minicurso

ARDUINO

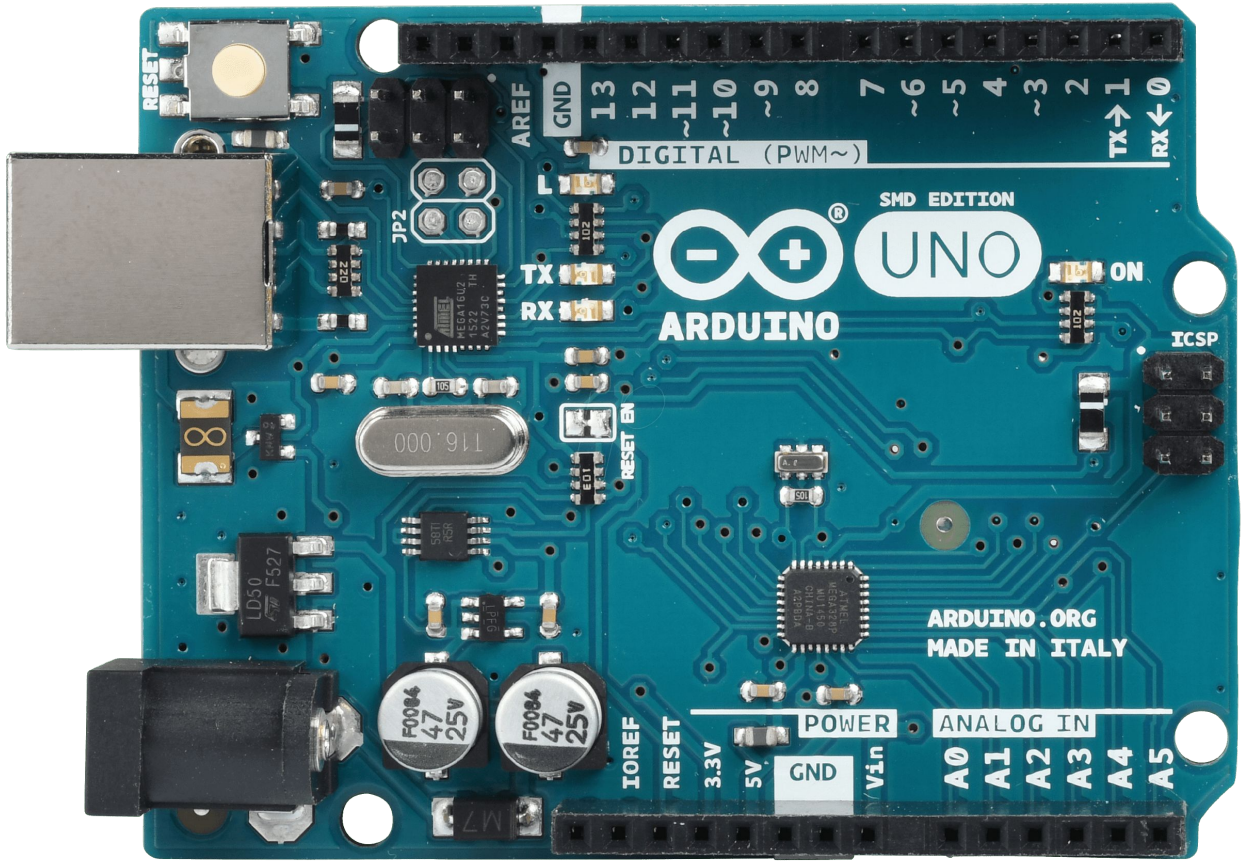
SEMANA DE ATUALIZACAO E TREINAMENTO DE ENGENHARIA

PET MECANICA

Horário de Início	segunda	terça	quarta	quinta	sexta
08:00	Arduino	-	Arduino	Machine Learning	Arduino
09:00		Biomecânica			
10:00	LaTeX	Autocad	LaTeX	Autocad	LaTeX
11:00					
12:00	almoço				
13:00					
14:00	Excel	Solid Edge	Excel	Solid Edge	Excel
15:00					
16:00	Octave	Impressão 3D	Octave	Impressão 3D	Octave
17:00					

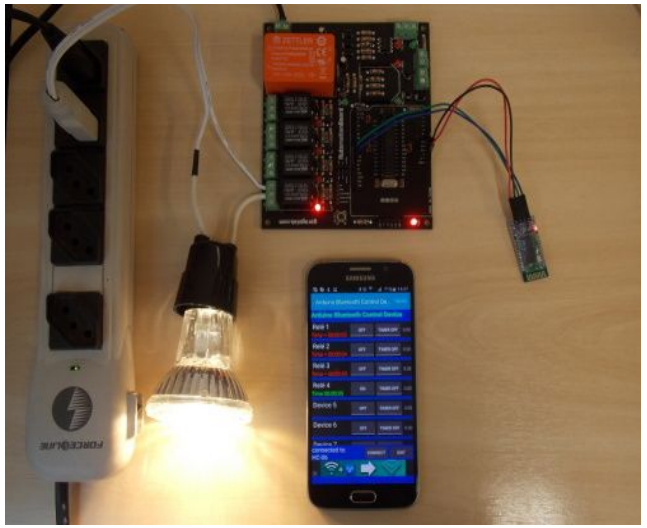
O Arduino

- Plataforma open-source
- Como funciona?
- Popularidade entre estudantes
- Diversas aplicações



Algumas 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

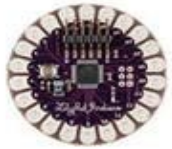
Arduino Mega 2560

Arduino Mini

LilyPad Arduino USB

LilyPad Arduino Simple

LilyPad Arduino SimpleSnap



LilyPad Arduino

Arduino Nano

Arduino Pro Mini

Software

PROFESSIONAL | EDUCATION | STORE | Search on Arduino.cc | SIGN IN

HARDWARE | SOFTWARE | CLOUD | DOCUMENTATION | COMMUNITY | BLOG | ABOUT

Arduino IDE 1.8.15

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the **Getting Started** page for Installation instructions.

SOURCE CODE

Active development of the Arduino software is **hosted by GitHub**. See the instructions for **building the code**. Latest release source code archives are available **here**. The archives are PGP-signed so they can be verified using **this** gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file
Windows app Win 8.1 or 10 Get
Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits
Mac OS X 10.10 or newer

Release Notes Checksums (sha512)

Hourly Builds

Download a **preview of the incoming release** with the most updated features and bugfixes.

DOWNLOAD OPTIONS

Windows

Previous Releases

Download the previous version of the current release, the classic 1.0.x, or old beta releases.

