Educação com tecnologia.

Aplicando Física e Programação em Sistemas Eletrônicos

E-book

KIT Alva V1.0

Sumário

Alva

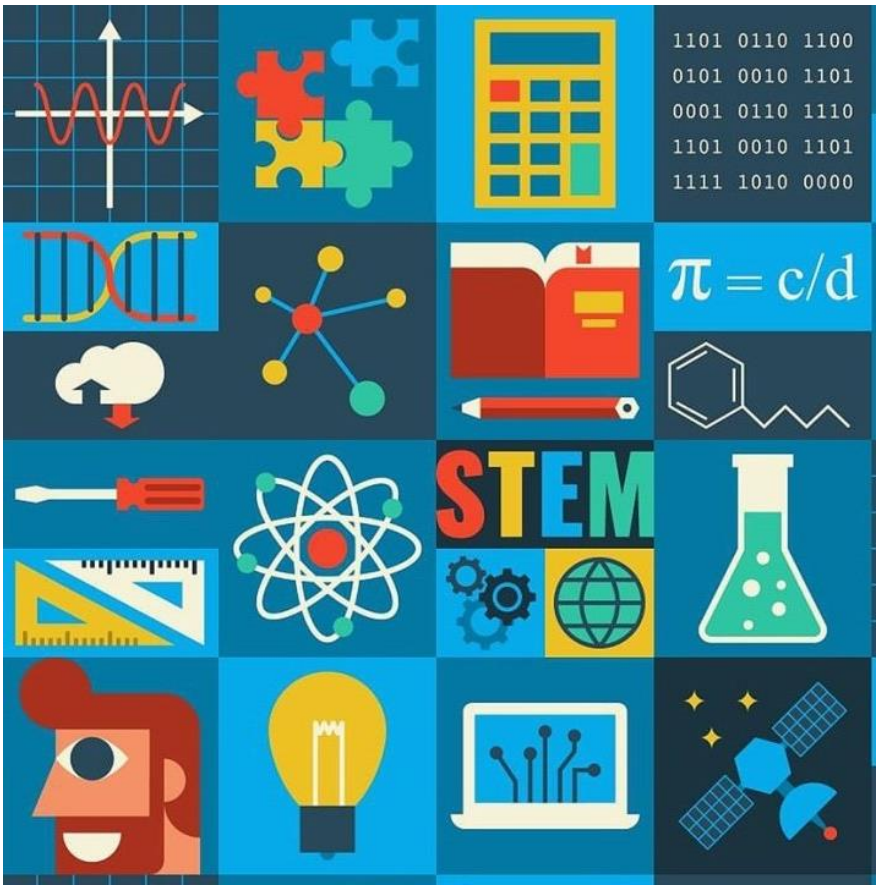
Educação com tecnologia

Introdução

A abordagem baseada na resolução de problemas tem como principal objetivo mesclar alguns dos princípios básicos da educação, ou seja, a teoria e a prática. A intenção é fazer com que o aprendizado seja dinâmico e aconteça de maneira simultânea, fazendo com que o aluno tenha as bases teóricas e teste-as ao mesmo tempo.

São apresentados problemas cotidianos e, a partir deles, as disciplinas são estudadas simultaneamente. Por exemplo: um ambiente que não precisa de luminosidade o tempo todo, apenas quando há presença de indivíduos. Instiga o desenvolvimento de sistemas para evitar que haja despercidio de energia e tenha menor gasto de recursos.

Neste exemplo várias questões de física, economia, e tecnologia podem ser abordadas, tendo a proposta pedagógica organizada e integrada com habilidades para criar soluções e não com a separação de disciplinas.



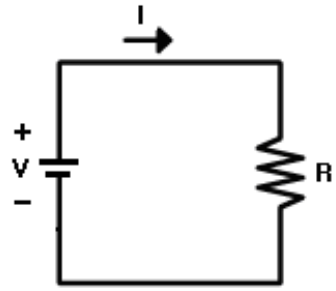
Desenvolvendo o Projeto 1

O primeiro desafio proposto é a construção de um protótipo de sinalizador de entrada e saídas de veículos, que funciona a partir da eletricidade.

O termo **Eletricidade** é usado para referir-se a uma grande área de conhecimento da Física, além disso, aplica-se à energia elétrica — uma forma de energia que possibilita o funcionamento de inúmeros dispositivos elétricos.

A Física Aplicada

Esse sistema contém duas luzes que são ligadas e desligadas de maneira alternada para atingir o efeito de alerta. Sua construção e operação são baseadas na **Eletricidade**, ramo da Física associado ao estudo de circuitos elétricos e de suas propriedades.



Os circuitos elétricos são constituídos de elementos interligados de maneira específica e devem ter, no mínimo, um percurso fechado.

A figura ao lado representa um circuito elétrico contendo uma fonte de tenão (v) e uma resistência (R) interligadas por um condutor, que possibilita a corrente (I).

Alva
Educação com tecnologia

Um condutor ligado a uma fonte estabelece **uma ddp, ou tensão**, nas pontas desse condutor, que por sua vez provoca a passagem de uma corrente elétrica (I). Isto é, a diferença de potencial movimenta as cargas elétricas (elétrons livres), que constituem a **corrente elétrica**.

O movimento das cargas elétricas só é possível se a ddp V entre dois pontos for mantida. Então, podemos considerar que a ddp é a causa da passagem da corrente elétrica (I).

Durante o movimento, as cargas elétricas realizam colisões contra os átomos e moléculas do condutor, que assim oferecem uma **resistência** à passagem da corrente elétrica através do condutor.

O cientista alemão Georg Ohm, no século XIX, realizou várias experiências com diversos fios condutores submetendo-os a diversas tensões, e verificou que para muitos materiais, principalmente os metais, o valor da resistência permanecia constante, independentemente da ddp. Resultando na **Lei de Ohm,** expressa por**:**

$$R = \frac{V}{I}$$

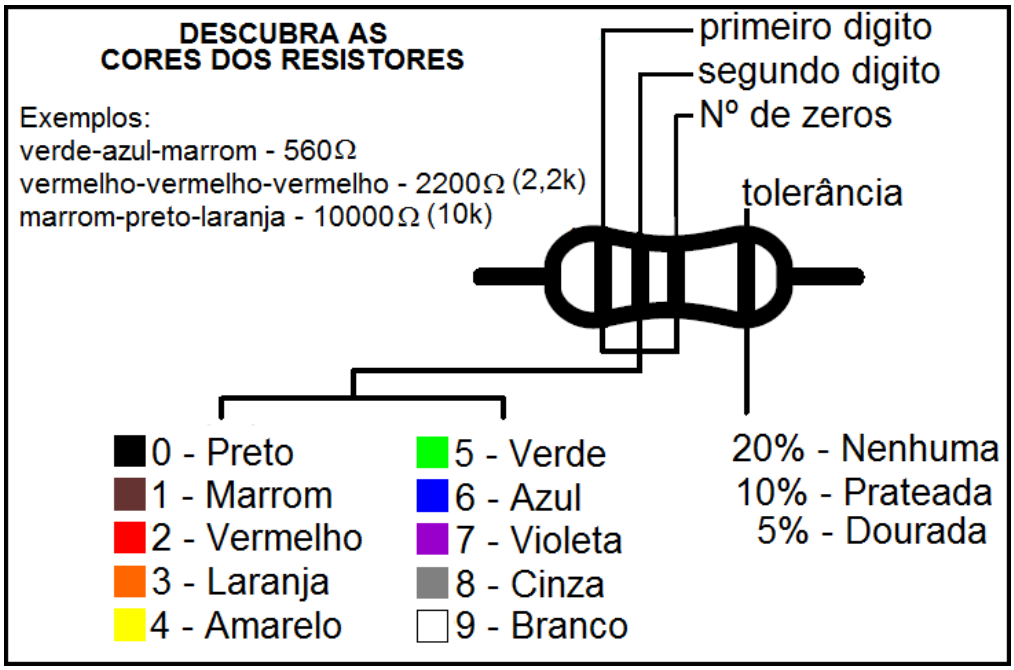
Em circuitos elétricos dos mais diversos tipos, elementos chamados resistores são usados com a função de diminuir a tensão e limitar a corrente para o adequado funcionamento de dispositivos.

- RESISTOR

Funcionalidade: Limita a corrente elétrica que passa pelo circuito. Para limitar mais ou menos corrente, o valor deste componente pode variar.

