Hack.ATOMIC

hackerspace

Apostila volume 1

ARDUINO

Amanda Pimenta

Sumário

Introdução

Prefácio

Esta apostila é um produto dos trabalhos didáticos realizados por hack.ATOMIC hackerspace. Dedicada à educação tecnológica e esperança em desenvolver um ambiente estimulante para a criatividade e iniciativa, a entidade vem a oferecer o material para estudos sobre o tema Arduino, para que os alunos estudem e se aprofundem nos experimentos executados.

Hack.ATOMIC

Hack.ATOMIC *hackerspace* ,criado em 2014 por alunos de Engenharia Física da Universidade EEL-USP, foi uma iniciativa para integração e compartilhamento de conhecimentos sobre eletrônica e automação.

No início do ano de 2015, iniciaram-se cursos de *Arduino* gratuitos oferecidos aos alunos da USP. É interessante notar que muitos, se não a maioria, dos participantes do curso nunca haviam tido contato com a eletrônica na vida, mas, após terem frequentado o curso, tornaram-se bem instruídos na área e hoje são capazes de realizar importantes projetos.

Turmas formadas:

- ✓ Curso de Arduino básico (2014)
  Instrutores: Rodrigo Trindade, Jhunu Fernandes, Leandro Kellermann
- ✓ Curso extensivo de Arduino (2015)
  Instrutores: Rodrigo Trindade, Amanda Pimenta e Gabriel Massa
- ✓ Curso intensivo de Arduino (2015)
  Instrutores: Lukas Joseph e Amanda Pimenta

Projetos executados ou em execução:

- ✓ Projeto economia de energia das salas da USP (2015)
- ✓ Projeto piano-escada (2015)
- ✓ Projeto controle de temperatura no laboratório de supercondutividade (2015)
- ✓ Projeto robô econômico feito de palitos picolé (2015)

O objetivo principal do *hackerspace* é melhorar tecnologicamente a faculdade EEL-USP: automatizando laboratórios e salas, bem como realizar projetos de eletrônica que possam ajudar o desenvolvimento universitário.

Agradecimentos

Agradecimentos aos professores Eduardo Ferro e Carlos Shigue que tiveram importante papel no desenvolvimento da entidade.

Ao líder desta entidade, Rodrigo Trindade, que a ministra com garra, esperança e determinação.

Muito obrigada

Capítulo 1

Arduino

É uma plataforma de computação física ou embarcada, ou seja, um sistema capaz de interagir com o ambiente por meio do *hardware* e *software*. Pode ser utilizado para criar objetos interativos independentes, ou conectado a um computador ou uma rede como a *Internet* para comunicação e atuação do *Arduino* sobre eles.

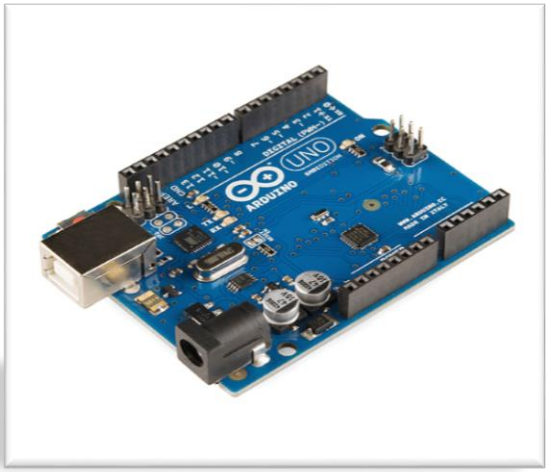
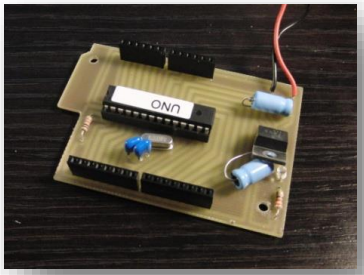
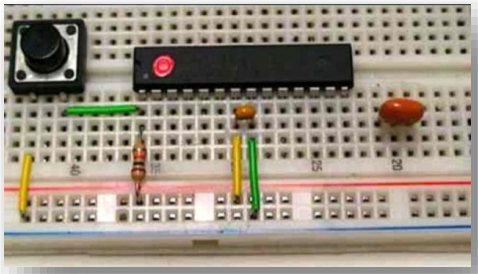


Figura 1: Arduino

Possui hardware e software livres (de fonte aberta). Software livre é uma expressão utilizada para designar qualquer programa de computador que pode ser executado, copiado, modificado e redistribuído pelos usuários gratuitamente. Os usuários possuem livre acesso ao código-fonte do *software* e fazem alterações conforme as suas necessidades. Hardware livre (em inglês, *Open source hardware*) é um *hardware eletrônico* projetado e oferecido da mesma maneira que um software de código livre.

A grande vantagem é que o código, os esquemas e o projeto podem ser utilizados livremente por qualquer pessoa e com qualquer propósito. Por este motivo existem muitas placas clonadas disponíveis para compra ou possibilita que você crie seu próprio *Arduino* em uma matriz de contatos(*protoboard*) ou em uma PCB (*Printed Circuit Board*, placa de circuito impresso).





Figuras 2 e 3: Arduinos clones

Surgimento

Arduino foi um projeto que iniciou na Itália no ano de 2005 com o objetivo de ser uma plataforma de baixo custo e de fácil aprendizado. Os desenvolvedores são: Massimo Banzi, David Cuartielles, Tom Igoe, Gianlica Martino e David Mellis.

Plataforma

Existem várias versões de *Arduino* no mercado: Uno, Mega 2560, Leonardo, Ethernet, Nano, etc. Cada tipo tem sua característica e vantagens específicas para os projetos que se deseja realizar.

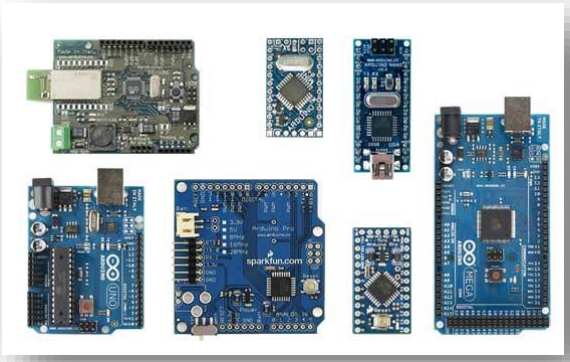


Figura 4: Modelos de Arduino

Hardware

A placa do *Arduino* é composta de um microcontrolador *Atmel AVR*, um cristal ou oscilador (relógio simples que envia pulsos de tempo em específica freqüência, para permitir a operação na velocidade correta), um regulador linear de 5 volts, saída USB (permitindo conectar a um PC ou Mac para *upload* ou visualização de dados) e portas de entrada/saída do microcontrolador (para conectar a outros circuitos ou sensores).

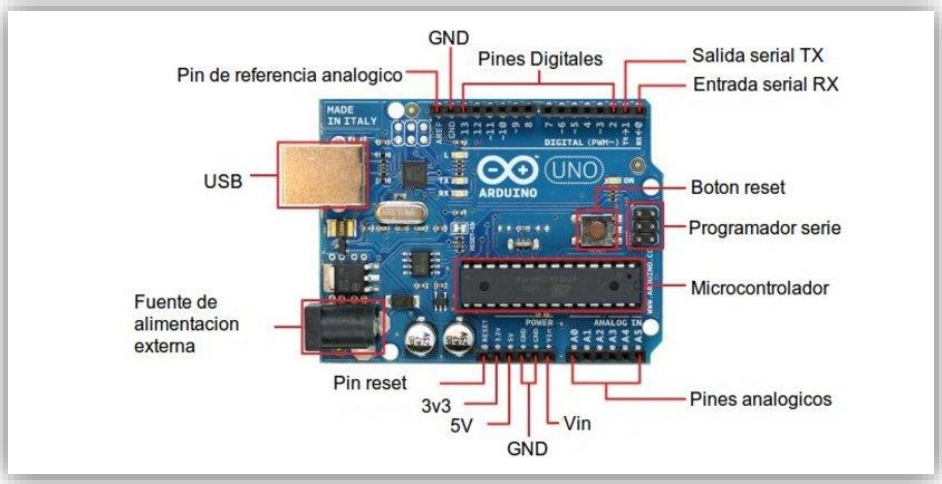


Figura 5: Componentes do Arduino

O Arduino possui tanto portas digitais como analógicas, sendo que estas portas servem para comunicação entre a placa e dispositivos externos, por exemplo, ler um botão, acender um led, entre outros muitos exemplos. Arduino UNO no total são 14 portas digitais e 6 analógicas.

Os conversores analógico-digitais (ADC) são de 10 bits, isso quer dizer que os valores lidos numa porta analógica variam de 0 a 1023 (2^10), onde 0 representa 0 Volts e 1023, que é o valor máximo, representa 5 Volts.

Portas Digitais: Trabalham com valores bem definidos, ou seja, no caso do Arduino esses valores são 0 e 5 Volts. Zero volts significa a ausência de um sinal e 5 volts indica a presença de um sinal.

Portas analógicas: Portas que recebem um tipo de sinal contínuo que varia em função do tempo. A representação do sinal analógico é uma curva.

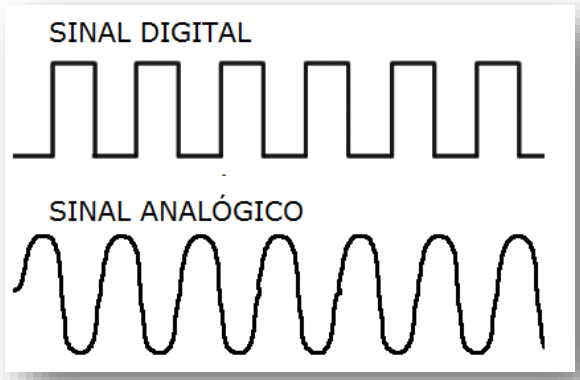


Gráfico 1: Diferença entre sinal digital e analógico

Software

O IDE *Arduino* é utilizada para programar o *Arduino* (fazer com que ele faça o que você deseja) que é um software livre que utiliza uma linguagem de simples compreensão e aprendizado (baseada na linguagem C). Após escrever sua programação, deve-se fazer *upload* para a placa e só assim, então, executará suas instruções interagindo com o que estiver conectado a ele. Os programas são chamados de *sketches* que significa rascunho ou esboço.

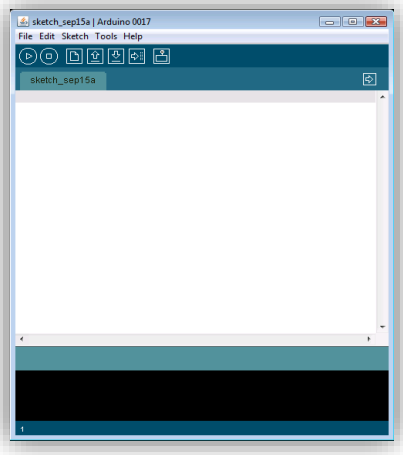


Figura 6: IDE Arduino

Funções primordiais:

```
void setup() {
// Escreva o código que será executado apenas uma vez.
}
void loop() {
// Escreva o código que será executado infinitas vezes
```

- A função setup () é chamada quando o código (sketch) é executado.

Use-a para:

a) Iniciar variáveis;
b) O modo como os pinos devem operar: entrada (INPUT) ou saída (OUTPUT);
c) Bibliotecas;
d) Cabeçalhos.

Tudo o que estiver no setup () será executado apenas uma vez, imediatamente após o microcontrolador ser energizado.

Quando o botão de reset for pressionado ou quando a energia cair, o código será reinicializado e nessa condição especial o setup () é novamente executado.

- A função loop () tem um propósito fundamental de repetição infinita do que estiver escrito.

Ao repetir a mesma função o microntrolador nunca para de funcionar.

Basicamente o loop () será o seu "escravo" e o setup () "dirá como o escravo se comporta".

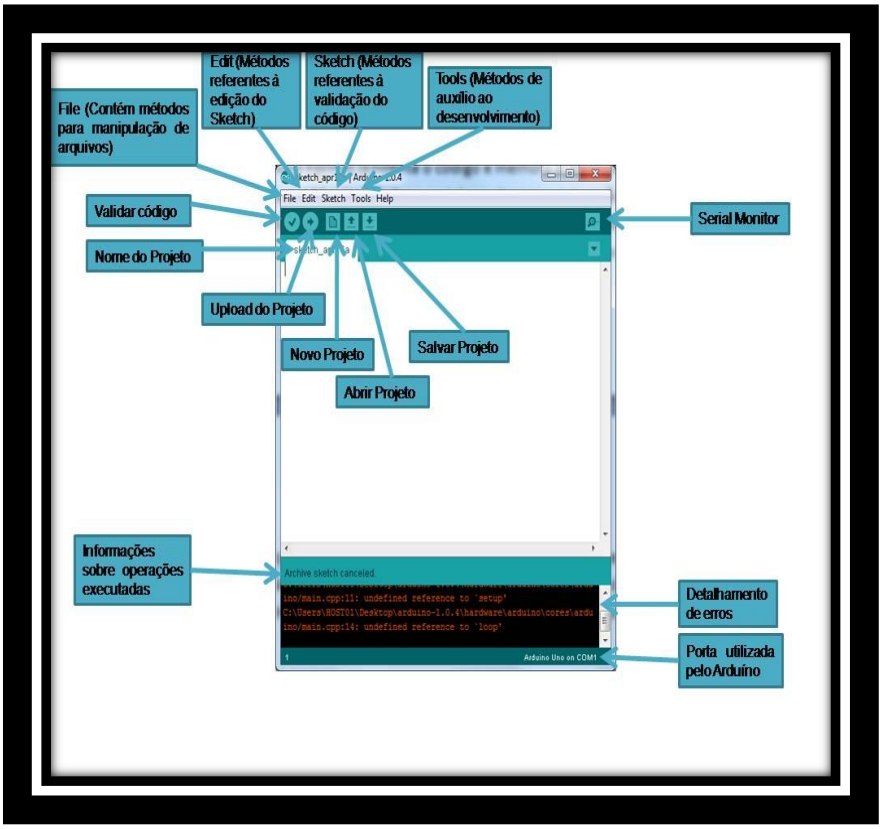


Figura 7: Apresentação da IDE

Primeiros passos

I. *Baixar a IDE Arduino:*

- Vá até o site www.arduino.cc/en/Main/Software;
- Faça o download da versão compatível ao sistema operacional do seu PC: Windows, Linux ou MAC;
- Descompacte o arquivo ".zip".

Desta forma a interface de programação estará pronta para funcionar e ao clicar no ícone principal teremos a tela abaixo:

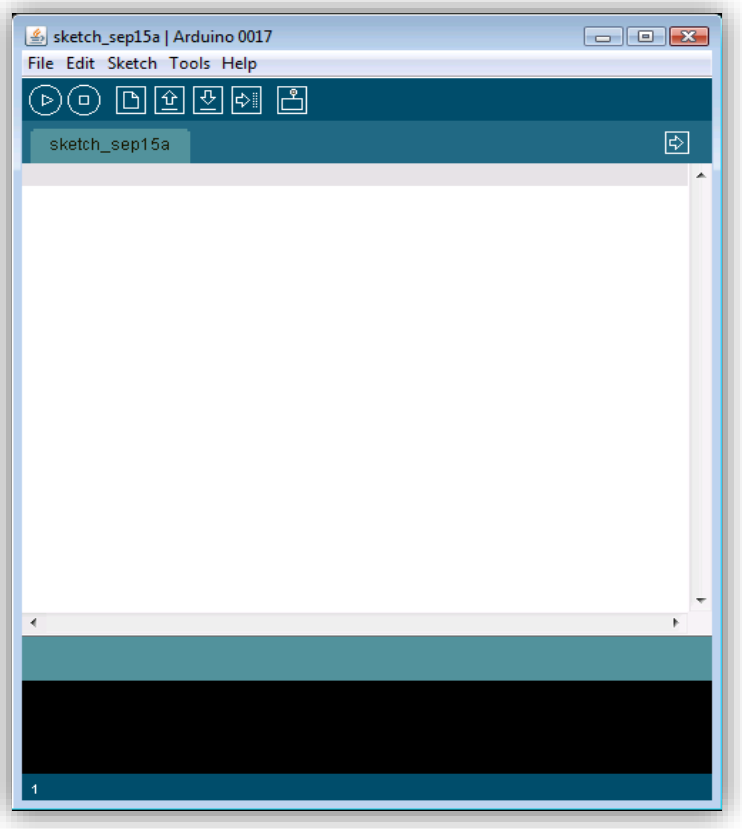


Figura 8: IDE

Temos os ícones principais:


Verify: Para Verificar se o programa está sem erros.


Upload: Carrega o programa no Microcontrolador. (Grava código no *Arduino*).


Serial Monitor: Ativa a leitura da porta serial. (Mostra valores numa tela especial).

II. *Preparar a sua IDE para gravação do Arduino*

- Aperte em *Tools* (ferramentas);
- Aperte em *Board;*
- Informe qual modelo está usando.

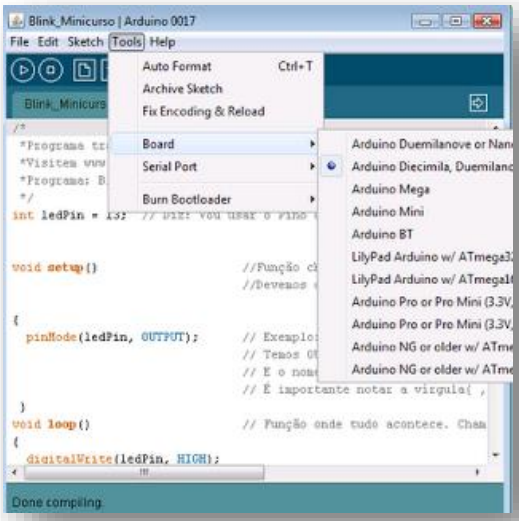


Figura 9: Seleção do modelo usado

III. *Informar a COM*

- Aperte em *Tools*;
- Aperte em *Serial Port*;
- Escolha a porta que é referente ao seu Arduino

Obs: COM é a porta onde o cabo USB está conectado ao seu PC.

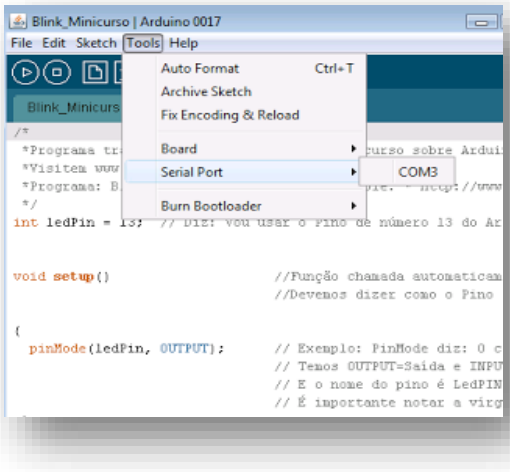


Figura 10: Seleção da porta USB

Figura 11: Programa feito na IDE

Após compilar (clicando em *Upload* ou *Verify*) uma informação aparece mostrando o tamanho do código e o espaço disponível para mais código.

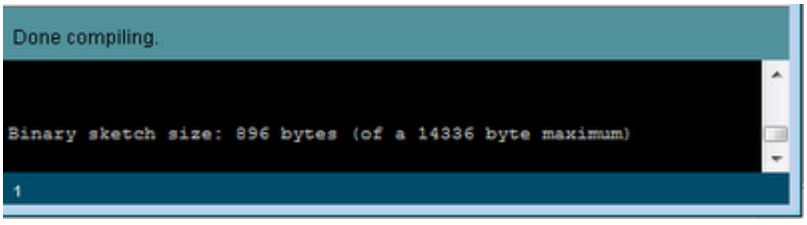


Figura 12: Informação do tamanho do código

Ao clicar em *upload* e o código for gravado no *Arduino* os LEDs RX e TX que se encontram na placa *Arduino* deverão piscar informando que o código está sendo carregado. Obviamente o cabo USB deve estar ligado à placa *Arduino* e ao PC.