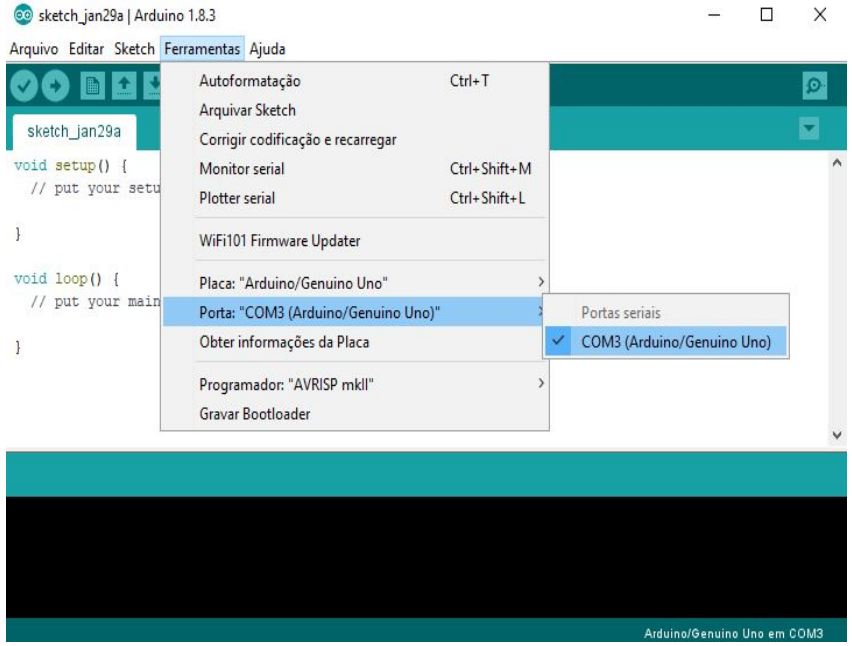
DOWNLOAD OPTIONS

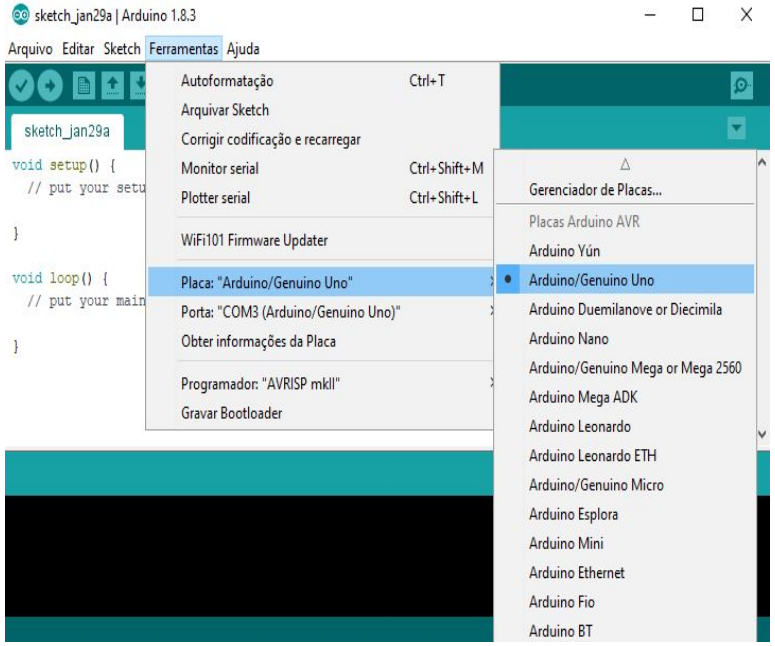
Previous Release (1.8.14)

Help

Arduino IDE

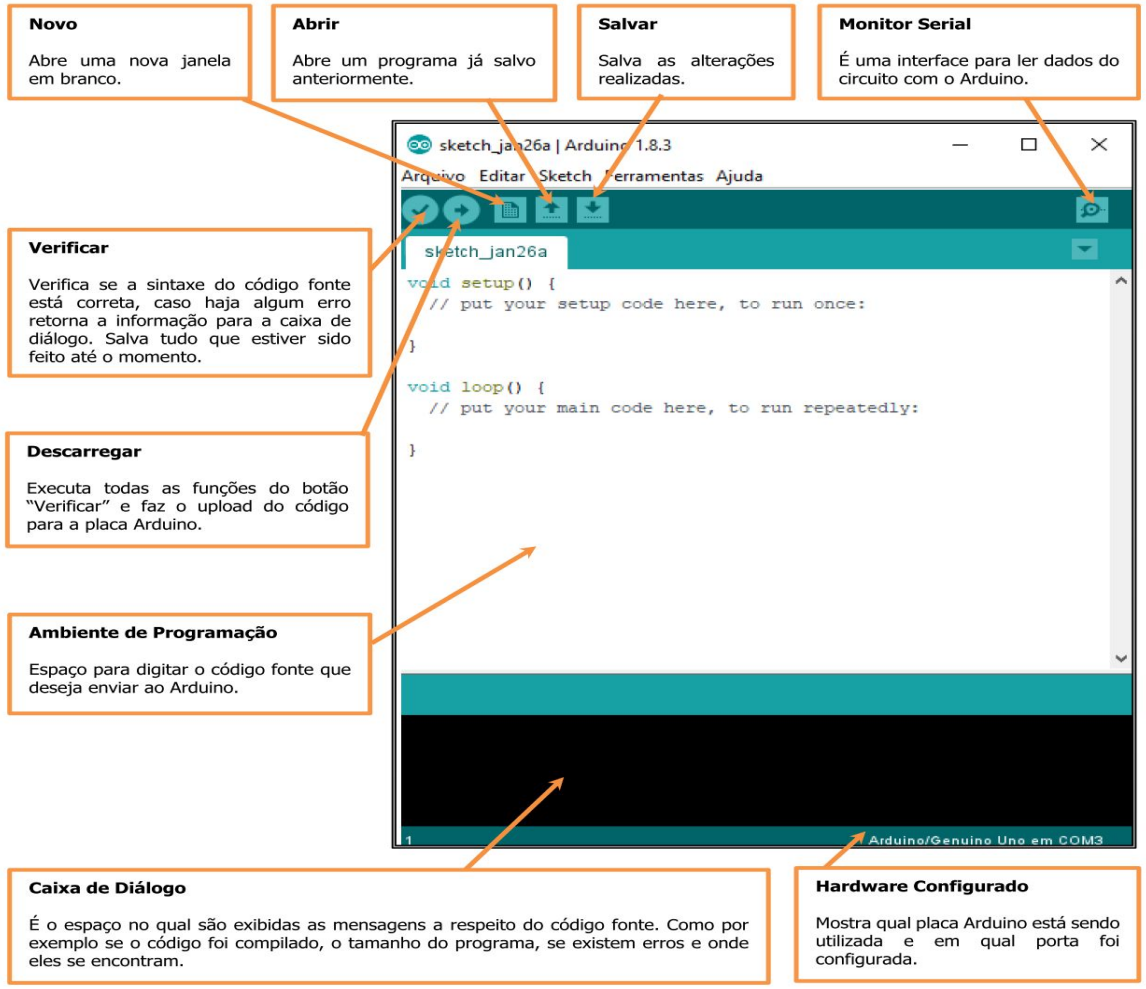
- Plug and play
- Escolha das portas
- Escolha da placa





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 re-inicialização da placa. Após a reinicialização, o sketch passa a ser executado pela placa.

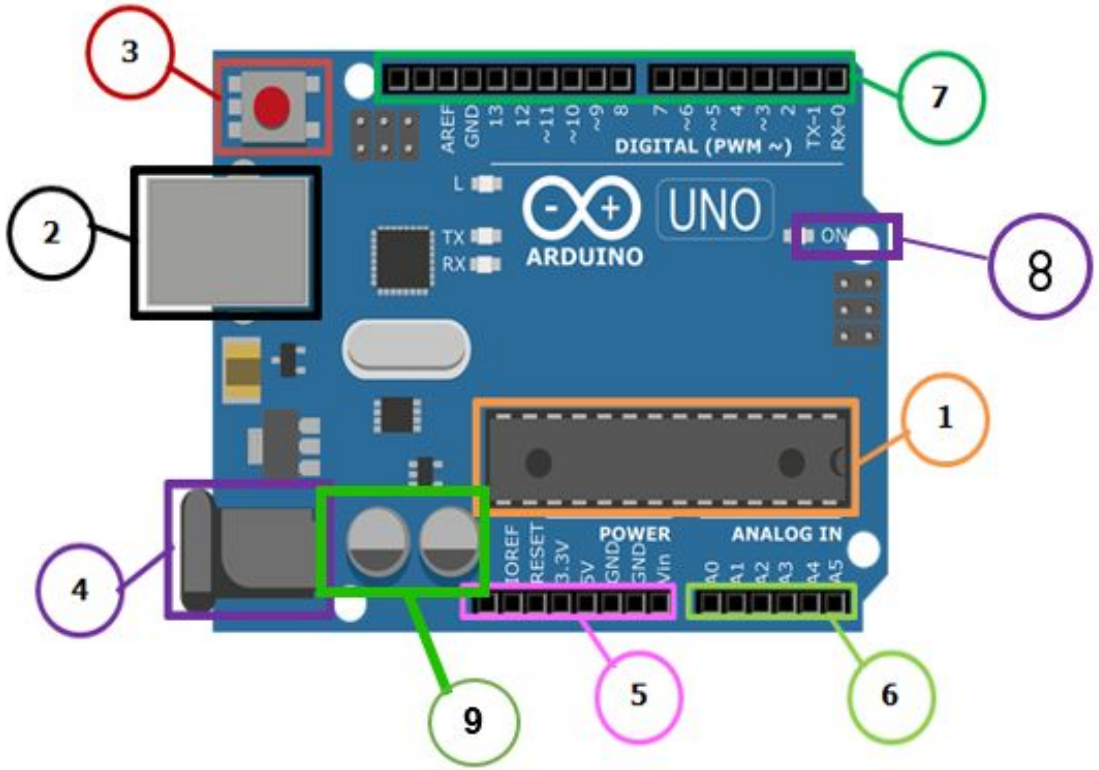


Gráficos no Arduino

```
Serial.print(variável1);
Serial.print(" ");
Serial.print(variável2);
Serial.print(" ");
Serial.print(variável3);
Serial.print(" ");
Serial.println(variável4);
```

