RMW\ET-18\c:\www\arduino\PartiuArduino.DOC em 25/03/2018 10:26.
RMW\ET-18\c:\www\arduino\PartiuArduinoR1.DOC em 07/01/2020 07:40.

Tutorial para leigos, isto é, pessoas que estão tomando contato com o Arduíno pela primeira vez na vida e que também são totalmente leigas em eletrônica mas dotados de incrível criatividade e enorme poder de concentração e com uma grande vontade de criar dispositivos úteis para o dia a dia das pessoas, sem depender de especialistas.

O autor desta apostila, *Roberto Massaru Watanabe*, é formado em engenharia pela Universidade de São Paulo, turma de 1972, e desejando projetar uma casa “customizada” um *Lar Doce Lar Sob Medida* para cada um dos moradores, uma casa dotada de dispositivos que automaticamente movimentam armários, pias, vaso sanitário, trancam/destrancam gavetas a um simples toque na maçaneta deixando tudo na medida da ergonomia (estatura, peso, altura, etc.) da pessoa e até de preferências pessoais como a temperatura da água do banho.

No seu desenvolvimento, Watanabe encontrou muitas dificuldades em aprofundar-se no *Mundo Arduíno*. Adquiriu dezenas de livros, acessou centenas de sites na internet mas não encontrou explicações claras para detalhes os mais simples como o código de cores de um resistor ou a polaridade das pernas de um transistor.

Anotou tudo que ía descobrindo, muitas vezes re-escrevendo de maneira mais didática e, de depois de um tempo, tinha em mãos material suficiente para compor uma apostila.

Seguindo o caráter pedagógico e a diretriz Open-Source da ARDUÍNO, todo o conteúdo deste tutorial pode também ser livremente copiado, impresso e distribuído. Àqueles que se interessarem pelo tema, sugira fotografar QR-CODE ao lado para obter, gratuitamente, um exemplar em formato PDF e assinado digitalmente via ICP-Brasil.

ESTUDANDO O MUNDO ARDUINO E ANTECIPANDO O FUTURO:

ACABEI DE GANHAR (OU COMPRAR)
UM KIT ARDUÍNO

acessar com www.ebanataw.com.br/arduino/ganheiumkit.htm ou
fotografe e guarde o QR-CODE ao lado.

MEU PRIMEIRO KIT

1 - BOAS VINDAS:

Parabéns para você que tem em maõs um kit ARDUÍNO.

Se você comprou o kit, parabéns, você tem em mãos uma ferramenta que é bastante poderosa porém que é muito fácil de usar.