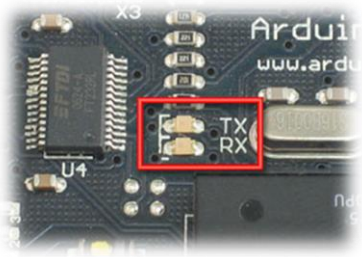


Figura 13: TX e RX

Capítulo 2

Componentes eletrônicos primordiais

Resistores

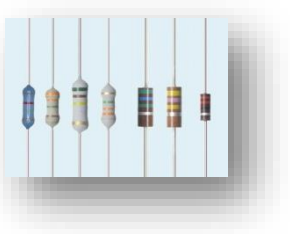


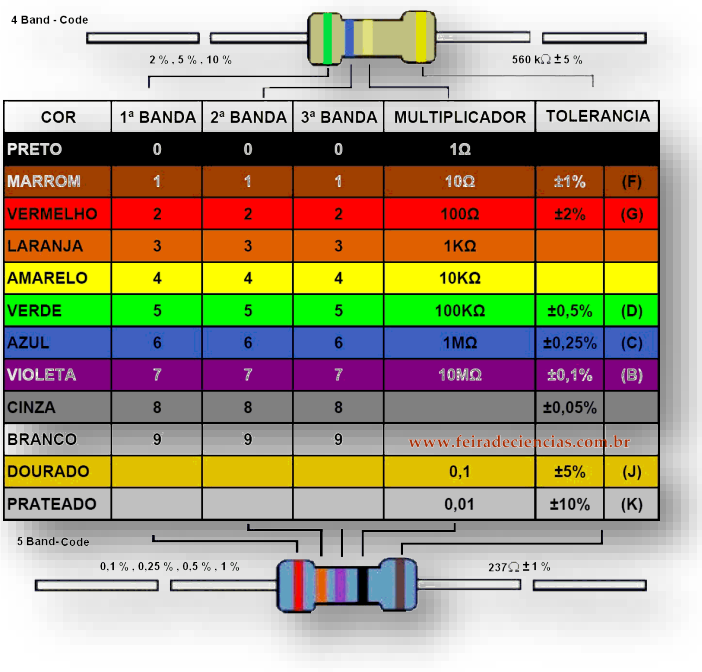
Figura 14: Resistores de resistência fixa

São dispositivos projetados para limitar o fluxo de corrente. São especificados em Ohms (Ω), kiloohms (kΩ) ou megaohms (MΩ).

A corrente (i) que passa em um resistor com resistência (R), devido a uma tensão aplicada (V), é igual a $i = V/R$, que é a Lei de Ohm.

Sempre que uma corrente flui em um resistor, ele dissipa potência. Os mais comuns, de carvão, têm a capacidade de dissipar potências máximas de 1/8, 1/4 ou 1/2 W (potências baixas) e esta é a razão pela qual se apresentam dispositivos tão finos. Porém, existem também resistores maiores capazes de dissipar potências maiores (2W, 5W).

Com relação à forma, os resistores são, normalmente, cilíndricos e pequenos com faixas coloridas que indicam o seu valor. Estas cores são uma espécie de codificação de resistência, assim como mostra na tabela a seguir:



COR	1ª BANDA	2ª BANDA	3ª BANDA	MULTIPLICADOR	TOLERANCIA	
PRETO	0	0	0	1Ω		
MARROM	1	1	1	10Ω	±1%	(F)
VERMELHO	2	2	2	100Ω	±2%	(G)
LARANJA	3	3	3	1KΩ		
AMARELO	4	4	4	10KΩ		
VERDE	5	5	5	100KΩ	±0,5%	(D)
AZUL	6	6	6	1MΩ	±0,25%	(C)
VIOLETA	7	7	7	10MΩ	±0,1%	(B)
CINZA	8	8	8		±0,05%	
BRANCO	9	9	9			
DOURADO				0,1	±5%	(J)
PRATEADO				0,01	±10%	(K)

Tabela 1: Código de resistores

Resistores de resistência variável:

Potenciômetro

Figura 15: Potenciômetro

Um potenciômetro é simplesmente um resistor cuja resistência pode ser variada. É disponível em uma grande variedade de valores de resistência e potência máxima dissipada. Possui um terminal móvel, que permite variar sua resistência além de dois terminais fixos.

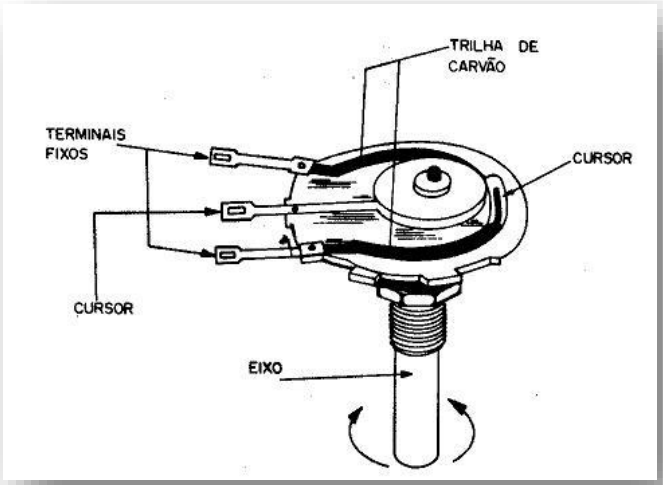


Figura 16: Esquema potenciômetro

LDR

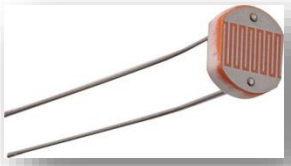


Figura 17: LDR

Também chamado de célula fotovoltaica ou fotoresistência é um dispositivo semicondutor de dois terminais, cuja resistência diminui quando sobre ele incide energia luminosa.

A resistência do LDR varia de forma inversamente proporcional à luz, segundo a equação: $R = C.L.a$, onde L é luminosidade (em Lux), C e a são constantes do material.

Quando a luminosidade no ambiente é alta, a resistência diminui e quando a luminosidade é baixa a resistência é alta.

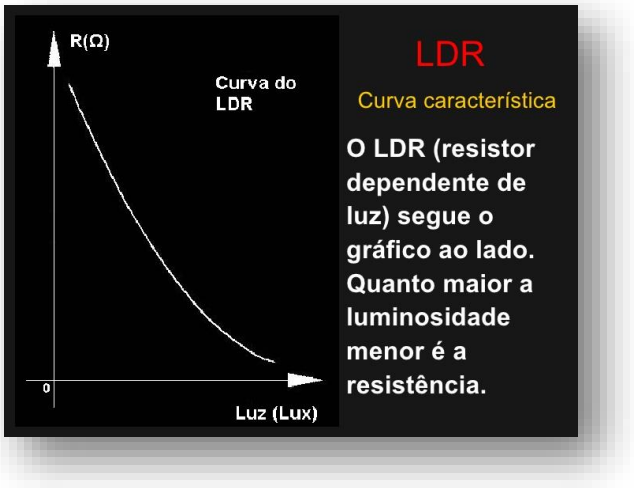


Gráfico 2: Resistência X Luminosidade do LDR

Com o LDR é possível fazer controle automático de portas, alarmes contra ladrão, controle de iluminação, contagem industrial entre outros.

Capacitor

Figura 18: Tipos de capacitores

É um dispositivo que pode ser utilizado para bloquear a passagem de corrente contínua (DC), mas permitir a passagem de corrente alternada (AC).

Quando é conectado a uma fonte de tensão tal como uma bateria, a corrente flui no capacitor até que ele tenha recebido tanta carga quanto possível. Esta capacidade de receber cargas é medida em unidades de µF ou pF. Se a fonte é removida do capacitor, a carga armazenada mantém a tensão constante através do capacitor, até que a carga se esgote e a tensão vai à zero. Este é o motivo para que os capacitores sejam utilizados como células de memória.

Indutores

Figura 19: Tipos de indutores

Assim como o capacitor, o indutor também é capaz de armazenar energia. Como a tensão versus corrente em um indutor é dada por: $v(t) = L.di(t)/dt$, pode ser utilizado tanto para armazenar energia quanto para impedir a variação brusca de corrente em um circuito, e também para produzir altos valores de tensão (no caso de fechamento ou abertura de uma chave liga-desliga).

Transistores

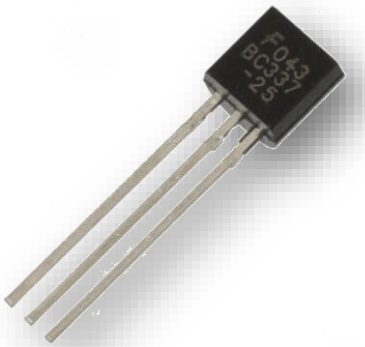


Figura 20: Transistor

São dispositivos, do tipo polarizado, que possuem três terminais. São utilizados como amplificadores ou chaves, também usados como retificadores elétricos em um circuito podendo ter variadas funções. Inventado em 1947, foi fruto do prêmio Nobel da de Física em 1956.

Muito simples de ser usado, um transistor é basicamente composto por três filamentos: Base, Emissor e Coletor. O emissor é o pólo positivo, o coletor, o pólo negativo, enquanto a base é quem controla o estado do transistor, que como vimos, pode estar ligado ou desligado. Um transistor ao ser desligado acaba não tendo carga na base, provocando consequentemente a não existência de corrente elétrica entre o emissor e o receptor. Assim cada transistor funciona como uma espécie de interruptor, na qual, pode estar ligado ou simplesmente desligado.

Diodos

Figura 21: Diodos

Um diodo é um dos mais simples e mais importantes componentes eletrônicos. É um dispositivo que possui dois terminais, onde o terminal "+" é o ânodo que está ligado á região p (onde há mais portadores positivos), e o terminal "-" é o cátodo que está ligado à região n (onde há mais portadores negativos).

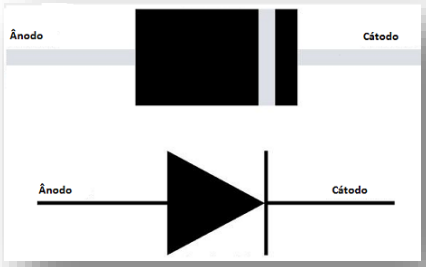


Figura 22: Símbolo de um diodo

O diodo tem três estados: Polarização direta, Polarização inversa e Disrupção; assim como mostra na figura 22.

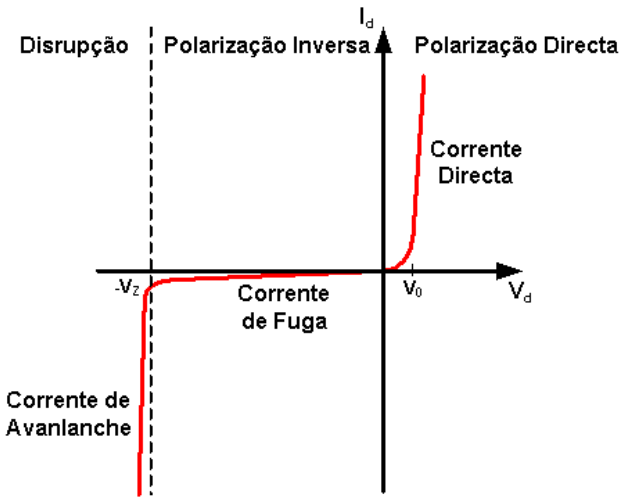


Gráfico 3: Estados de um diodo

Se aplicar uma tensão diretamente no diodo, ou seja, o positivo da fonte no ânodo (junção P) do diodo, e o negativo da fonte no cátodo (junção N) do diodo; diz-se que este está polarizado diretamente. Se a conexão, diz-se que está polarizado inversamente. Disrupção significa que o diodo queima a certa tensão (-V2 no gráfico).

Capítulo 3

Aula 1 - LED

Esta aula consiste em ligar LEDs ao Arduino podendo, desta maneira, controlar o ligar e desligar deste diodo emissor de luz.

Pisca LED

Materiais necessários

- *Protoboard*
- LED
- Resistor de 220 a 10k *ohms*
- *Jumpers*
- *Arduino*

Figura 23: Materiais necessários

Estudo dos materiais

Protoboard, também chamada de matriz de contatos, é uma placa de ensaio que contêm furos e conexões condutoras para montagem de circuitos elétricos experimentais. A protoboard é um dispositivo reutilizável, sem solda, utilizado para fazer um protótipo de um circuito eletrônico ou para experimentar projetos de circuitos. A placa consiste em uma série de furos em uma grade; sob a placa, esses furos são conectados por uma tira de metal condutivo. No sentido coluna da matriz, os furos encontram-se internamente conectados, ou seja, caso você coloque dois componentes em furos de mesma coluna, estes estarão ligados de maneira que não seja necessário o uso da soldagem para uni-los.O sentido linha é o sentido se passagem de corrente.

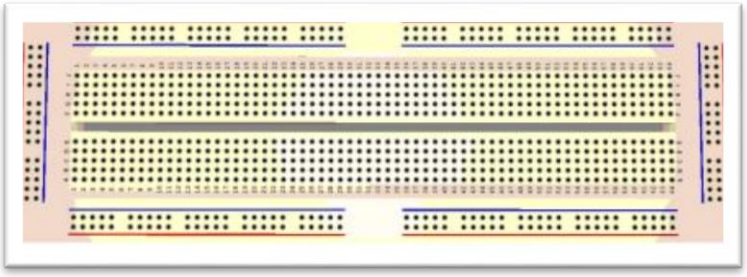


Figura 24: Matriz de contatos

As tiras ao longo do topo e da base correm em paralelo à placa, e são projetadas para carregar o barramento de alimentação e o barramento do terra. Os componentes no meio da placa convenientemente conectam com os 5 V (ou a voltagem que você estiver utilizando) ou com o terra. Algumas protoboards têm uma linha vermelha e outra preta correndo paralelas a esses furos, para mostrar qual é a alimentação (vermelho) e qual é o terra (preto).

LED (*Light emitter diode*), diodo emissor de luz, é um componente eletrônico semicondutor que possui a propriedade de transformar energia elétrica em luz. É um componente polarizado que deve ser corretamente conectado. Se observar irá perceber que o LED possui um terminal maior, chamado de ânodo, e o outro será o cátodo. Alguns apresentam o pólo negativo (cátodo) com um chanfro (parte ligeiramente plana) em seu encapsulamento.
Temos no *Arduino*:

- VCC(+)
- GND ou *Ground* como (-).

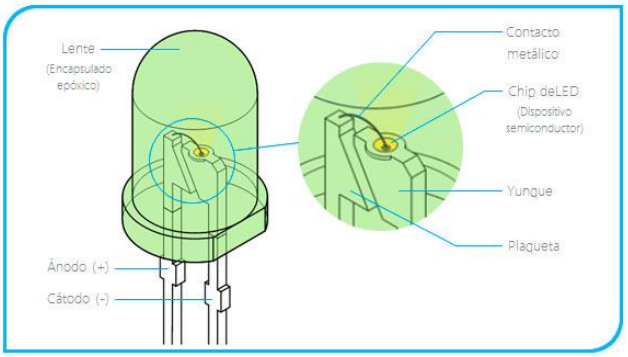
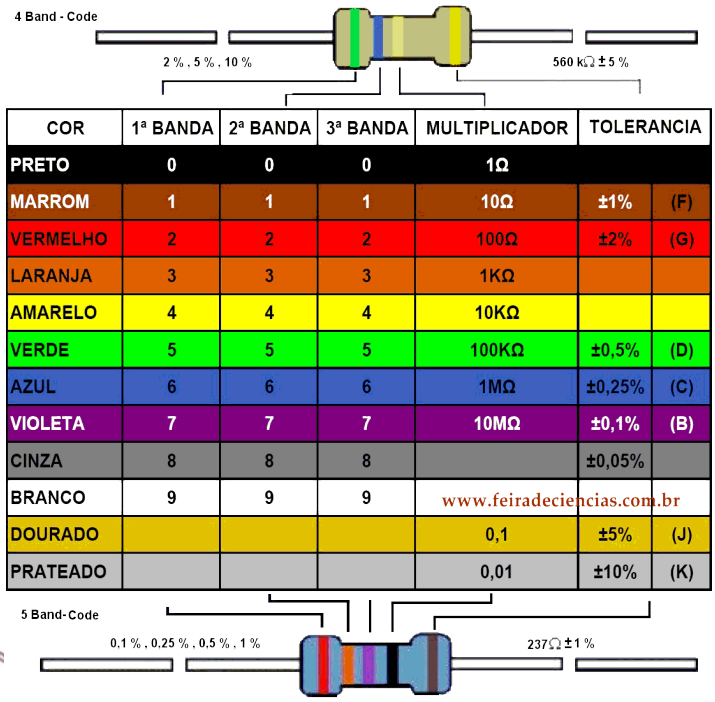


Figura 25: LED e sua estrutura

Resistor é um dispositivo elétrico muito utilizado na eletrônica que possui duas principais finalidades: transformar energia elétrica em energia térmica ou limitar a corrente elétrica de um circuito, impedindo que componentes queimem. No caso do projeto 1, utilizaremos o resistor para a segunda finalidade citada.



COR	1ª BANDA	2ª BANDA	3ª BANDA	MULTIPLICADOR	TOLERANCIA	
PRETO	0	0	0	1Ω		
MARROM	1	1	1	10Ω	±1%	(F)
VERMELHO	2	2	2	100Ω	±2%	(G)
LARANJA	3	3	3	1KΩ		
AMARELO	4	4	4	10KΩ		
VERDE	5	5	5	100KΩ	±0,5%	(D)
AZUL	6	6	6	1MΩ	±0,25%	(C)
VIOLETA	7	7	7	10MΩ	±0,1%	(B)
CINZA	8	8	8		±0,05%	
BRANCO	9	9	9	www.feiradeciencias.com.br		
DOURADO				0,1	±5%	(J)
PRATEADO				0,01	±10%	(K)

Figuras 26 e 27: Resistor 220 ohms e tabela de resistência

Utilizando a tabela acima, pode-se descobrir qual a real resistência de um resistor.

- ✓ PRIMEIRA FAIXA (mais próxima da extremidade): Indica o primeiro algarismo do valor da resistência;
- ✓ SEGUNDA FAIXA: indica o segundo algarismo do valor da resistência;
- ✓ TERCEIRA FAIXA: indica o número de zeros que devem ser colocados aos algarismos;
- ✓ QUARTA FAIXA: indica a imprecisão ou tolerância do valor da resistência.

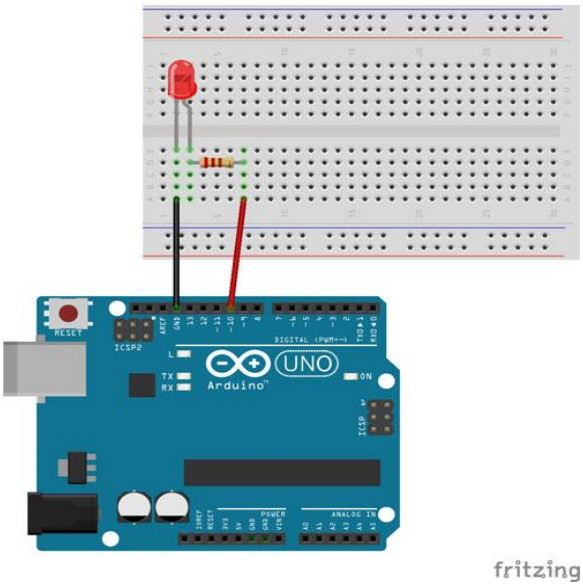
OBS: A inexistência da quarta faixa pressupõe-se uma tolerância de 20% no valor da resistência, para mais ou para menos. Para os resistores de precisão encontramos cinco faixas, onde as três primeiras representam o primeiro, o segundo e o terceiro algarismos significativos, e as demais o fator multiplicativo e a tolerância, respectivamente.

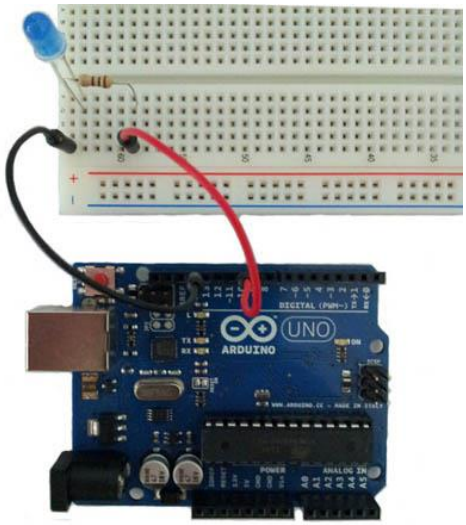
Jumpers são fios com pontas moldadas para facilitar sua inserção na protoboard ligando pontos distantes de um circuito.

Figura 28: fios jumper

Montagem na *protoboard*

Faça a montagem dos componentes na *protoboard* da maneira mostrada a seguir:





Figuras 29 e 30: Montagem do circuito na *protoboard*

O LED deve estar conectado corretamente na *protoboard*, com o terminal mais longo conectado a um pino digital. O terminal longo é o ânodo e deve sempre ir para a alimentação de 5V (neste caso, saindo do pino digital); o terminal curto é o cátodo e deve ir para o terra (GND).

Quando tudo estiver seguro, ligue seu *Arduino* conectando-o, utilizando o cabo USB, ao computador.