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



Linguagem

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

Variáveis

- Cada variável tem a função de armazenar um respectivo dado, podendo usá-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.
```

- Variáveis 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
- Variáveis constantes (Pré-Definidas no Arduino)
  - True/False
  - INPUT/OUTPUT
  - HIGH/LOW

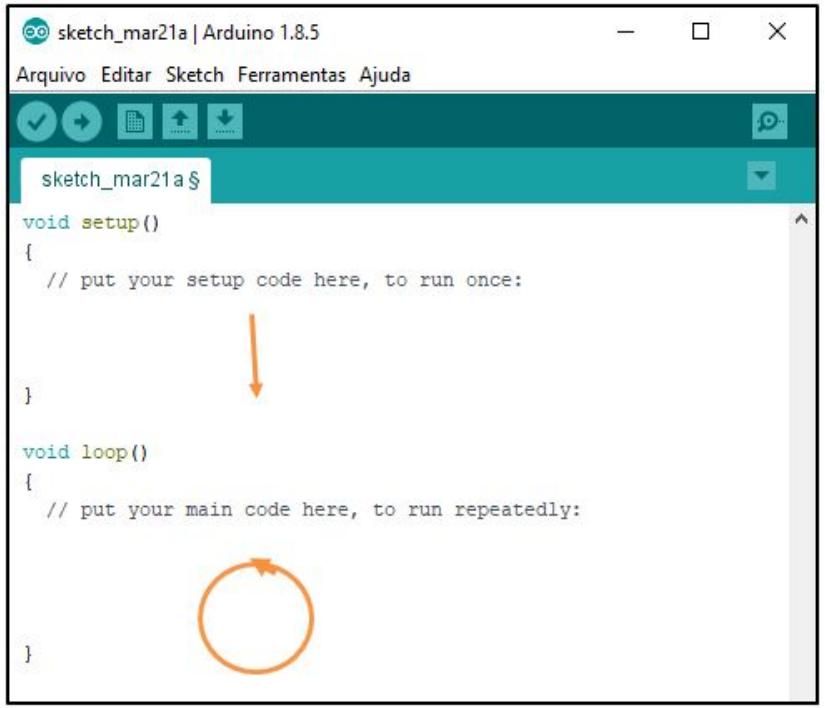
EXEMPLO STRING:

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

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

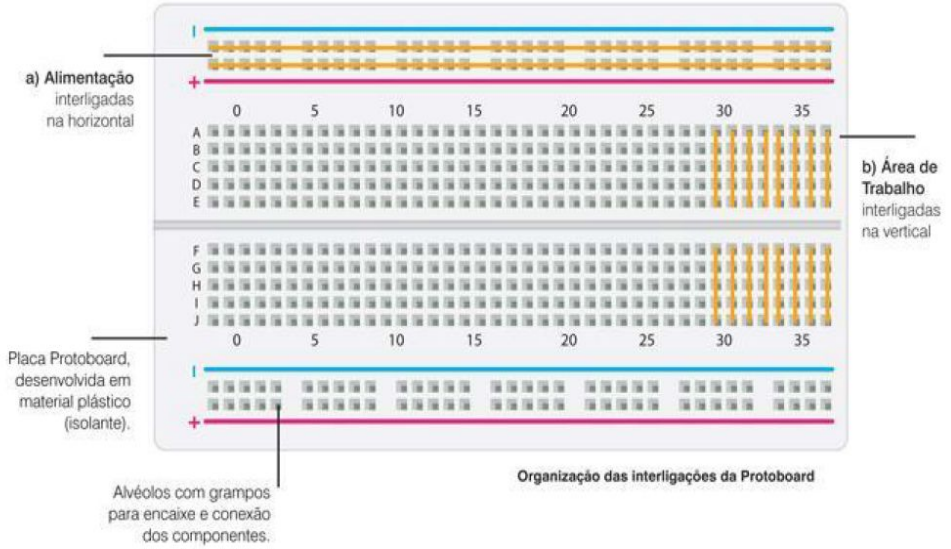
Funções Principais

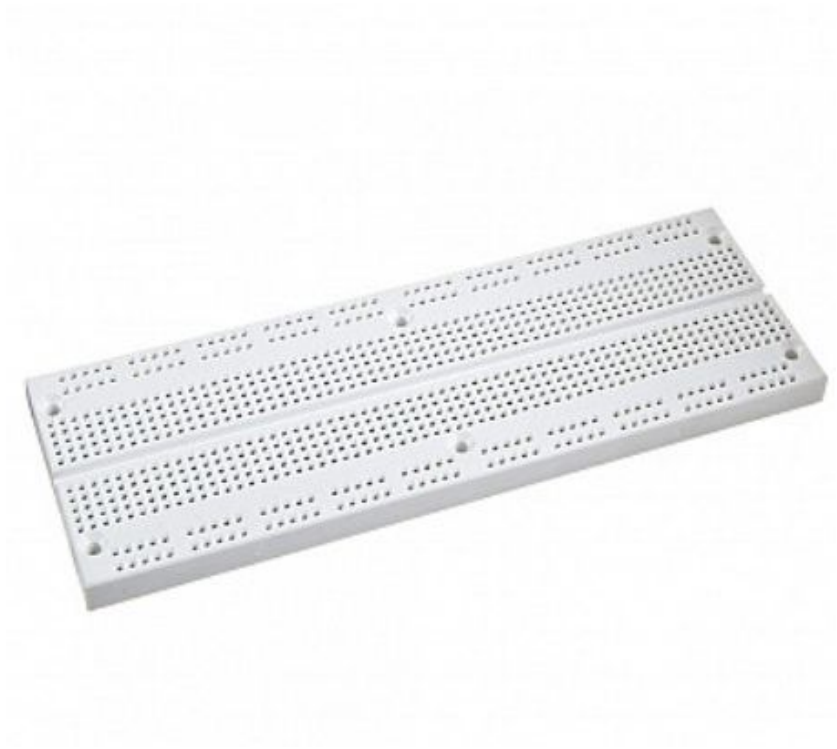
- Função Setup
  - Tipo void
  - Executa uma única vez
  - Declaração de pinos e inicializações
- Função Loop
  - Tipo void
  - Repete o código programado como num ciclo
  - Comando a serem executados



Protoboard

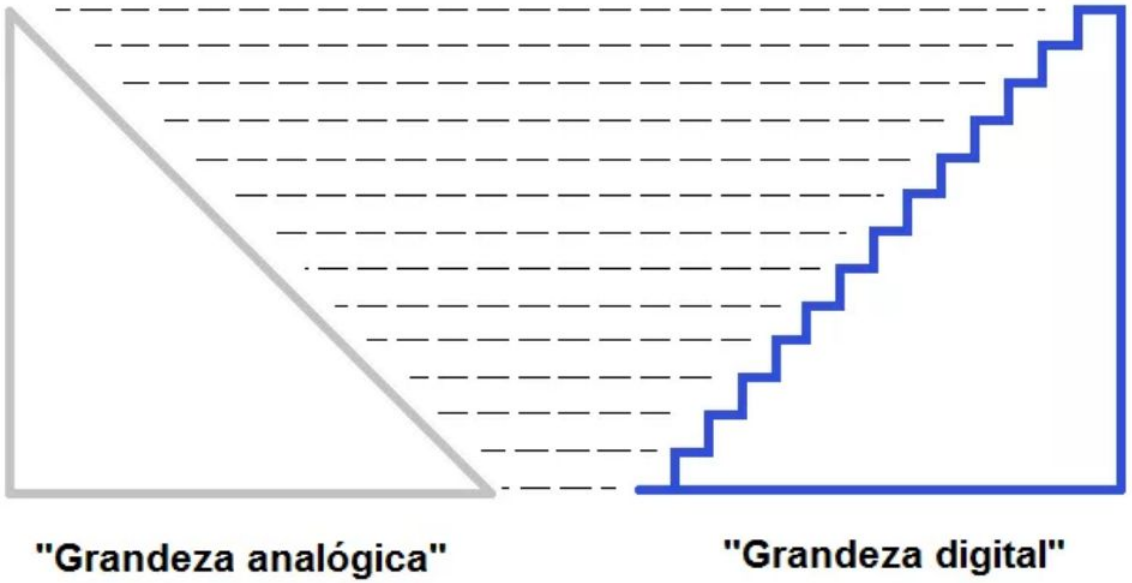
É uma placa para a montagem dos circuitos eletrônicos. Ela possui diversos furos que estão interligados de forma a facilitar as conexões dos componentes, como mostra a figura a seguir.