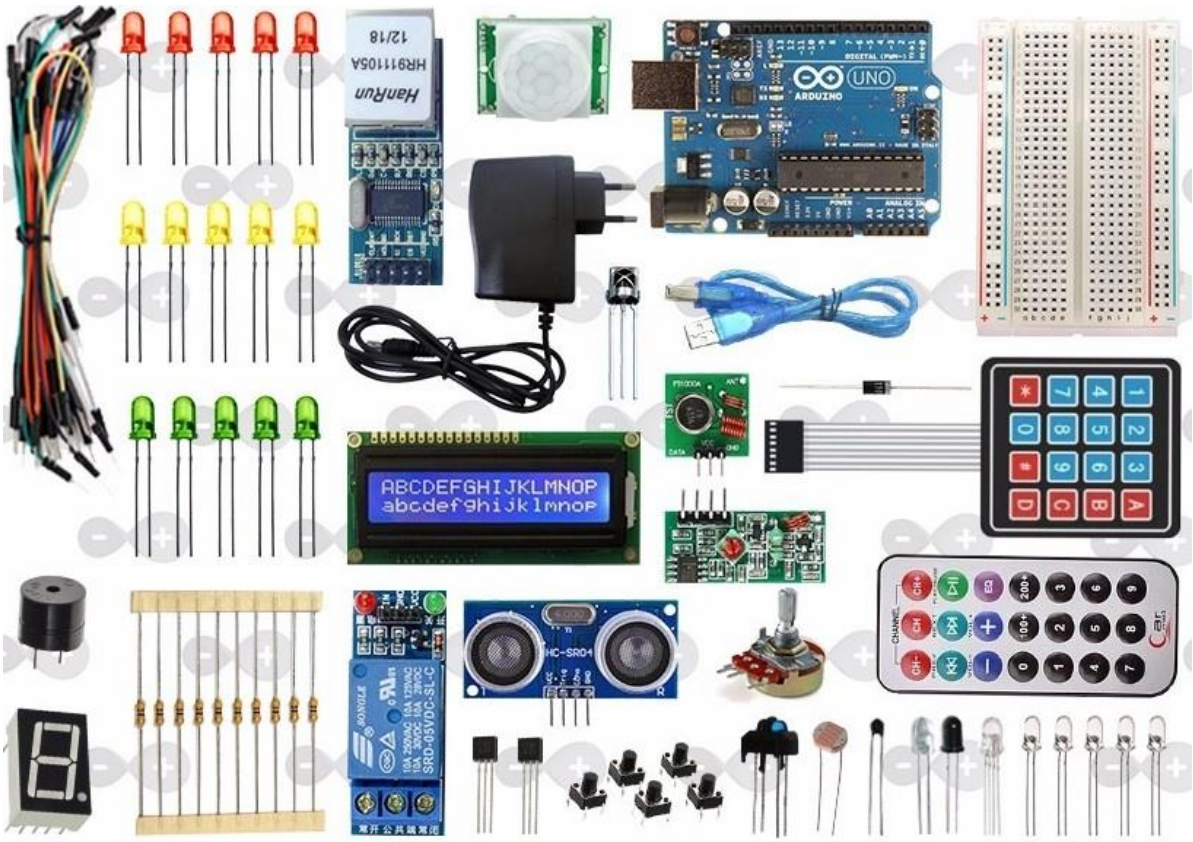
Se em vez de comprar você ganhou de presente o kit, parabéns também. O kit ARDUINO é muito fácil de usar quando você compreender cada uma das partes que compõem o kit. Existem muitos kits para iniciantes e todos eles tem um ARDUINO, um PROTOBOARD, muitos JUMPERS e diversos COMPONENTES, dependendo da área que se pretende iniciar.

Há kits mais voltados para Automação Residencial, outros para Automação Industrial, outros para Modelos Reduzidos como aeromodelismo, outros para Comunicações à Distância, outros para comando de motores e assim muitos outros tipos de kits. A pessoa que vai adquirir um kit Arduino deve pesquisar os tipos de kits existentes para ver qual deles é o mais apropriado para a pessoa que vai ser iniciada no Mundo Arduino. Os componentes que fazem parte do kit são relativos à área de trabalho e possuem sensores específicos. Veja os tipos mais comuns de sensores: Sensor de umidade do solo, Sensor de temperatura de líquidos, Sensor de proximidade por Ultrassom, Sensor de Luminosidade, Sensor de raio infra-vermelho, Sensor de GPS, Sensor de Internet e milhares de outros tipos.

Só para vocês terem uma ideia da variedade de tipos de KIT, vai aí uma pequena relação com os preços aproximados:

		PREÇO APROXIMADO, em março de 2018
1	ARDUINO Starter Kit	R$ 89,90
2	KIT Robótica	R$ 294,90
3	KIT Big Jack	R$ 329,90
4	KIT Impressora 3D	R$ 159,90
5	KIT Intermediate	R$ 179,90
6	KIT Iniciante V7	R$ 199,00
7	KIT Iniciante	R$ 84,00
8	KIT Arduino Uno R3 Automação Residencial	R$ 279,00

O kit que serve de base para o presente tutorial é o ***Kit Arduino Uno R3 Automação Residencial - Ethernet/Wifi*** e é formado pelos seguintes componentes:



Olhando para todos esses componentes você pode até ficar meio assustado mas vamos com calma pois é andando devagar que se vai ao longe, dando um passo de cada vez. Só para constar, vai aqui a lista de todos os componentes que fazem parte da figura acima:

01 – CD com software e material de apoio
01 – Arduino Uno R3 + cabo USB
01 – Fonte chaveada 9v 1A
01 – Módulo Wireless Esp8266 WIFI
01 – Chave gangorra 2 polos KCD1/101
01 – Barra de pinos preta 1x40 180 Graus
50 – Jumper fêmea fêmea premium 20cm
01 – Protoboard 400 Pontos
01 – Clip bateria com plug saída P4
05 – Led difuso Verde 5mm
05 – Led difuso Amarelo 5mm
05 – Led difuso Vermelho 5mm
05 – Led Difuso Azul 5mm
01 – Display lcd 16x2 + barra de pinos
01 – Módulo Serial I2C para display
01 – Módulo relógio RTC DS3231
01 – Bateria CR2032 para RTC
01 – Módulo bluetooth HC-05
01 – Sensor ultrassônico HC-SR04
01 – Sensor de umidade e temperatura DHT11

01 – Micro Servo motor SG90
01 – Buzzer / Sirene 5v
05 – Chave táctil
10 – Resistor 10K
10 – Resistor 330R
10 – Resistor 100R
10 – Resistor 2K
10 – Resistor 680R
01 – Módulo leitor cartão micro SD
01 – Módulo relé 1 canal
01 – Barra de pinos preta 1x40 180 Graus
01 – Potenciômetro de 100k

2 - ORGANIZANDO A SUA BANCADA DE TRABALHO:

Os primeiros passos no Mundo Arduíno já envolve, logo de cara, um quantidade muito grande de componentes, a maioria muito pequenos e que se perde com facilidade. Então é bom organizar a sua *Bancada de Trabalho*, arranjando um local adequado com mesa, tomada, notebook, estante e caixas para guardar tudo.

Eu vou te ensinar tudo, as montagens mais simples e as montagens mais complicadas, mas vamos começar pelas montagens mais simples. Você tem em mãos o kit, fornecido dentro de um organizador com divisórias.

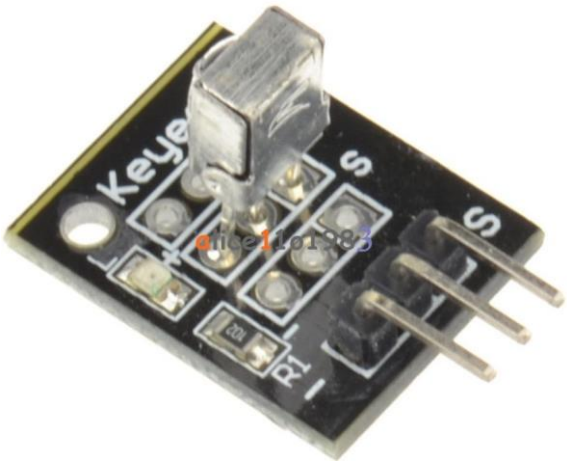
Se o kit que você comprou ou ganhou veio tudo num pacote, então adquira uma caixa organizadora com divisórias. A mais prática é a Caixa Organizadora Tamanho G de tamanho 27X34cm da Plasmont:

Organizador com Divisórias G

PLASMONT

Identifique, com uma etiqueta, cada componente do kit. Você que não está, ainda, familiarizado vai ter uma certa dificuldade de saber quem é quem.