Número de pinos: 2 pinos de mesmo comprimento

Para saber o valor de cada resistor, a identificação pode ser consultada na tabela abaixo:



É importante entender como deve ser a **escolha do valor do resistor** para um circuito. Para o sistema de sinalização, utilizaremos LED's como elemento emissor de luz, funciona com valor específico de tensão. Se for ligado com tensão inadequada provavelmente ele irá queimar. Portanto, fazemos uso do resistor, que tem a função de diminuir a tensão aplicada no LED.

LED

O que isto faz: Emite uma luz quando uma pequena corrente o excita (apenas em uma direção, do pino mais longo para o pino mais curto)

É importante notar que o LED possui polaridade, ou seja, terminal positivo (Anodo) e negativo (Catodo). O terminal maior do LED é o positivo e o menor é o negativo. Ou veja também pelo chanfro, que é o lado negativo.

Iremos projetar o circuito do sistema que estamos desenvolvendo. A fonte de tensão será de 5V, que é a tensão que o Arduino fornece nos seus pinos. Ou seja, quando ligamos um pino temos 5V e quando desligamos temos 0V.

Analisando o **datasheet** (ducumento com informações de um dispositivo) de um LED , podemos ver que ele funciona com uma tensão de 2V e corrente de 20mA. Agora precisamos encontrar um valor de resistor que fará o circuito chegar próximo de 2V e 20mA.

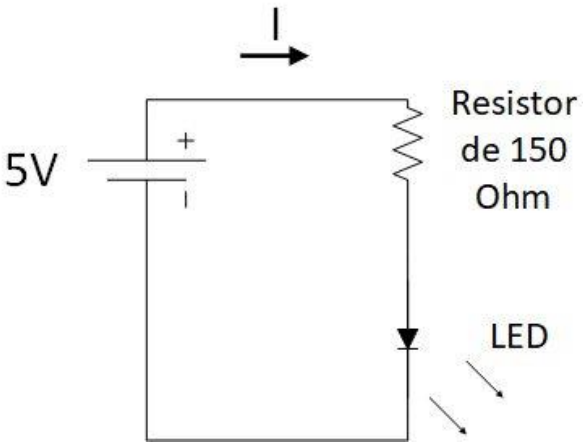
Colocaremos o resistor em série com o LED, e com isso podemos concluir que:

- A tensão total (soma das tensões no resistor e no LED) será de 5V, ou seja: VLED + VR = 5V

- A corrente total que passa pelo resistor e pelo LED é igual, ou seja, 20mA: ILED = IR = 20mA

- Precisamos colocar uma tensão de 2V no LED, ou seja: VLED = 2V

Sabendo desses detalhes, podemos concluir que a tensão no resistor será de: VR = 5V - VLED)
VR = 5V - 2V → VR = 3V.



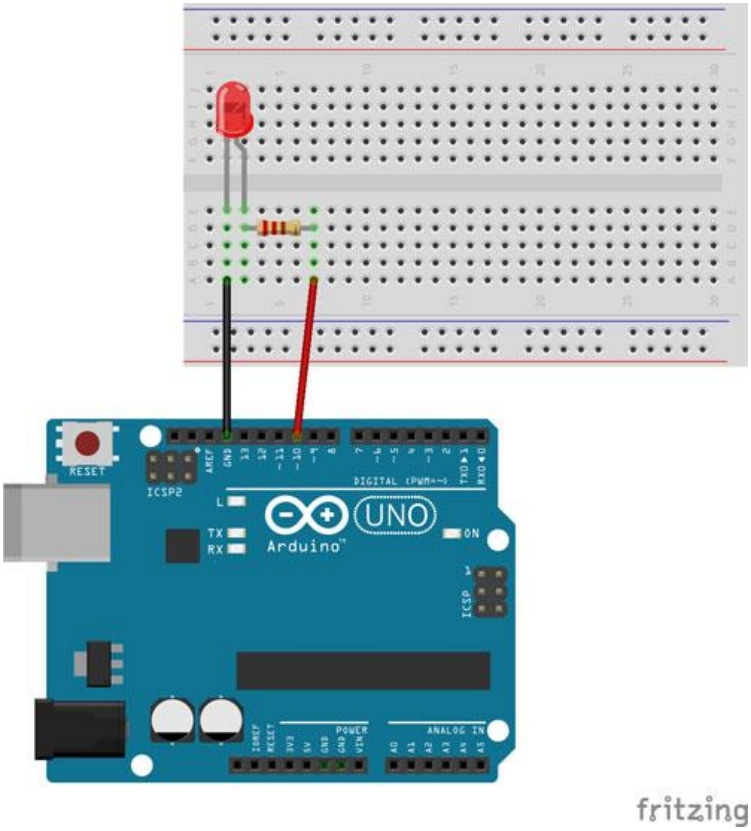
Como IR = 20mA e VR = 3V, podemos calcular o valor da resistência R do resistor que iremos utilizar através da Lei de Ohm:

$$V = RI$$

Assim, temos: 3V = R * 0.020A

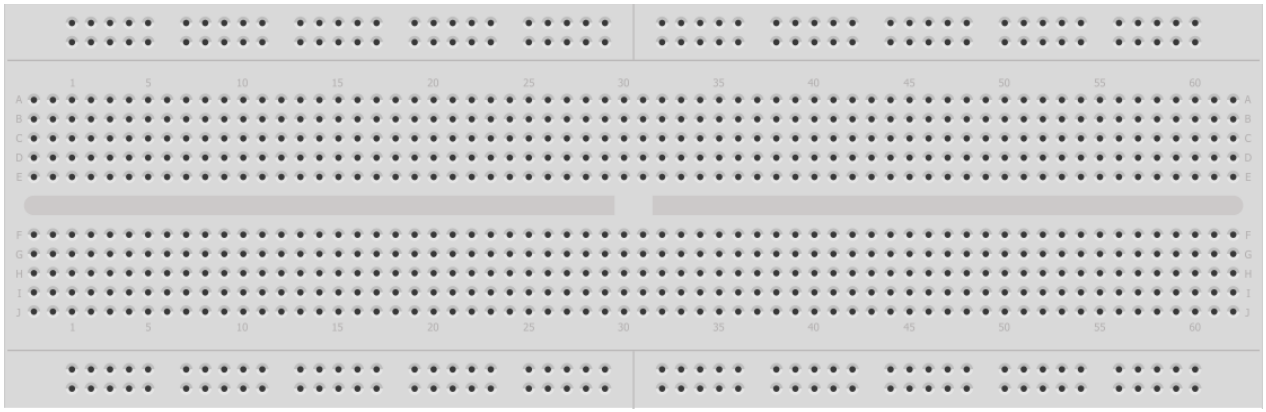
R = 3V / 0.020A → R = 150 *ohm*

Alva
Educação com tecnologia

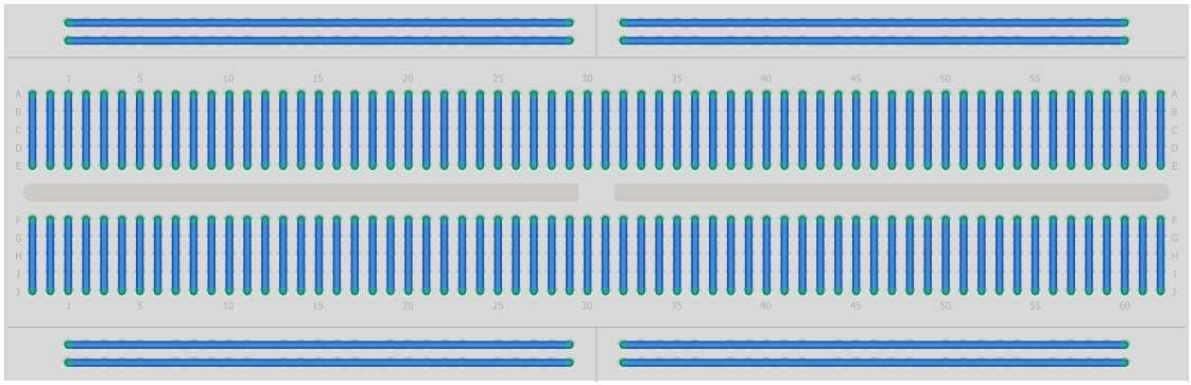


Circuito com Arduino

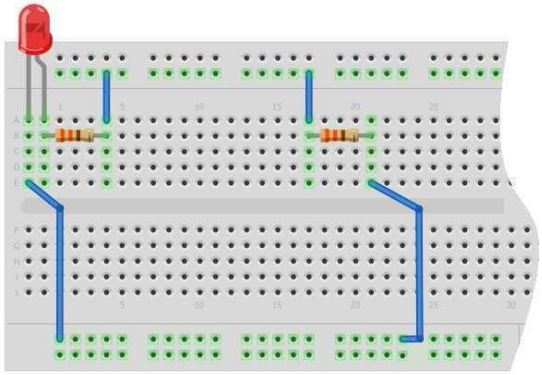
Montagem do circuito utilizamos uma **PROTOBOARD**