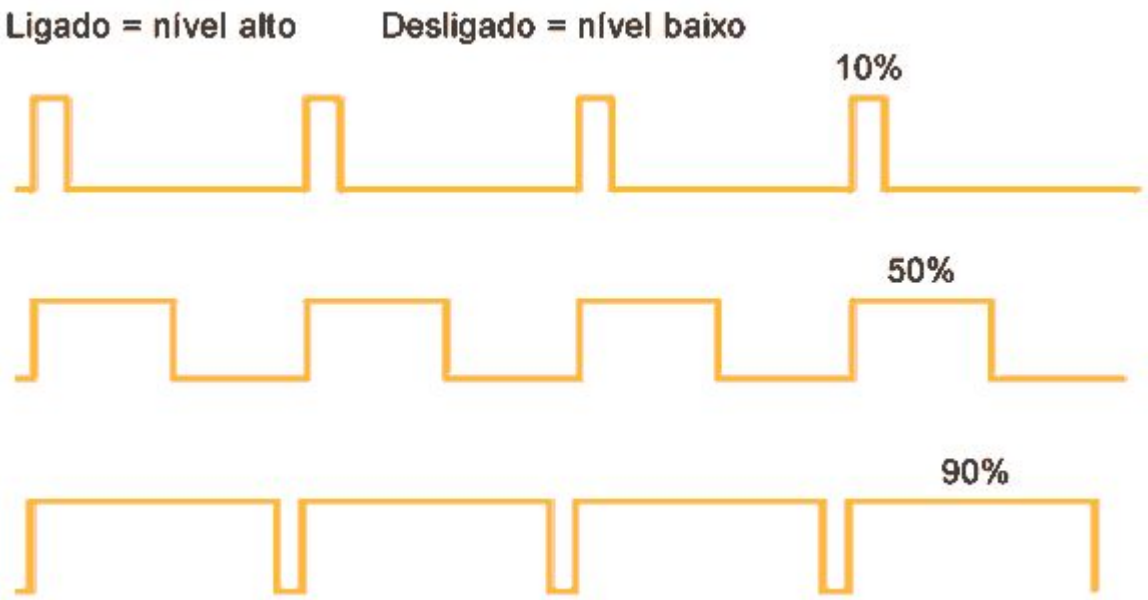
Grandezas analógicas e digitais

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



PWM

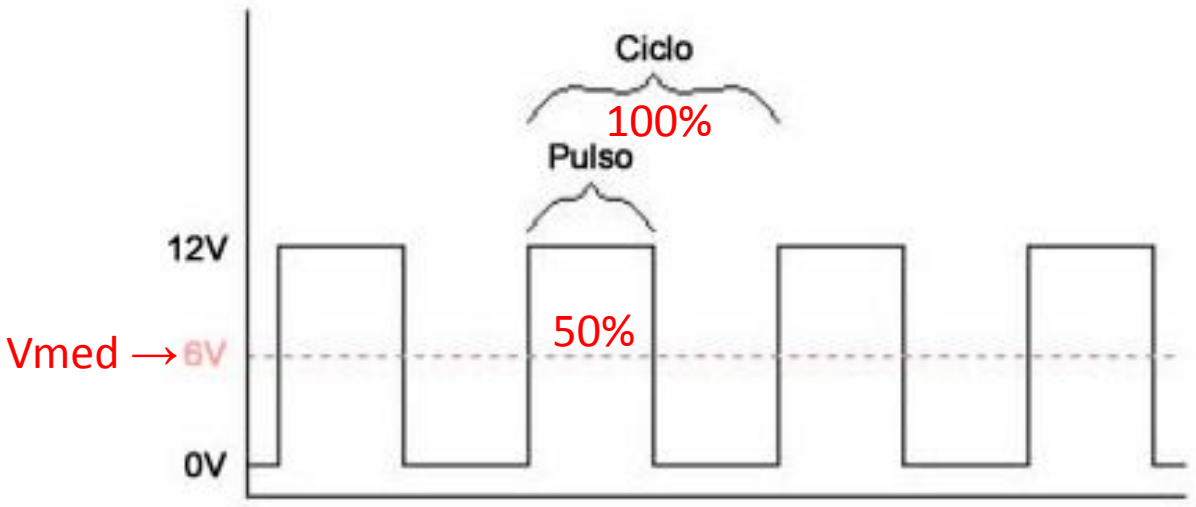
- Modulação por largura de pulso
- Sinais analógicos através de sinais digitais
- Controle de potência ou velocidade
- Aplicações: motores, aquecedores, LEDs
- Controle da tensão média sobre a carga



PWM no Arduino

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

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



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)

Pinos Analógicos

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

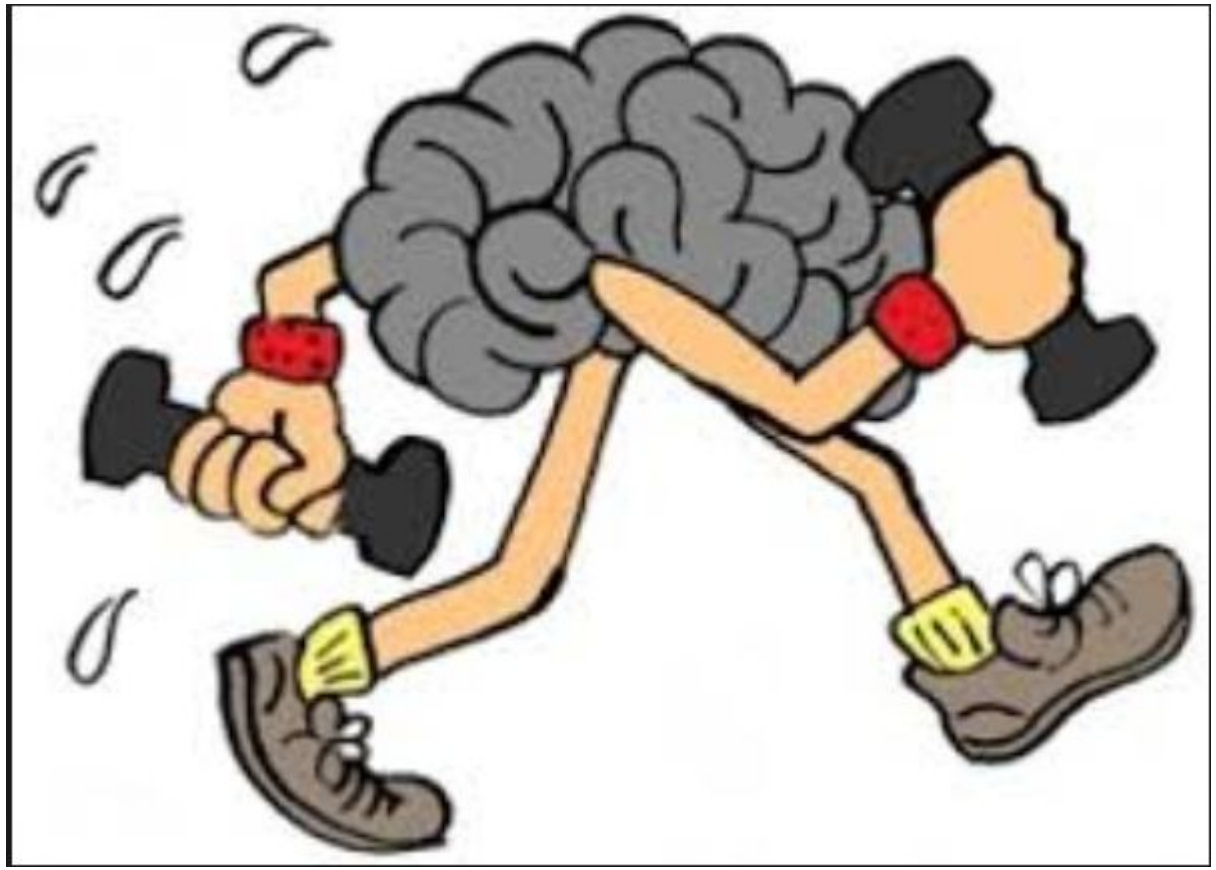
Comunicação Serial e outras funções

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]

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

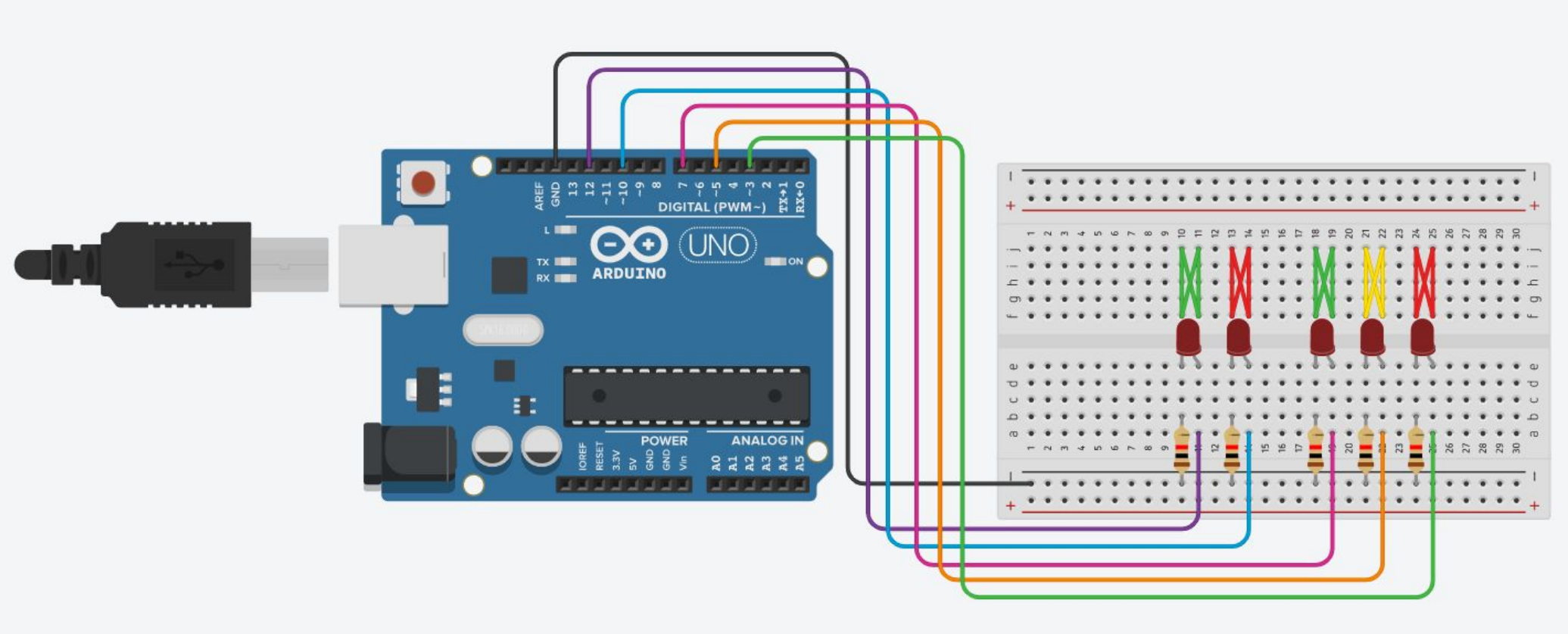
Solução: https://youtu.be/Zcrir_m6COM

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 3 segundos. Em horário de pico, para não haver engarrafamento, exige um tempo mínimo de 5 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

