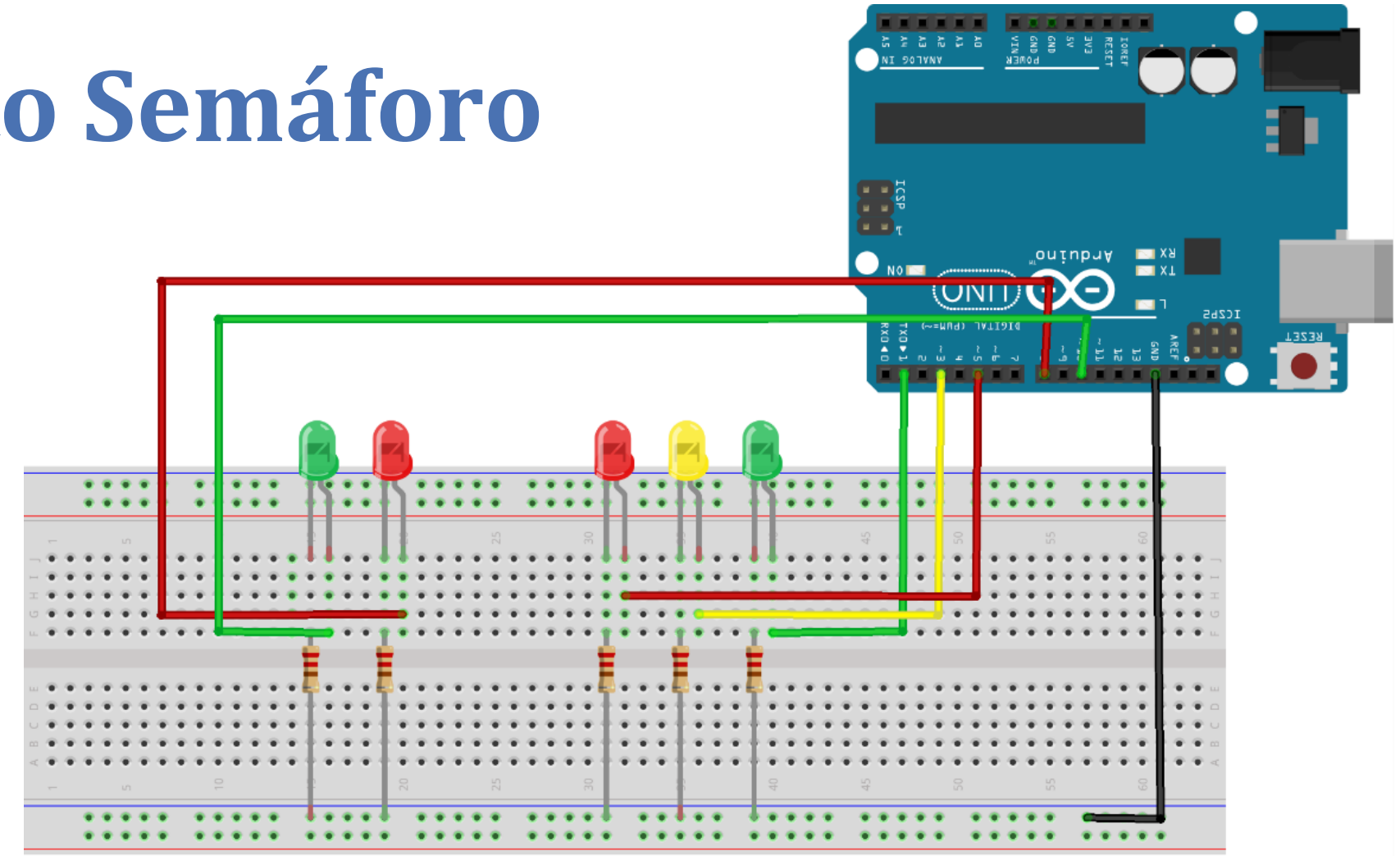
- Placa Arduino
- Protoboard
- LEDs
- Cabos Jumper
- Resistores (opcional)

Projeto Semáforo

A Av. Fernando Ferrari possui, no ponto analisado para a construção do semáforo, uma largura de 12,5 m e a travessia dura, em média, o total de 5 segundos. Em horário de pico, para não haver engarrafamento, exige-se um tempo mínimo de 10 segundos.

Antes de sinalizar a parada dos carros, você deve sinalizar um alerta, tanto para os carros, como também para os pedestres. Um semáforo para carros, dispõe de LED vermelho, amarelo e verde, enquanto para pedestres, LEDs verde e vermelho. Pisque o sinal vermelho intermitentemente para sinalizar alerta para os pedestres, use o sinal amarelo para sinalizar para os carros.

Projeto Semáforo



Projeto Semáforo

```
int R1 = 2; //sinal vermelho para carros
int G1 = 4; //sinal verde para carros
int Y = 6;  //sinal amarelo para carros
int R2 = 8; //sinal vermelho para pedestres
int G2 = 10;//sinal verde para pedestres

void setup() {
  // Pinos de saida
  pinMode(R1,OUTPUT);
  pinMode(R2,OUTPUT);
  pinMode(G1,OUTPUT);
  pinMode(G2,OUTPUT);
  pinMode(Y,OUTPUT);
}
```

Projeto Semáforo

```
void loop() {
  // Carros: Verde
  // Pedestres: Vermelho
   digitalWrite(G1,HIGH);
   digitalWrite(R2,HIGH);
   delay(10000);

// Carros: Amarelo
// Pedestres: Vermelho
   digitalWrite(G1,LOW);
   digitalWrite(Y,HIGH);
   delay(2000);

 // Carros: Vermelho
  // Pedestres: Verde
  digitalWrite(Y,LOW);
  digitalWrite(R2,LOW);
  digitalWrite(R1,HIGH);
  digitalWrite(G2,HIGH);
  delay(5000);

// Carros: Vermelho
//Pedestres: Alerta
  digitalWrite(G2,LOW);
  for(int i=0;i<10;i++)
  {
    if (i%2!=0){
      digitalWrite(R2,HIGH);
      delay(200);
      }
    else{
      digitalWrite(R2,LOW);
      delay(200);
      }
    }
   digitalWrite(R1,LOW);
}
```

Projeto Buzzer

Você percebeu que anda tendo muita dificuldade para acordar todos os dias às 5 horas da matina para pegar um ônibus lotado e ir para a faculdade estudar. O despertador do seu celular estragou depois que ele caiu na água e você dispõe de pouco recurso para manda-lo para uma assistência técnica especializada.

No entanto, para um projeto de alguma disciplina passada, você teve de comprar alguns dispositivos eletrônicos para sinalizar ou medir qualquer coisa. E foi aí que você teve um grande Insight!

Projeto Buzzer

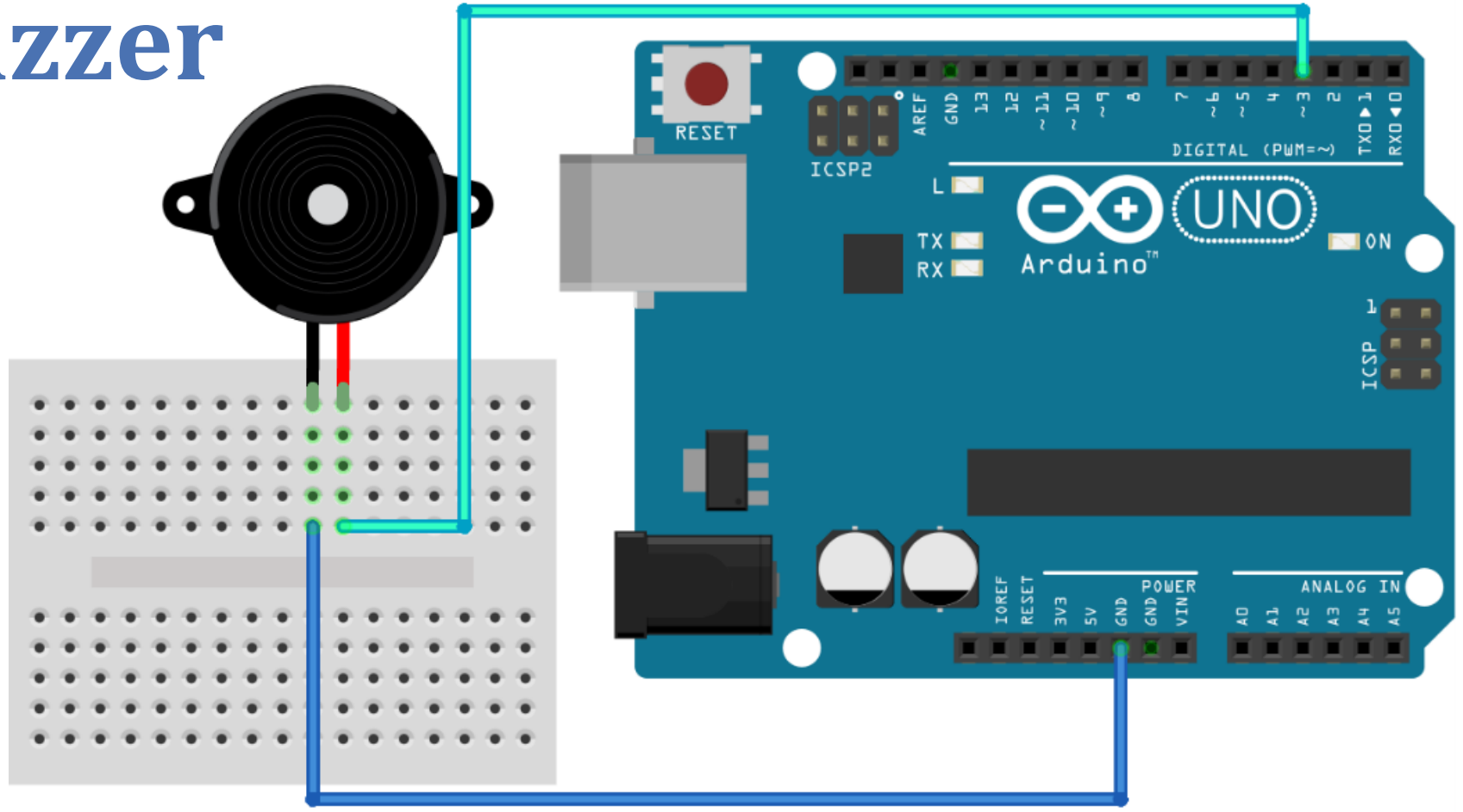
Construa um alarme despertador usando apenas:

- Uma placa Arduino
- Um protoboard
- Um buzzer
- Alguns cabos Jumper

Projeto Buzzer

Você pode começar com um alarme de frequência única, que seja intermitente, mas com certeza você vai levar um baita susto quando acordar de manhã e vai passar o dia mal humorado. Então tente algo mais suave, parecido como uma sirene de ambulância, aproveite para acordar os seus vizinhos também.
Depois que você definir a melodia, consegue definir o alarme para tocar em 1 minuto a partir do tempo de início do programa?