A programação

Escreva na interface de programação o código a seguir:

```
void setup() {

pinMode(10, OUTPUT); //Declara que o pino 12 do Arduino é de Saída. Vai mandar dados, energia...}

void loop() {

digitalWrite(10, HIGH);  // Diz que o pino 12 do Arduino está Ligado. Logo LED ON

delay(1000); // Espera por 1s

digitalWrite(10, LOW); //  Diz que o pino 12 do Arduino está Desligado. Logo: LED OFF

delay(1000); // Espera por 1s}
```

Comentários: No setup () declara-se que usaremos o pino 12 digital do *Arduino* e como se trata de uma saída de dados escreve-se OUTPUT. No loop () existem duas tarefas a serem executadas, a de ligar e a outra desligar o LED. Para ligá-lo utilizamos HIGH e para desligá-lo utilizamos LOW. O tempo que cada comando deve ocorrer informa-se no *delay* e escreve-se o tempo em milissegundos.

Semáforo comum

Neste projeto iremos simular um sinal de trânsito na utilização de LEDs coloridos.

Materiais necessários

- Arduino
- Protoboard
- LED vermelho
- LED verde
- LED amarelo
- 3 resistores 220 ohms
- Jumpers

Montagem na protoboard

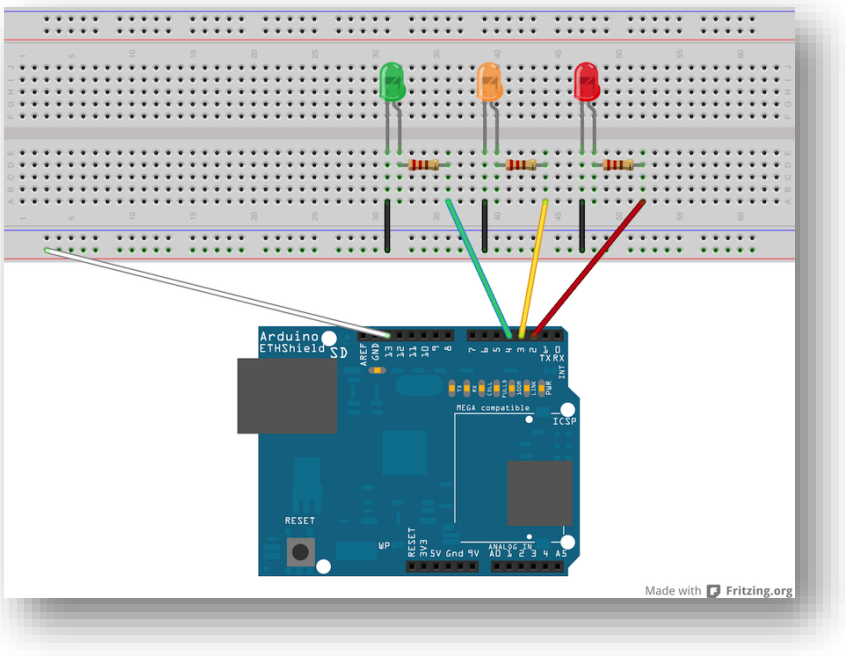


Figura 31: Montagem do semáforo

A programação

```
int verde = 4;

int amarelo = 3;

int vermelho = 2;

void setup(){

  pinMode(verde,OUTPUT);

  pinMode(amarelo, OUTPUT);

  pinMode(vermelho,OUTPUT);

}
```

```
void loop(){
  digitalWrite(verde,HIGH);
  digitalWrite(amarelo,LOW);
  digitalWrite(vermelho,LOW);
  delay(2000);
  digitalWrite(verde,LOW);
  digitalWrite(amarelo,HIGH);
  digitalWrite(vermelho,LOW);
  delay(2000);
  digitalWrite(verde,LOW);
  digitalWrite(amarelo,LOW);
  digitalWrite(vermelho,HIGH);
  delay(2000);
}
```

Aula 2 - Botão

Botão para ligar LED

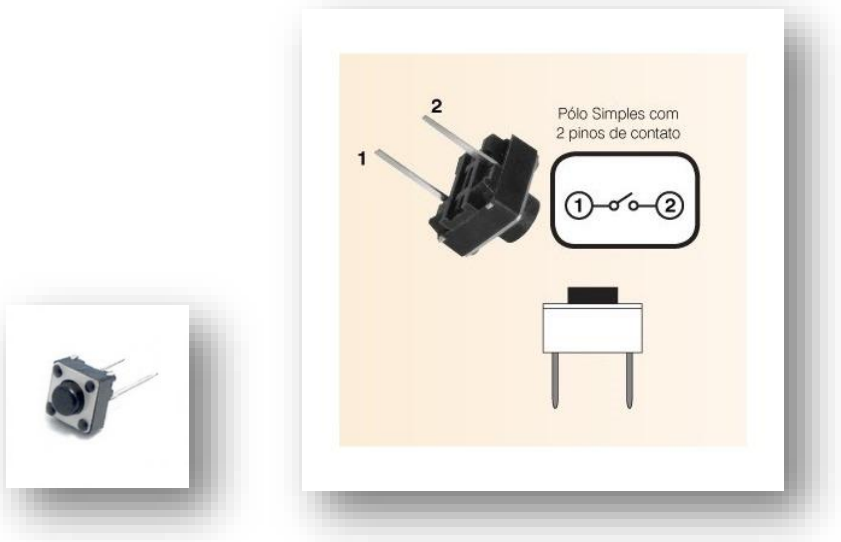
Este projeto consiste em controlar o funcionamento de um LED, ligando e desligando o mesmo através de um *push button*. Este projeto básico introduz a utilização de botões, ou *push buttons*, em projetos *Arduino*, permitindo a você dar um passo a mais no seu aprendizado.

Materiais necessários

- *Protoboard*
- Botão
- LED
- Resistor de 220 *ohms* (para LED) e 10k *ohms* (para botão)
- *Jumpers*
- *Arduino*

Estudo dos componentes

O **botão** que iremos utilizar neste projeto é do tipo micro chave *push Button dip,* ideal para *protoboard*. O interruptor momentâneo é um componente que conecta dois pontos de um circuito ao pressioná-lo.



Figuras 32 e 33: botões (*push Button*)

Montagem na *protoboard*

Faça a seguinte montagem na sua *protoboard:*

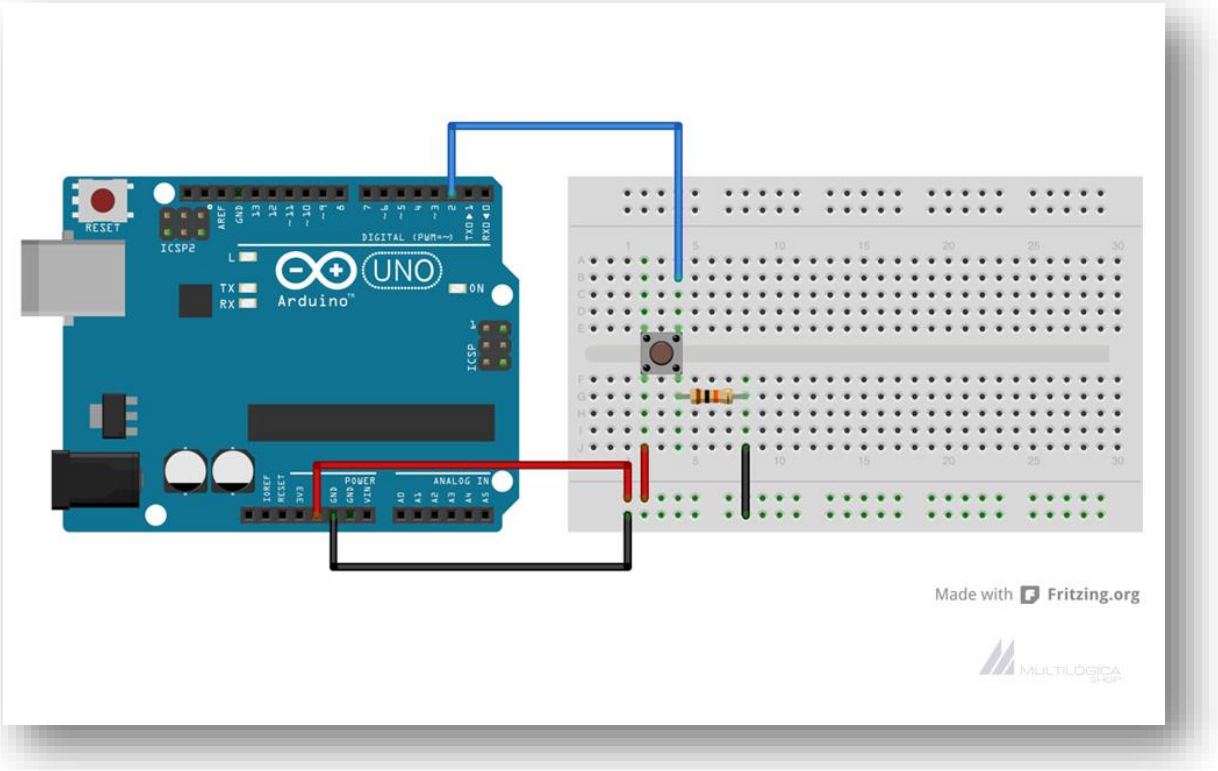


Figura 34: circuito montado na *protoboard*

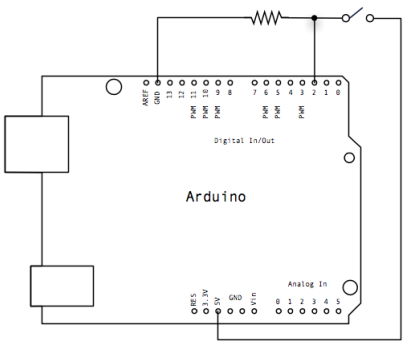


Figura 35: Circuito representado (botão)

Conecte, inicialmente, dois fios ao Arduino; o primeiro vai de um dos terminais do botão ao 5V e o segundo do outro terminal até um pino digital qualquer. Ligue um terminal do resistor de 10k ohms nesse último terminal e o outro terminal do resistor ao GND.

Ligue agora o LED da mesma maneira que foi ensinado no projeto anterior:

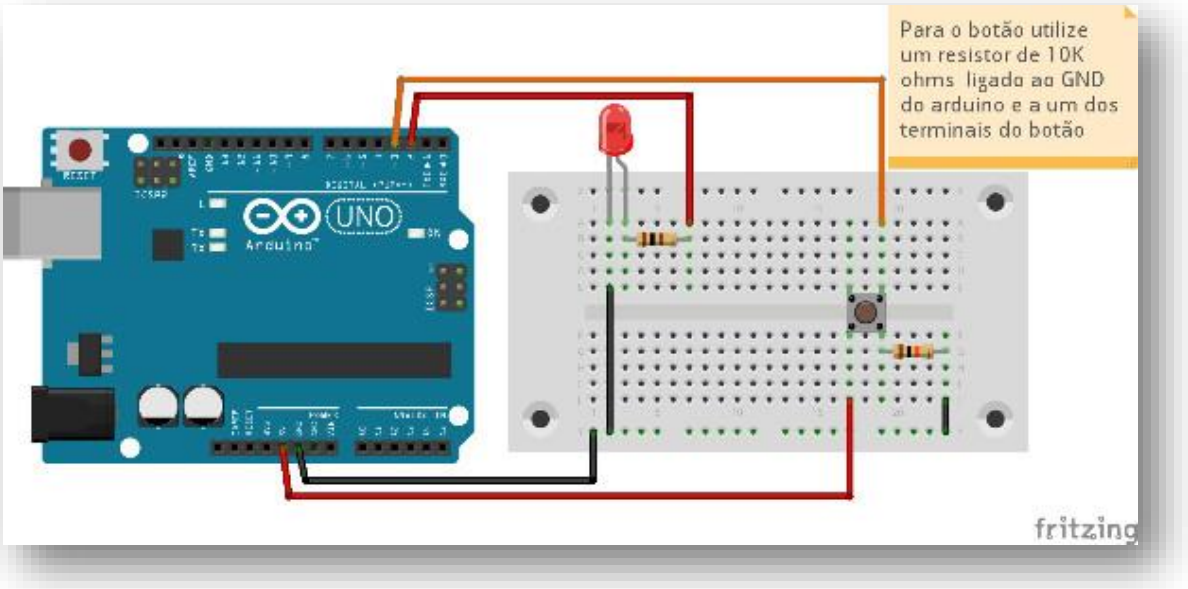


Figura 36: Montagem do botão para ligar led

A programação

```
int ledPin = 2; // escolha o pino para o LED

int inPin = 3;   // escolha o pino de entrada (para o interruptor)

int val = 0;     // variável para ler o estado do interruptor

void setup() {

  pinMode(ledPin, OUTPUT);  // declara-se o LED como saída

  pinMode(inPin, INPUT);    // declara-se o interruptor como entrada de dados

}
```

```
void loop(){
 val = digitalRead(inPin);  // ler o valor de entrada
 if (val == HIGH) {        // verificar se a entrada é HIGH (interruptor livre)
   digitalWrite(ledPin, HIGH);  // ligar LED }
else {
   digitalWrite(ledPin, LOW);  // desligar LED }}
```

Controlando cores de um LED RGB

Neste projeto veremos como controlar as cores de um led RGB utilizando botões.

Materiais necessários

- Arduino
- Protoboard
- 3 botões
- Jumpers
- LED RGB
- 6 resistores: 3 de 150 ohms e 3 de 10k ohms

Montagem na protoboard

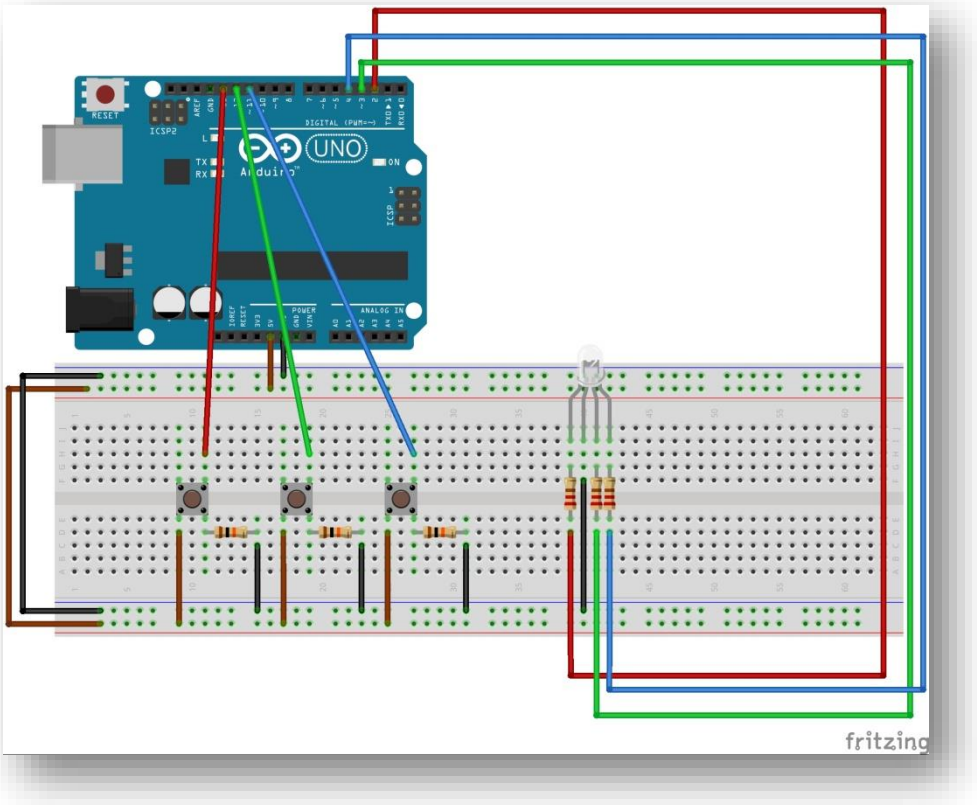


Figura 37: circuito rgb e botões

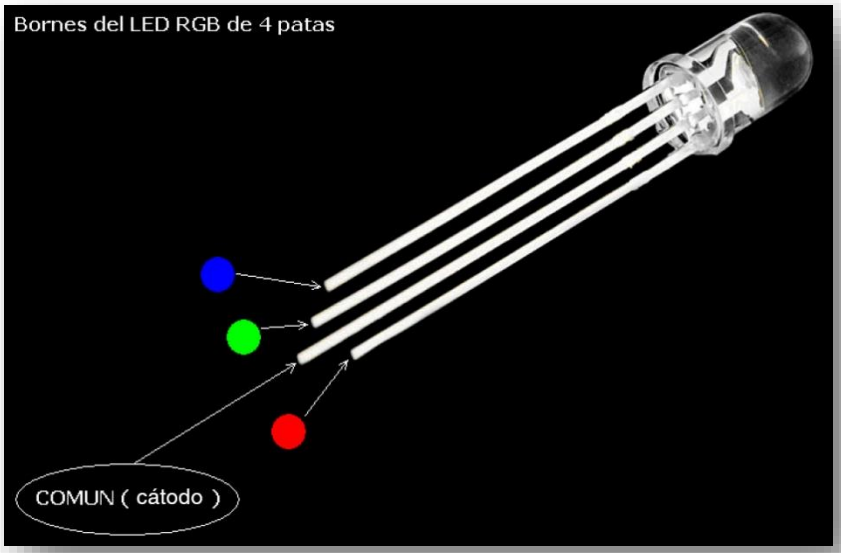


Figura 38: Terminais do led RGB

A programação

```
//Declaração das constantes
const int ledAzul = 4;      //ledAzul refere-se ao pino digital 8.
const int ledVerde = 3;     //ledVerde refere-se ao pino digital 9.
const int ledVermelho = 2; //ledVermelho refere-se ao pino digital 10.
 const int botaoAzul = 11;     //botaoAzul refere-se ao pino digital 2.
const int botaoVerde = 12;    //botaoVerde refere-se ao pino digital 3.
const int botaoVermelho = 13; //botaoVermelho refere-se ao pino digital 4.
 //Variáveis que conterão os estados dos botões (pressionados ou não).
int estadoBotaoAzul;
int estadoBotaoVerde;
int estadoBotaoVermelho;
 //Método setup, executado uma vez ao ligar o Arduino.
void setup() {


  pinMode(ledAzul,OUTPUT);
  pinMode(ledVerde,OUTPUT);
  pinMode(ledVermelho,OUTPUT);
```

```
  pinMode(botaoAzul,INPUT);
  pinMode(botaoVerde,INPUT);
  pinMode(botaoVermelho,INPUT);
}
 //Método loop, executado enquanto o Arduino estiver ligado.
void loop() {
  //Lendo o estado dos botões através dos pinos digitais.
 estadoBotaoAzul = digitalRead(botaoAzul);
 estadoBotaoVerde = digitalRead(botaoVerde);
 estadoBotaoVermelho = digitalRead(botaoVermelho);
    //Acendendo o led RGB conforme os botões pressionados.
 //Tratando a cor azul
 if (estadoBotaoAzul == HIGH) {
  digitalWrite(ledAzul,HIGH);
 } else {
  digitalWrite(ledAzul,LOW);
 }
   //Tratando a cor verde
 if (estadoBotaoVerde == HIGH) {
  digitalWrite(ledVerde,HIGH);
 } else {
  digitalWrite(ledVerde,LOW);
 }

 //Tratando a cor vermelha
 if (estadoBotaoVermelho == HIGH) {
  digitalWrite(ledVermelho,HIGH);
 } else {
  digitalWrite(ledVermelho,LOW);
 }
}
```

Semáforo interativo

Este projeto consiste em fazer um semáforo que possui botão para pedestre. Quando o sinal está verde para os carros e o pedestre quer passar, ele aperta o botão e automaticamente o sinal para carros fica vermelho, podendo assim atravessar a rua.