```
#define REDC 3
#define YELC 5
#define GREC 7
#define REDP 10
#define GREP 12

void setup()
{
  pinMode(REDC, OUTPUT);
  pinMode(YELC, OUTPUT);
  pinMode(GREC, OUTPUT);
  pinMode(REDP, OUTPUT);
  pinMode(GREP, OUTPUT);
}
```

```
void loop()
{
  digitalWrite(REDC, HIGH); //ACENDE VERMELHO CARRO
  digitalWrite(GREP, HIGH); // ACENDE VERDE PEDESTRE

  delay(3000); // ESPERA 3 SEGUNDOS

  digitalWrite(REDC, LOW); //APAGA VERMELHO CARRO
  digitalWrite(YELC, HIGH); //ACENDE AMARELO CARRO

  delay(1000); //ESPERA 1 SEGUNDO

  digitalWrite(YELC, LOW); //APAGA AMARELO CARRO
  digitalWrite(GREP, LOW); //APAGA VERDE PEDESTRE
  digitalWrite(REDP, HIGH); //ACENDE VERELHO PEDESTRE
  digitalWrite(GREC, HIGH); //ACENDE VERDE CARRO

  delay(5000); // ESPERA 5 SEGUNDOS

  digitalWrite(GREC, LOW); //APAGA VERDE CARRO
  digitalWrite(REDP, LOW); //APAGA VERMELHO PEDESTRE
}
```

Projeto Buzzer

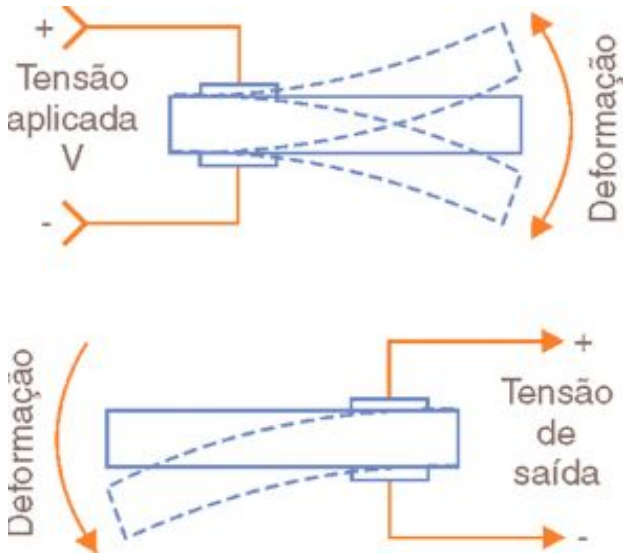
Para a construção de uma cidade mais inclusiva, você resolveu complementar o projeto de semáforo anterior acrescentando um dispositivo sonoro que indique aos deficientes visuais quando sua travessia for segura.

Tarefa: Complemente o projeto anterior adicionando bips periódicos, por meio do dispositivo eletrônico buzzer, quando o sinal verde de pedestre estiver ativado.

Tarefa Extra: Faça com que os bips sejam mais intensos com o decorrer do tempo até o momento em que o sinal de pedestre mude para vermelho.

Buzzer

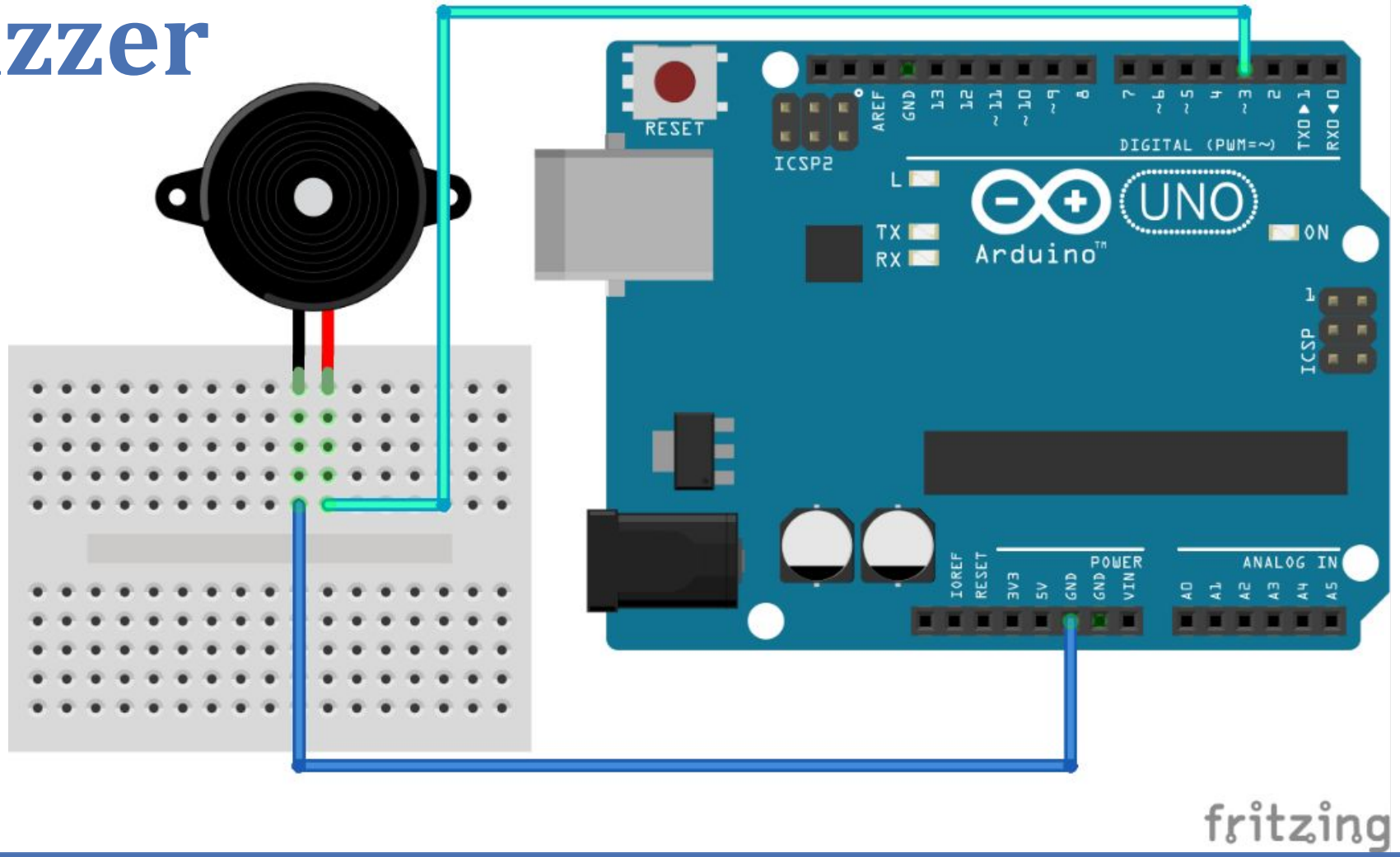
- Dispositivo utilizado para produção de som de baixa potência.
- Funciona a partir do efeito piezelétrico que ocorre na célula piezelétrica em seu interior.
- Som produzido pelas vibrações da célula piezelétrica.