Projeto Buzzer



Projeto Buzzer

Para mudar a frequência do sinal enviado para a porta escolhida, você deve se lembrar da função **tone(***porta,frequência***)**, que já foi explicada anteriormente nesta apostila**.** Para variar o valor de uma função suavemente, você vai precisar de usar uma função harmônica, tente a função **sin(***valor em radiano***)**.
Para programar um horário, você possui vários caminhos. Mas a função **millis()** será útil se você quiser medir o tempo de execução do seu programa.

Projeto Buzzer

```
float seno;
int frequencia;

void setup() {
 //define o pino 9 como saída
 pinMode(3,OUTPUT);
 pinMode(5,OUTPUT);
}
```

```
void loop() {
 for(int x=0;x<180;x++){
  //converte graus para radiando e depois obtém
o valor do seno
  seno=(sin(x*3.1416/180));
  //gera uma frequência a partir do valor do
seno
  frequencia = 2000+(int(seno*1000));
  tone(3,frequencia);
  tone(5,2*frequencia);
  delay(2);
 }
}
```

Projeto Alarme Refrigerador

Problema: Uma câmara de refrigeração possui um sensor de temperatura e um LDR, e uma lâmpada fica acessa sempre que o refrigerador está ligado, nesse caso você precisa fazer algo usando o NTC 10k e o LDR de forma que seja informado caso uma queda de energia ocorra (lâmpada apague) e a temperatura comece a subir, para que medidas sejam tomadas rapidamente de forma que não estrague os produtos.

Dados: Com a biblioteca indicada pode-se ler o valor coletado pelo NTC com o comando thermistor.read(), e é necessário criar um objeto no ambiente Setup do tipo THERMISTOR com os seguintes parâmetros; pino, valor da resistência do termistor, coeficiente do termistor (usaremos 3950), valor da resistência do resistor utilizado.

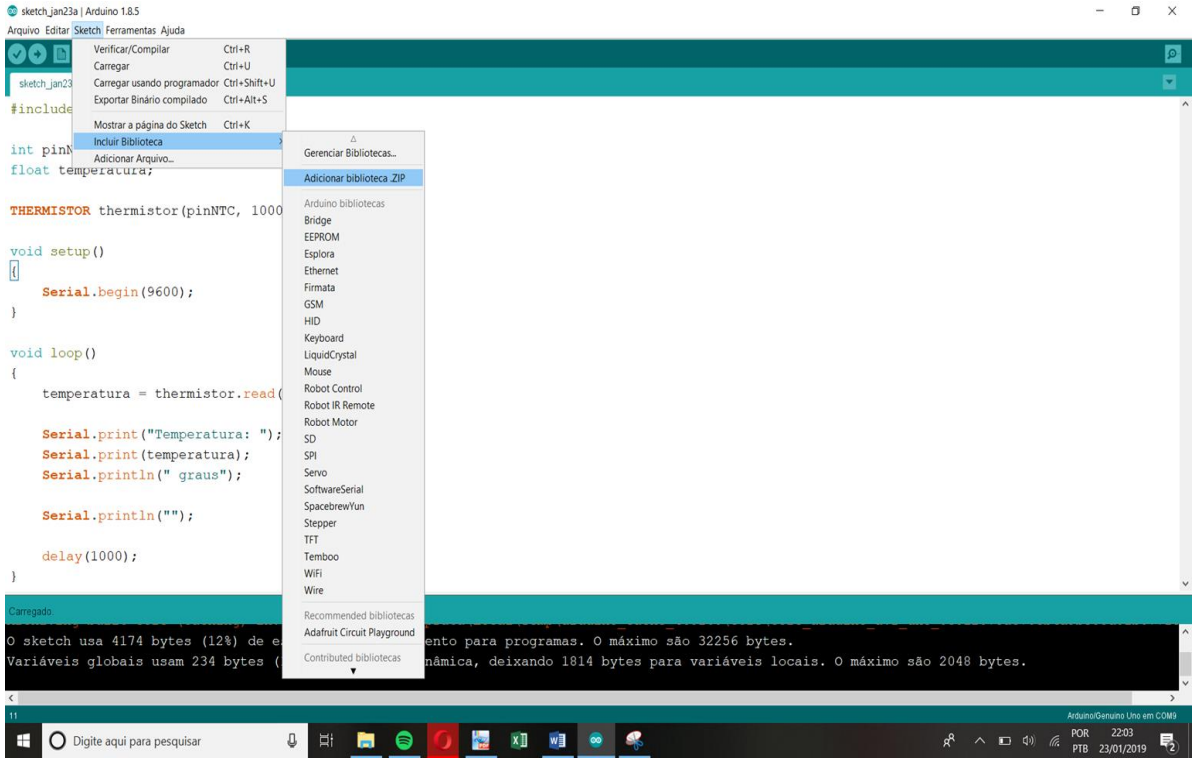
Exemplo: THERMISTOR thermistor (pinNTC, 10000, 3950, 10000);

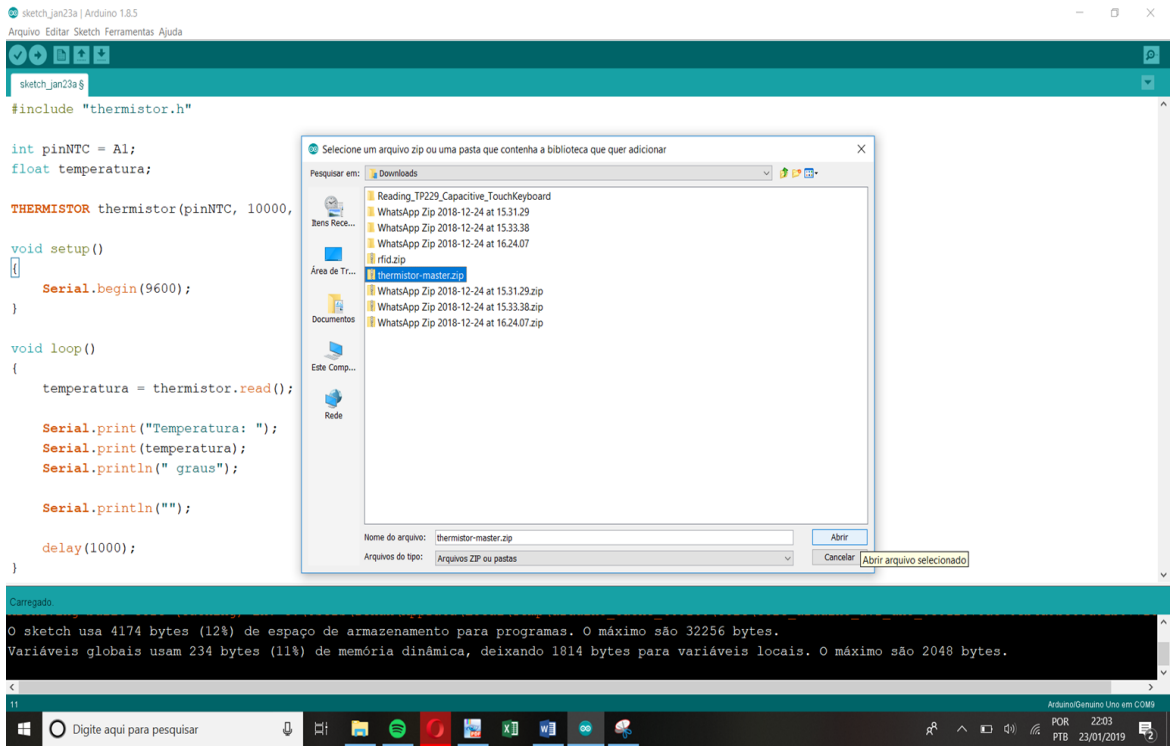
Componentes:

- 1 LED
- 1 LDR
- 1 NTC 10K
- 2 RESISTORES 1K
- 1 RESISTOR DE 10K
- 1 PROTOBOARD
- JUMPERS

Biblioteca:
https://github.com/NormanFriemann/thermistor

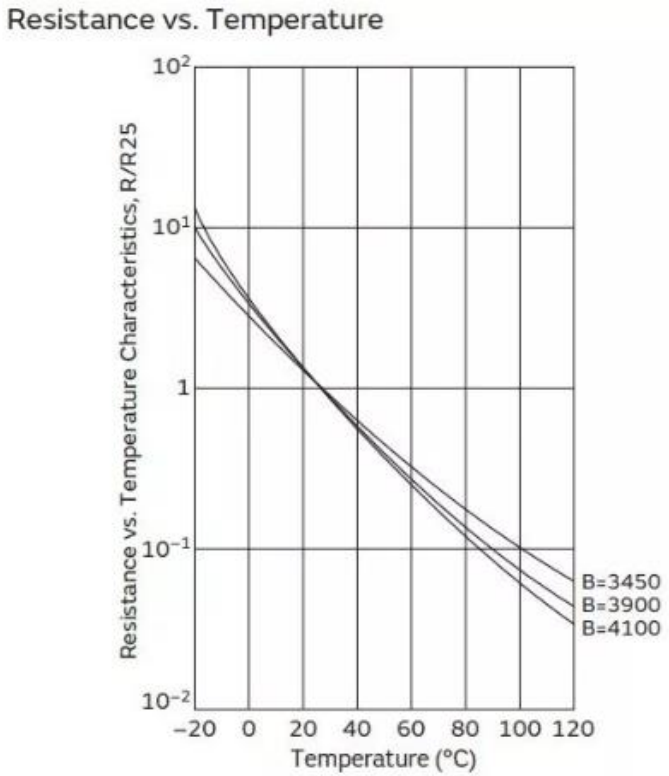
Biblioteca thermistor





Termistor

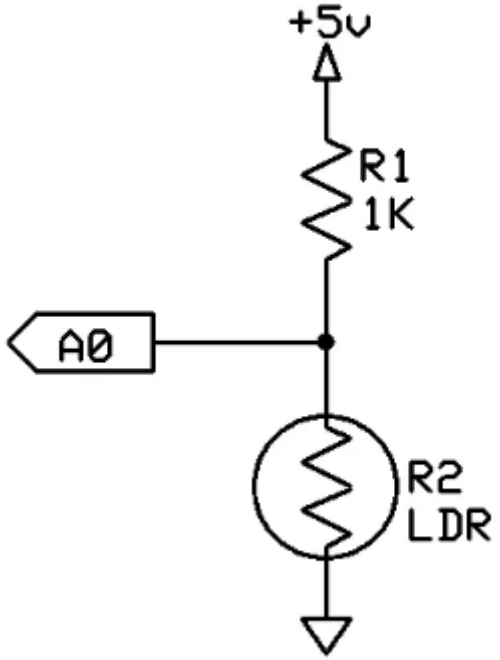
- Esses componentes quando experimentam variações de temperatura no ambiente em que estão variam também a sua resistência
- Baixo custo, são encontrados em impressoras, eletrodomésticos e etc.
- Termistores do tipo NTC (Negative Temperature Coefficient) tem a sua resistência diminuída quando experimentam um aumento na temperatura.