Materiais necessários

- Arduino
- Protoboard
- Botão
- LED vermelho
- LED verde
- LED amarelo
- 3 resistores 220 ohms e 1 resistor 10k ohms
- Jumpers

Montagem na protoboard

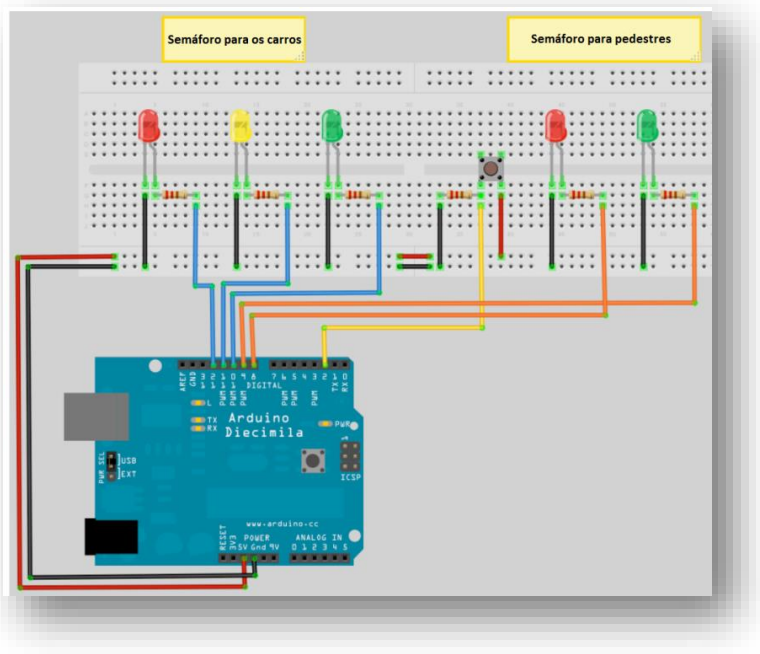


Figura 39: Montagem do semáforo interativo

A programação

```
int carRed = 12; // definindo os leds dos carros
int carYellow = 11;
int carGreen = 10;
int pedRed = 9; // definindo os leds de pedestres
int pedGreen = 8;
int button = 2; // pino do botao
int crossTime = 10000; // tempo de travessia

unsigned long changeTime; // tempo desde a ultima vez que o botao foi pressionado
```

```
void setup() {
  pinMode(carRed, OUTPUT);
  pinMode(carYellow, OUTPUT);
  pinMode(carGreen, OUTPUT);
  pinMode(pedRed, OUTPUT);
  pinMode(pedGreen, OUTPUT);
  pinMode(button, INPUT);

 digitalWrite(carGreen, HIGH);
  digitalWrite(pedRed, HIGH);}

void loop() {
  int state = digitalRead(button);

 /*verifica se o botao foi pressionado e se faz mais de 5 seg desde a ultima vez*/
  if (state == HIGH && (millis() - changeTime) > 5000) {

  //chama a funcao que altera as luzes
    delay(4000);
    changeLights();  }}
void changeLights(){
  digitalWrite(carGreen, LOW); //apaga carGreen
  digitalWrite(carYellow, HIGH); //acende carYellow
  delay(4000);
  digitalWrite(carYellow, LOW); //apaga carYellow
  digitalWrite(carRed, HIGH); //acende carRed
  digitalWrite(pedRed, LOW); //apaga pedRed
  digitalWrite(pedGreen, HIGH); //acende pedGreen
  delay(crossTime); //aguarda crossTime

   //pisca pedGreen
  for (int x=0; x<10; x++) {
    digitalWrite(pedGreen, HIGH);
    delay(250);
    digitalWrite(pedGreen, LOW);
    delay(250); }

  digitalWrite(pedRed, HIGH); //acende pedRed
  digitalWrite(carRed, LOW); //apaga carRed
  digitalWrite(carGreen, HIGH); //acende carGreen
   changeTime = millis();}
```

Aula 3 - LDR

Sensor de Luz (LDR)

Este projeto consiste em fazer a leitura, no serial monitor, de um sensor de luz.

Componentes necessários

- *Protoboard*
- LDR
- Resistor 1k ohm
- *Jumpers*
- *Arduino*

Estudo dos componentes

O **LDR** (Light Dependent Resistor) é um componente eletrônico que tem a característica de variar a resistência com a luminosidade. A relação é inversamente proporcional: quando a luminosidade é alta a resistência é baixa e tem limite zero Ohm, e, analogamente, quando a luminosidade é baixa a resistência é alta, por volta de 1k Ohms.

Por causa desta característica interessante, a partir do LDR foi possível criar um sensor que é ativado ou desativado quando incide certa quantidade de luz.

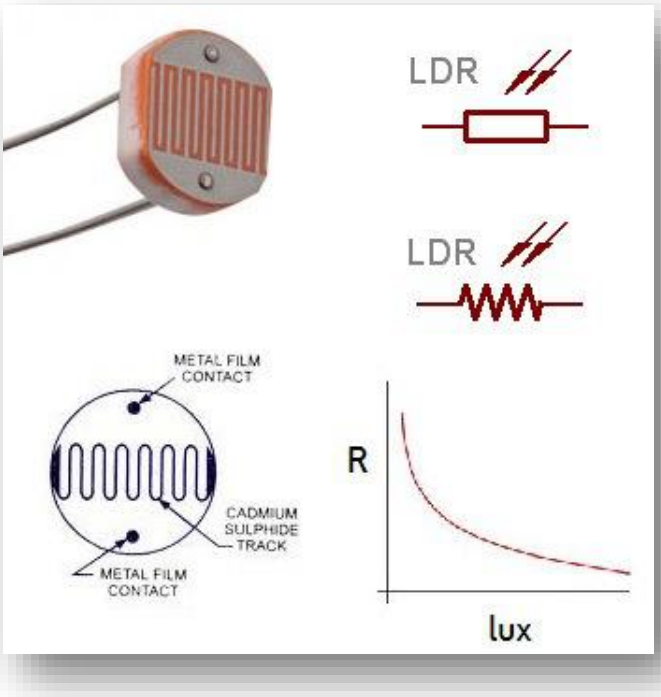


Figura 40: Sensor LDR

Montagem na Protoboard

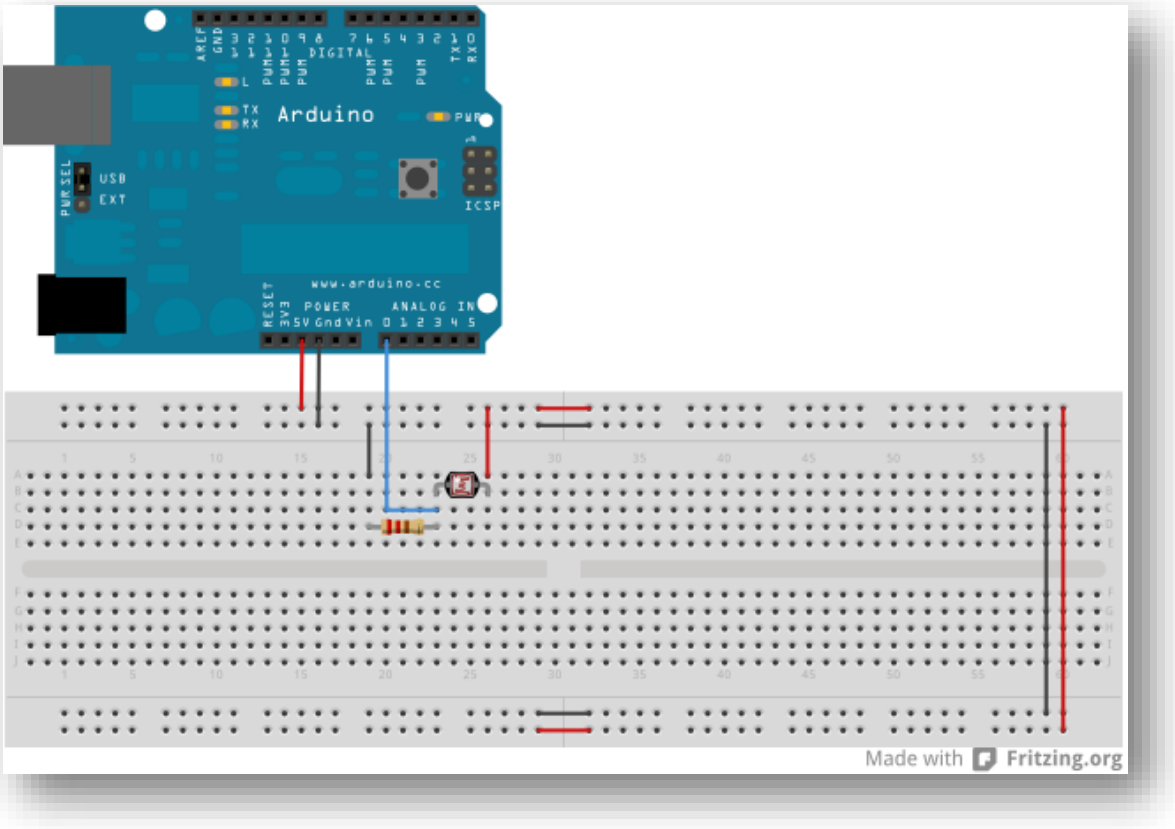


Figura 41: Sensor de luz (montagem)

O programa

A leitura do sensor LDR é através de uma entrada analógica do Arduino, que irá converter o sinal analógico em digital, entre 0 e 1023 conforme a quantidade de luz no ambiente. Com este valor pode-se verificar a variação luminosa no terminal serial, e testar diferentes intensidades luminosas para ver o funcionamento do sensor.

```
int sensor = A0; //Pino analógico em que o sensor está conectado.

int valor = 0; //Usada para ler o valor do sensor em tempo real.

void setup(){

Serial.begin(9600);

}

void loop(){

valor = analogRead(sensor);

Serial.println (valor);

delay(100);}
```

Controle de luminosidade

Utilizando LED

Como primeiro projeto de automação, este tem a função de controlar o ligar e desligar do LED na dependência da luminosidade ambiente.

Componentes necessários

- *Protoboard*
- LDR
- LED
- Resistor 220 e 10k ohm
- *Jumpers*
- *Arduino*

Montagem na Protoboard

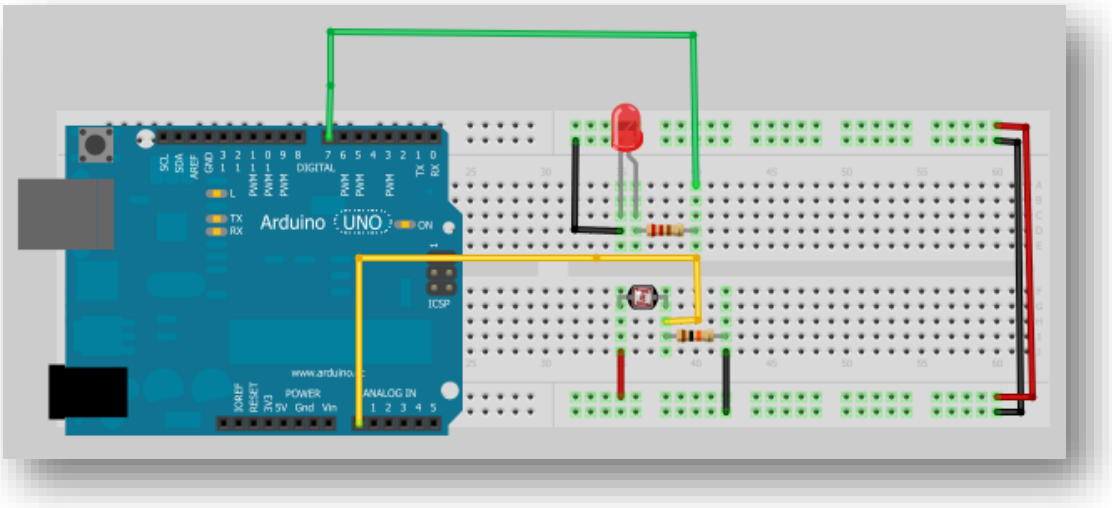


Figura 42: Montagem na protoboard

O programa

```
int sensor = A0;

int valor =0;

int led = 7;

void setup(){

Serial.begin(9600);

pinMode(led, OUTPUT);}

void loop(){
```

```
valor = analogRead(sensor);

Serial.print (valor);

delay(100);

if ( valor <= 500){

digitalWrite(led, HIGH);}

else {

digitalWrite(led,LOW);}}
```

O valor 500 foi obtido experimentalmente de acordo com o projeto anterior, utilizando o monitor serial para fazer leituras. Percebeu-se que o número 500 significa o ponto médio entre a luminosidade e a penumbra. Valores acima de 500 são caracterizados por ambientes luminosos e abaixo por lugares escuros. Sabendo disso, automatizamos o LED para ligar quando estiver escuro, iluminando o lugar. Perceba que foi utilizado o comando IF, que é um comando condicional.

Obs: É aconselhável testar os valores antes de fazer a programação para que a automação ajuste-se perfeitamente.

Utilizando lâmpada

Este projeto tem como objetivo controlar a luminosidade ambiente utilizando um sensor de luz (LDR) e uma lâmpada. O projeto que faremos a seguir é o mesmo utilizado nos postes de luz nas cidades:

Figura 43: poste de luz

Componentes necessários

- *Protoboard*
- LDR
- Lâmpada

- Módulo relé (eletromecânico)
- Bucal para lâmpada e conector para tomada
- Resistor 220 e 10k ohm
- *Jumpers*

Figura 44: Módulo relé

Estudo dos componentes

O **relé** é um dispositivo eletromecânico, formado por um magneto móvel, que se desloca unindo dois contatos metálicos. O funcionamento de um relé é bem simples, quando uma corrente circula pela bobina esta cria um campo magnético que atrai um ou uma série de contatos fechando ou abrindo circuitos.

Ao fechar corrente da bobina o campo magnético também fecha, fazendo com que os contatos voltem á posição inicial. Possui contatos do tipo NA (normal aberto), NF (normal fechado) e um C (comum ou central). Os contatos NA são os que estão abertos quando a bobina não está energizada e que fecham quando a bobina recebe corrente. Os NF abrem-se quando a bobina recebe corrente, ao contrário dos NA. O C é o contato que estabelece condução ao fechar com NA ou NF.

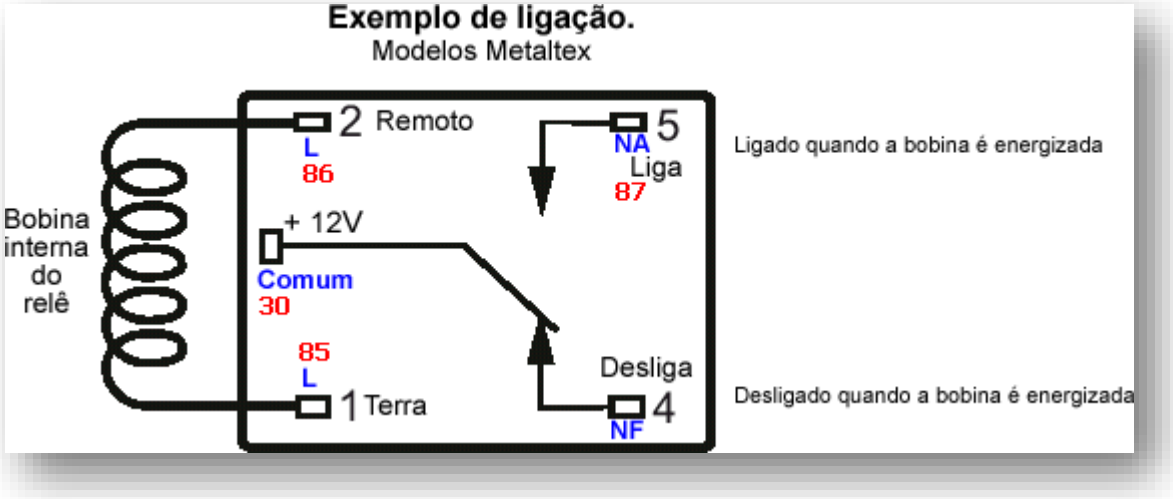


Figura 45: Composição do relé eletromecânico

Montagem na protoboard

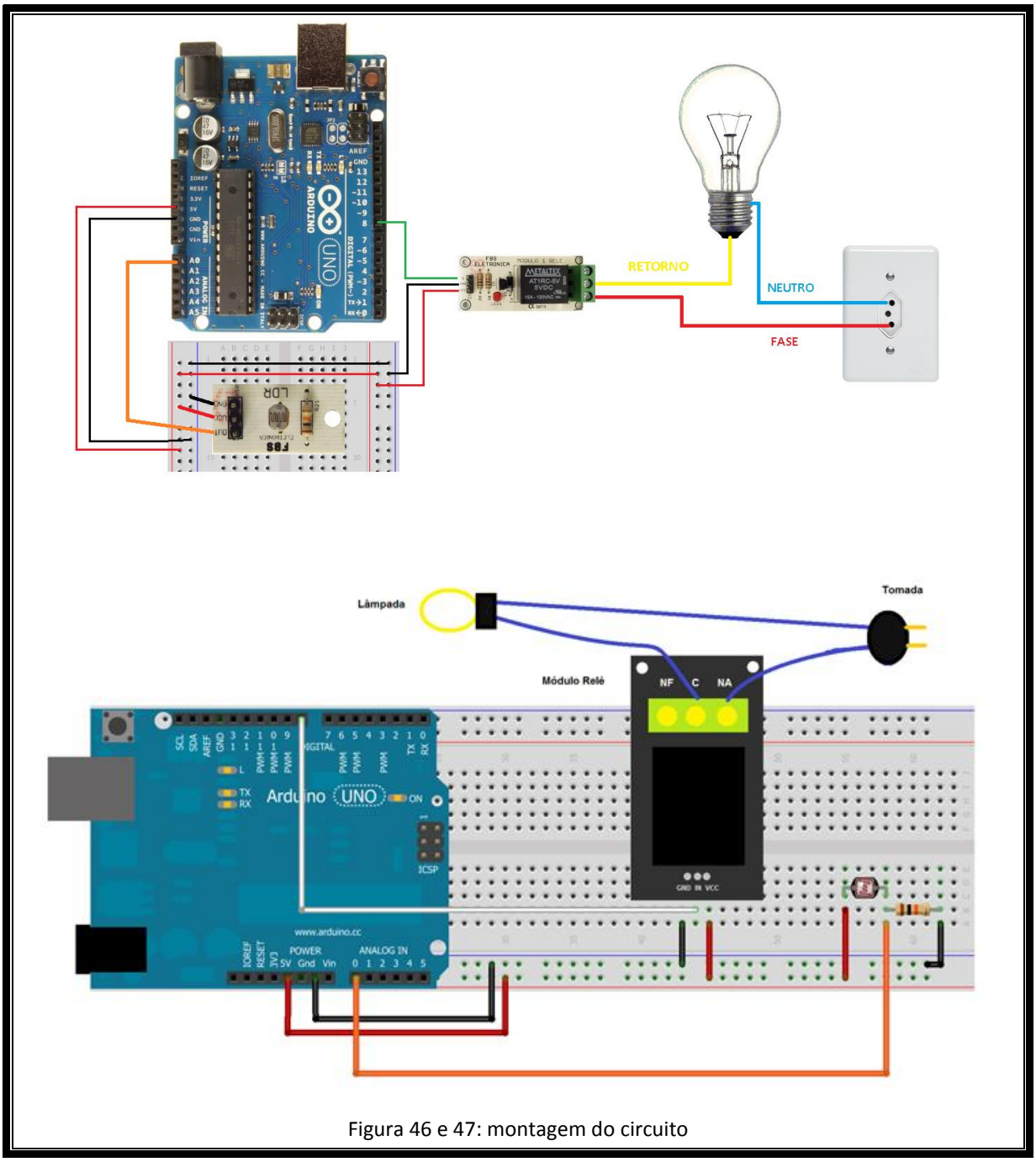


Figura 46 e 47: montagem do circuito

O programa

```
#define LDR A0 //Define LDR como A0
#define rele 8 //Define lâmpada como 8

float luminosidade; //Variável para armazenar o valor da luminosidade
float amostra; //Variável para armazenar a amostra

void setup()
{
```

```
Serial.begin(9600); //Inicia a Serial do Arduino

pinMode(rele, OUTPUT); //Configura pino 8 como saída
digitalWrite(rele, LOW); //Desliga a lâmpada

}

void loop()
{

amostra = 0; //Zera a variável amostra   //Pega 100 amostras

for(int i=0; i < 100 ; i++)
{
luminosidade = analogRead(LDR); //Armazena o valor lido pelo LDR
luminosidade = (luminosidade / 1024) * 100; //Transforma o valor lido em porcentagem
amostra = amostra + luminosidade; //Armazena na variável amostra
}

amostra = amostra/100; //Tira a média das amostras
Serial.print("Luminosidade = "); //Imprime na serial "Luminosidade = "
Serial.print(amostra, 0); //Imprime a amostra sem casas decimais
Serial.println("%"); //Imprime o símbolo de porcentagem
if(amostra < 30) //Se a luminosidade estiver abaixo de 30%
{
digitalWrite(rele,HIGH); //Liga a lâmpada
}
else //Senão
{
digitalWrite(rele,LOW); //Desliga a lâmpada
}
delay(250); //Delay de 250 milissegundos

}
```

É importante destacar que foi tirada uma média dos valores lidos pelo sensor para diminuir o desvio e fazer a medição com maior precisão. O sketch usado no projeto 4 parte 1 pode ser usado igualmente para a parte 2, mas você irá perceber um melhor funcionamento utilizando o programa acima.

Outras diferenças notáveis são: utilização o #define ao invés do int para definir variáveis inteiras e a escrita do valor da luminosidade em porcentagem e por isso foi necessário criar vaiáveis float (utilizada para valores reais).