- Deve-se atentar que o buzzer possui polaridade, isto é, o local onde os pinos do Arduino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



Buzzer ativo

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

Projeto Buzzer



Projeto Buzzer

pinMode(*porta,OUTPUT*);	Declaração do modo de operação do Buzzer;
tone(*porta,frequência*);	Ativa o Buzzer que está conectado na PORTA com determinada FREQUÊNCIA.
noTone(*porta*);	Desativa o Buzzer que está conectado na PORTA.
delay(*milissegundos*);	Faz com que o código “trave” nessa parte por determinado tempo.

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 algum 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.

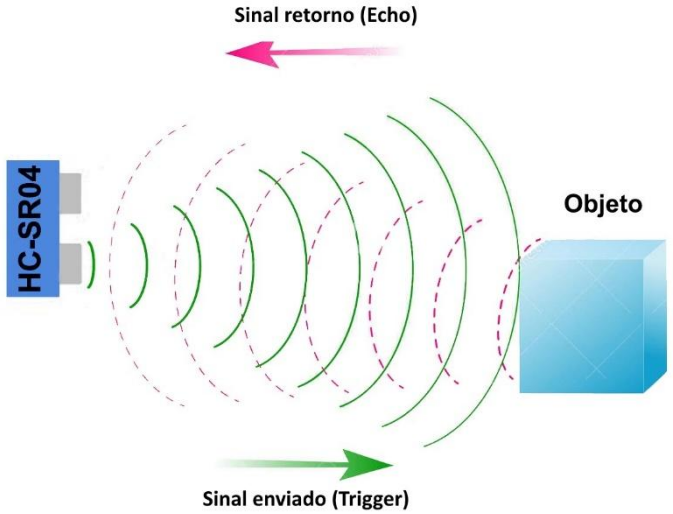
Componentes:

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

Solução: https://youtu.be/wLKkP1kI1ac

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

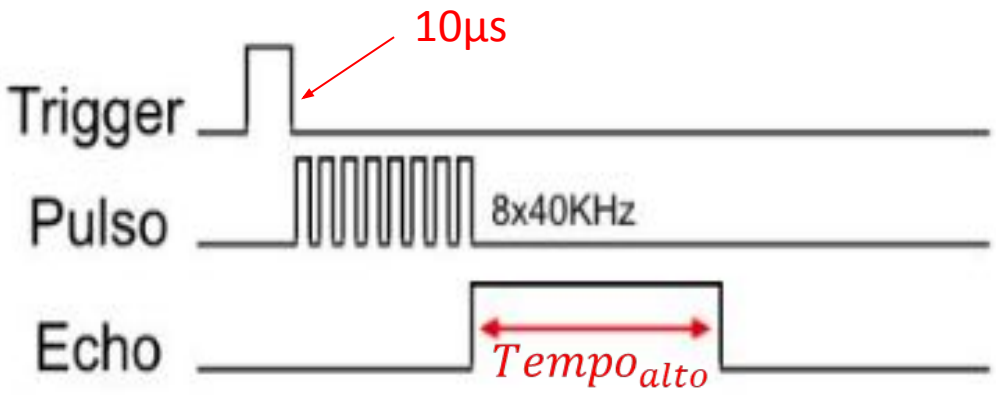
Pino Trig: HIGH (10µs) → Pino Echo: HIGH (emissão sinal) → Pino Echo: LOW (recepção sinal)

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

Sensor ultrassônico

- O circuito externo(Arduino) envia um pulso de trigger de pelo menos 10µs em nível alto;
- Ao receber o sinal de trigger, o sensor envía 8 pulsos de 40khz e detecta se há algum sinal de retorno ou não;
- Se algum sinal de retorno for identificado pelo receptor, o sensor gera um sinal de nível alto no pino de saída cujo tempo de duração é igual ao tempo calculado entre o envio e o retorno do sinal ultrassônico;

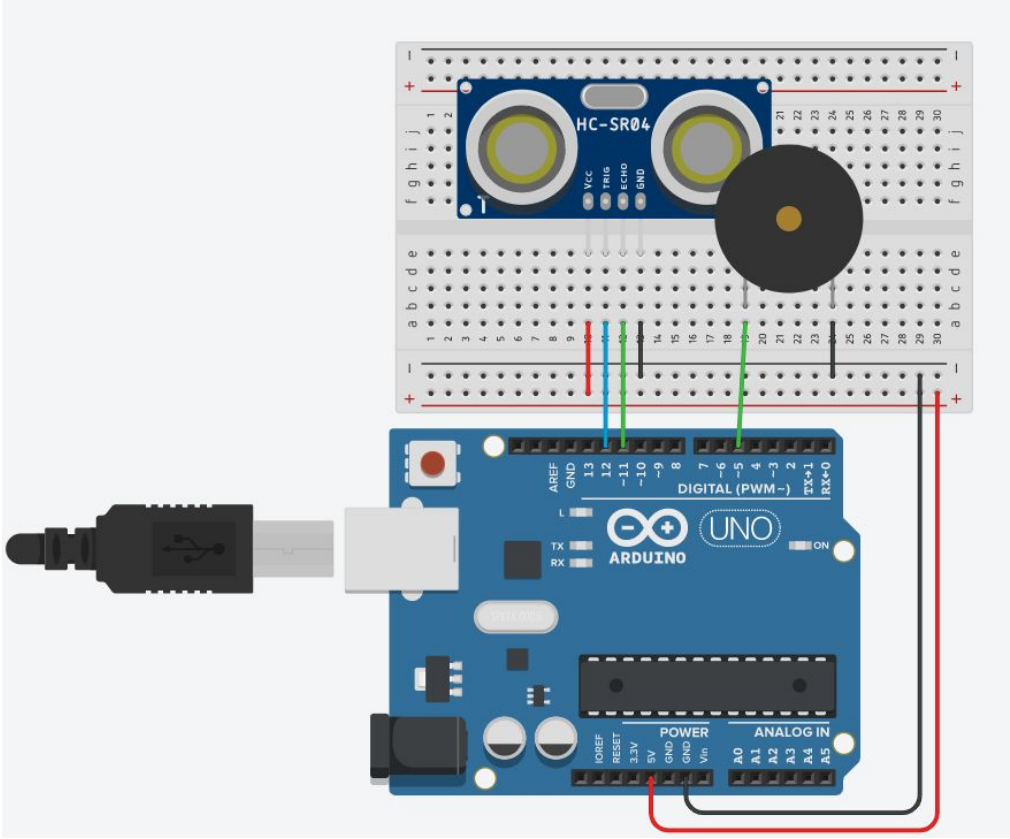
```
pinMode(TRIGGER,OUTPUT);
pinMode(ECHO,INPUT);
```



Sinais nos pinos do HC-SR04

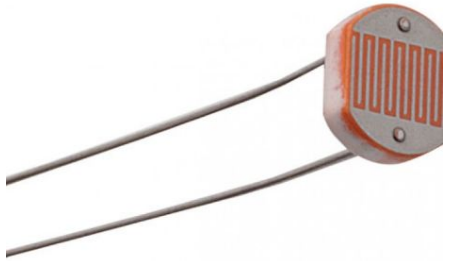
```
digitalWrite(TRIG,HIGH);
delayMicroseconds(10);