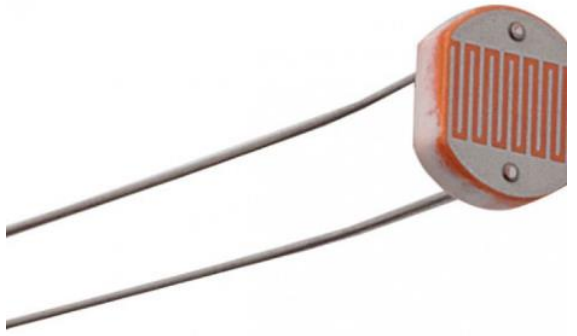


NTC 10K

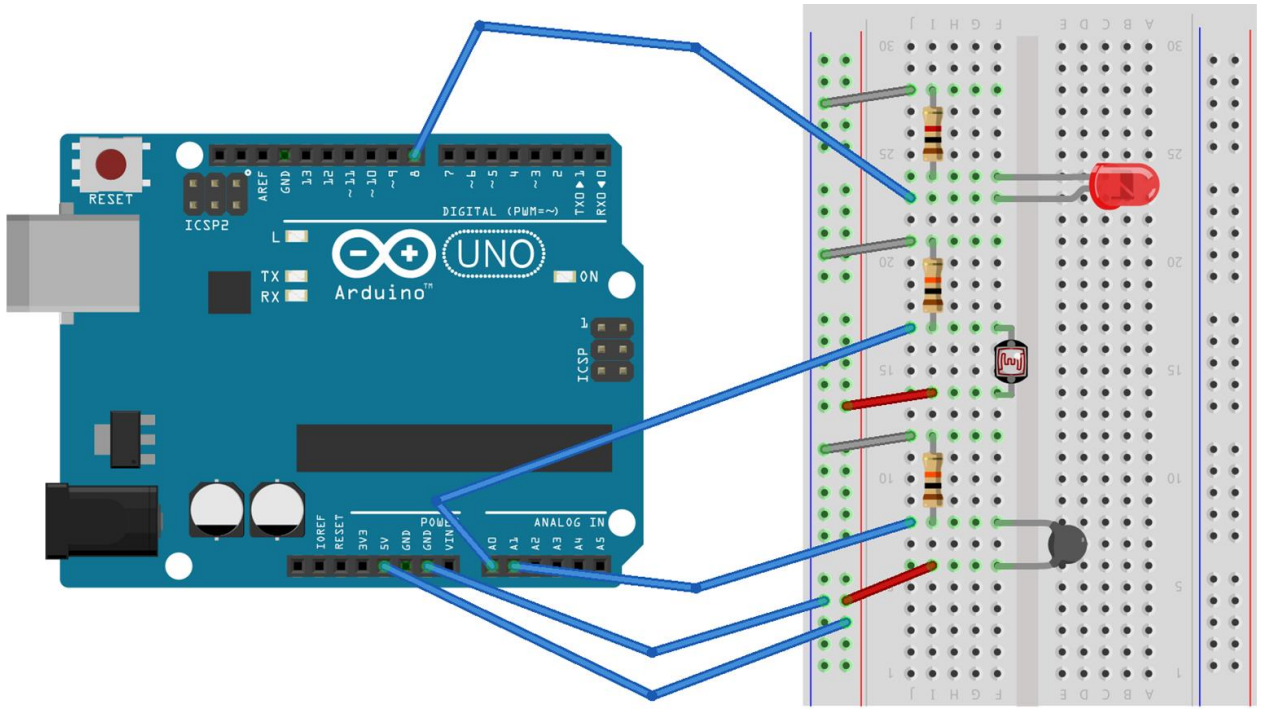
LDR

- O LDR é um semicondutor de alta resistência que tem sua resistividade quase zero quando está exposto a uma grande quantidade de luz.
- Quanto menor luminosidade maior a resistência do LDR;
- Quanto maior a resistência do LDR, maior a tensão em A0;
- Então: quanto menor a luminosidade maior é a tensão em A0.





Circuito Alarme Refrigerador



Código Alarme Refrigerador

```
#include "thermistor.h"
int pinoLED = 8; // o pino 8 será utilizado para acender o LED
int pinoLDR= A0; //  o pino A0 será utilizado para receber os dados do LDR
int pinNTC = A1; // o pino A1 será utiliado para receber os dados do NTC
float temperatura; // variavel para armazenar a leitura do NTC
int luminosidade; // variavel para armazenar a leitura do LDR

THERMISTOR thermistor(pinNTC, 10000, 4300, 10000); // objeto thermistor com parametros

void setup()
{
  pinMode(pinoLED,OUTPUT);// pino 8 como pino de saida
    Serial.begin(9600);// incialização da leitura de dados do arduino via USB
}

void loop()
{
 temperatura = thermistor.read(); //armazena o valor lido numa variavel
    Serial.print("Temperatura: ");// escreve nos serial monitor
    Serial.println(temperatura);//escreve o valor lido pelo NTC
  Serial.print("Luminosidade: ");//escreve no serial monitor
  luminosidade = analogRead(pinoLDR); //armazena o valor lido numa variavel
  Serial.println(luminosidade);  //escreve o valor lido pelo LDR
  if(luminosidade<650)//se a luminosidade estiver baixa
```

```
if(luminosidade<650)//se a luminosidade estiver baixa
{
  digitalWrite(pinoLED,HIGH); // acende o LED
  if(temperatura >= 34){ // se a temperatura estiver alta
  Serial.print("Sinal: ");
  Serial.println("Alerta, a temperatura esta muito alta");
  Serial.println("");
}
else{ // se a temperatura estiver baixa
  Serial.print("Sinal: ");
  Serial.println("Ambiente ainda refrigerado");
  Serial.println("");
}
}
else // se a luminosidade estiver alta
{
  digitalWrite(pinoLED,LOW);// apaga o LED
Serial.print("Sinal: ");
Serial.println("Luz alta"); // a lampada ta acessa, portanto tudo esta funcionando
Serial.println("");
}
delay(10000);
}
```

Projeto Alarme de presença

Problema: Buscando tornar sua residência familiar um ambiente mais seguro enquanto todos dormem, deseja-se implementar um sistema que identifique a aproximação de alguém intruso da porta de entrada de casa e que esse sistema possa sinalizar de alguma forma essa movimentação. Sabendo que o Arduino possui potencial para implementar esse tipo de programação, construa um projeto no Arduino aplicando um sensor ultrassônico HC-SR04 e um Buzzer para resolver o problema.

Dados:

1 - A função *pulseIn(pino, HIGH/LOW)* deve ser utilizada para contagem do tempo de propagação da onda. Caso o dado utilizado nela seja HIGH, a função aguarda o pino ir de LOW para HIGH, inicia a contagem, aguarda o pino ir para LOW e termina a contagem, retornando o valor do tempo em microssegundos.

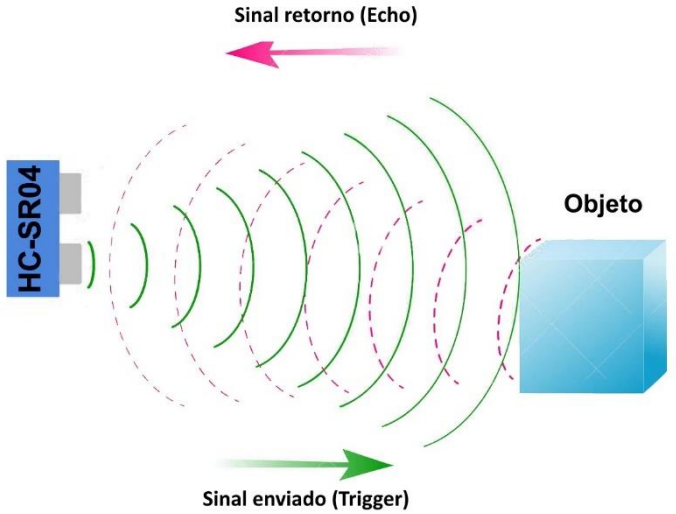
2 - Para simular o alarme, recomenda-se que seja utilizada a função tone(), já explicada no projeto da sirene.

Componentes:

- 1 HC-SR04
- 1 BUZZER ATIVO
- 1 PROTOBOARD
- JUMPERS

Sensor ultrassônico

- Utilizado para em situações que é necessário medir distâncias, como em alarmes, ou evitar colisões, como em robôs móveis.
- Mede distâncias entre 20mm e 4000mm.



HC-SR04

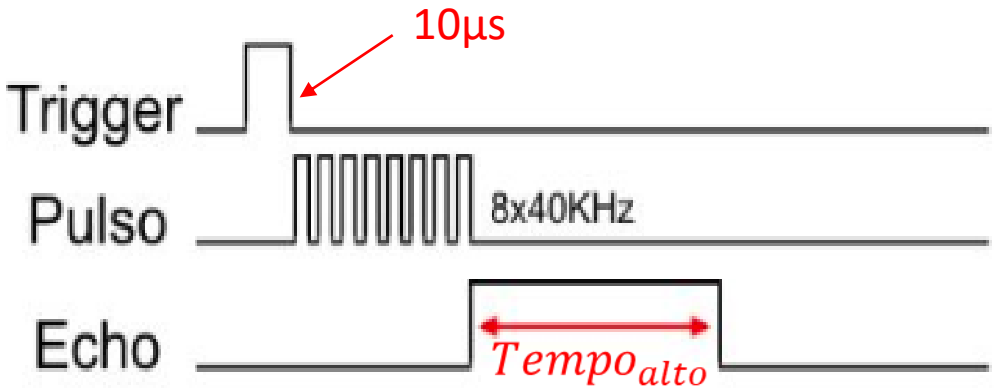
Funcionamento

Pino Trig: HIGH (10µs) → Pino Echo: HIGH (emissão sinal) → Pino Echo: LOW (recepção sinal)

Equação: $Distância = \frac{Tempo_{alto} \cdot v_{som}}{2}$

Sensor ultrassônico

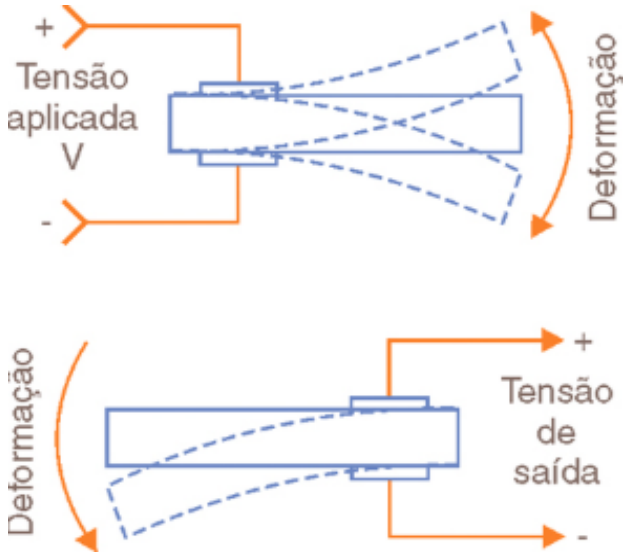
- Pulsos emitidos: 8 pulsos com 40kHz de frequência.
- Deve-se atentar em dividir o tempo de propagação por 2 visto que o sinal atinge o objeto e retorna, percorrendo a distância duas vezes.