Pode-se perceber que há diversas maneiras de escrever um programa que excuta uma mesma função, alguns mais simples de escrever e outros que executam um melhor desempenho.

Aula 4 - LM35

Sensor de Temperatura (LM35)

Componentes necessários

- *Protoboard*
- LM35
- *Jumpers*

Estudo dos componentes

O sensor **LM35** é um sensor de precisão, fabricado pela National Semiconductor, que apresenta uma saída de tensão linear proporcional à temperatura a que se encontra, no nosso caso esta tensão varia entre 0V e 5V, pois o sensor é alimentado com 5V.

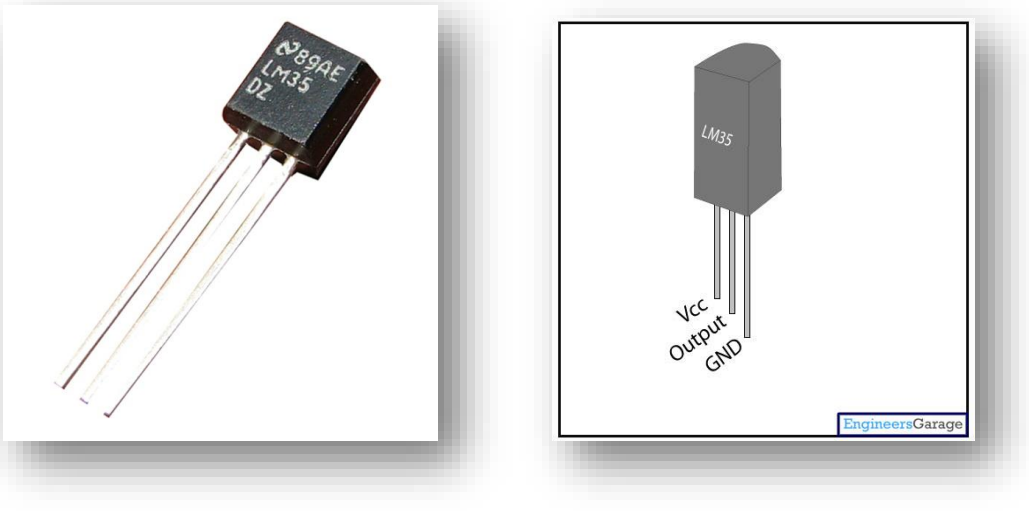


Figura 48 e 49: LM35

Montagem na protoboard

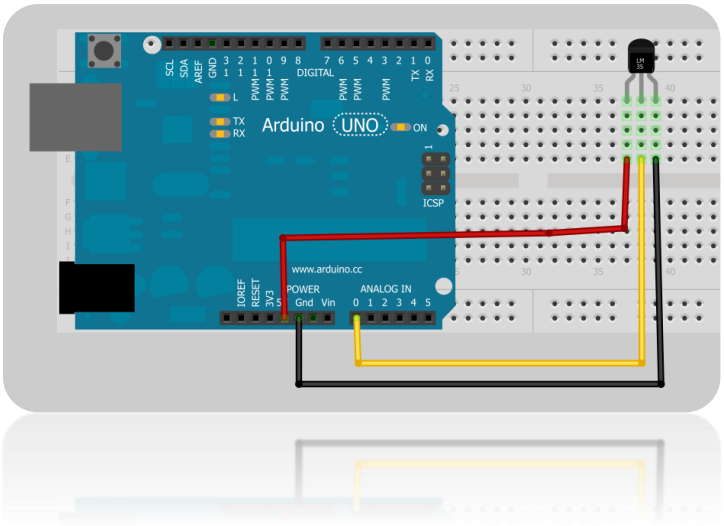


Figura 50: Montagem do projeto 5 na protoboard

O programa

```
//Programa que mede temperaturas por meio de um sensor, visualização via Serial Monitor.

int sensor = 0;

int temperatura;

void setup() {

Serial.begin(9600);

}

void loop() {

temperatura= 5*100* analogRead(sensor); //converte em graus o valor lido

Serial.print("Temperatura:  ");

Serial.println(temperatura);

}
```

As entradas e saídas digitais possuem somente dois estados, Alto (1) e Baixo (0). Para a leitura, para ser considerado estado Alto o pino precisa estar entre 3V e 5V e estado Baixo entre 0V e 1V. Caso o esteja entre 1V e 3V o estado atribuído não muda, ou seja, continua o mesmo estado antes do pino entrar nessa faixa de valores. Algumas vezes necessitamos que o Arduino interprete com mais precisão a tensão aplicada a um pino. Para isso utilizamos a Leitura Analógica.
A Leitura Analógica permite que o Arduino traduza a tensão em valores que variam de 0 até 1023. Isso é realizado por um circuito interno, chamado Conversor Analógico Digital (CAD). O CAD faz a conversão da tensão e seta 10 bits de acordo com a tensão, ou seja, resulta em um valor com 10 bits de resolução.
O Arduino UNO, por exemplo, possui 6 pinos que permitem a Leitura Analógica. Para realizá-la devemos utilizar a função "analogRead(pino)", onde "pino" deve ser o número do pino onde se deseja fazer a leitura. Essa função retornará um valor inteiro de 0 a 1023.

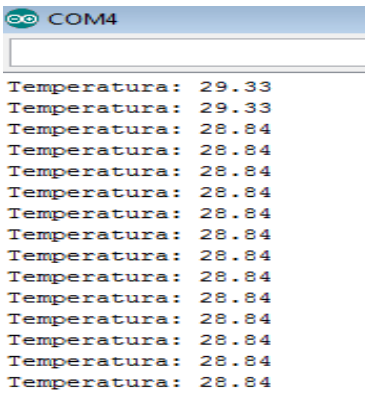


Figura 51: Temperatura via serial monitor

Controle de temperatura

Utilizando um cooler

Utilizando um cooler, iremos neste projeto controlar a temperatura ambiente. Este é mais um pequeno projeto de automação muito útil para nosso dia a dia.

Componentes necessários

- *Protoboard*
- LM35
- Cooler
- Relé de estado sólido
- Fonte de alimentação 12v 1A
- *Jumpers*

Figura 52: Componentes do projeto 6

Estudo dos componentes

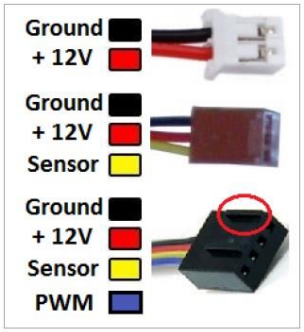
Relé de estado sólido é um dispositivo semicondutor capaz de desempenhar as mesmas funções dos relés eletromecânicos comuns (citados no projeto 4 parte 2), porém seu sistema de funcionamento é completamente diferente.

O relé de estado sólido não possui partes mecânicas, pois opera por meio de *tiristores* (chaveamento do estado de bloqueio e estado de condução) que comutam quando determinada corrente passa por eles, este processo ocorre não só com o *tiristor*, mas também nos transistores ou *triacs*. A necessidade de contatos metálicos no interior do relé é eliminada o que aumenta exponencialmente sua vida útil e a segurança da operação além de eliminar o barulho e requer cargas menores para a alimentação.

Figura 53: Relé de estado sólido

Geralmente, os coolers vêm com três ou quatro fios de diferentes cores e cada um representa uma funcionalidade distinta.

Se o cooler tiver três fios com as cores vermelho, amarelo e preto:



Preto: terra

Amarelo: sensor

Vermelho: 12 v

Figuras 54 e 55: cooler ; funções das cores dos fios

Se o cooler tiver quatros fios com as cores amarelo, preto, verde e azul:

Preto: terra

Amarelo: 12 v

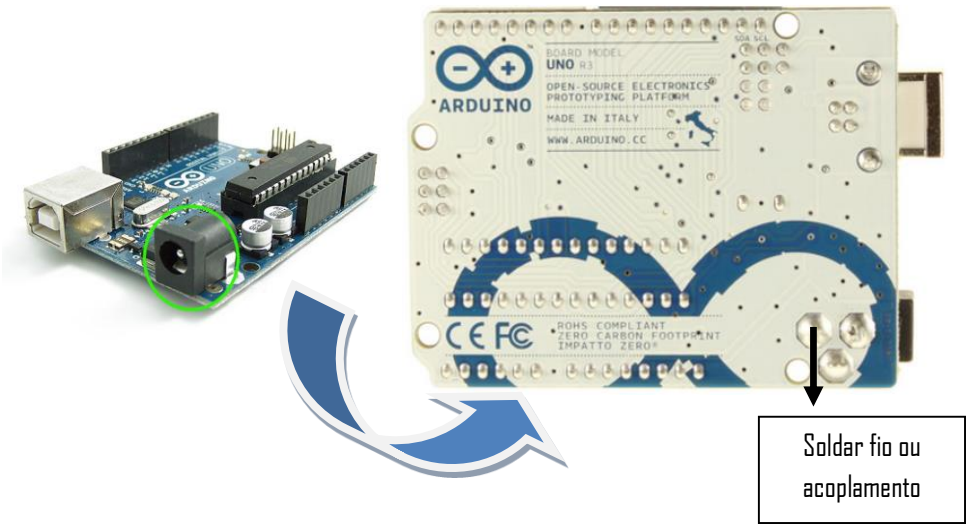
Verde: sensor de velocidade

Azul: Controlador PWM

Figura 56: cooler

Montagem na protoboard

- 1º Passo: Solde um fio ou um acoplamento fêmea para fio na parte de trás do Arduino, especificamente onde mostra a figura a seguir:



OBS: Este passo é necessário para conseguirmos acesso a 12 V diretamente da fonte, já que este ponto exato corresponde ao positivo da entrada de fonte de alimentação.

- 2º Passo: Conecte a fonte de alimentação 12v ao Arduino

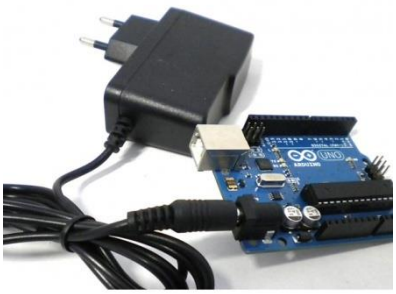


Figura 57: Arduino ligado à alimentação

- 3º Passo: faça a seguinte montagem na protoboard:

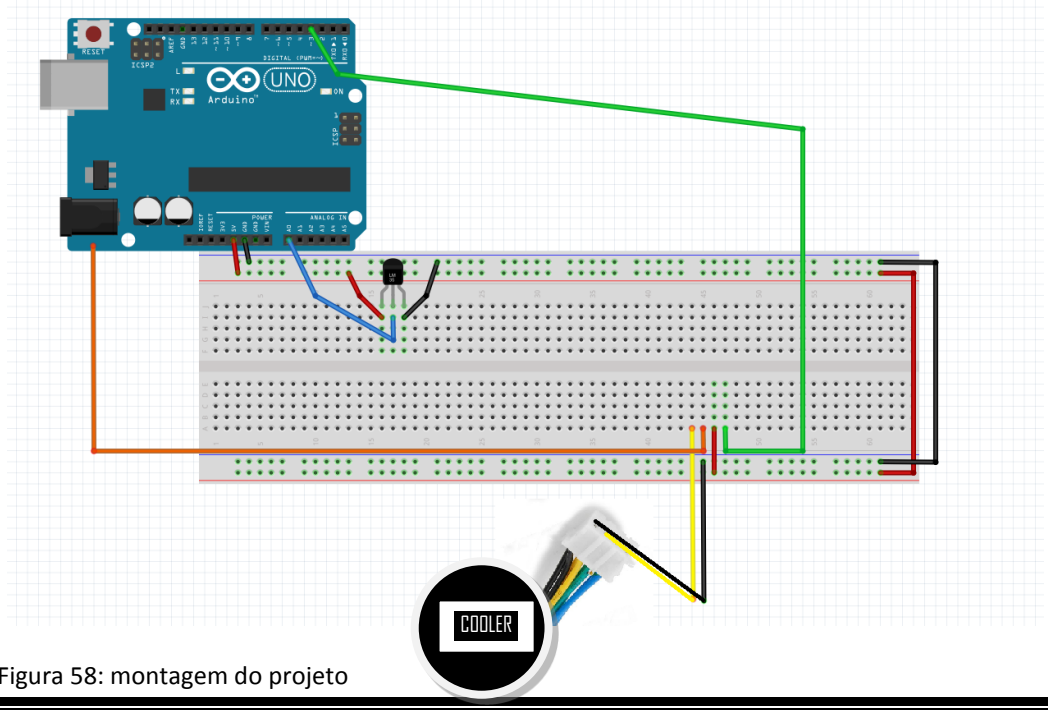


Figura 58: montagem do projeto

O programa

```
int cooler = 3;

int sensor = A0;

int temperatura;

void setup(){

pinMode(cooler,OUTPUT);

Serial.begin(9600);
```

```
}

void loop(){

temperatura=5*100*analogRead(sensor);

Serial.println(temperatura);

if(temperature>=30){

digitalWrite(cooler,HIGH);}

else{

digitalWrite(cooler,LOW);}

}
```

Utilizando um ventilador

Utilizando um ventilador, iremos neste projeto controlar a temperatura ambiente. Este é mais um pequeno projeto de automação muito útil para nosso dia a dia.

Componentes necessários

- *Protoboard*
- LM35
- Ventilador
- Módulo relé
- *Plug para tomada*
- *Fios*

Montagem na protoboard

Primeiro faça a seguinte montagem:

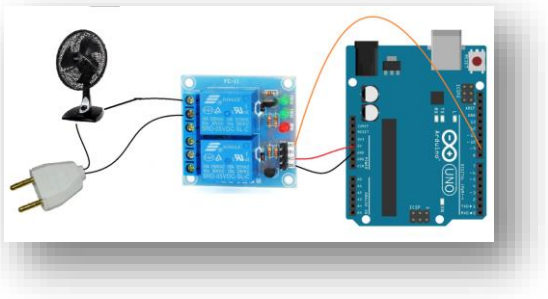


Figura 59: Montagem do circuito (parte 1)

Depois faça a montagem na protoboard do LM35 igual ao projeto anterior.

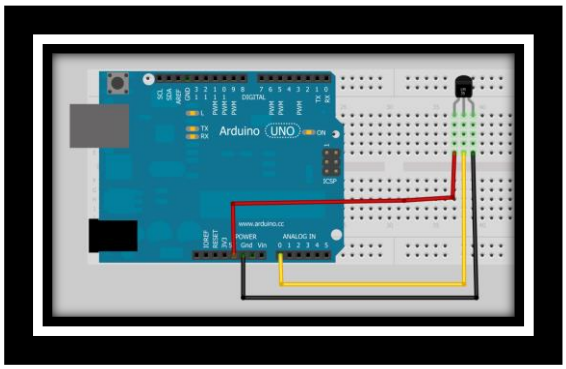


Figura 60: lm35 na protoboard

O programa

O programa é exatamente o mesmo que o anterior:

```
int vent= 3;

int sensor = A0;

int temperatura;

void setup(){

pinMode(vent,OUTPUT);

Serial.begin(9600);

}

void loop(){

temperatura=5*100*analogRead(sensor);

Serial.println(temperatura);

if(temperature>=30){

digitalWrite(vent,HIGH);}

else{

digitalWrite(vent,LOW);}

}
```

Aula 5 - LCD

Escrevendo no LCD

Este projeto utiliza o display LCD 16x2 para escrever mensagens.

Componentes necessários

- Protoboard
- Display LCD 16 colunas e 2 linhas
- Potenciômetro de 10k ohms
- Jumpers

Estudo dos componentes

Um **LCD** (*liquid crystal display)* é um líquido polarizador de luz, eletricamente controlado, que se encontra comprimido dentro de celas entre duas lâminas transparentes polarizadoras. Os eixos polarizadores das duas lâminas estão alinhados perpendicularmente entre si. Cada cela é provida de contatos elétricos que possibilitam um campo elétrico ser aplicado ao líquido no interior.

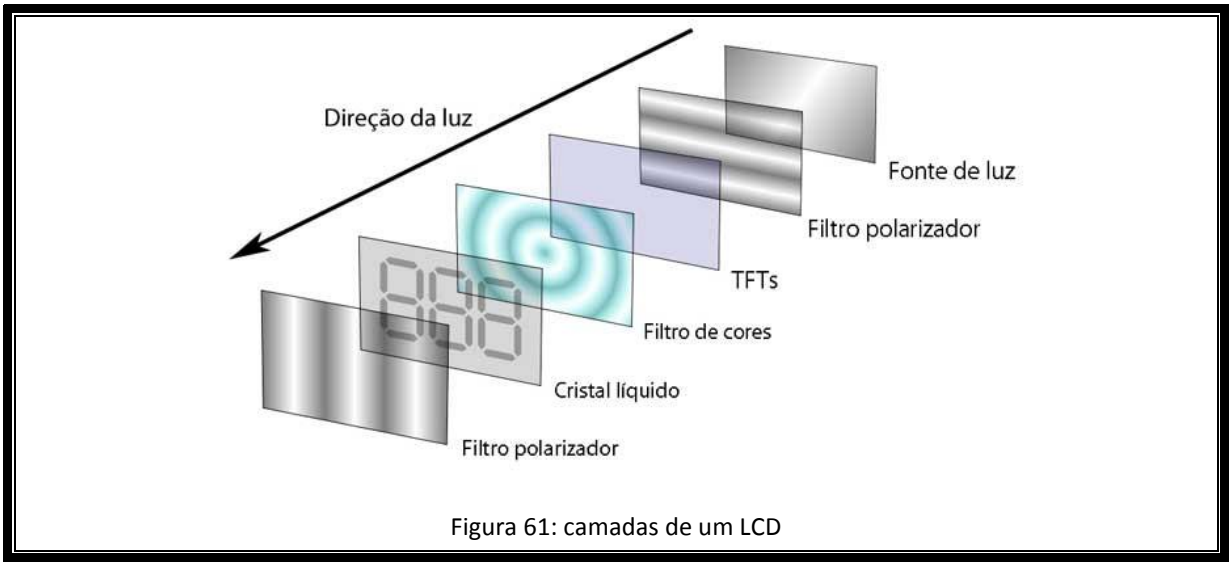


Figura 61: camadas de um LCD

Ao receber uma corrente, um LCD tem sua estrutura molecular alterada. Em seu estado normal as substâncias são transparentes, mas ao receberem uma carga elétrica tornam-se opacas, impedindo a passagem da luz.