O que isto faz: trata-se de uma placa de plástico, cheia de pequenos furos com ligações internas, onde você irá fazer as ligações elétricas. Os furos nas extremidades superior e inferior são ligados entre si na horizontal, enquanto que as barras do meio são ligadas na vertical. Para ilustrar isto, veja abaixo como são as ligações internas da protoboard:



Cada fio azul acima representa uma ligação interna. Para deixar este componente totalmente entendido, veja o exemplo abaixo:



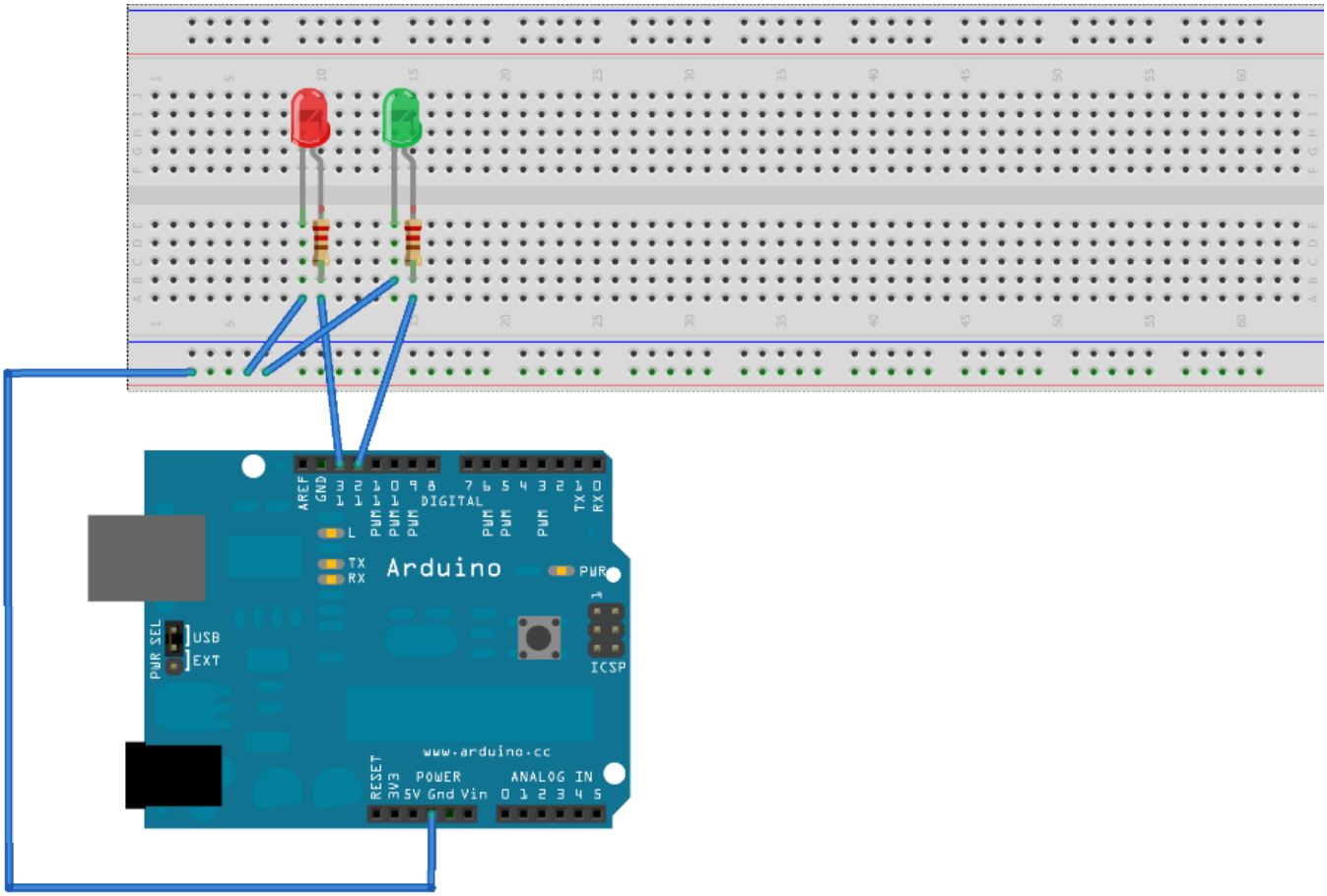
O led vermelho tem a extremidade direita ligada a um resistor. Este resistor está ligado a outro resistor por meio de uma das ligações internas superiores da protoboard. Este último resistor, por sua vez, está ligado à extremidade esquerda do led, utilizando uma das ligações internas inferiores da protoboard.

Número de pinos: na protoboard que acompanha o kit existem 840 furos, porém existem protoboards com menos e com mais furos.

Agora que já sabemos como o circuito deve ser montado, iremos realizar a montagem dos componentes na protoboard.

Material necessário

1x LED Vermelho 5 mm
1x LED Verde 5 mm
2x Resistor 150 ohm
5x Jumper Macho-macho
1x Cabo USB
1x Placa Arduino Uno



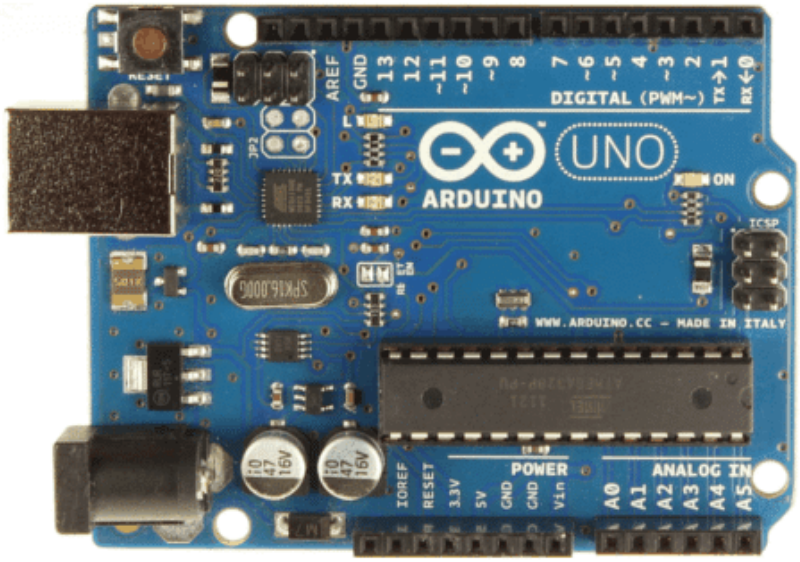
Circuito do sistema de sinalização

Arduino

O site da plataforma o define como: O Arduino é uma plataforma de prototipagem eletrônica *open-source* que se baseia em hardware e software flexíveis e fáceis de usar. É destinado a artistas, designers, hobbistas e qualquer pessoa interessada em criar objetos ou ambientes interativos.

O Arduino pode *sentir* o estado do ambiente que o cerca por meio da recepção de sinais de sensores e pode interagir com os seus arredores, controlando luzes, motores e outros atuadores. A plataforma de desenvolvimento é formada por dois componentes principais: Hardware e Software.

O ***hardware*** é composto por uma placa de prototipagem na qual são construídos os projetos. Existem diversas placas oficiais de Arduino e muitas outras não oficiais. Vamos abordar a placa oficial Arduino Uno:



O Arduino UNO possui vários conectores com função de interface com o mundo externo. Vejamos como estão organizados os pinos na placa:

- 14 pinos de entra e saída digital (pinos 0-13):

  Esses pinos podem ser utilizados como entradas ou saídas digitais de acordo com a necessidade do projeto e conforme foi definido no *sketch* criado na IDE.

- 6 pinos de entradas analógicas (pinos A0 - A5):

  Esses pinos são dedicados a receber valores analógicos, por exemplo, a tensão de um sensor. O valor a ser lido deve estar na faixa de 0 a 5 V onde serão convertidos para valores entre 0 e 1023.

- 6 pinos de saídas analógicas (pinos 3, 5, 6, 9, 10 e 11):

  São pinos digitais que podem ser programados para ser utilizados como saídas analógicas, utilizando modulação PWM, o que será tratado em outro momento.

A alimentação da placa pode ser feita a partir da porta USB do computador ou através de uma fonte DC de 9 a 12 volts.