Sinais nos pinos do HC-SR04

Buzzer

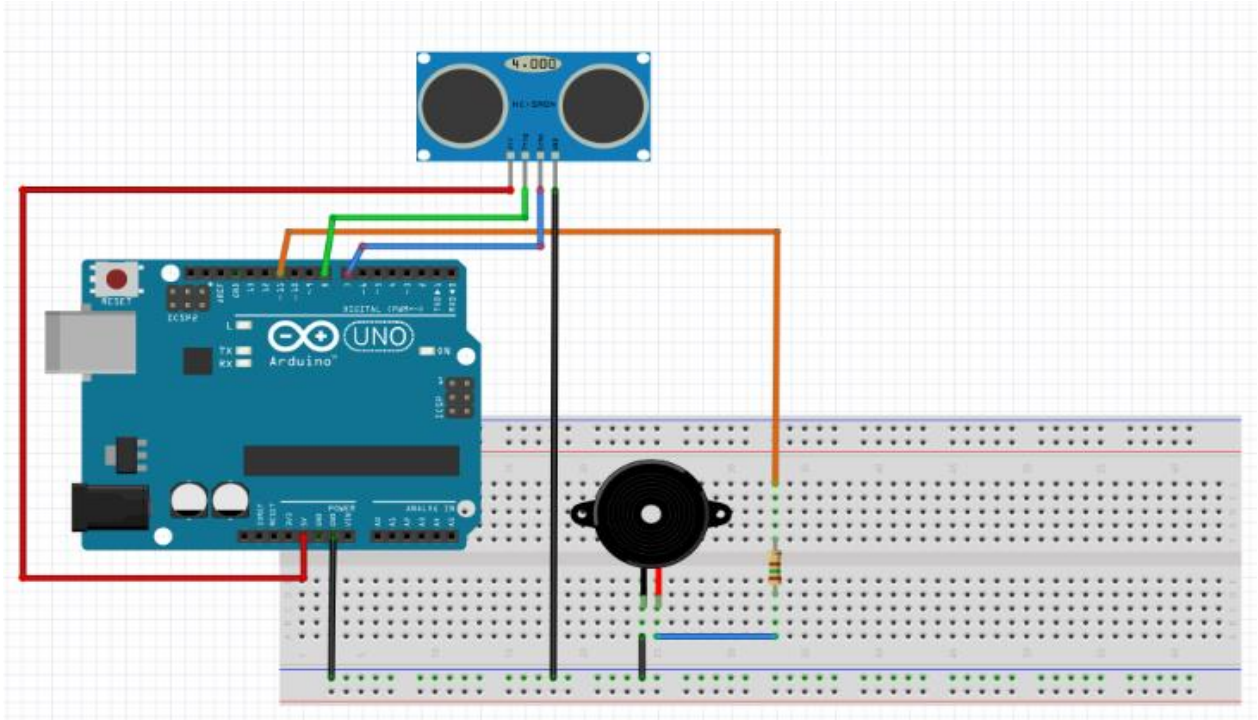
- Dispositivo utilizado para produção de som de baixa potência.
- Funciona a partir do efeito piezelétrico que ocorre na célula piezelétrica em seu interior.
- Som produzido pelas vibrações da célula piezelétrica.
- Deve-se atentar que o buzzer possui polaridade, isto é, o local onde os pinos do Arduino são conectados faz diferença.



Buzzer ativo

Efeito Piezelétrico Consiste no surgimento de uma tensão elétrica a partir de um esforço mecânico ou vice-versa.

Montagem do Circuito



Código

```
int trigPin = 8 // Declara constante trigPin
int echoPin = 7 // Declara constante como echoPin
int tempo = 10 // Declara constante de tempo
int frequencia = 0; // Inicializa variável de frequência em 0
int Buzzer = 11; // Inicializa buzzer no pino 11
int duracao, distancia; // Variável de distância e tempo

void setup() {
 Serial.begin(9600);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 pinMode(Buzzer,OUTPUT);
}

void loop() {
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
 duracao = pulseIn(echoPin, HIGH);
 distancia = (0.0343*duracao)/2; // Define base do cálculo de
conversão
 if (distancia >= 40 || distancia <= 0) { // Define as distâncias bases de
verificação
 Serial.println("Alarme desligado");
 digitalWrite(Buzzer, LOW);
 }

else {
 Serial.println("Pessoa Detectada");
 Serial.println("Distância = ");
 Serial.print(distancia); // Imprime o valor da distância no Monitor Serial

 for (frequencia = 150; frequencia < 1800; frequencia += 1) { // Tone que
produz sirene de polícia
 tone(Buzzer, frequencia, tempo);
 delay(1);
  }

 for (frequencia = 1800; frequencia > 150; frequencia -= 1) { // Tone que
produz sirene de polícia
 tone(Buzzer, frequencia, tempo);
 delay(1);
  }
 }
}
```

Projeto Trena Eletrônica

Problema: Suponha que você irá realizar uma reforma na sua casa e você precise utilizar uma trena para efetuar algumas medidas. Para usar esta ferramenta é necessário da ajuda de uma segunda pessoa, a fim de que esta segure a trena em uma ponta enquanto você lê as medidas. Infelizmente, não há uma outra pessoa contigo para te ajudar nessa empreitada. Uma alternativa bastante inteligente é montar uma trena eletrônica, já que apenas apontando um sensor de distância a um objeto é possível calcular as distâncias.

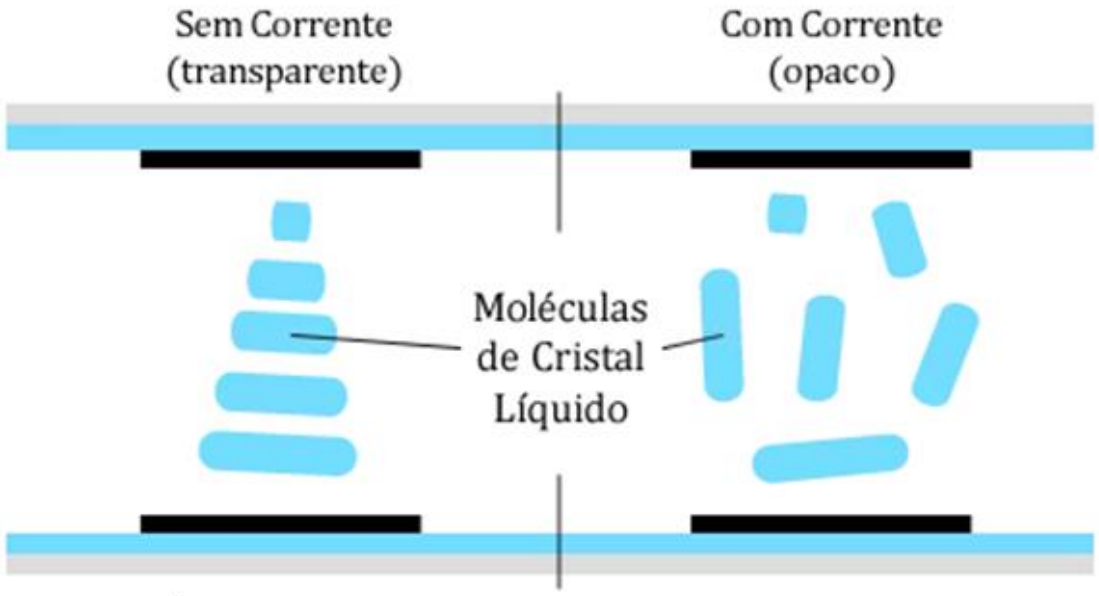
Componentes:

- 1 HC-SR04
- 1 POTÊNCIOMAETRO 10K
- 1 DISPLAY LCD 16X2
- JUMPERS

Biblioteca:
http://blog.vidadesilicio.com.br/wp-content/uploads/2014/06/Ultrasonic.zip.

Display LCD (Liquid Crystal Display)

- É formado por cristais líquidos compreendidos entre duas placas de vidro;
- Podem possuir um conjunto de LEDs por baixo da estrutura (back-light) ou não;
- Com a aplicação de uma tensão o cristal líquido deixa de ser transparente e se torna opaco.



Fonte: Apostila Vida de Silício Vol.2.

Display LCD (Liquid Crystal Display)

Os LCDs encontrados no mercado são descritos por AxB, em que, A é o número de colunas e B é o número de linhas.

Podem ser de dois tipos

- **Caracter:** Permite escrever apenas caracteres, números e pequenos símbolos cridos pelo usuário;
- **Gráficos:** Apresenta resoluções bem maiores e permite exibir figuras e imagens.

Display LCD (Liquid Crystal Display)

Display LCD Gráfico 128x64

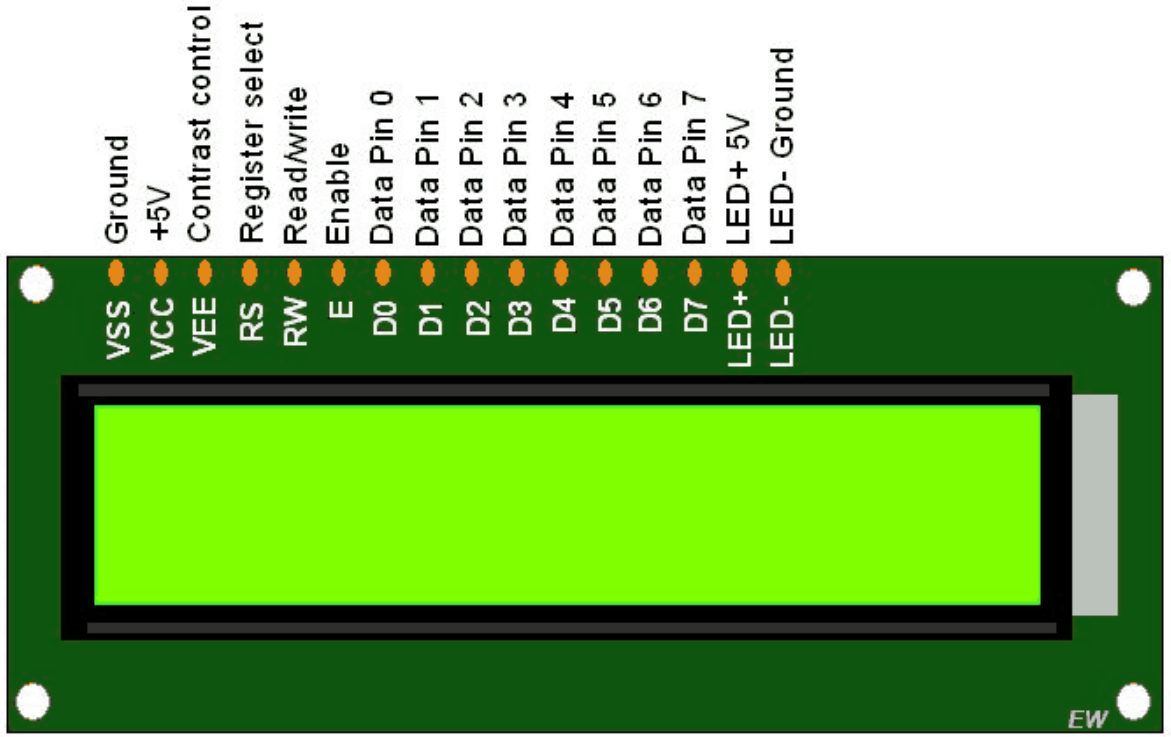
Display LCD Caracter 16x2

Display LCD Caracter 20x4

Display LCD (Liquid Crystal Display)