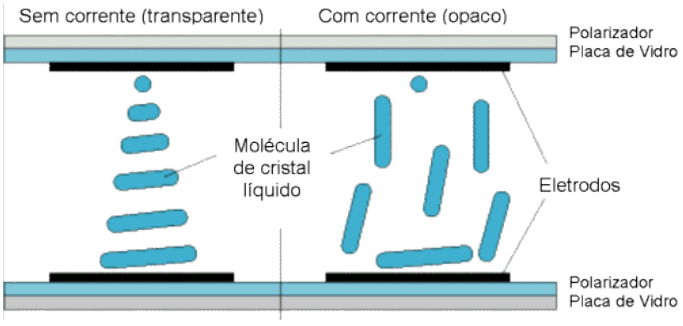


Figura 62: estruturação das moléculas dentro de um LCD

Montagem na protoboard

1. Potenciômetro para controlar contraste de luz

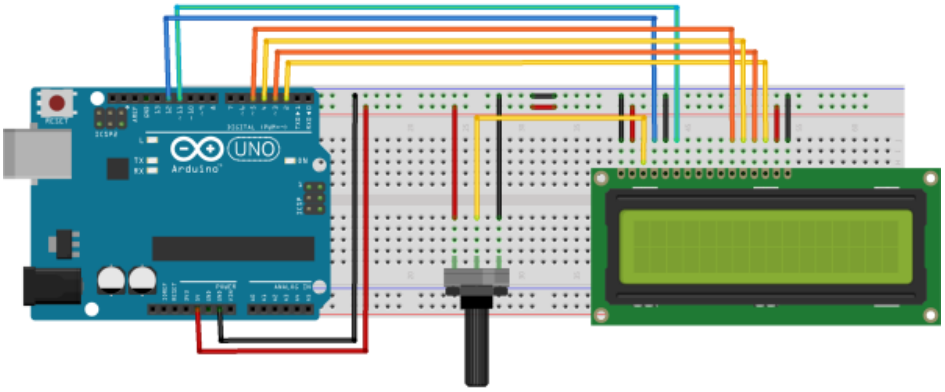


Figura 63: LCD com um potenciômetro

2. Potenciômetros para controlar contraste e luminosidade de fundo

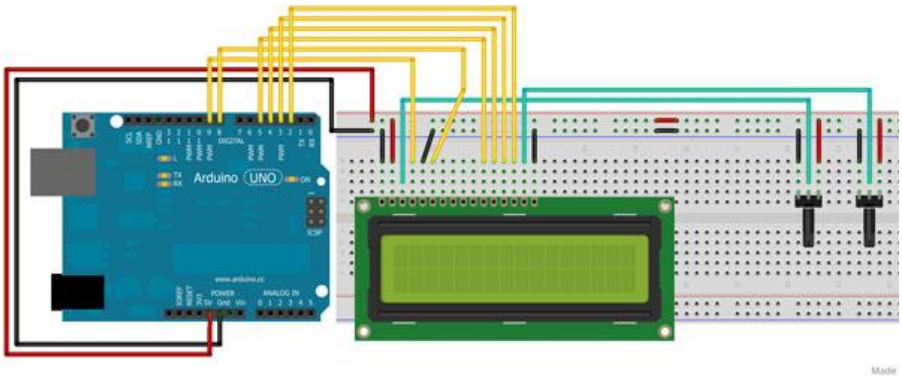


Figura 64: LCD com 2 potenciômetros

3. Sem a utilização de potenciômetro

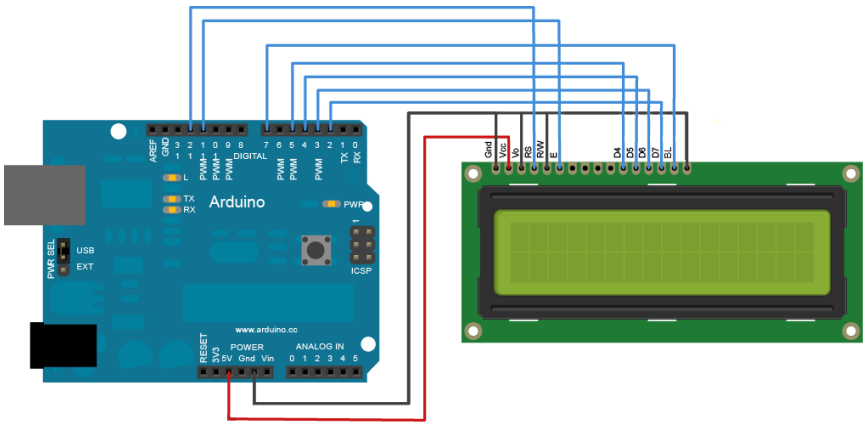


Figura 65: LCD sem controle de luminosidade e ou contraste

Em todas as opções acima o LCD funciona perfeitamente, mas caso queira regular a luz do display prefira as montagens com o potenciômetro. Abaixo mostra o datasheet, detalhando a funcionalidade de cada pino do display. Os pinos são numerados da direita para a esquerda.

PIN No	Name	Function
1	VSS	Ground voltage
2	VCC	+5V
3	VEE	Contrast voltage
4	RS	Register Select 0 = Instruction Register 1 = Data Register
5	R/W	Read/ Write, to choose write or read mode 0 = write mode 1 = read mode
6	E	Enable 0 = start to latch data to LCD character 1= disable
7	DB0	Data bit 0 (LSB)
8	DB1	Data bit 1
9	DB2	Data bit 2
10	DB3	Data bit 3
11	DB4	Data bit 4
12	DB5	Data bit 5
13	DB6	Data bit 6
14	DB7	Data bit 7 (MSB)
15	BPL	Back Plane Light +5V or lower (Optional)
16	GND	Ground voltage (Optional)

Tabela: Datasheet LCD

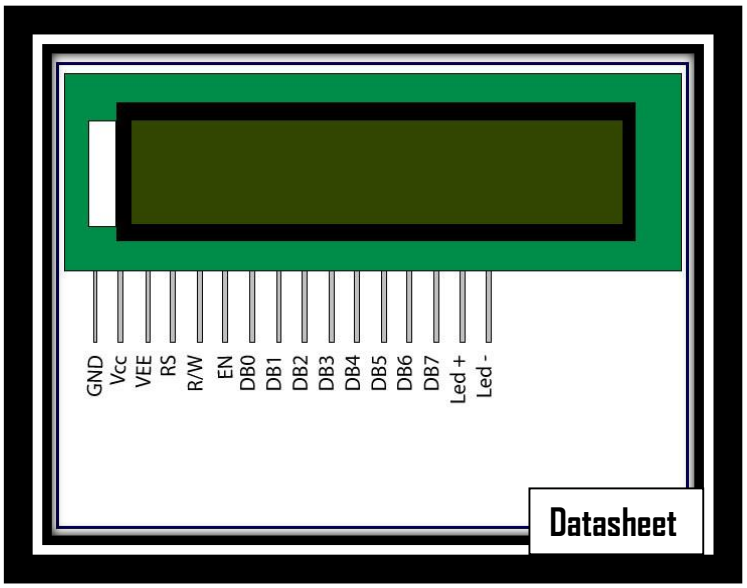


Figura 66: LCD esquemático

O programa

Por incrível que pareça, a programação de um display LCD não é nada complicada, mas isso só ocorre pelo fato de existir uma biblioteca específica para esta função.

A biblioteca se chama “LiquidCrystal.h” e para incluí-la em um programa deve-se escrever “#include <LiquidCrystal.h>”.

```
//Define a utilização da biblioteca para controle de telas LDCs.
 #include <LiquidCrystal.h>

//Criando um objeto da classe LiquidCrystal e
//inicializando com os pinos da interface.
LiquidCrystal lcd(9, 8, 5, 4, 3, 2);

void setup() {
```

```
  //Inicializando o LCD e informando o tamanho de 16 colunas e 2 linhas
  //que é o tamanho do LCD JHD 162A usado neste projeto.
  lcd.begin(16, 2);
}

void loop() {
  lcd.clear();        //limpa o display do LCD.
  lcd.print("Oi ");  //imprime a string no display do LCD.
  delay(2000);

  lcd.setCursor(0,1);   //posiciona cursor na coluna 0 linha 1 do LCD
  lcd.print("Tudo Bem???"); //imprime a string no display do LCD.
  delay(2000);

  lcd.clear();
  lcd.print("Quer aprender");
  lcd.setCursor(0,1);
  lcd.print("eletrônica?");
  delay(4000);

  lcd.clear();
  lcd.print("Vem com o hack.ATOMIC que é sucesso");
  delay(1000);

  //Rolando o display para a esquerda 18 vezes
  for (int i = 0; i <18; i++) {
    lcd.scrollDisplayLeft();
    delay(600);
  }

  delay(1000);
}
```

Relógio com hora e data

Este projeto consiste em fazer um relógio utilizando o LCD.

Materiais necessários

- Arduino
- Protoboard
- Potenciômetro
- LCD 16x2
- Jumpers

Montagem na protoboard

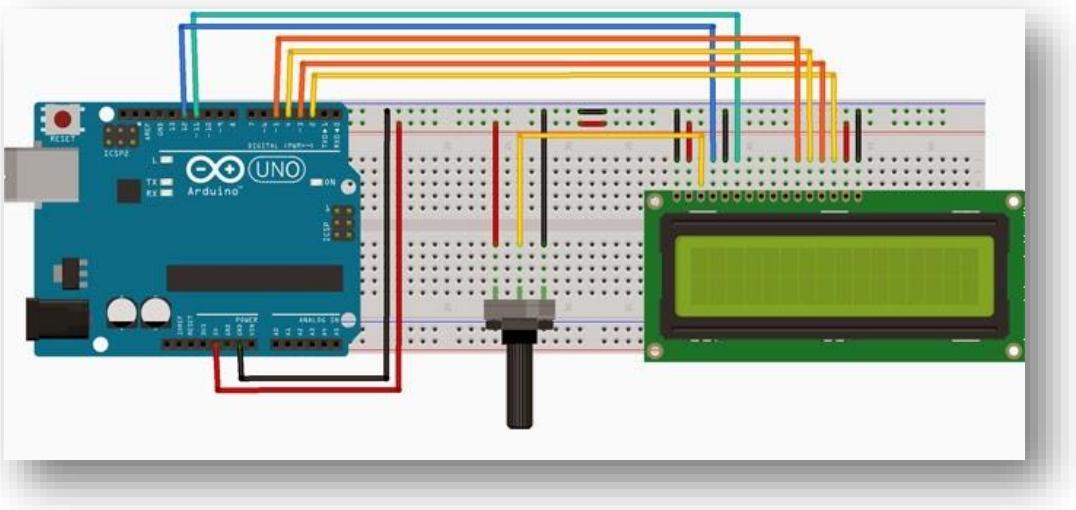


Figura 67: montagem do LCD na protoboard

A programação

```
#include <LiquidCrystal.h> //Inclui a biblioteca do LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int segundo,minuto, hora, dia, mes,ano;
unsigned long UtlTime;
void setup(){
UtlTime=0;
minuto=0;
hora=0;
dia=0;
mes=0;
ano=0;
Serial.begin(9600);
lcd.begin(16, 2);
lcd.setCursor(0,0);
lcd.print(" Data e hora ");
lcd.setCursor(0,1);
lcd.print(" com Arduino");
delay (2000);
//Configura o minuto
```

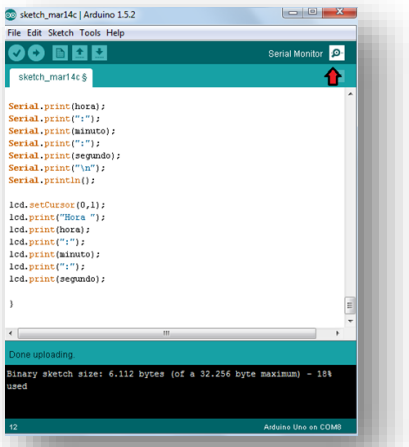
```
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Minuto: ");
Serial.print("\nEntre Minuto:");
while(minuto==0)  {
if (Serial.available() > 0)   {
minuto= Serial.parseInt();   } }
lcd.print(minuto);
delay(1000);
//Configura a hora
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Hora: ");
Serial.print("\nEntre Hora:");
while(hora==0)  {
if (Serial.available() > 0)   {
hora= Serial.parseInt();   } }
lcd.print(hora);
delay(1000);
//Configura o Dia
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Dia: ");
Serial.print("\nEntre Dia:");
while(dia==0)  {
if (Serial.available() > 0)   {
dia= Serial.parseInt();   } }
lcd.print(dia);
delay(1000);
//Configura o mês
lcd.clear();
lcd.setCursor(0,0);
```

```
lcd.print("Mes: ");
Serial.print("\nEntre Mes:");
while(mes==0)  {
if (Serial.available() > 0)    {
mes= Serial.parseInt();    } }
lcd.print(mes);
delay(1000);
//Configura o Ano
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Ano: ");
Serial.print("\nEntre ano:");
while(ano==0)  {
if (Serial.available() > 0)    {
ano= Serial.parseInt();    } }
lcd.print(ano);
delay(1000);
lcd.clear();}
void loop(){
if(millis()-UtlTime<0)  {
UtlTime=millis();  }
else {
segundo=int((millis()-UtlTime)/1000);  }
if(segundo>59)  {
segundo=0;
minuto++;
UtlTime=millis();
if(minuto>59)    {
hora++;
minuto=0;
if(hora>23)     {
dia++;
```

```
hora=0;
if(mes==1||mes==3||mes==5||mes==7||mes==8||mes==10||mes==12)      {
if(dia>31)       {
dia=1;
mes++;
if(mes>12)         {
ano++;
mes=1;         }       }      }
else if(mes==2)      {
if(ano%400==0)       {
if(dia>29)         {
dia=1;
mes++;         }       }
else if((ano%4==0)&&(ano%100!=0))       {
if(dia>29)         {
dia=1;
mes++;         }       }
else       {
if(dia>28)         {
dia=1;
mes++;         }       }      }
else      {
if(dia>30)       {
dia=1;
mes++;       }      }     }   } }
Serial.print(dia);
Serial.print("/");
Serial.print(mes);
Serial.print("/");
Serial.print(ano);
Serial.println();
lcd.setCursor(0,0);
```

```
lcd.print("Data ");
lcd.print(dia);
lcd.print("/");
lcd.print(mes);
lcd.print("/");
lcd.print(ano);
Serial.print(hora);
Serial.print(":");
Serial.print(minuto);
Serial.print(":");
Serial.print(segundo);
Serial.print("\n");
Serial.println();
lcd.setCursor(0,1);
lcd.print("Hora ");
lcd.print(hora);
lcd.print(":");
lcd.print(minuto);
lcd.print(":");
lcd.print(segundo);}
```

Depois da programação feita, clique no serial monitor para configurar seu relógio:



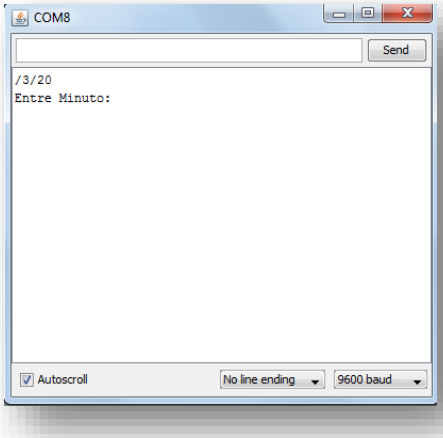


Figura 68: Configuração pelo serial monitor

Finalização:

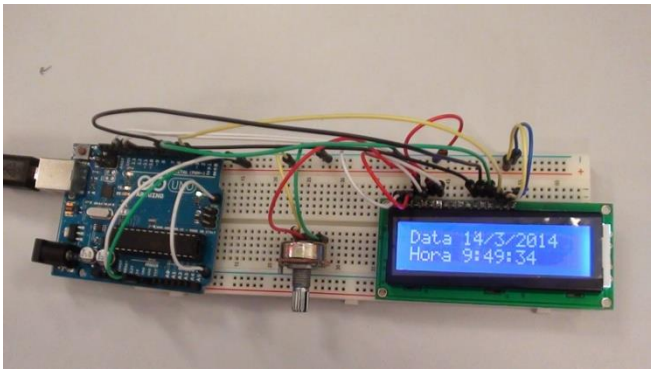


Figura 69: finalização

Aula 6 - Servo Motores

Comandando um servo

O projeto consiste em controlar um servo de rotação 180 graus.

Componentes necessários

- Micro servo
- Jumpers
- Arduino

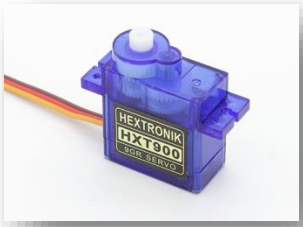


Figura 70: servo motor

Estudo dos componentes

Servos são motores de posição muito utilizados em aeromodelos, carrinhos, robôs e também em projetos de automação. As posições de um servo são determinadas em ângulo e, geralmente, o motor tem a limitação de 180 graus de rotação.

São precisos quanto à posição por possuírem um sistema constituído de três componentes básicos:

- Sistema atuador: Motor elétrico de corrente contínua e conjunto de engrenagens;
- Sensor: é um potenciômetro capaz de medir a posição angular com base no valor da sua resistência elétrica;
- Circuito de controle: oscilador e controlador PID.

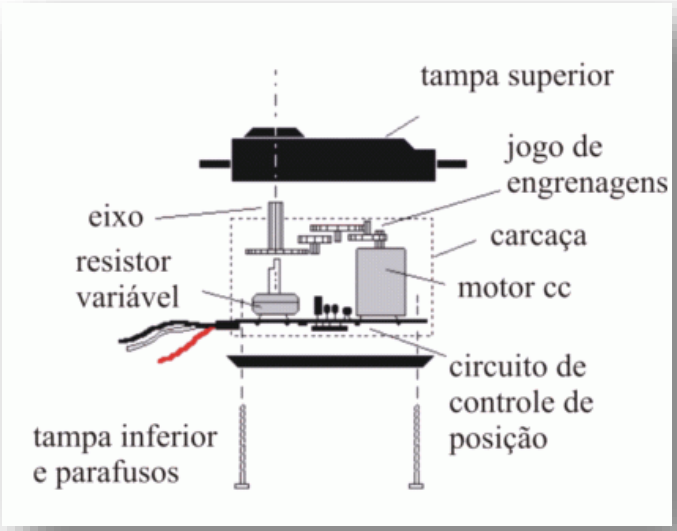


Figura 71: interior de um servo

Servos possuem três fios: dois para alimentação e um para sinal de controle, o qual utiliza o protocolo modulação por largura de pulso (PWM).

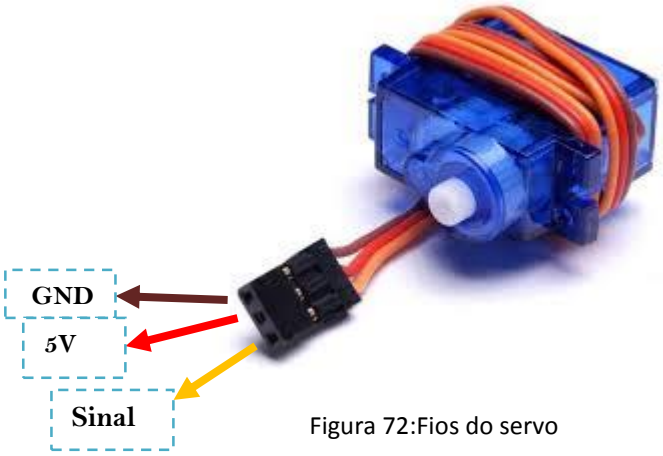
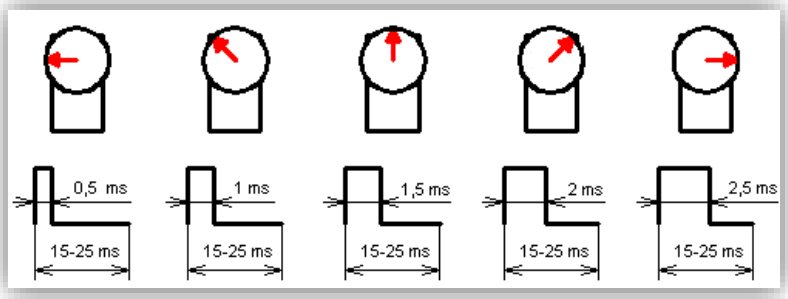
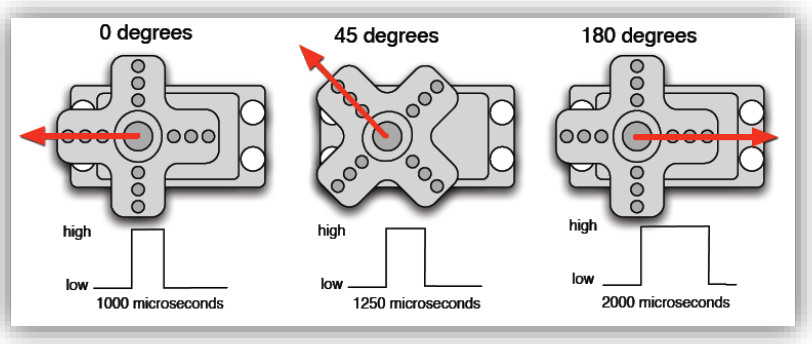


Figura 72:Fios do servo

PWM – características: largura máxima (+90º da posição central), largura mínima (-90º da posição central) e taxa de repetição (frequência).