O ***software*** é uma IDE, que é executado em um computador onde é feita a programação, conhecida como *sketch*, na qual será feita *upload* para a placa de prototipagem Arduino, através de uma comunicação serial, com um cabo USB. O *sketch* feito pelo projetista dirá à placa o que deve ser executado durante o seu funcionamento.

A IDE possui uma linguagem própria baseada na linguagem C e C++.

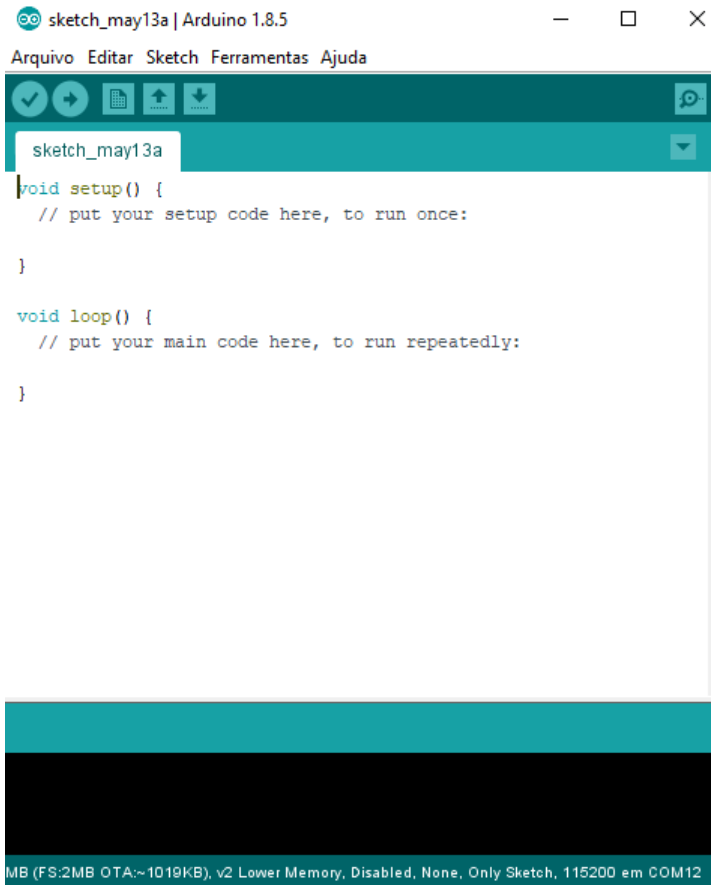
O Ciclo de programação do Arduino pode ser dividido da seguinte maneira:

1. Conexão da placa a uma porta USB do computador;
2. Desenvolvimento de um *sketch* com comandos para a placa;
3. Upload do *sketch* para a placa, utilizando a comunicação USB.
4. Aguardar a reinicialização, após ocorrerá à execução do *sketch* criado.

A partir do momento que foi feito o *upload* o Arduino não precisa mais do computador: o Arduino executará o *sketch* criado, desde que seja ligado a uma fonte de energia.

A IDE pode ser baixada gratuitamente no site do Arduino https://www.arduino.cc/en/Main/Software

Pode ser escolhida a melhor opção de download de acordo com o sistema operacional utilizado. A instalação é feita com a execução do arquivo baixado. Após a instalação a janela inicial é semelhante à figura abaixo:



A IDE é dividida em três setores: A barra de ajustes no topo, o código ou a Sketch no centro, e a área de mensagens na base, conforme mostrado na figura anterior.

Na barra de ajustes há uma guia, ou um conjunto de guias, com o nome do *sketch*. Ao lado direito há um botão que habilita o serial monitor. No topo há uma barra de menus, com os itens Arquivo, Editar, Sketch, Ferramentas e Ajuda. Os botões da barra fornecem acesso rápido às funções mais utilizadas dentro desses menus.

Abaixo são identificados os ícones de atalho da IDE:

Verify Verifica se existe erro no código digitado.

Upload Compila o código e grava na placa Arduino se corretamente conectada;

New Cria um novo *sketch* em branco.

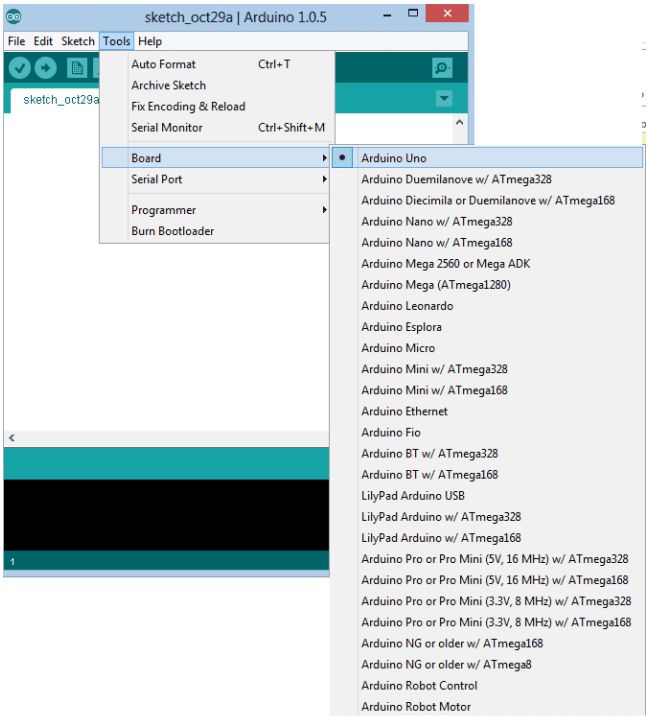
Open Abre um *sketch*, presente no sketchbook.

Save Salva o *sketch* ativo

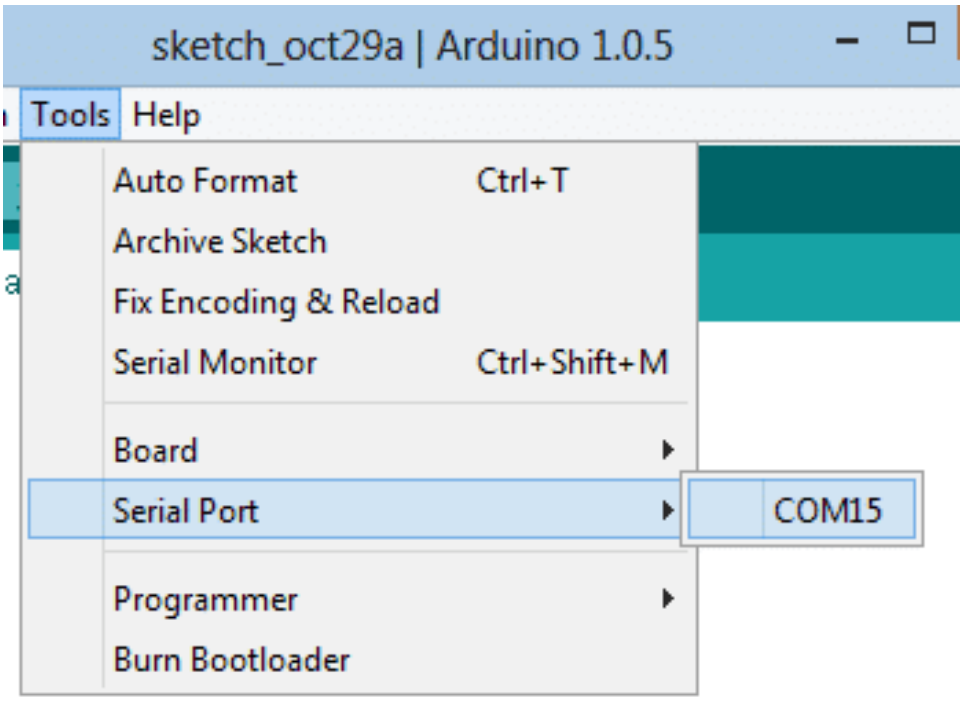
Seria monitor Abre o monitor serial.

Os demais comandos presentes na barra de menus podem ser consultados através do menu <help>

Após a conexão do Arduino ao computador, é atribuído a placa uma COM. A primeira vez que o programa Arduino for executado deve-se selecionar o modelo de placa utilizado, no nosso caso escolheremos Arduino Uno, conforme figura abaixo:



Após a definição do modelo, deve-se selecionar em qual COM a placa foi atribuída:



Após estas configurações o ambiente está preparado para uso e pode-se testar qualquer um dos exemplos que acompanham a IDE ou até mesmo com um novo sketch.