VEE (V0)
Tensão Ajustável

RS
Configurações (Nível 0)
Dados (Nível 1)

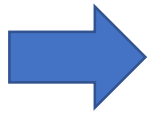


R/W
Read (Nível 1)
Write (Nível 0)

E
Ativa (Nível 1)
Desativa (Nível 0)

Biblioteca LiquidCrystal.h

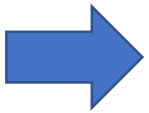
LiquidCrystal tela (RS, Enable, DB4, DB5, DB6, DB7);



```
LiquidCrystal lcd(2,3,4,5,6,7);
/*Cria objeto lcd da classe LiquidCrystal
RS 2
Enable 3
DB4 4
DB5 5
DB6 6
DB7 7
*/
```

Biblioteca LiquidCrystal.h

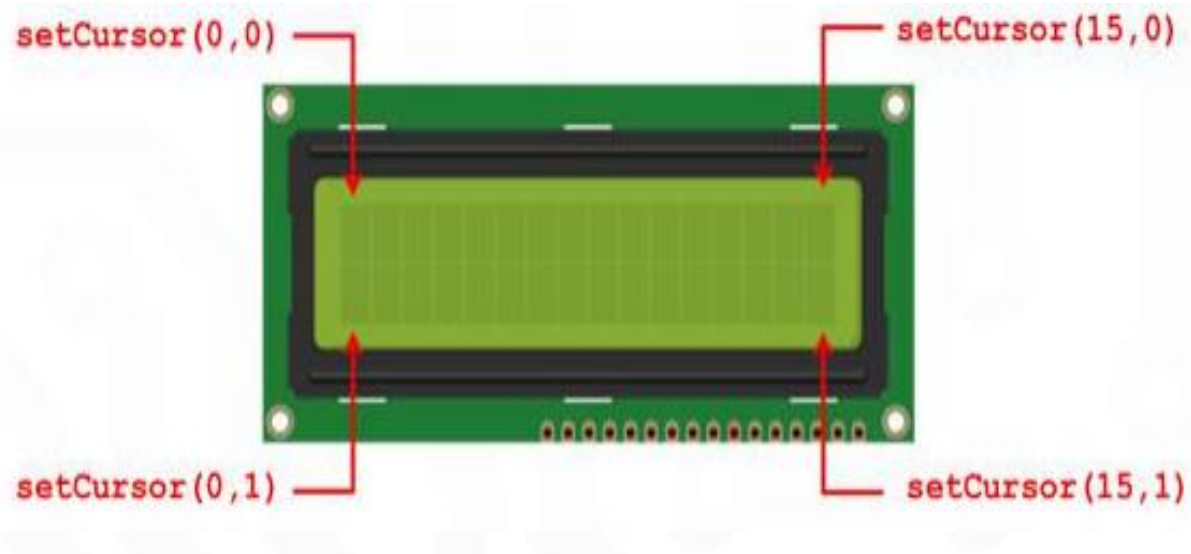
begin (colunas, linhas);



```
lcd.begin(16,2);
```

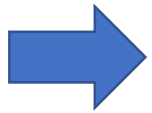
Biblioteca LiquidCrystal.h

setCursor(coluna, linha);



Biblioteca LiquidCrystal.h

print (conteúdo, base);

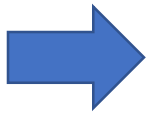


```
lcd.print("Hello World");
```

BIN	1101
HEX	F2A7
OCT	4701
DEC	9825

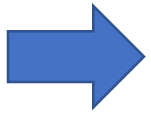
Biblioteca LiquidCrystal.h

clear();



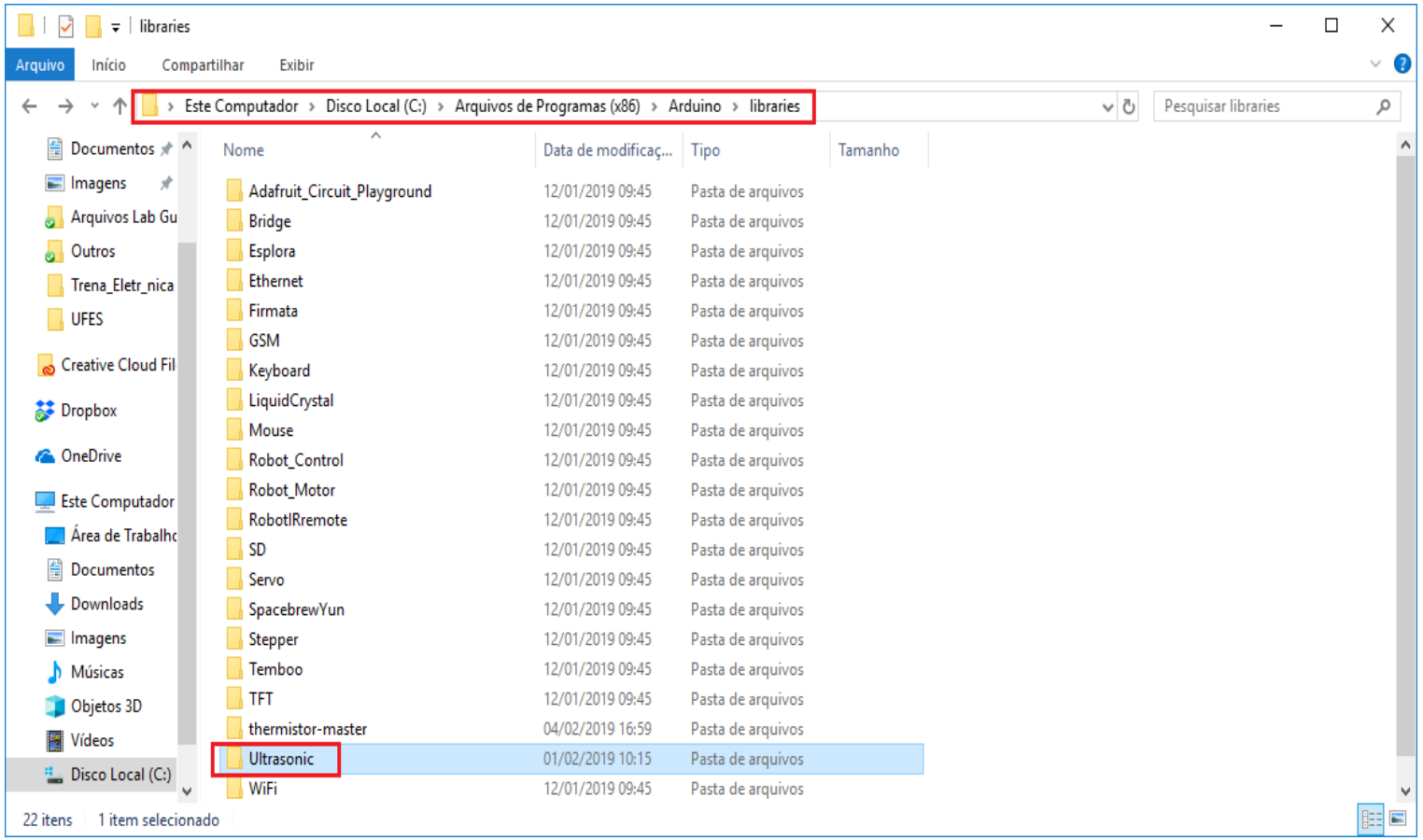
```
lcd.clear();
```

home();