Gráficos 3 e 4: PWM

Montagem

Colocar o servo ligado a uma porta digital PWM, uma das portas indicadas abaixo:

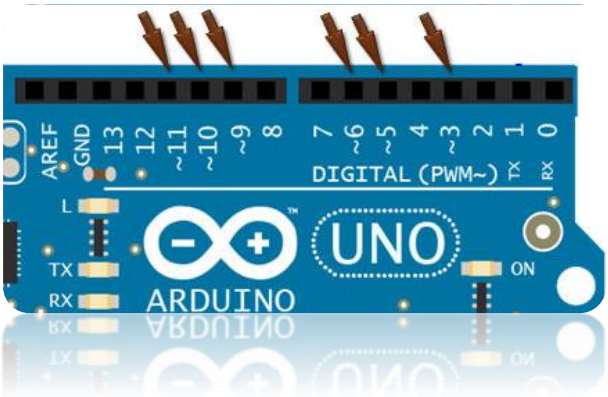


Figura 73: Portas digitais PWM

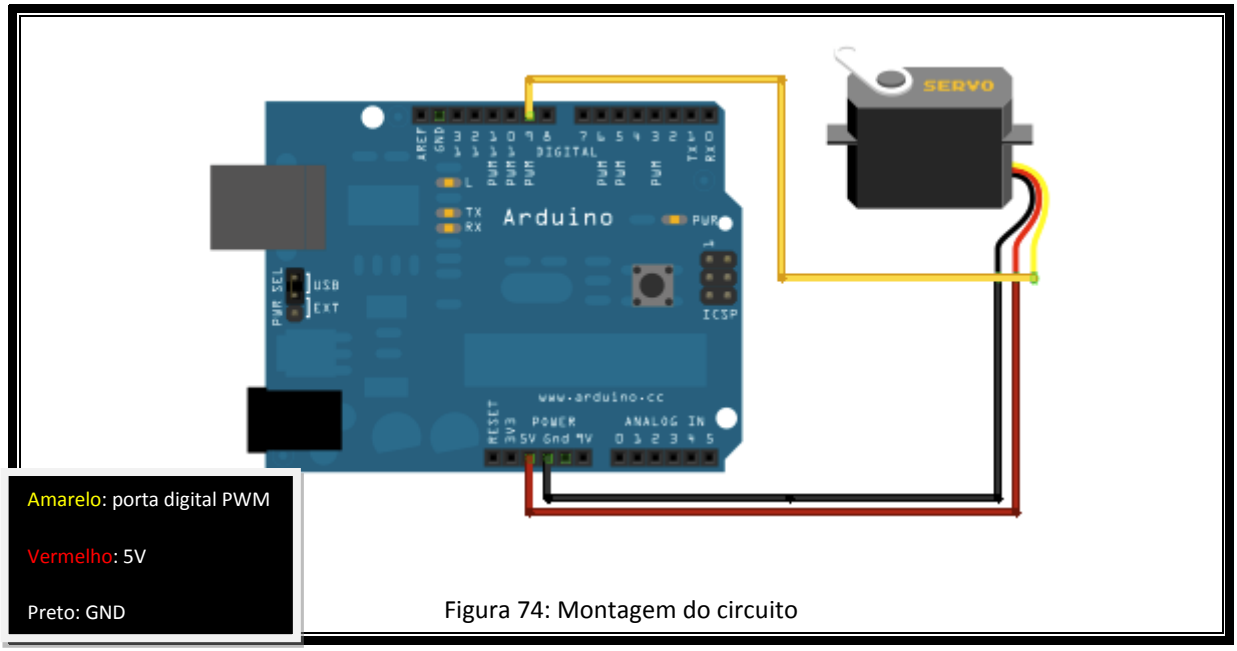


Figura 74: Montagem do circuito

O programa

Para programar o servo motor utilizamos uma biblioteca chamada “Servo.h”.

```
#include <Servo.h>//inclui a biblioteca

Servo motor ;

void setup(){

motor.attach(9);  //declara o número da porta digital PWM utilizada

}

void loop(){

motor.write(45); //posição inicial : 45 graus

delay(1000); //fica parado na posição 45 graus durante 1 segundo

motor.write(125); //depois do 1 s, passa para a posição 125 graus

delay(1000);//parado na posição 125 graus durante 1 segundo}
```

É importante destacar que ao executar este programa não estamos controlando a velocidade com que o servo gira, mas apenas o tempo que ficará em determinada posição. O servo passa de um ângulo para o outro em cerca de 1milissegundo o que é extremamente rápido.

Caso você queira controlar a velocidade com que o servo gira (optando por velocidades mais amenas), execute o seguinte programa:

```
#include<Servo.h>

int i; // variável para contador

Servo motor;

void setup(){

 motor.attach(9);

}

void loop(){

 for(i=45;i<=125;i++){ //o “i” recebe um valor maior a cada vez que o semi loop recomeça

  motor.write(i); //o primeiro valor de i é 45, depois 46, depois 47... e assim até 125

  delay(100); // para velocidade mais amenas coloque um delay maior

 }

 for(i=125;i>=45;i--){ // o i recebe um valor menor a cada vez que o semi loop recomeça

  motor.write(i); // o primeiro valor de I é 125, depois 124, 123... até 45
```

```
  delay(100); // para velocidades mais amenas coloque um delay maior
 }
 delay(100); //para durante 100 milissegundos
}
```

Braço Robótico

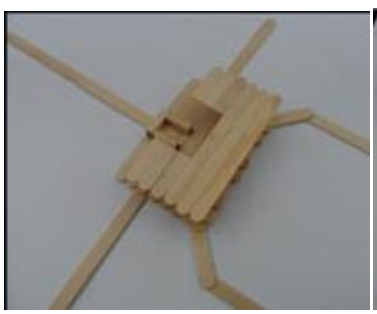
Este é o primeiro projeto de robótica. A idéia é fazer um braço robótico utilizando palitos de picolé.

Componentes necessários

- Protoboard
- Cerca de 200 palitos de picolé
- Super cola com secagem rápida
- 3 servomotores
- 3 potenciômetros
- Jumpers

Montagem da estrutura

A estrutura do braço robótico foi feita com palitos de picolé. A montagem pode ser dividida em três etapas: base, articulações e garra.



Figuras 75, 76 e 77: etapas de montagem

Depois que juntar as três partes, a estrutura ficará assim:

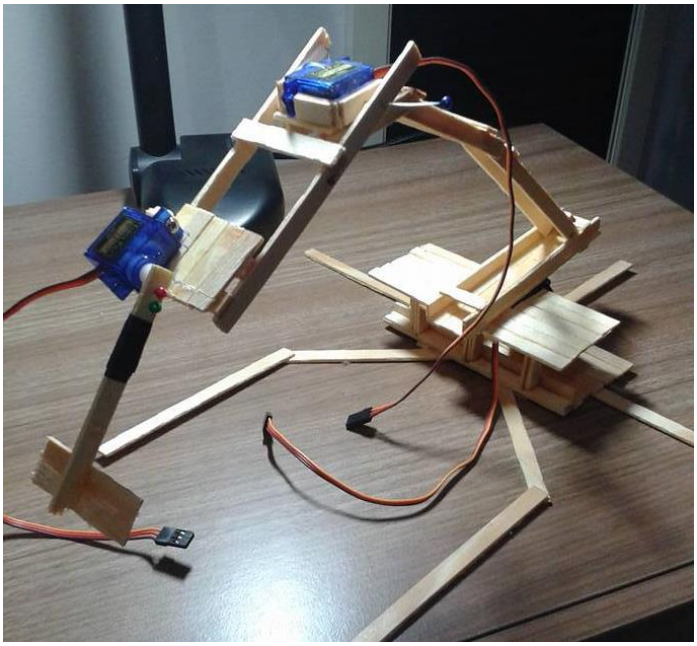


Figura 78: finalização

Montagem na protoboard

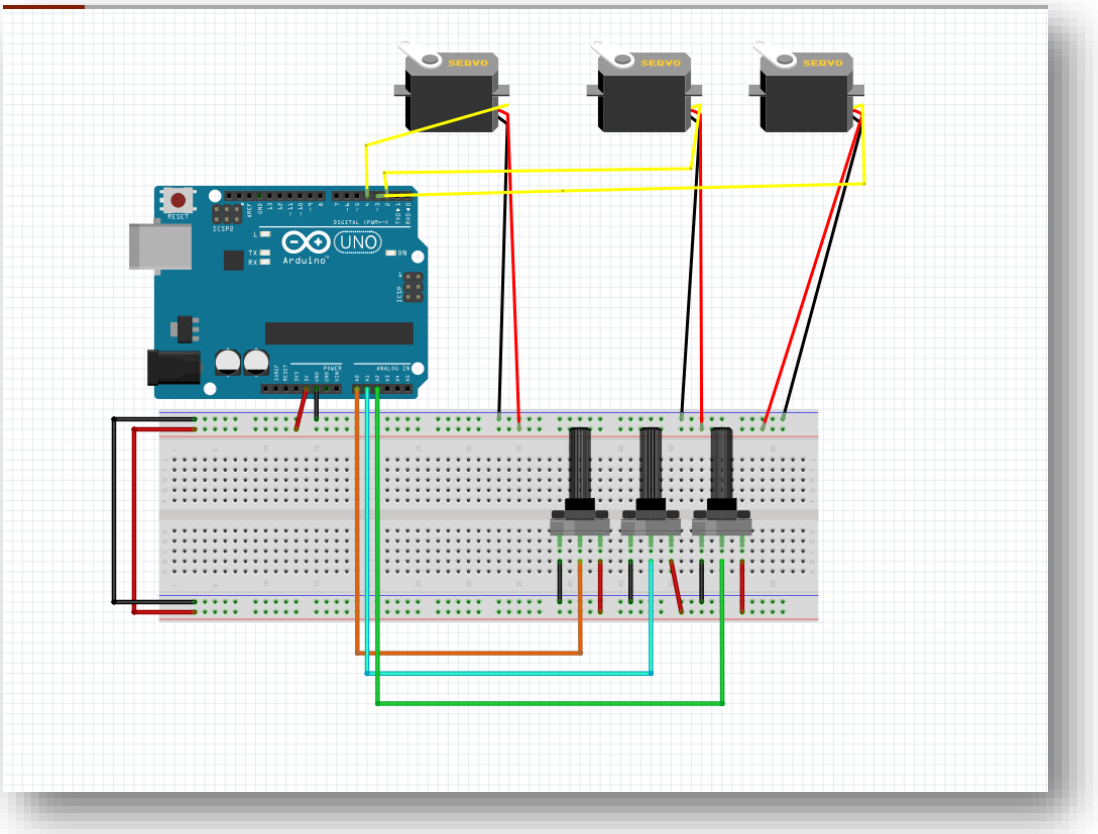


Figura 79: Montagem dos motores na protoboard

A programação

```
#include "Servo.h" //biblioteca para controle do servomotor

//Criando objeto da classe Servo
Servo servoMotor1Obj;  //para controlar o servo 1
Servo servoMotor2Obj;  //para controlar o servo 2
Servo servoMotor3Obj;  //para controlar o servo 3

//pinos analógicos dos potenciômetros
int const potenciometro1Pin =A0; //potenciômetro 1
int const potenciometro2Pin = A1; //potenciômetro 2
int const potenciometro3Pin = A2; //potenciômetro 3

//pino digital associado ao servomotor
int const servoMotor1Pin = 4; //controle do servomotor 1
int const servoMotor2Pin = 3; //controle do servomotor 2
int const servoMotor3Pin = 2; //controle do servomotor 3

//variáveis usadas para armazenar o valor lido nos potenciômetros
int valPotenciometro1; //potenciômetro 1
int valPotenciometro2; //potenciômetro 2
int valPotenciometro3; //potenciômetro 3

//variáveis para armazenar os valores em graus dos servomotores
int valServo1; //servomotor 1
int valServo2; //servomotor 2
int valServo3; //servomotor 3

void setup() {
  //associando o pino digital ao objeto da classe Servo
  servoMotor1Obj.attach(servoMotor1Pin); //Servo 1
  servoMotor2Obj.attach(servoMotor2Pin); //Servo 2
  servoMotor3Obj.attach(servoMotor3Pin); //Servo 3  }

void loop() {
  //lendo os valores dos potenciômetros
  //o (intervalo do potenciômetro é entre 0 e 1023)
  valPotenciometro1 = analogRead(potenciometro1Pin);
  valPotenciometro2 = analogRead(potenciometro2Pin);
  valPotenciometro3 = analogRead(potenciometro3Pin);

  //mapeando os valores dos potenciômetros para a escala
  //do servo (intervalo entre 5 e 175 graus)
  valServo1 = map(valPotenciometro1, 0, 1023, 5, 175);
  valServo2 = map(valPotenciometro2, 0, 1023, 5, 175);
  valServo3 = map(valPotenciometro3, 0, 1023, 5, 175);

  //definindo o valor/posição dos servomotores em graus
  servoMotor1Obj.write(valServo1);
  servoMotor2Obj.write(valServo2);
  servoMotor3Obj.write(valServo3);

  delay(15);   }
```

Aula 7 - Motor de passo

Controlando motor de passo

Neste projeto será explicado como conectar um Motor de Passo 28BYJ-48 5v ao Arduino utilizando um driver ULN2003.

Materiais necessários

- Arduino
- Motor de passo
- Driver
- Jumpers

Estudo dos materiais

O **motor de passo** é um dispositivo eletromecânico capaz de converter impulsos elétricos em movimentos mecânicos. É um tipo de motor elétrico utilizado para projetos que necessitam de alta precisão. O eixo do motor gira em incrementos discretos quando os impulsos de comando são aplicados na correta sequência.

Possui um número de pólos magnéticos que determinam o número de passos por revolução. É classificado pelo torque que produz e para atingir todo o seu torque suas bobinas devem receber toda a corrente marcada durante o passo.

A rotação do motor de passo tem relação direta com os pulsos de entrada aplicados. Para controlá-lo precisa-se aplicar tensão a cada uma das bobinas em uma sequência específica: a sequência dos impulsos aplicados é relacionada com a direção de rotação do eixo do motor, a velocidade de rotação com a freqüência de impulsos de entrada e o comprimento de rotação com o número de impulsos de entrada.

A posição do motor é conhecida simplesmente através do controle de impulsos de entrada, o que é uma grande vantagem, sendo dispensáveis os caros sistemas de detecção (por exemplo, codificadores ópticos).

Figura 80: motor de passo

Conectando o motor ao Arduino

Para fazer uma simples interface com motor de passo, utilizaremos o chip ULN2003A (presente no driver ULN2003). O CI suporta até 500 mA por canal e tem queda de tensão interna de cerca de 1V quando ligado.

Esquema de ligação das bobinas e cores dos fios do motor de passo:

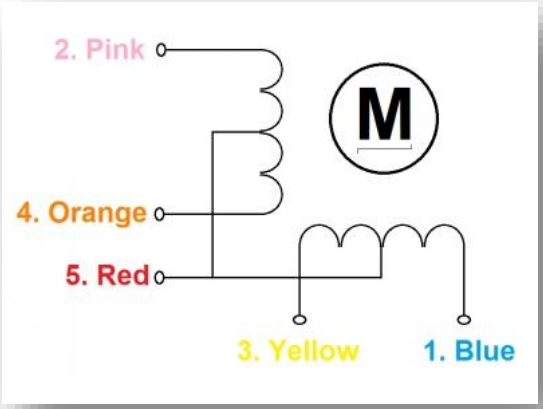


Figura 81: Ligação das bobinas

Tensão nominal: 5VDC
Número de Fase 4
Variação Velocidade Rácio 1/64
Stride Ângulo 5,625 ° / 64
100Hz freqüência
DC resistência 50Ω ± 7% (25 ℃)
Ocioso In- tração Frequency > 600Hz
Ocioso Out- tração Frequency > 1000Hz
In- tração Torque> 34.3mN.m (120Hz)
Auto- posicionamento Torque> 34.3mN.m
Atrito de torque 600-1200 gf.cm
Puxe no torque de 300 gf.cm
Resistência Duplas > 10MΩ (500V)
Duplas poder 600VAC electricidade / 1 mA / 1s
Isolamento classe A
Aumento da temperatura < 40K (120Hz)
Ruído < 35dB (120Hz , Sem carga , 10 centímetros)
Modelo 28BYJ -48 - 5V

Conecte o motor ao Arduino na seguinte sequência:

- IN1 ao pino 8
- IN2 ao pino 10
- IN3 ao pino 9
- IN4 ao pino 11

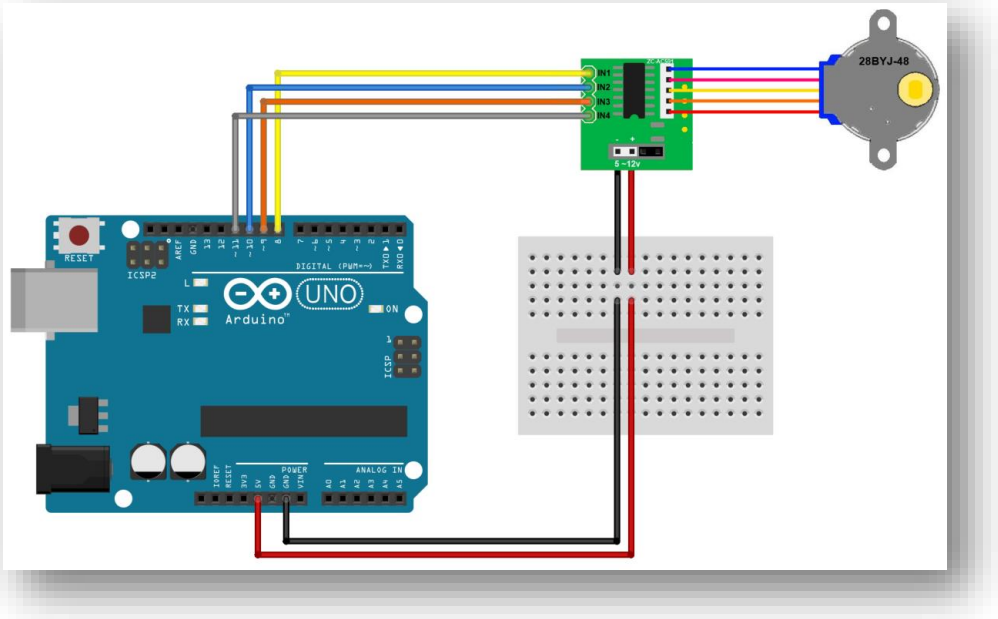


Figura 82: como ligar o motor de passo ao Arduino

A programação

A biblioteca que será utilizada é chamada *Stepper.* O programa abaixo fará com que o motor gire no sentido horário 4 vezes em ângulos de 90 graus, depois 3 vezes no anti-horário, e por último vai girar o motor e aumentar a velocidade gradativamente. E ao final um delay de 2 segundos para recomeçar o loop.

```
#include <Stepper.h>

const int stepsPerRevolution = 500;

//Inicializa a biblioteca utilizando as portas de 8 a 11 para
//ligacao ao motor
Stepper myStepper(stepsPerRevolution, 8,10,9,11);

void setup() {
 //Determina a velocidade inicial do motor
 myStepper.setSpeed(60);}

void loop() {
 //Gira o motor no sentido horario a 90 graus
 for (int i = 0; i<=3; i++) {
 myStepper.step(-512);
 delay(2000); }

 //Gira o motor no sentido anti-horario a 120 graus
 for (int i = 0; i<=2; i++) {
 myStepper.step(682);
 delay(2000); }

 //Gira o motor no sentido horario, aumentando a
 //velocidade gradativamente
 for (int i = 10; i<=60; i=i+10) {
```

```
myStepper.setSpeed(i);
myStepper.step(40*i); }
delay(2000); }
```

Aula 8 – Ultrassônico

Sensor de Distância

Neste projeto será ensinado como detectar a distância de obstáculos utilizando um sensor ultrassônico.

Materiais necessários

- Arduino
- Protoboard
- Sensor ultrassônico
- 3 LEDs
- 3 resistores 220 ohms
- Jumpers

Estudo dos materiais

O **sensor Ultrassônico** é capaz de detectar a distância de obstáculos que estão a sua frente. Muitas vezes usado em robôs ou carrinhos para que caminhem sozinhos evitando possíveis batidas.

Caracterizado por operar um tipo de radiação não sujeita a interferência eletromagnética e totalmente limpa, o que é muito importante para certas aplicações.

O princípio de operação é exatamente o mesmo do sonar (utilizado por morcegos para detectar objetos e presas em seu cego vôo).

O pequeno comprimento de onda das vibrações ultrassônicas permite a detecção de até mesmo pequenos objetos, podendo ser captadas por um sensor.

Figura 83: Diferença entre ondas de baixa e alta freqüência

O comprimento de onda usado e, por sua vez, a freqüência são extremamente importantes neste tipo de sensor, pelo fato deles determinarem as dimensões mínimas do objeto a ser detectado. Um sinal de 1000 HZ (34 cm de comprimento de onda), portanto, tem a capacidade de detectar no mínimo objetos de dimensão de 34 cm, considerando a velocidade 340 m/s (velocidade do som).