Para o sistema de sinalização, os LED's devem piscar de maneira alternada e o Arduino é ótimo para realizar essa tarefa de maneira automática. Devemos criar um conjunto de instruções para que o controlador trabalhe ligando e desligando os pinos que as luzes estão conectadas.

O conjunto de instruções é chamado também de algoritmo ou código de programação, conforme o código do sistema de sinalização abaixo:

```
//Sistema de sinalização de entrada e saída de veículos

//Declaração de variáveis
int led_vermelho = 12;
int led_amarelo = 13;

//Configuração de pinos
void setup() {
  pinMode(led_vermelho, OUTPUT);
  pinMode(led_amarelo, OUTPUT);
}
void loop() {
  digitalWrite(led_vermelho, HIGH);
  digitalWrite(led_amarelo, LOW);
  delay(1000);                    // Aguarda por 1 segundo
  digitalWrite(led_vermelho, LOW);
  digitalWrite(led_amarelo, HIGH);
 delay(1000);
}
```

Para iniciar o entendimento do código, devemos observar o que são e como são feitos os comentários em um código de linguagem C.

Para fazer um comentário que irá se desenvolver por mais de 1 linha, devemos usar os caracteres:

/* para começar um comentário de mais de 1 linha
*/ para finalizar os comentários que foram feitos anteriormente

Para fazer um comentário em 1 linha apenas, podemos utilizar:
// para fazer um comentário de apenas 1 linha

Vamos agora entender a estrutura dos programas. No inicio de todos os programas, uma ordem deve ser respeitada:

1 Estrutura de Inclusão de Bibliotecas

2 Estrutura de Declaração de Variáveis

3 Estrutura Setup

4 Estrutura Loop

5 Demais estruturas de funções

O que são estas 5 estruturas citadas acima?

Não iremos utilizar nenhuma biblioteca neste código porque nosso sistema não necessita de uma para funcionar, pois é um código básico e utiliza apenas escritas digitais e delays, funções que já estão inserida em todos os programas feitos no ambiente de desenvolvimento Arduino. Por este motivo, após os comentários iniciais, começa com a declaração de variáveis:

```
int led_vermelho = 12;
```

A linha anterior quer dizer o seguinte:

int
variável do tipo inteira

led_vermelho = 12;

Nome da variável. Neste caso, como o próprio nome diz, temos que a variável led_vermelho é atribuída ao pino digital 12.

```
int led_amarelo = 13;
```

Esta linha é equivalente a sua antecessora mudando apenas o nome da variável e o pino do Arduíno.

Vamos agora olhar a estrutura de Setup do programa:

Declaração que irá começar o Setup do programa. Sempre aberto com uma “{“ e fechada, no fim da declaração, por uma “}”.

A configuração da operação dos pinos (pinMode) pode ser como entrada (INPUT) ou saída (OUTPUT). Como neste caso queremos acender um led, a corrente elétrica irá sair do pino e não entrar. Logo, setamos os pinos 12 e 13 como saídas.

```
void setup() {

pinMode(led_vermelho, OUTPUT);          // Configura o pino digital como saída
pinMode(led_amarelo, OUTPUT);           // Configura o pino digital como saída

}
```

Por fim, neste programa, iremos analisar a estrutura Loop:

```
void loop() {

digitalWrite(led_vermelho, HIGH);

//Comando para energizar com 5V o pino que a variável led_vermelho representa, ou seja, o pino 12

digitalWrite(led_amarelo, LOW);

//Comando para deixar o pino que a variável led_amarelo representa, ou seja, o pino 13 com 0V

delay (1000);                    // Aguarda por 1 segundo

digitalWrite(led_vermelho, LOW);
digitalWrite(led_amarelo, HIGH);
delay(100);
 //As três linhas anteriores executam o inverno das três primeiras do loop
}
```

```
void loop()
```

De modo análogo ao setup, com o comando ao lado dizemos que irá começar o loop do programa, ou seja, o programa principal que ficará rodando por tempo indeterminado.
Também é aberto com uma “{“ e fechado com uma “}”.

```
digitalWrite(led_vermelho, HIGH);
```

Liga o pino com 5V Escrita digital. Por tratar-se de um pino digital, ou você terá nível lógico 1 ou terá nível lógico 0, no caso de um led, ou teremos led acesso (1) ou teremos led apagado (0). O comando então liga o led, ou seja, envia 1 para o pino 13

```
delay(1000);
```

Comando para aguardar um tempo Delay é mais uma função pronta de seu arduino. O número que for inserido entre os parêntesis será o valor, em milissegundos, que o Arduino irá esperar para seguir para a próxima instrução. No caso, temos um delay de 500 milissegundos, ou seja, uma espera de meio segundo para executar a próxima instrução.

```
digitalWrite(led_amarelo, LOW);
```

Desliga o pino

```
delay(500);
```

Aguarda 1 segundo

Estes dois comandos são análogos aos dois vistos anteriormente, com a única diferença que a escrita digital escreverá um 0 no pino do led, ou seja, um nível lógico baixo: o led apagará e o Arduino espera 1 segundo para fazer a próxima instrução.

Lembrete: Nunca se esqueça dos ponto e vírgula (;) no final dos comandos em seu programa em C.

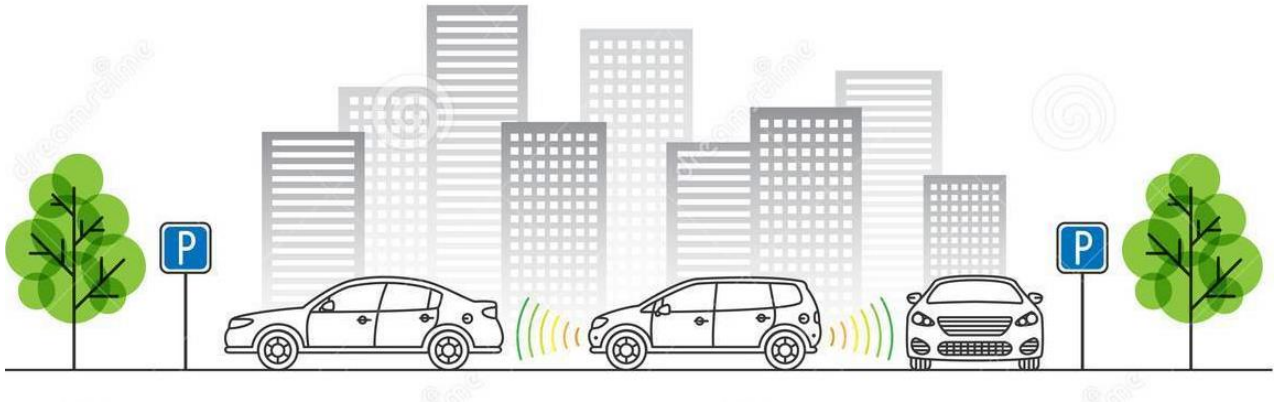
Se este programa está 100% entendido, já podemos compilar o mesmo e fazer o upload para nossa placa Arduino. Para compilar o programa devemos clicar no botão Verify do ambiente de desenvolvimento, para ver se não existe nenhum erro de código. O botão é o seguinte: 

Se na barra inferior aparecer a mensagem: *Done Compiling*, o programa está pronto para ser enviado ao Arduino. Para tanto, basta clicar no botão Upload que é o seguinte:  . Espere então o upload ser completado e pronto. Você deverá ver o sistema de sinalização funcionando com os LED's piscando com intervalos de 1 segundo.

Se houve algum problema, procure seu erro e tente consertá-lo. Se não, parabéns!

Desenvolvendo o Projeto 2

Considere que você precisa desenvolver um sistema para auxiliar no estacionamento de um veículo. Esse tipo de sistema faz aplicação de conhecimentos de ciências, tendo maior contribuição da física. Os sensores utilizados neste tipo de sistema funcionam emitindo e recebendo ondas sonoras e os sinais são condicionados e processados pelo computador do carro para identificar as distâncias em relação aos demais objetos em volta.



O desafio proposto é projetar e desenvolver um sistema eletrônico com esta funcionalidade. Para esta, e muitas outras tarefas, um projetista recorre aos conhecimentos de ciências, engenharia e tecnologias de maneira integrada, como será tratada aqui.