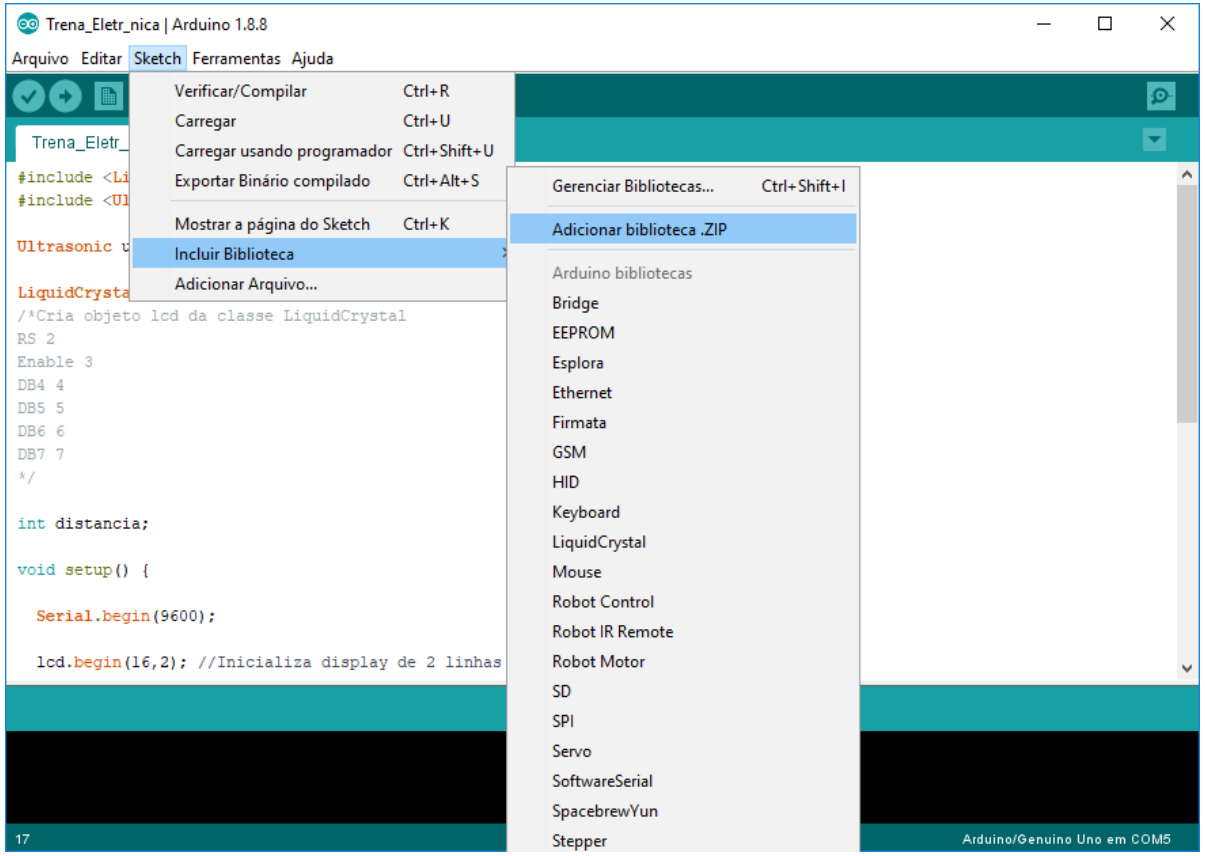


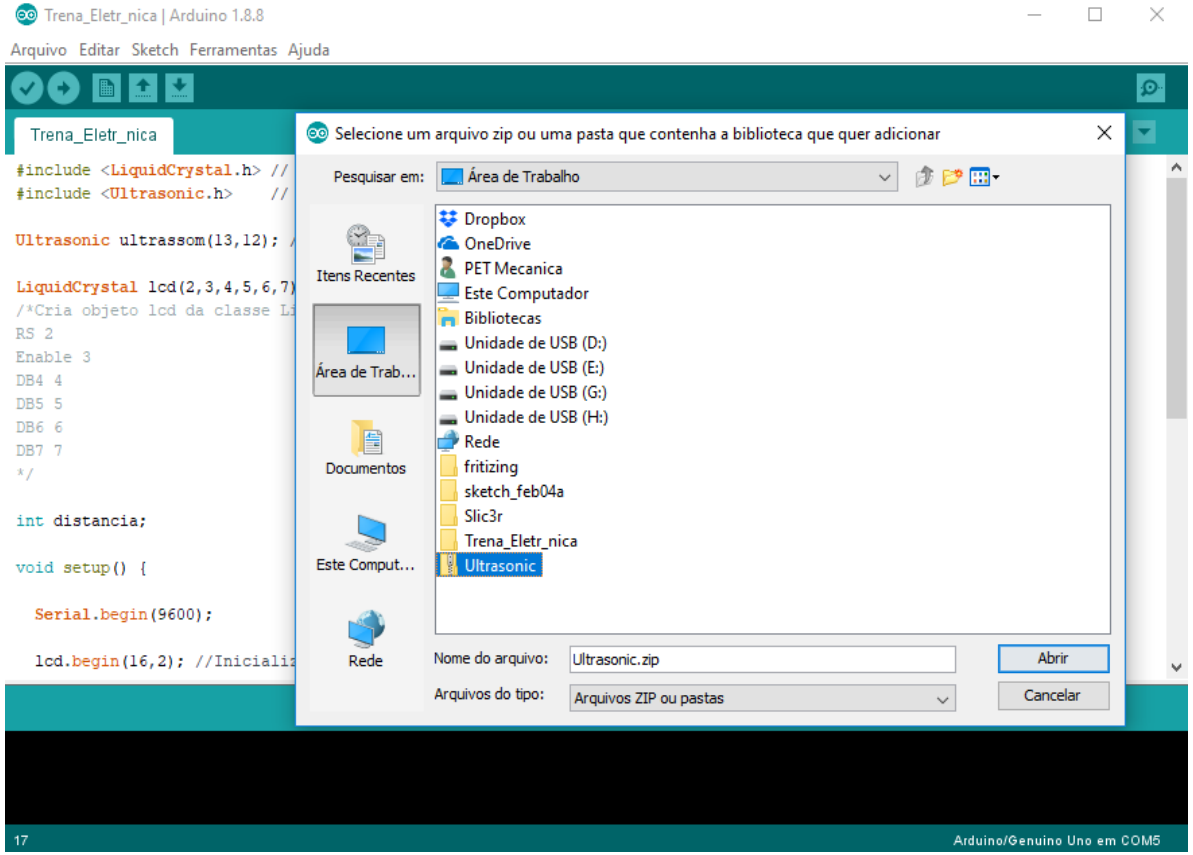
```
lcd.home();
```

Biblioteca Ultrasonic.h



Biblioteca Ultrasonic.h





Biblioteca Ultrasonic.h

Ultrasonic ultrassom (trig, echo);

```
Ultrasonic ultrassom(13, 12);
/*
Trig conectado ao pino digital 13
Echo conectado ao pino digital 12
*/
```

Biblioteca Ultrasonic.h

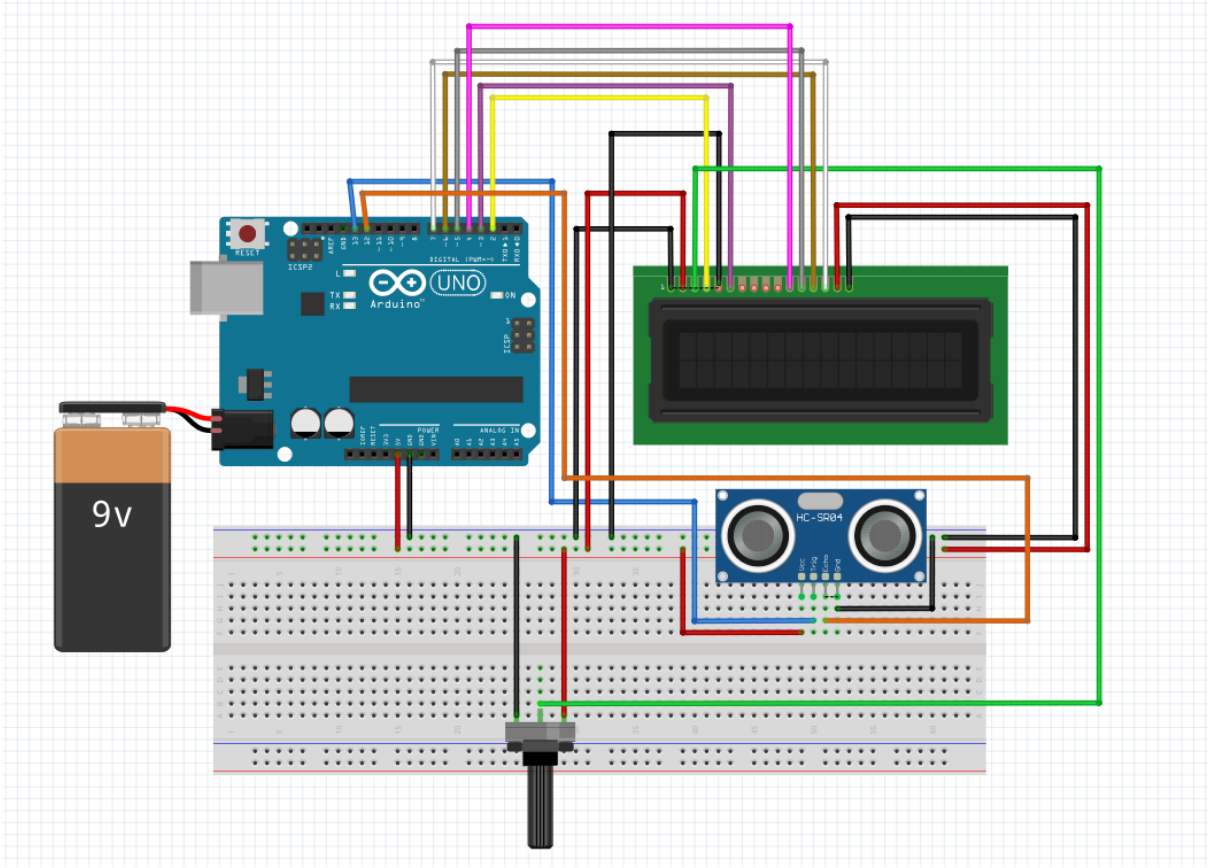
Ranging (unidade)

```
nome_sensor.Ranging(CM);
```

CM → Centímetros

INC → Polegadas

Montagem do Circuito



Código

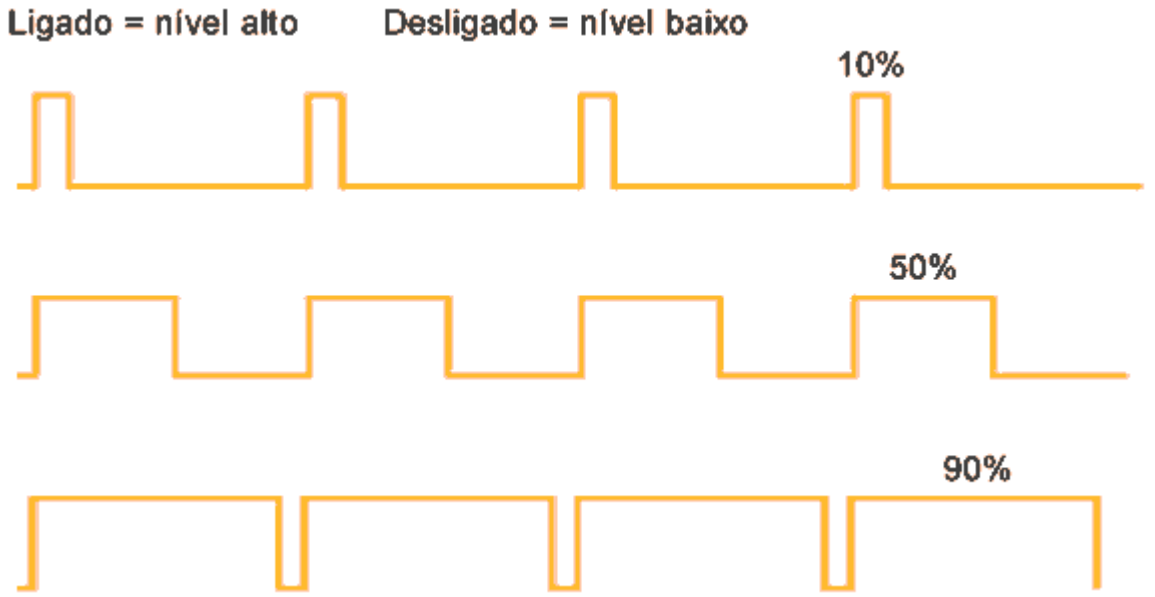
```
#include <LiquidCrystal.h> // Chama a biblioteca do Dysplay LCD
#include <Ultrasonic.h>    // Chama a biblioteca do Ultrassom

Ultrasonic ultrassom(13,12); // define o nome do
sensor(ultrassom)
/*
 Trig conectado ao pino 13
 Echo conectado ao pino 12
 */

LiquidCrystal lcd(2,3,4,5,6,7);
/*Cria objeto lcd da classe LiquidCrystal
RS 2
Enable 3
DB4 4
DB5 5
DB6 6
DB7 7
*/

int distancia;
```

```
void setup() {

  Serial.begin(9600);

  lcd.begin(16,2); //Inicializa display de 2 linhas x 16 colunas


}

void loop() {

  lcd.setCursor(4,0); //Posiciona o cursor na posição (6,1)
  lcd.print("Dist obj");

  distancia = ultrassom.Ranging(CM);
  Serial.println(distancia);
  delay(1000);

  lcd.setCursor(0,1);
  lcd.print("d = ");
  lcd.setCursor(4,1);
  lcd.print(distancia);
  lcd.setCursor(8,1);
  lcd.print(" cm");
  delay(500);
  lcd.clear(); //Limpa a tela do LCD
}
```