digitalWrite(TRIG, LOW);
duracao = pulseIn(ECHO, HIGH);
distancia = (duracao * 0.0343)/2;
```

Montagem do Circuito



Projeto Cortina Automática

Problema: Você gostaria de acordar cedo para receber os primeiros raios de sol e sintetizar vitamina D, mas o horário em que o sol nasce possui pequenas variações entre um dia e outro. Não deixando se abater, você lembro que possui um kit de arduino e, portanto, fará um circuito para que o servomotor abra a sua cortina quando o seu sensor de luminosidade for ativado com a luz do sol.



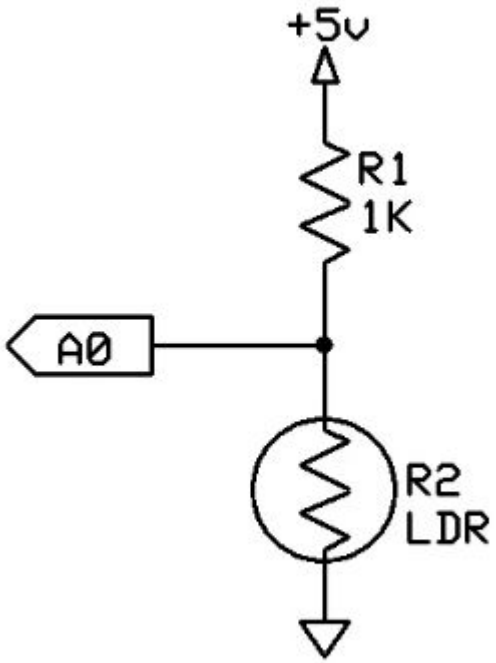
Componentes:

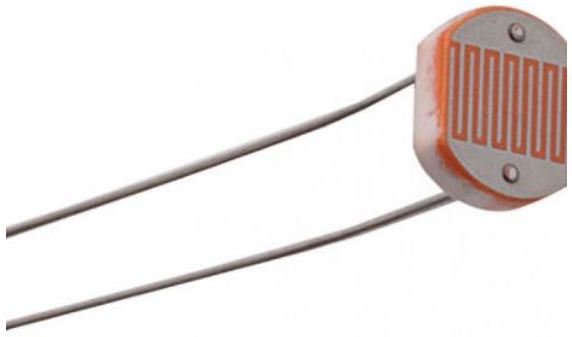
- 1 LDR
- 1 Servomotor
- 1 Resistor
- JUMPERS

Solução: https://youtu.be/e8vWwMfyKvo

LDR

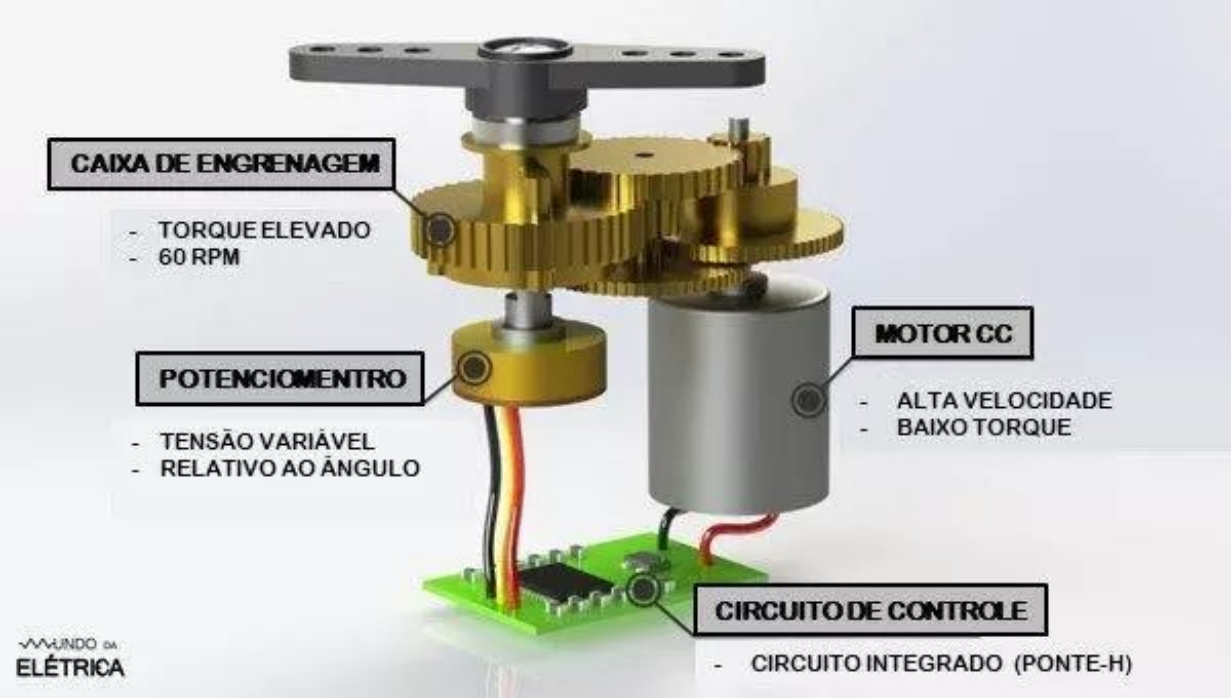
- O LDR é um semicondutor de alta resistência que tem sua resistividade quase zero quando está exposto a uma grande quantidade de luz.
- Quanto menor luminosidade maior a resistência do LDR;
- Quanto maior a resistência do LDR, maior a tensão em A0;
- Então: quanto menor a luminosidade maior é a tensão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





Servomotor

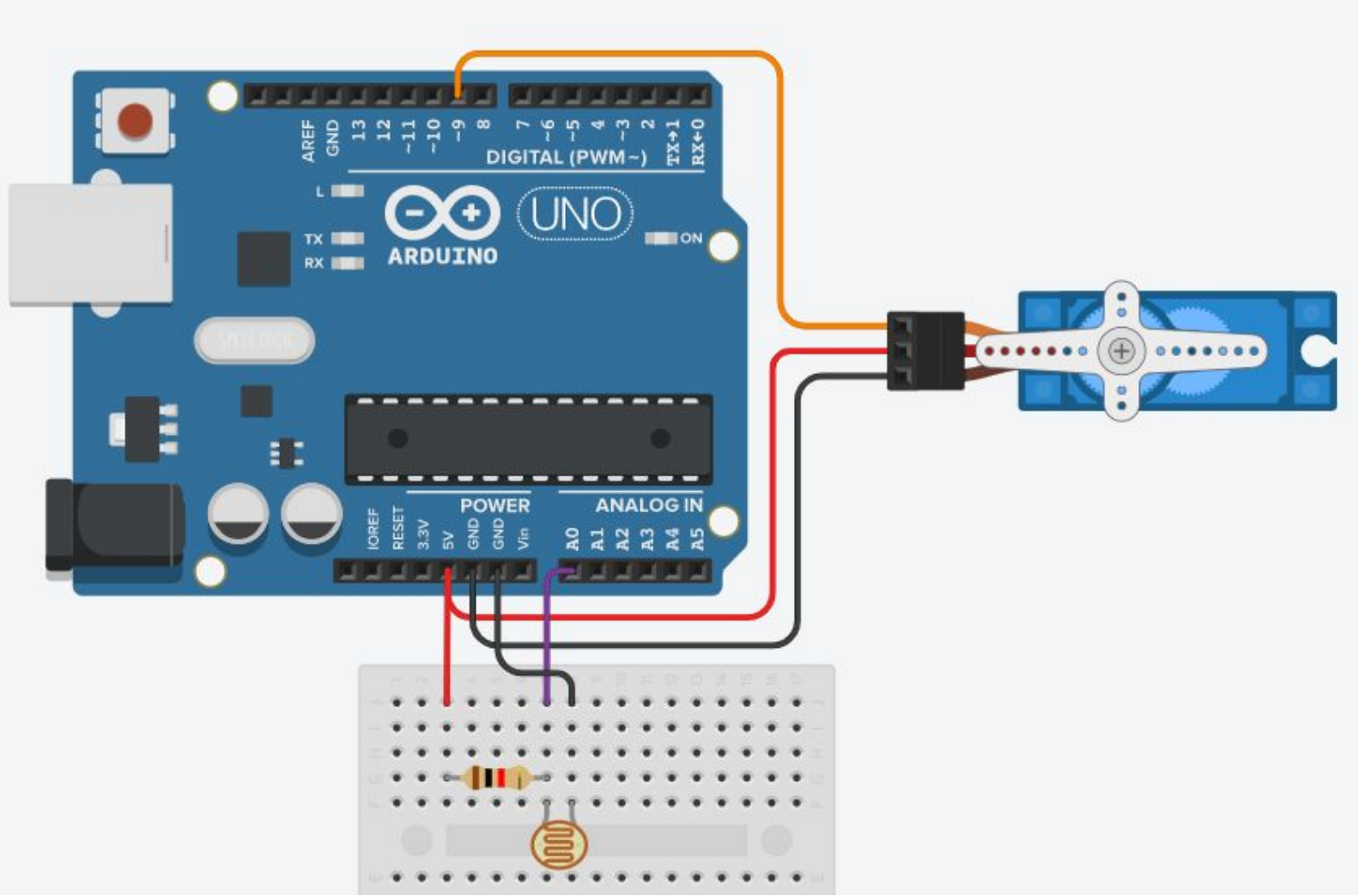
- Podem ser tanto motores CA quanto motores CC;
- Têm incorporado um encoder e um controlador. Este encoder é na verdade um sensor de velocidade que possui a função de fornecer a velocidade e posicionamento do motor;
- Os **Servos de CC** geralmente são usados em projetos menores, eles possuem um custo relativamente baixo e são eficientes;
- Os **Servo de CA** normalmente são usados em ambientes industriais, pois costuma ser de elevada potência, oferece maior exatidão no seu controle e pouca manutenção;
- Para utilizarmos no arduino, faremos uso da Biblioteca <Servo.h>.