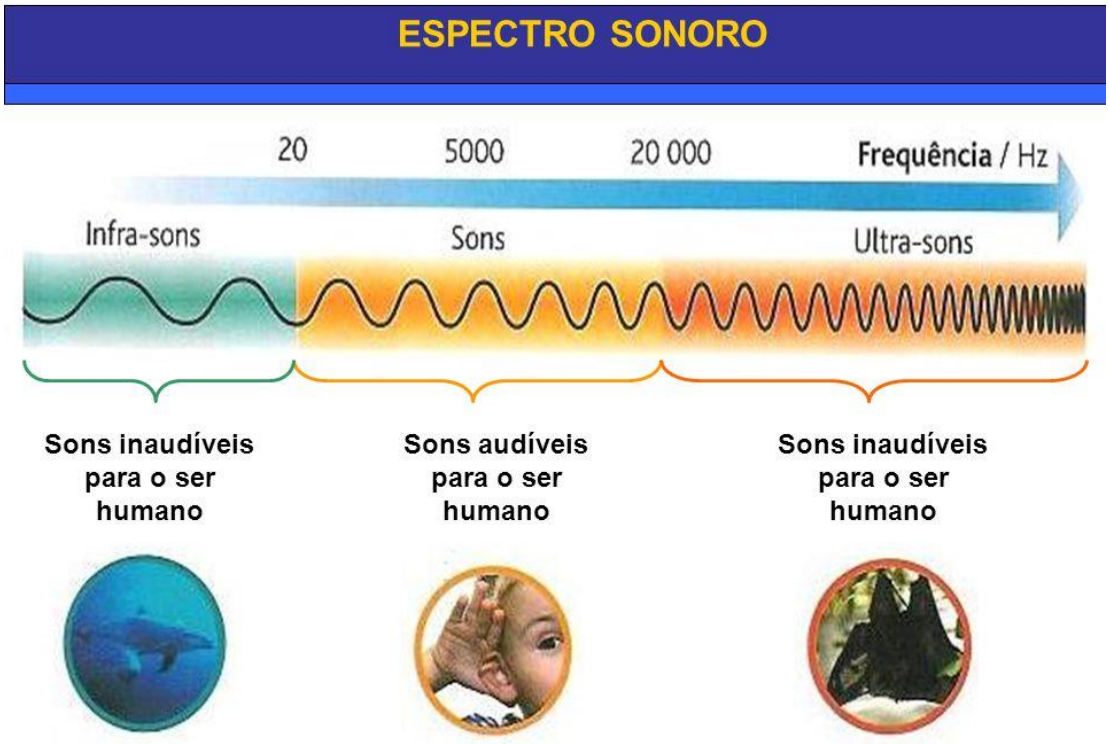
A Física Aplicada

É necessário que o veículo reconheça objetos próximos e avise o motorista por meio de sons e/ou luzes. Uma maneira eficiente encontrada é utilizando sensores ultrassônicos instalados na estrutura externa do carro, esses sensores têm princípio de funcionamento baseado na acústica, ramo da física associado ao estudo do som.

As **ondas sonoras são de natureza mecânica** e necessitam de um meio para se deslocar, seja sólido, líquido ou gasoso, e não se propagam no vácuo. Se propagam na mesma direção do deslocamento das partículas do meio, por isso são chamadas de **ondas longitudinais**. Classificadas como **tridimensionais**, com propagação em todas as direções.

Não se propagam no vácuo (por isso, o vácuo é o melhor isolante acústico que se conhece).

O ser humano é capaz de captar frequências sonoras que vão de 20Hz a 20.000Hz. Uma onda com frequência inferior a 20Hz é chamada infrassom; superior a 20.000Hz, ultrassom. Alguns animais, como os morcegos e os cachorros, são capazes de perceber os ultrassons.



O ultrassom é muito utilizado na Medicina em diagnósticos de gravidez e problemas de saúde. O sonar é utilizado na navegação de navios e na pesquisa do fundo do mar.

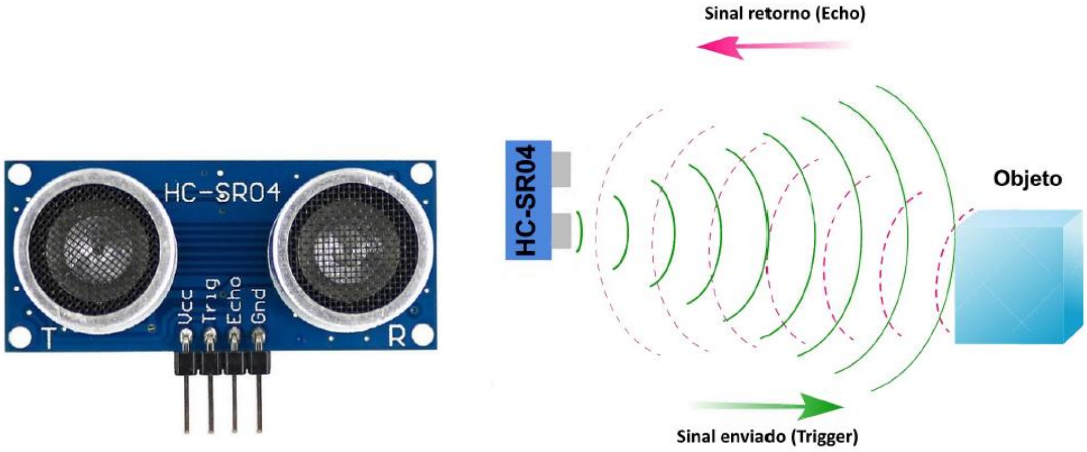
As ondas sonoras se propagam em um meio material e quanto mais próximas estiverem as moléculas do meio por onde passa a onda, tão mais rápida será a sua **velocidade**. Como nos sólidos as moléculas estão mais juntas, a velocidade é maior do que nos líquidos, cujas moléculas estão mais separadas. Os gases têm suas moléculas mais separadas ainda que os líquidos; portanto, sua velocidade é a mais lenta.

Exemplos de velocidade do som: Ar: 340 m/s Água: 1.450 m/s Ferro: 4.450 m/s

O sensor HC-SR04, utilizado neste projeto, envia sinais ultrassônicos de 40khz e aciona um relógio de alta precisão para cronometrar o tempo entre a onda sonora colidir com um obstáculo e refletir de volta ao receptor, para então calcular a distância entre o sensor e o obstáculo.

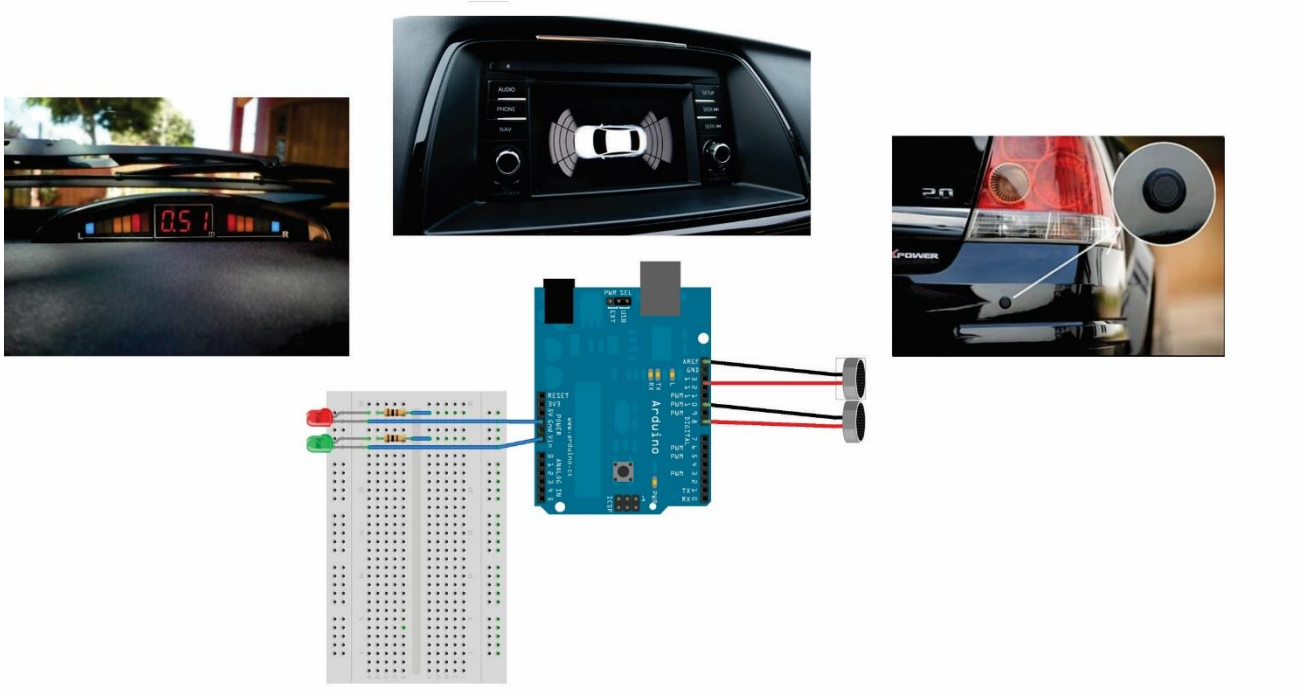
Uma vez que a velocidade do som no ar é 340 m/s, é possível, através do tempo que o sinal levou para colidir com o obstáculo e retornar, saber qual a distância percorrida entre sensor e obstáculo a partir da equação.

$$\mathbf{d} = \frac{\mathbf{V} * \mathbf{t}}{\mathbf{2}}$$



Os sensores ficam insalados na estrutura externa do carro, em posiçoes adequadas para que as ondas sejam operadas e assim objetos serem detectados.