PWM

- Modulação por largura de pulso
- Sinais analógicos através de sinais digitais
- Controle de potência ou velocidade
- Aplicações: motores, aquecedores, LEDs
- Duty cycle → percentual do ciclo que o pulso está em alto
- Controle da tensão média sobre a carga

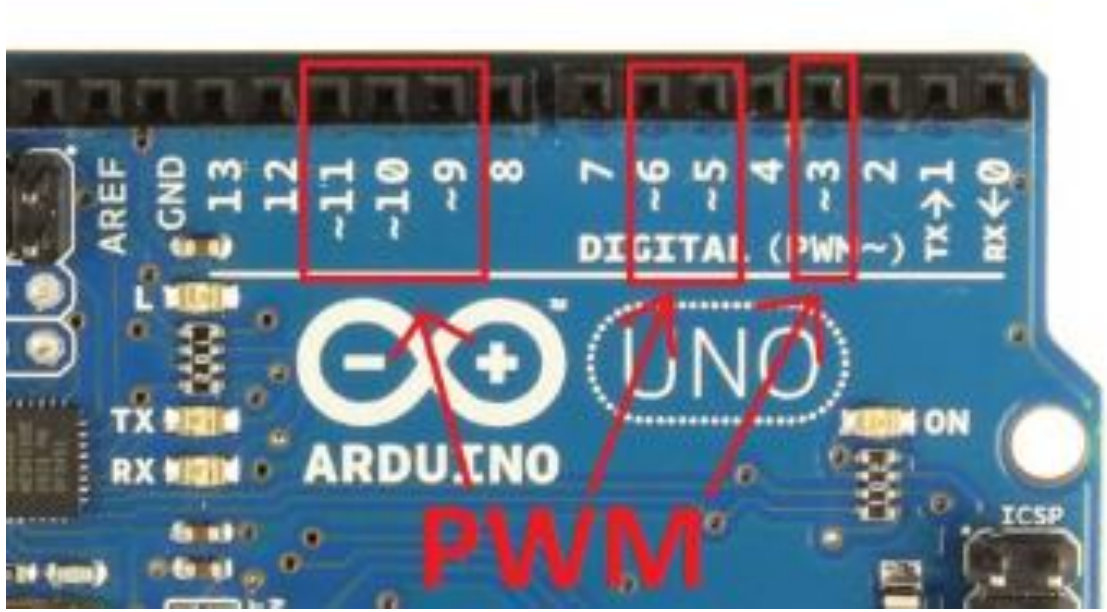
Equação: $Vout = \left(\frac{Duty\ cycle}{100}\right) \cdot Vcc$



PWM no Arduino

- Pinos digitais com o símbolo (~)
- Frequência dos pinos → 490Hz
- Resolução → 8 bits
- Duty cycle no Arduino → número entre 0 e 255

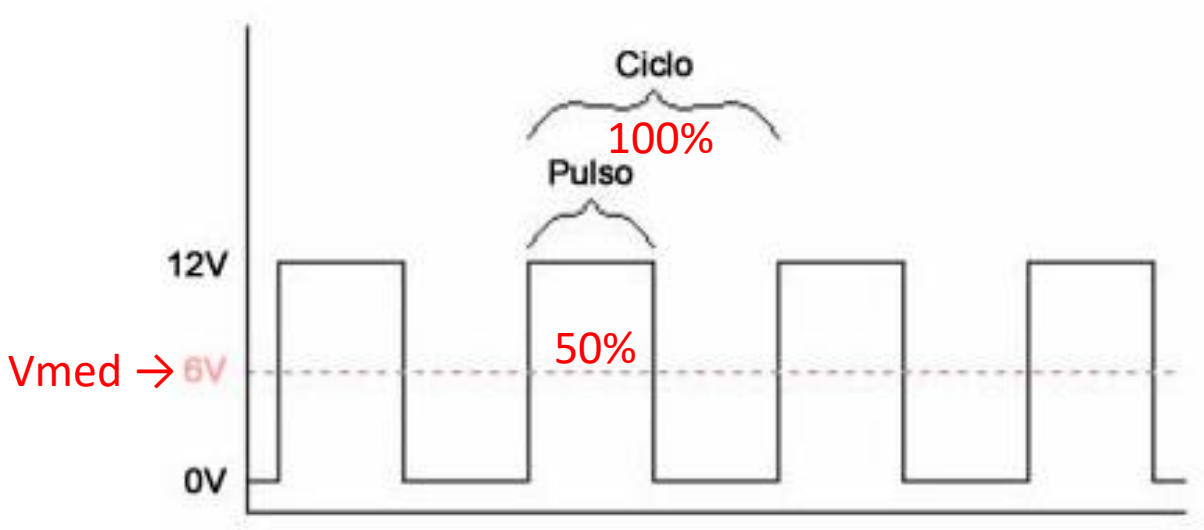
Equação: $Vout = \left(\frac{Bits}{255}\right) \cdot V_{5V}$



PWM no Arduino

- Função para enviar sinais PWM → analogWrite
- Sintaxe: analogWrite(pino,valor)

Onde:
pino – pino escolhido para enviar sinal
valor – duty cycle: 0 a 255



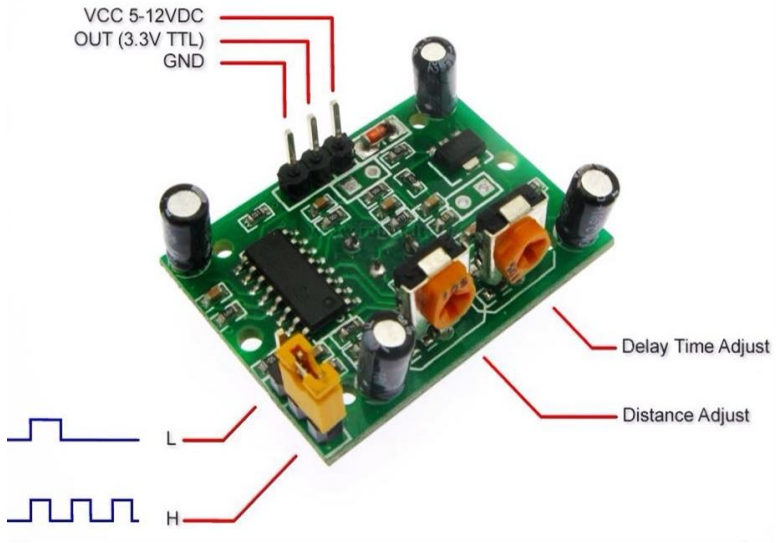
Projeto Relé e Sensor PIR

Problema: Para que se tenha visibilidade ao passar pela escadaria em um prédio, é necessário que o sensor de presença detecte quem esteja passando pela mesma e, em seguida, envie um sinal para o relé acender a lâmpada da escadaria. Monte um circuito que represente essa situação acendendo um LED.

Componentes:

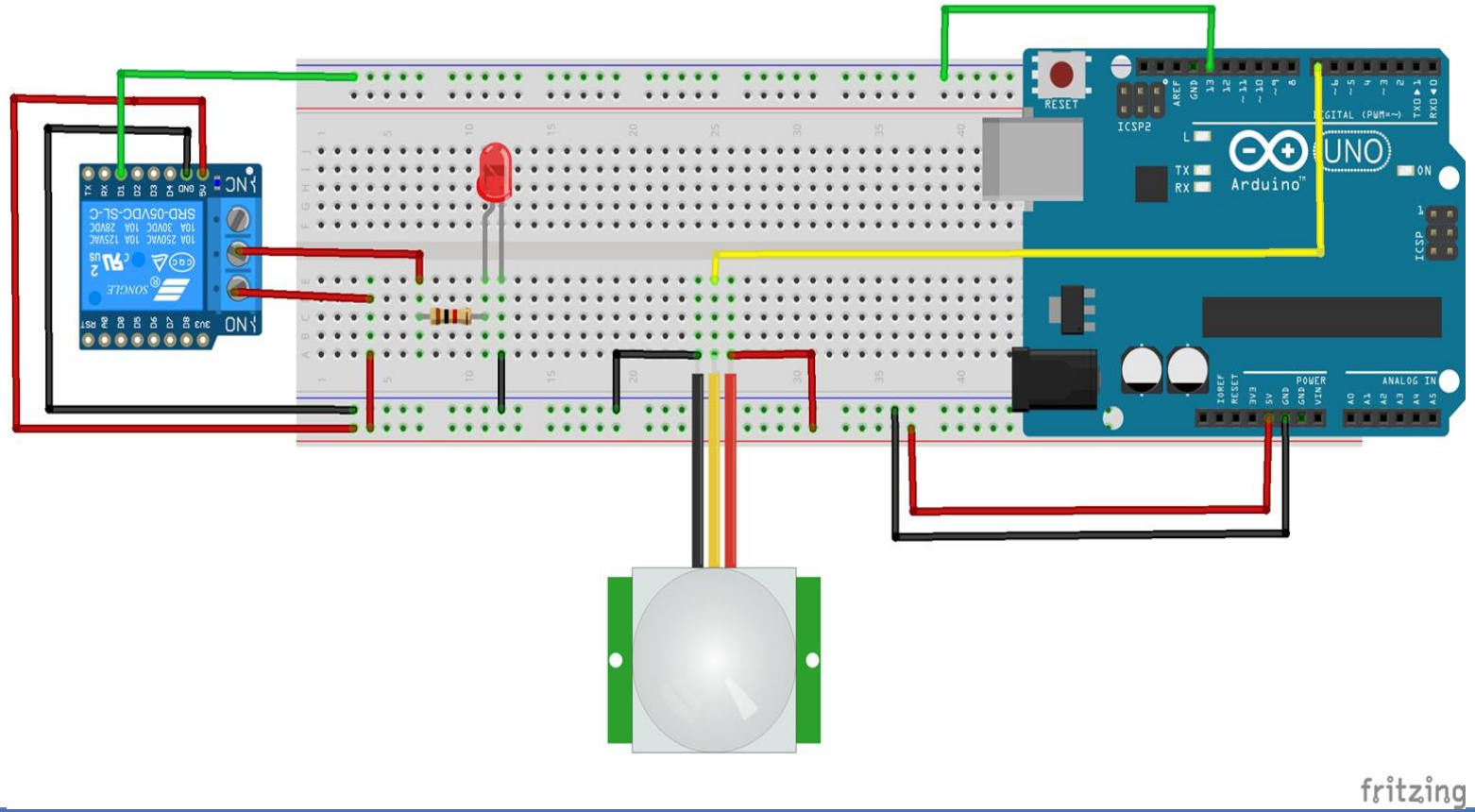
- 1 Módulo Sensor PIR HC-SR501;
- 1 resistor de 1kΩ;
- 1 Módulo Relé;
- Protoboard;
- Placa UNO;
- Jumpers.
- 1 LED;