Estrutura interna de um Servomotor de Corrente Contínua

Fonte: https://www.mundodaeletrica.com.br/o-que-e-servo-motor-e-como-funciona/

Circuito


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
IOREF
RESET
3.3V
5V
GND
GND
Vin
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5

Circuito

#include<*Servo.h*>	Adição da biblioteca para o servomotor;
Servo motor;	Criação da variável do tipo Servo;
Servo.attach(*porta*);	Define a porta na qual está ligado o servomotor;
var = **analogRead**(*porta_analógica*);	Faz a leitura do valor de resistência que o fotoresistor apresenta no momento;
motor.Write(*posição*);	Comando para o servomotor se mover até determinada posição.

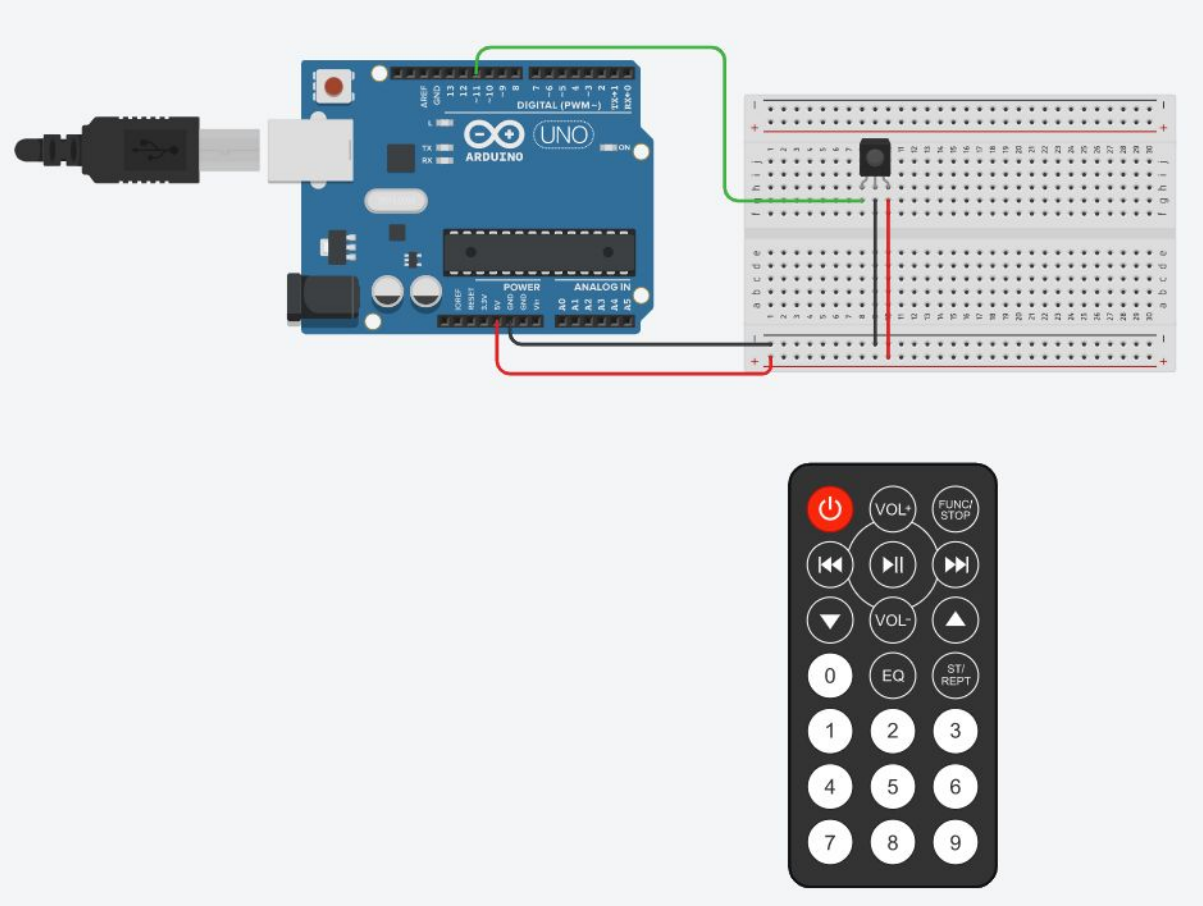
Projeto Controle Infravermelho

Problema: Você gostaria de controlar seus circuitos eletrônicos de forma remota. Uma possibilidade seria programá-los utilizando um controle remoto. Sua tarefa agora é criar um circuito para controlar um servo-motor com base em um controle remoto infravermelho.

Componentes:

- 1 Receptor Infravermelho
- 1 Servomotor
- 1 Controle remoto infravermelho
- JUMPERS

Projeto Controle Infravermelho



Projeto Controle Infravermelho

```
1  #include <IRremote.h>
2
3  #define IRS 11
4
5  IRrecv receptor(IRS);
6  decode_results leitura;
7
8  float armazenavalor;
9
10 void setup() {
11   pinMode(IRS, INPUT);
12   Serial.begin(9600);
13   receptor.enableIRIn();
14 }
```

```
16 void loop() {
17   if (receptor.decode(&leitura))
18   {
19     Serial.print("Valor lido : ");
20     Serial.println(leitura.value, HEX);
21     receptor.resume();
22     if(leitura.value == 0xFD906F){
23       Serial.println("VOL-");
24     }
25   }
26 }
```