Basicamente, o sensor é formado por um emissor e um receptor, tanto fixados num mesmo conjunto como separados, dependendo do posicionamento:

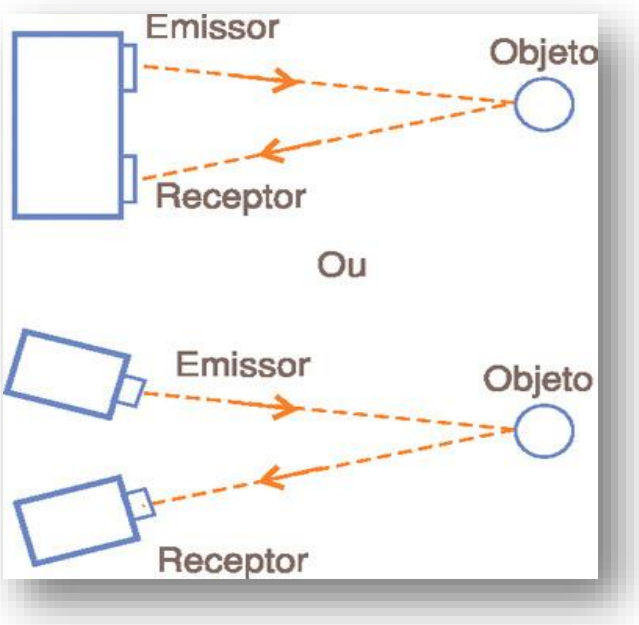


Figura 84: receptor e emissor de sensores ulrassonicos

Montagem na protoboard

Utilizaremos na montagem o sensor HC-SCO4

É capaz de datectar objetos entre 2 cm e 4 metros de distância;
Sua operação não é afetada pela luz solar;
Revestimentos acústicos podem atrapalhar a operação do sensor HC-SRO4;
Possui o transmissor e o

Figura 85: sensor ultrassom

Pinos:

VCC - 5V (podendo variar de 4,5V até 5,5V)
Trig - Entrada do sensor (gatilho)
Echo - Saída do sensor (Eco)
GND - Ground (terra)

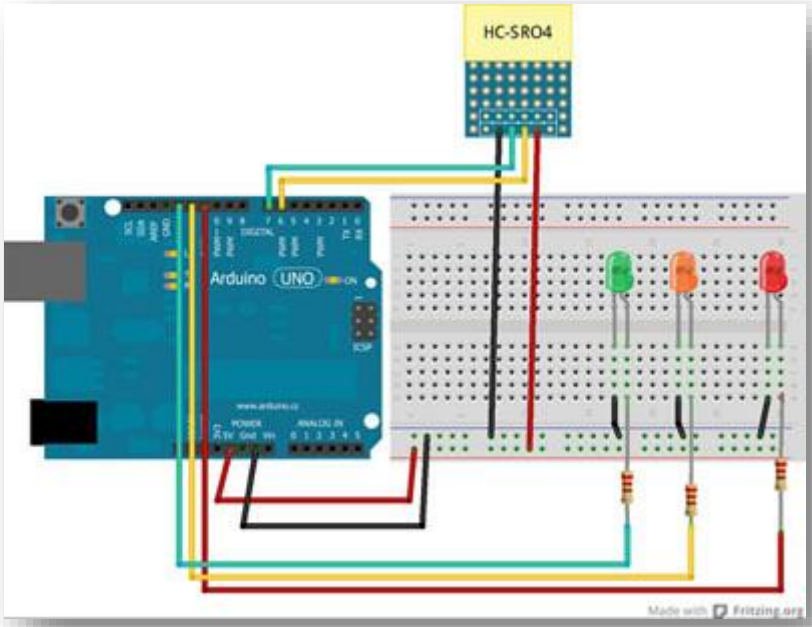


Figura 86: Montagem do sensor na protoboard

A programação

```
//Incluindo biblioteca Ultrasonic.h

#include "Ultrasonic.h"
```

```
//criando objeto ultrasonic e definindo as portas digitais
//do Trigger - 6 - e Echo - 7
Ultrasonic ultrasonic(6,7);


//Declaração das constantes referentes aos pinos digitais.
const int ledVerde = 13;
const int ledAmarelo = 12;
const int ledVermelho = 11;
 long microsec = 0;
float distanciaCM = 0;


void setup() {
  Serial.begin(9600); //Inicializando o serial monitor


  //Definindo pinos digitais
  pinMode(ledVerde,OUTPUT); //13 como de saída.
  pinMode(ledAmarelo,OUTPUT); //12 como de saída.
  pinMode(ledVermelho,OUTPUT); //11 como de saída.}


void loop() {
  //Lendo o sensor
  microsec = ultrasonic.timing();


//Convertendo a distância em CM
  distanciaCM = ultrasonic.convert(microsec, Ultrasonic::CM);
   ledDistancia();


  Serial.print(distanciaCM);
  Serial.println(" cm");
  delay(1000);}


//Método que centraliza o controle de acendimento dos leds.
```

```
void ledDistancia() {


 //Apagando todos os leds
 digitalWrite(ledVerde,LOW);
 digitalWrite(ledAmarelo,LOW);
 digitalWrite(ledVermelho,LOW);


 //Acendendo o led adequado para a distância lida no sensor
 if (distanciaCM > 20) {
  digitalWrite(ledVerde,HIGH);   }
 if (distanciaCM <=20 and distanciaCM >= 10) {
  digitalWrite(ledAmarelo,HIGH);  }
 if (distanciaCM < 10) {
  digitalWrite(ledVermelho,HIGH);  }}
```

Carrinho autônomo

Neste projeto iremos fazer um robô capaz de desviar de obstáculos sem nenhum controle humano.

Materiais necessários:

- Arduino
- 2 Servos Motores (HK15138 Standard Analog Servo 38g / 4.3kg / 0.17s)
- Ferro de solda
- Solda
- Fita isolante
- Estilete
- Chave Philips
- Alicate
- Resistor 2,2k
- Protoboard
- Roda giratória (Rodízio giratório)
- Rodas
- Bateria 9V
- Jumpers

Hackeando o sevo motor

Os servos que usaremos no carrinho serão os motores das rodas do carrinho autônomo. Porém temos que hackeá-lo para que ele possa girar livremente e não até 360 graus (como vem de fábrica).

1-Retire a tampa superior do servo com o auxílio de uma chave Philips:

2- Memorize a ordem das engrenagens para que o processo de remontagem seja feito com sucesso;

3- Retire todas as engrenagens do conjunto, pegue a maior engrenagem a qual possui um bloqueio e corte-o;

Figura 87: engrenagem maior com uma trava

4-Retire a tampa inferior, a placa com o motor e o potenciómetro e corte os três fios do potenciômetro;

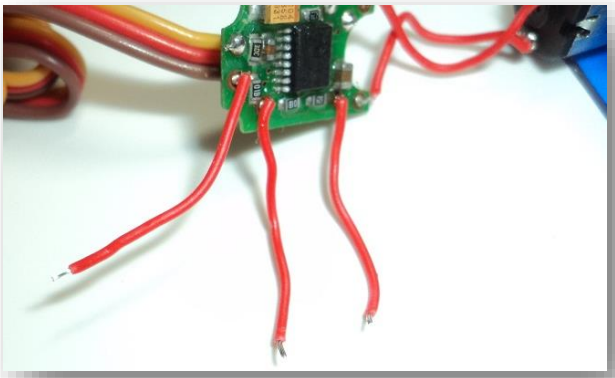


Figura 88: corte dos fios do potenciômetro

5- Retire a trava interna do potenciômetro com um alicate de corte;

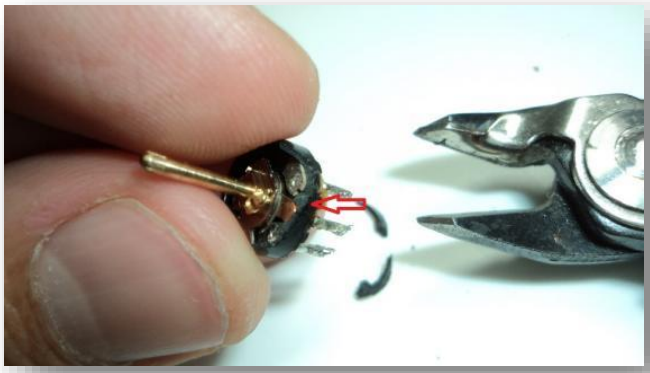


Figura 89: corte da trava do potenciômetro

6- Pegue 2 resistores de 2,2k , os solde nas extremidades dos fios que foram desprendidos e coloque o potenciômetro (agora sem sua trava de giro) no seu local inicial.

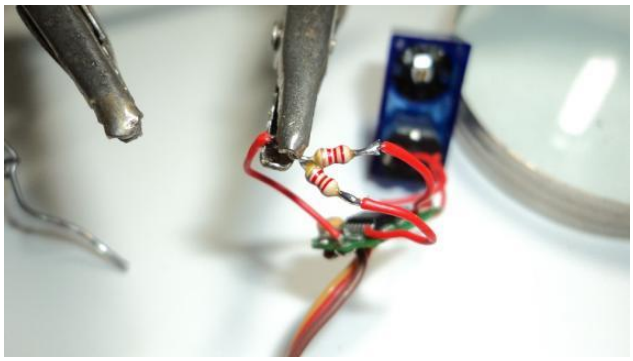


Figura 90: soldagem dos resistores nos fios

7- Use a fita isolante para evitar curto;

8-Remonte o servo.

Montagem na protoboard

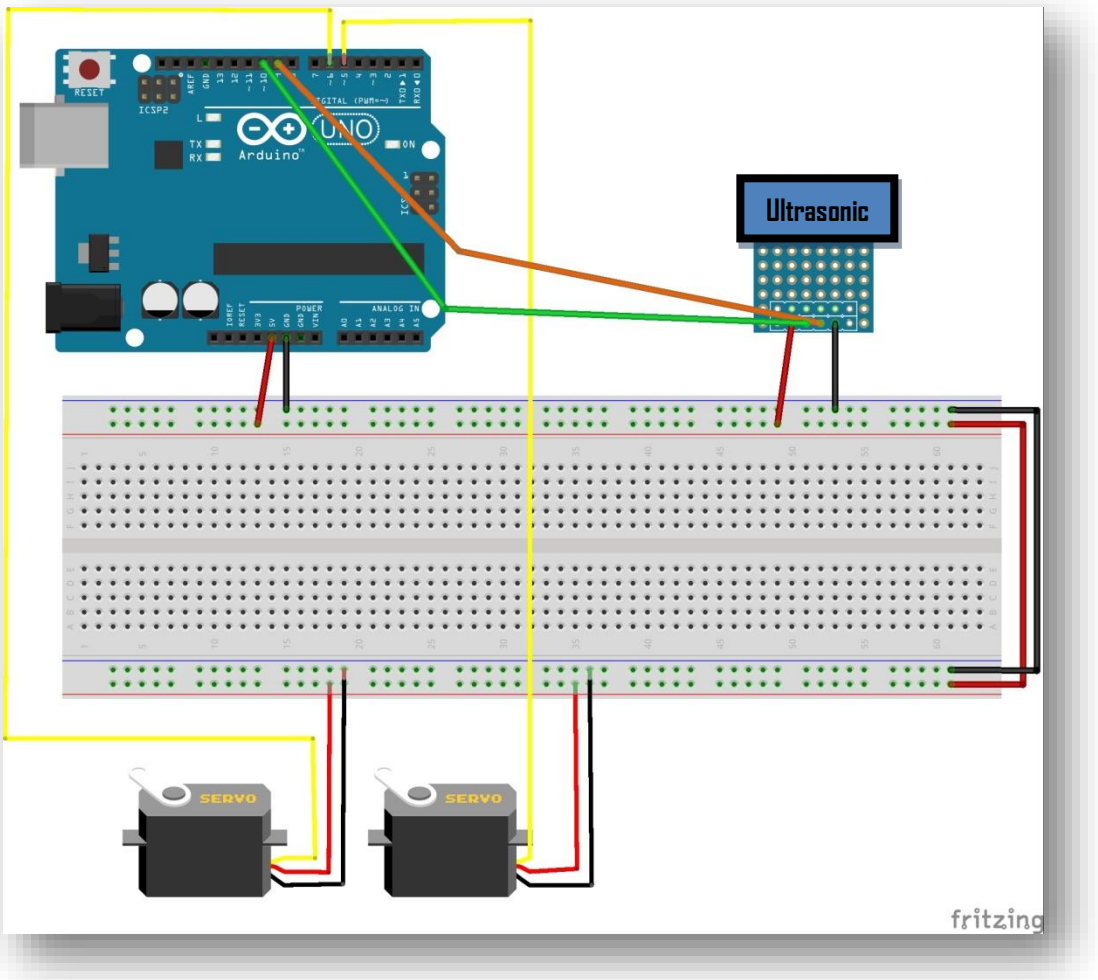


Figura 91: servos (das rodas) e ultrassônico na protoboard

Montagem da estrutura

Figura 92: base do carrinho

A programação

```
#include <Servo.h>

#include <Ultrasonic.h>//Inicia a biblioteca do ultrassom

Ultrasonic ultrasonic(6, 5); // Seta o pino triger como 6 e o echo como 5

Servo motorR; //Variavel servo direito

Servo motorL; //Variavel servo esquerdo

float DCm;  //Variavel distancia em CM

int x,y=0;  //Variaveis para tomada de decisao

void setup(){

 motorR.attach(8);

 motorL.attach(9);}

void loop(){

 DCm = ultrasonic.Ranging(CM); //Comando para calcular distancia em CM

 //Faz o carro ir para a frente

 while(DCm >= 10){

  motorR.write(-180);

  motorL.write(180);

  DCm = ultrasonic.Ranging(CM);  }

 x = random(1,3); // Escolhe um numero aleatoriamente dentro do range

 if(x==1){

  //Vira para a direita

  motorR.write(-180);

  motorL.write(-180);

  y = 0;

  delay(500); }

 if(x==2){

  //Vai pra direita

  motorR.write(180);

  motorL.write(180);

  y = 0;

  delay(500);  }}
```

Aula 9 - Comunicação sem fio

Módulo RF

Nesta aula iremos discorrer sobre como comunicar dois Arduinos distantes, como um conseguir mandar mensagens a uma distância razoável.

Materiais necessários

- 2 Arduinos
- 2 protoboards
- Módulo RF 433 MHz Transmissor
- Módulo RF 433 MHz Receptor
- push Button
- LED
- Resistor 220 e 470 ohms

Estudo dos materiais

Os **módulos RF** (transmissor e receptor) são componentes básicos para comunicação via rádio freqüência, presente em sistemas de alarmes, controle remoto, aquisição de dados e robótica em geral.

Os módulos alcançam até 200 metros sem obstáculos com modulação AM e freqüência 433 MHz.

Figura 93: Módulo RF transmissor

Transmissor

Alcance: 20-200 metros

Tensão de operação: 3,5 – 12V

Modo de operação: AM (Modulação em Amplitude)

Taxa de transferência: 4KB/s

Potência de transmissão: 10 mW

Freqüência de transmissão: 433 MHz

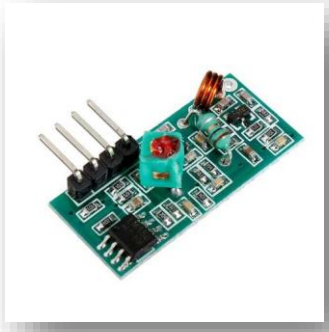


Figura 94: Módulo RF receptor

Receptor

Tensão de operação: 5V DC

Corrente de operação: 4mA

Freqüência de recepção: 433 MHz

Sensibilidade: 105 dB

Montagem na protoboard

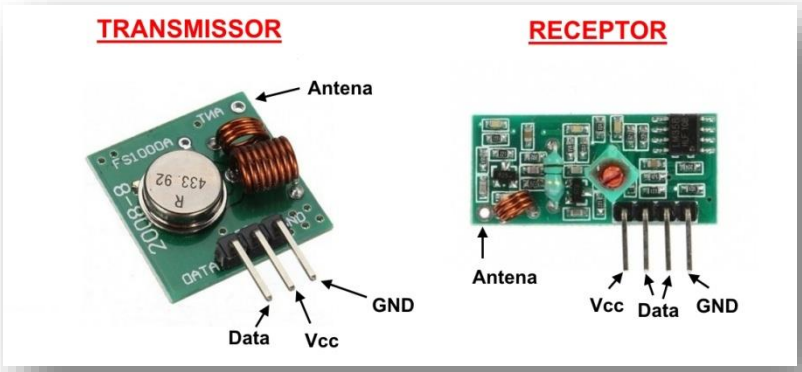


Figura 95: Finalidade dos terminais dos módulos RF

Faça as seguintes montagens na protoboard:

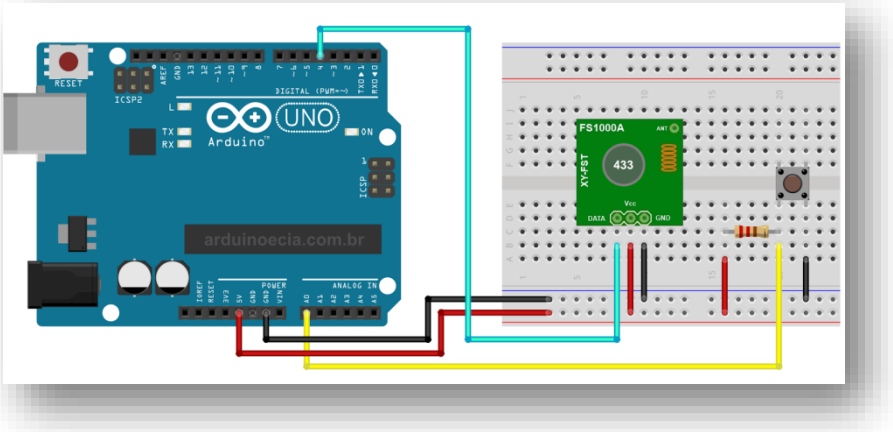


Figura 96: circuito transmissor

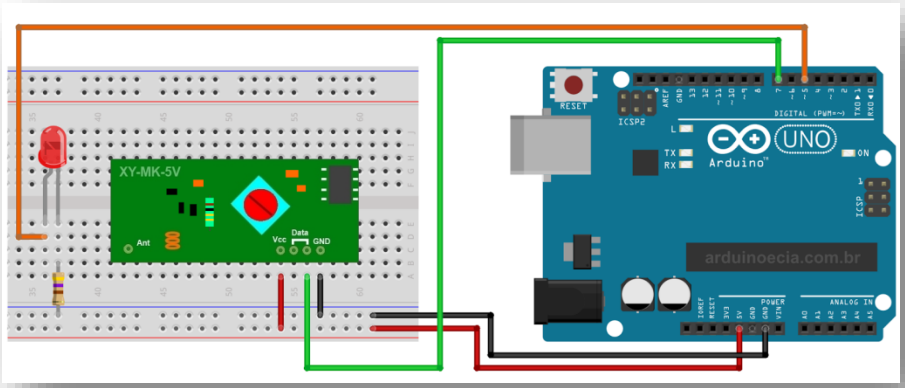


Figura 97: circuito receptor

A programação

Para transmissor:

```
#include <VirtualWire.h>

//Define pinos Led e Botao
const int ledPin = 13;
const int pino_botao = A0;

int valor_botao;
char Valor_CharMsg[4];
//Armazena estado led = ligar/desligar
int estado = 0;

void setup()
{
  Serial.begin(9600);
  pinMode(ledPin,OUTPUT);
  pinMode(pino_botao,INPUT);
  //Pino ligado ao pino DATA do transmissor RF
  vw_set_tx_pin(4);
  //Velocidade de comunicacao (bits por segundo)
  vw_setup(5000);
  Serial.println("Trasmissao modulo RF - Acione o botao...");
}

void loop()
{
  //Verifica o estado do push-button
  valor_botao = digitalRead(pino_botao);

 //itoa(valor_botao,Valor_CharMsg,10);

  //Caso o botao seja pressionado, envia dados
  if (valor_botao == 0)
  {
    //Altera o estado do led
    estado = !estado;
    //Converte valor para envio
    itoa(estado,Valor_CharMsg,10);
    //Liga o led da porta 13 para indicar envio dos dados
    digitalWrite(13, true);

 //Envio dos dados
    vw_send((uint8_t *)Valor_CharMsg, strlen(Valor_CharMsg));
    //Aguarda envio dos dados
    vw_wait_tx();

  //Desliga o led da porta 13 ao final da transmissao
    digitalWrite(13, false);
    Serial.print("Valor enviado: ");
    Serial.println(Valor_CharMsg);
    delay(500);
  }
}
```

Para receptor:

```
#include <VirtualWire.h>

//Define pino led
int ledPin = 5;

int valor_recebido_RF;
char recebido_RF_char[4];

void setup()
{
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
  //Pino ligado ao pino DATA do receptor RF
  vw_set_rx_pin(7);
  //Velocidade de comunicacao (bits por segundo)
  vw_setup(5000);
  //Inicia a recepcao
  vw_rx_start();
  Serial.println("Recepcao modulo RF - Aguardando...");
}

void loop()
{
   uint8_t buf[VW_MAX_MESSAGE_LEN];
   uint8_t buflen = VW_MAX_MESSAGE_LEN;

   if (vw_get_message(buf, &buflen))
   {
   int i;
      for (i = 0; i < buflen; i++)
     {
       //Armazena os caracteres recebidos
       recebido_RF_char[i] = char(buf[i]);
     }
     recebido_RF_char[buflen] = '\0';

     //Converte o valor recebido para integer
     valor_recebido_RF = atoi(recebido_RF_char);

     //Mostra no serial monitor o valor recebido
     Serial.print("Recebido: ");
     Serial.print(valor_recebido_RF);
     //Altera o estado do led conforme o numero recebido
     if (valor_recebido_RF == 1)
     {
       digitalWrite(ledPin, HIGH);
       Serial.println(" - Led aceso !");
     }
     if (valor_recebido_RF == 0)
     {
       digitalWrite(ledPin, LOW);
       Serial.println(" - Led apagado !");
     }
   }
}
```