Os alertas de proximidade são feitos com luzes e sons no painel do carro. Neste projeto são usados LED's para essa função.



Os carros possuem computadores de bordo responsáveis pelo gerenciamento e controle dos mais diversos tipos de sensores e sistemas. O Arduino é um controlador que representará o computador de bordo do carro.

Material necessário

1x LED Verde 5 mm

1x LED Vermelho 5 mm

2x Resistor 300 ohm

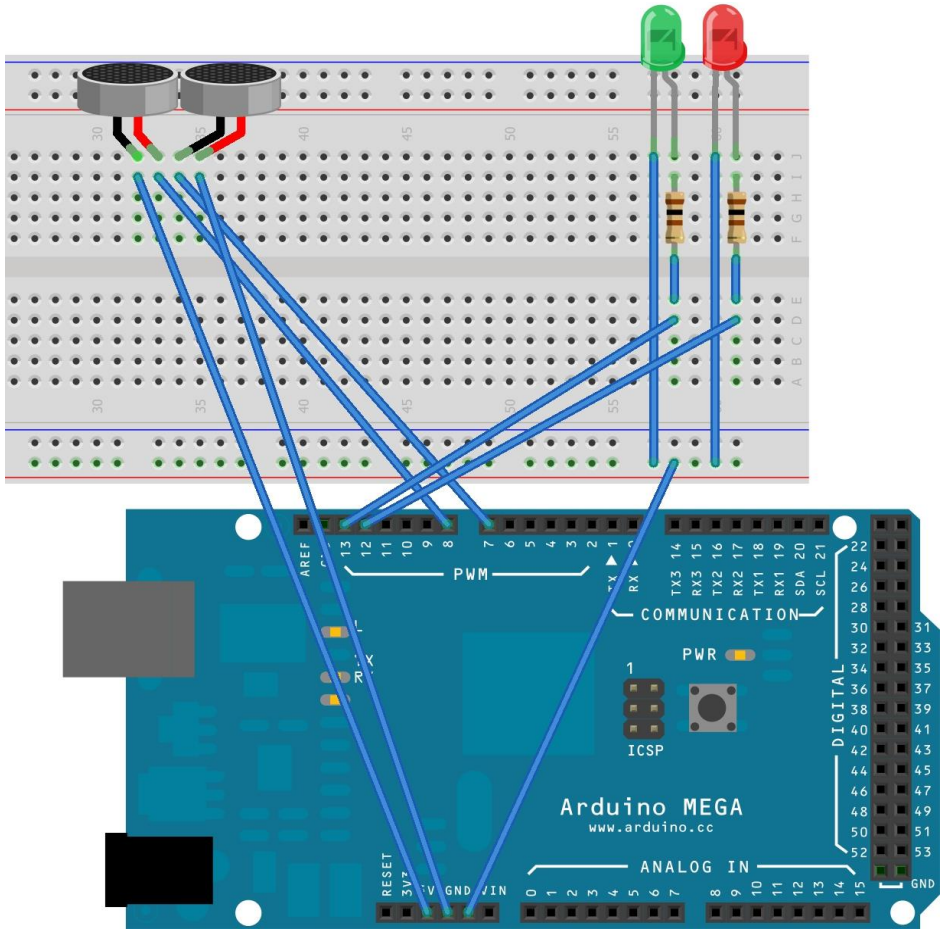
1x Sensor Ultrassônico

10x Jumper Macho-macho

1x Cabo USB

1x Placa Arduino Uno

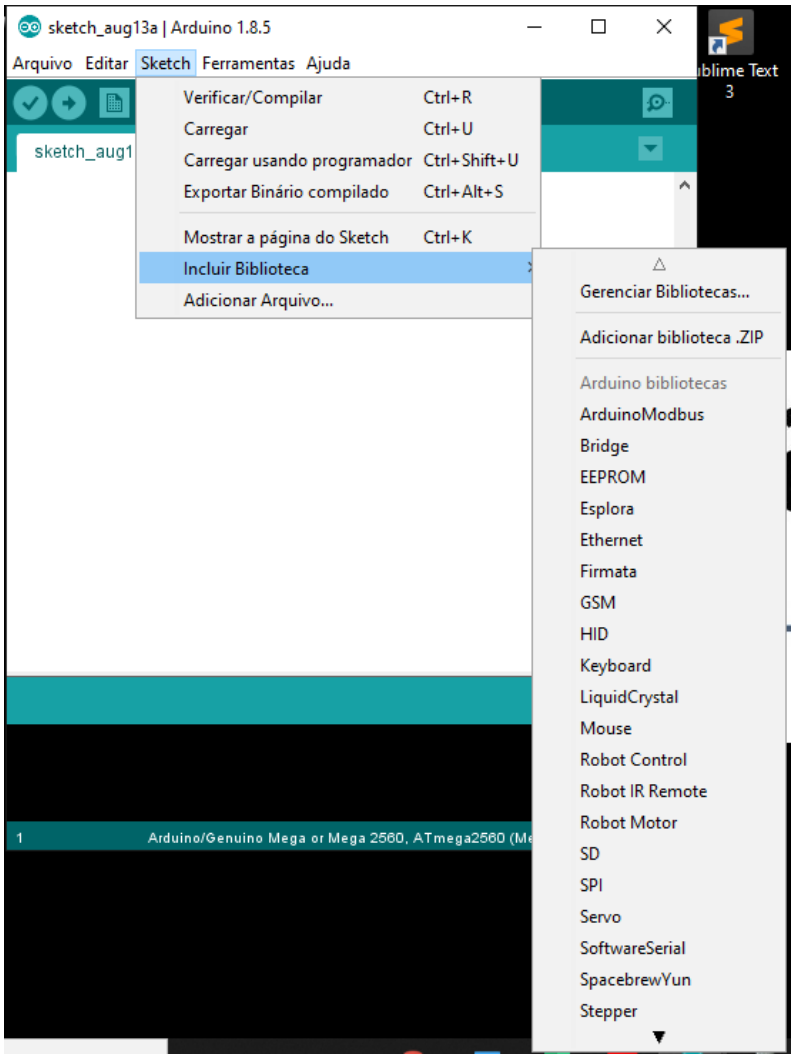
Montagem do circuito



O diferencial de uma placa como o Arduino está profundamente ligada à possibilidade de utilização

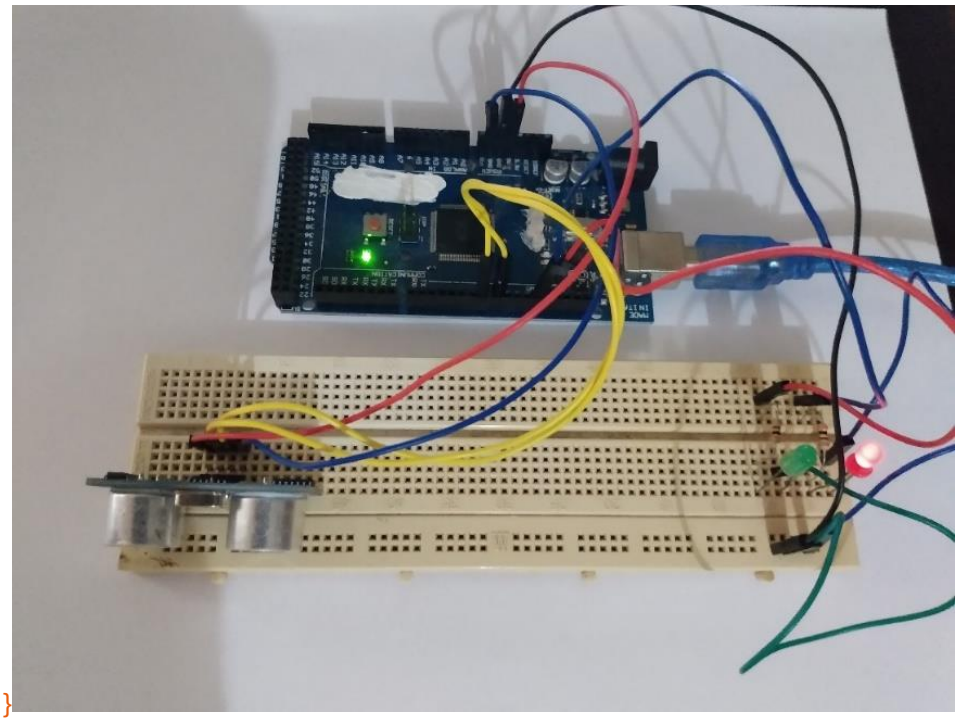
da estrutura de bibliotecas. Neste projeto utilizaremos uma biblioteca para fazer o sensor ultrassônico funcionar.