Módulo Relé



Sensor PIR

Montagem do Circuito



Código

```
int pinoRele = 13;
int sensor = 7;
bool estadoSensor;

void setup(){
  pinMode(pinoRele, OUTPUT);
  pinMode(sensor, INPUT);
}

void loop(){
  estadoSensor = digitalRead(sensor);

  if (estadoSensor == HIGH)
  {
      digitalWrite(pinoRele, LOW);
  }
  else
  {
      digitalWrite(pinoRele, HIGH);
  }
}
```

Projeto Controle de Motor

Problema: Uma perna mecânica é articulada com um motor DC e deseja-se controlar o sentido de rotação e a velocidade de rotação utilizando, respectivamente, dois push buttons e um potenciômetro. Você deve projetar um circuito que controle esta perna atendendo os seguintes objetivos:
- Quando um dos botões for pressionado, o motor gira no sentido horário;
- Quando pressionar o outro botão, o motor gira no sentido anti-horário;
- A velocidade de rotação deve ser controlada pelo potenciômetro.

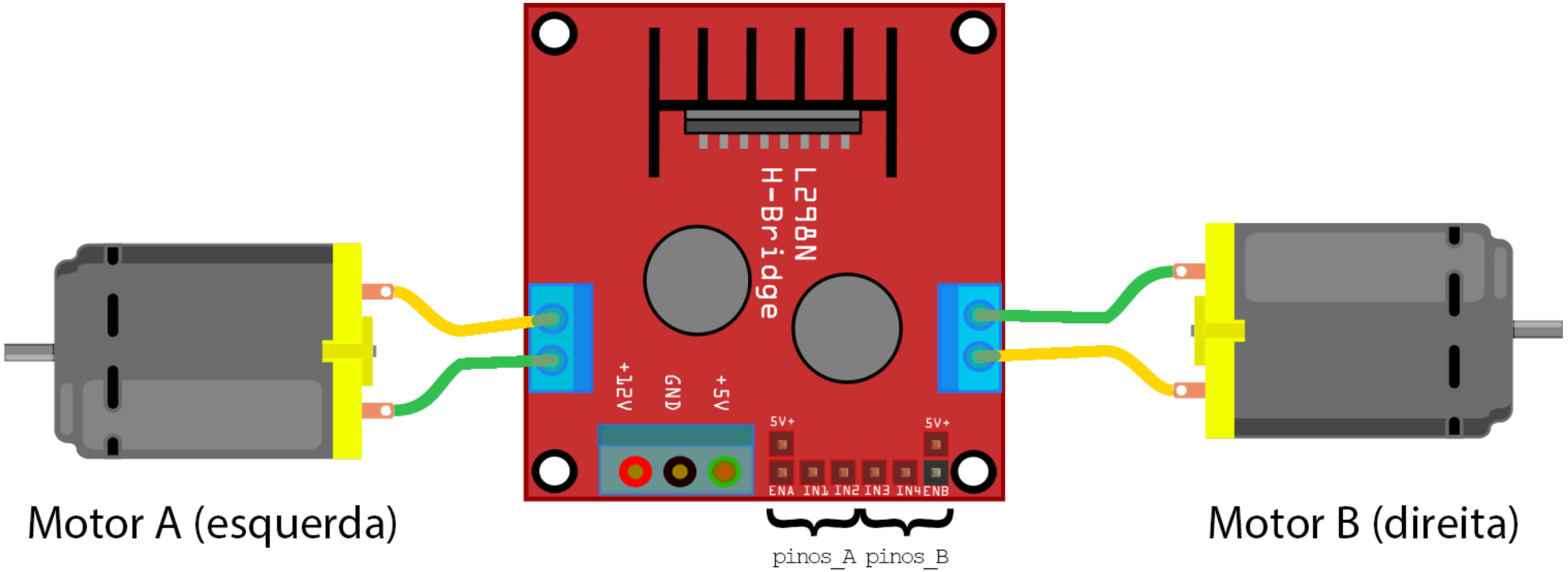
Dados: Para ler os botões é possível utilizar o modo `INPUT_PULLUP` com o botão conectado ao `GND`. Atenção: este modo faz com que a leitura do pino seja mantida no estado `HIGH` até que botão seja pressionado e mude seu estado para `LOW`.

Para usar este modo, basta utilizar a função `pinMode(pino, INPUT_PULLUP)` e ligar o botão ao GND e ao pino desejado.

Exemplo: `pinMode(8, INPUT_PULLUP);`

Componentes:
- 1 MOTOR DC
- 1 PONTE-H L298N
- 1 POTENCIÔMETRO 10K
- 2 PUSH-BUTTONS
- 1 PROTOBOARD
- JUMPERS MACHO-MACHO
- JUMPERS FÊMEA-MACHO

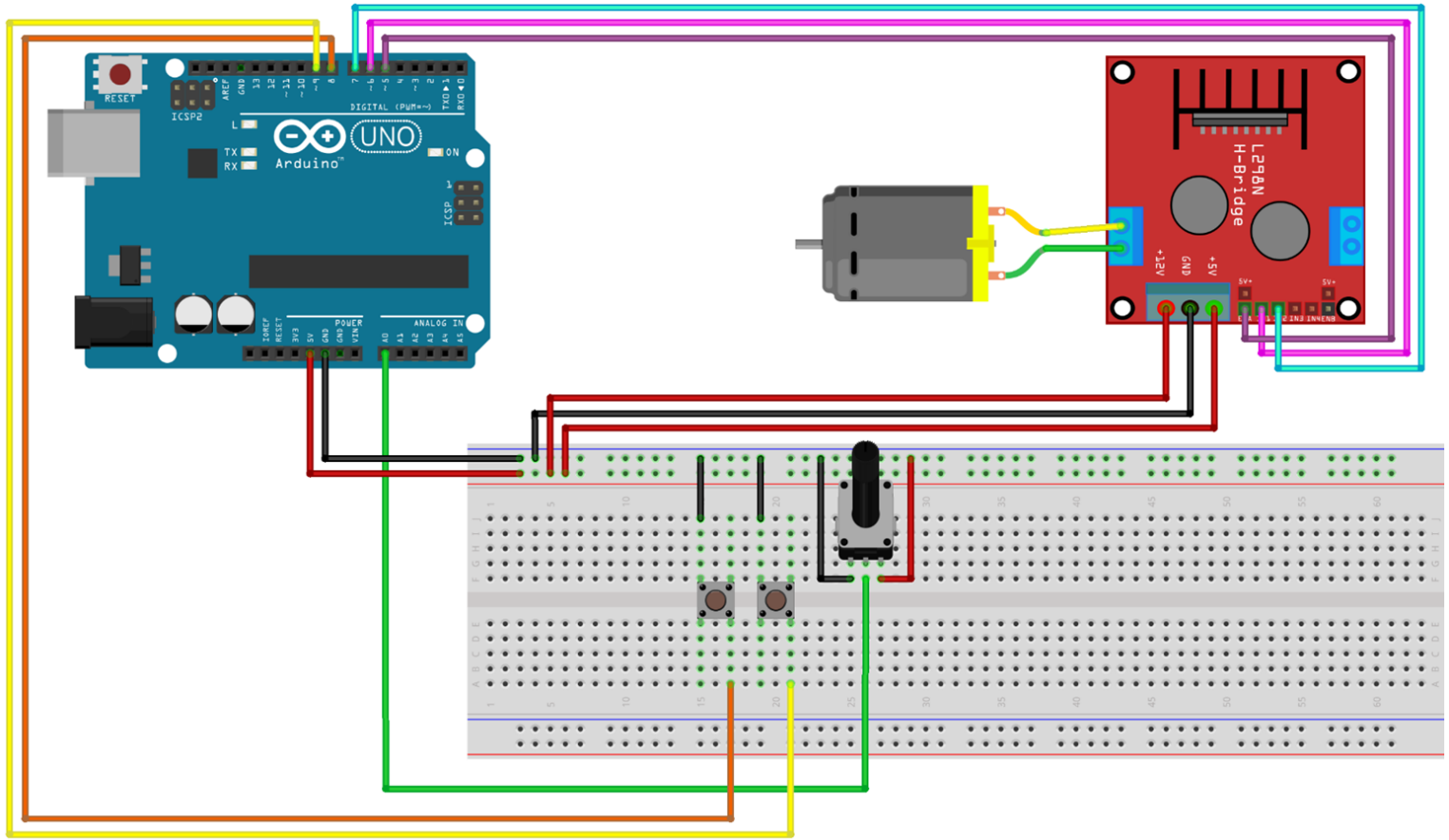
Ponte-H L298N



Ponte-H L298N

IN1	IN2	Sentido de Giro
LOW (0V)	LOW (0V)	Desligado
HIGH (5V)	LOW (0V)	Horário
LOW (0V)	HIGH (5V)	Anti-horário
HIGH (5V)	HIGH (5V)	Travado

Montagem do Circuito



Código

```
int leit_pot;
int enableA;
int but_hora;
int but_antihora;

void setup() {
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,INPUT_PULLUP);
  pinMode(9,INPUT_PULLUP);
}

void loop() {

  // Leitura dos botões

  but_hora = digitalRead(8);
  but_antihora = digitalRead(9);

  // Sentido de giro

  if(but_hora == 1 & but_antihora == 1){
    digitalWrite(6,LOW);
    digitalWrite(7,LOW);
    } // motor desligado

  if(but_hora == 0 & but_antihora == 1){
    digitalWrite(6,HIGH);
    digitalWrite(7,LOW);
    } // motor gira no sentido horário

  if(but_hora == 1 & but_antihora == 0){
    digitalWrite(6,LOW);
    digitalWrite(7,HIGH);
    } // motor gira no sentido anti-horário

  if(but_hora == 0 & but_antihora == 0){
    digitalWrite(6,HIGH);
    digitalWrite(7,HIGH);
    } // motor travado

  // Velocidade do motor

    leit_pot = analogRead(A0);
    enableA = map(leit_pot,0,1023,0,255);
    analogWrite(5,enableA);

}
```