Portanto, o que são Bibliotecas? São conjuntos de funções desenvolvidas para uma aplicação particular. Seu ambiente de desenvolvimento Arduino já vem com algumas bibliotecas instaladas. Para vê-las, simule que você quer importar uma biblioteca (apenas simule, não precisa clicar em nenhuma por enquanto). Para tanto, clique em SKETCH > INCLUIR BIBLIOTECA... e veja quantas bibliotecas prontas para seu uso já existem:



Para o sistema auxiliar de estacionamento, é necessário baixar a biblioteca Ultrasonic.h, essa biblioteca está disponível para download clicando aqui (https://github.com/evsystems/ultrasonic), após realizar o download da biblioteca, você deve descompactar o arquivo e adicionar à pasta **libraries** da IDE Arduino. Após instalar a biblioteca Ultrasonic.h, iremos criar um sketch e desenvolver o código para o sistema.

```
#include <Ultrasonic.h>

Ultrasonic ultrasonic(7, 8);
//pino trig do sensor é conectado ao pino 8 do Arduino
//pino echo do sensor é conectado ao pino 7 do Arduino

int distance;

int ledPin13 = 13;
int ledPin12 = 12;

void setup() {
  Serial.begin(9600);
  pinMode( ledPin13, OUTPUT);  //LED Verde
  pinMode( ledPin12, OUTPUT);  //LED Vermelho

}

void loop() {
  // Pass INC as a parameter to get the distance in inches
  distance = ultrasonic.read();

  Serial.print("Distance in CM: ");
  Serial.println(distance);

    if (distance <= 50)
  {
    digitalWrite(ledPin12, HIGH);
    digitalWrite(ledPin13, LOW);
  }
  else {
    digitalWrite(ledPin12, LOW);
    digitalWrite(ledPin13, HIGH);

  delay(1000);
  }
```



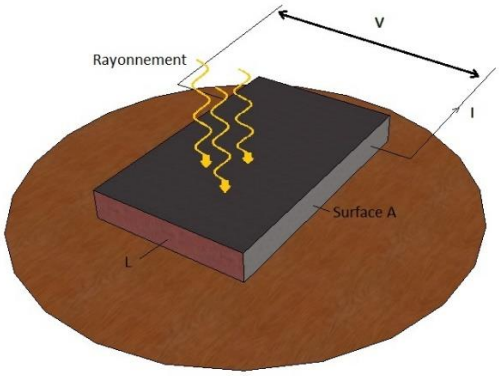
}

Desenvolvendo o Projeto 3

Os postes de iluminação pública são equipados com sensores chamados de fotocélulas ou relês fotoelétricos, que detectam a intensidade de luz natural no ambiente. Quando o sol se põe e diminui a incidência de luz, os sensores enviam sinais para um sistema de ligamento automático das luzes.

Física Envolvida

A Física envolvida no LDR lembra bastante o efeito fotovoltaico. Quando os fótons (partículas de luz) incidem sobre a superfície do componente, os elétrons presentes no material semicondutor são liberados. Com isso, sua condutividade aumenta e a resistência cai.



E, como o material possui alta resistência em seu estado normal, sua resistência fica baixa quando incide muita luz nele. Ou seja, se está escuro a resistência é máxima, e se está claro, a resistência é minima.

O desafio é identificar a quantidade de luz presente em um ambiente utilizando o Arduino e o sensor de luminosidade LDR (*Light Dependent Resistor*).

O sistema de ligamento automátoco de luz deve funcionar conforme a intensidade de luz no ambiente. Com isso conseguimos manter uma luz ligada apenas quando a quantidade de luz presente em um lugar for baixa, com objetivto de economizar energia elétrica.

Material necessário

1x LED Vermelho 5 mm

1x Resistor 220 ohm

1x Resistor 10K ohm

1x Sensor de luminosidade LDR

7x Jumper Macho-macho

1x Cabo USB

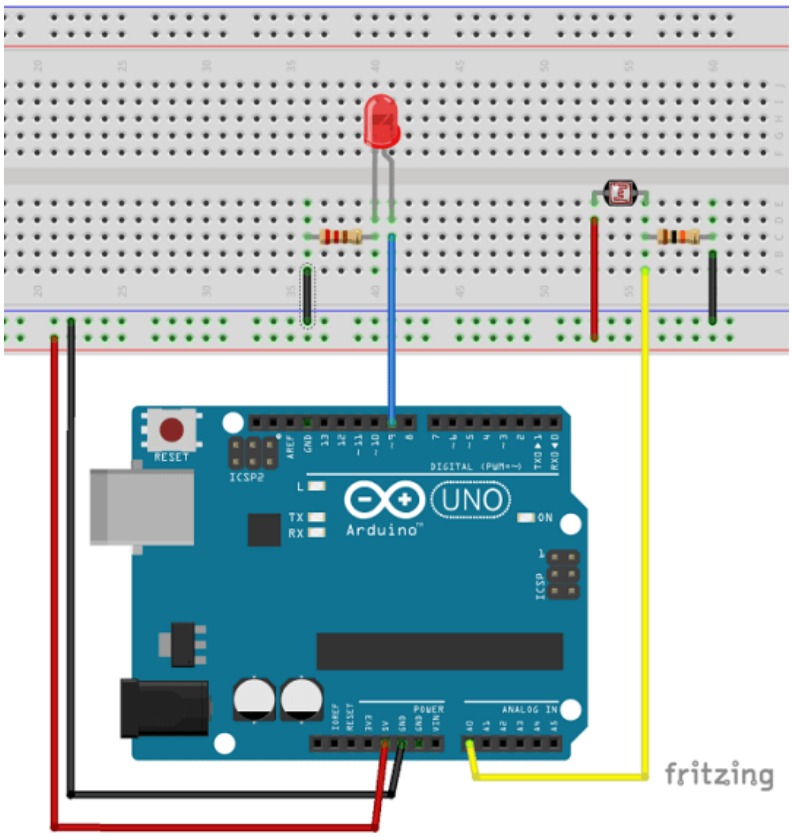
1x Placa Uno

- SENSOR DE LUMINOSIDADE LDR

O que isto faz: É uma resistência que varia conforme a luminosidade se altera sobre ele

Número de pinos: 2 pinos de mesmo comprimento

Montagem do circuito



Note que o LDR usa um resistor de 10K ohm e o LED um resistor de 100 ohm.

Programação

```
// Pode ser necessário alterar o "valor limite de disparo" para encontrar um valor adequado
// para ligar e desligar o LED quando "mover a mão" sobre o fotoresistor (LDR)

int limiteDisparo = 65;

// Ligue o LED ao pino digital 9
int led = 9;

// O fotoresistor (LDR) é conectado ao pino analógico 0
int sensor = A0;

// Armazena o valor de leitura analógica
int sensorValue = 0;

void setup() {

  // Define o LED como uma saída
  pinMode(led, OUTPUT);

  // Define o fotoresistor como uma entrada
  pinMode(sensor, INPUT);

  // Inicia a comunicação serial com uma taxa de transmissão de 9600 boud rate
  Serial.begin(9600);
}

void loop(){

  // Lê o valor atual do fotoresistor
  sensorValue = analogRead(sensor);

  // Se o valor estiver abaixo de um determinado "limite de disparo", então o LED liga, caso
contrário o LED permanece desligado

  if (sensorValue < limiteDisparo) {
     digitalWrite(led, HIGH);
  }
  else {
     digitalWrite(led,LOW);
  }

  // Imprime as leituras atuais no monitor serial da IDE do Arduino
  Serial.print ("Leitura atual do sensor: ");
  Serial.println(sensorValue);
  delay(130);
}
```