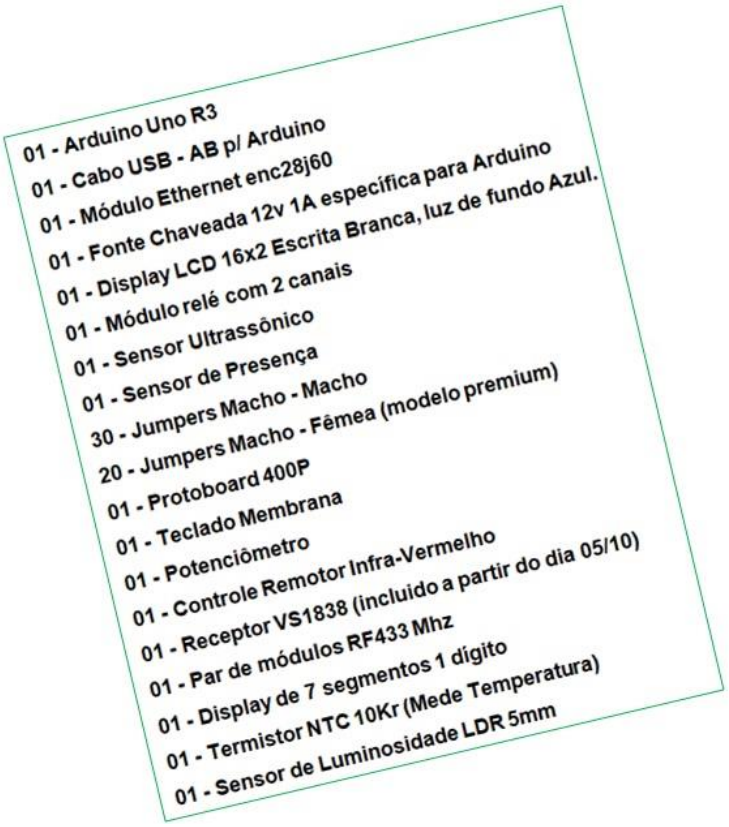
Ao ver um componente como o seguinte:



você, que é novo no pedaço, não vai fazer ideia do ele faz nem muito menos como utilizá-lo. Mas, não se desespere. Na pagina CATALOGO DE SENSORES (www.ebanataw.com.br/arduino/catalogosensores.htm) eu apresento a fotos dos sensores mais empregados. Consulte a página e localize quem é quem.

Lá, você vai descobrir que o componente acima é um *Sensor Receptor VS-1838* e dicas de onde e como empregá-lo.

Para facilitar a identificação do componente, eu recomendo que você confeccione pequenas etiquetas autocolantes com o nome dos componentes. Para isso, eu costumo usar a "lista de componentes" que geralmente vem com o kit. Copie a lista para um editor de texto tipo Word, altere o tipo de letra pelo tipo ARIAL e no tamanho 10 em negrito e imprima a lista num papel autocolante usando, por exemplo, a etiqueta *#6285 da PIMACO*.



Depois é só recortar com uma tesoura ou com um estilete e colar no componente.

Alguns componentes possuem partes sensíveis à RF e devem ser manuseados com cuidado pois ao pegar numa das pernas o componente pode captar a Radio Frequência existente no ar e, por ser muito sensível, pode queimar, inutilizando o componente. Por isso, recomendo guardar o componente num envelope plástico e colar a identificação por cima.

Além das caixas para guardar componentes é bom ter várias caixas para guardar "montagens em andamento". Algumas montagens, pela complexidade ou pelo trabalho que dá para montar podem levar um longo tempo para serem concluídas. Depois da montagem finalizada, nem sempre se deseja desmontá-la. Então sugiro ter

caixas para guardar essas montagens. Eu uso as caixas marca BEL de 4 litros pois são providas e tampas e podem ser empilhadas, além da disponibilidade de várias cores.

Caixa organizadora BEL de 4 litros medidas 19X27X12cm

3 - AS PARTES DO ARDUINO:

O sistema ARDUINO funciona por meio de um "programa", que o pessoal chama de ***sketch***.

Esse programa é feito em um computador que pode ser um desktop ou um notebook.

O melhor é montar a estrutura de programas num notebook pois algumas das montagens que faremos precisa ser executada ao ar livre como o "velocímetro", um medidor de velocidade baseado num shield GPS, velocímetro que a gente pode usar no carro, no aeromodelo ou num drone.

COMO É UM SKETCH?

O sketch é uma sequência de comandos que a gente digita num editor especial denominado ***Editor ARDUINO***.

Aparentemente complicado, na verdade o "programa" é uma coisa bem simples pois é uma sequencia lógica de comandos que a gente passa para o Arduíno. Veja um exemplo:

```
void loop() {
    digitalWrite(LED, HIGH);
    delay(500);
    digitdigitalWrite(LED, HIGH);lWrite(LED,LOW);
    delay(500);
    }
```

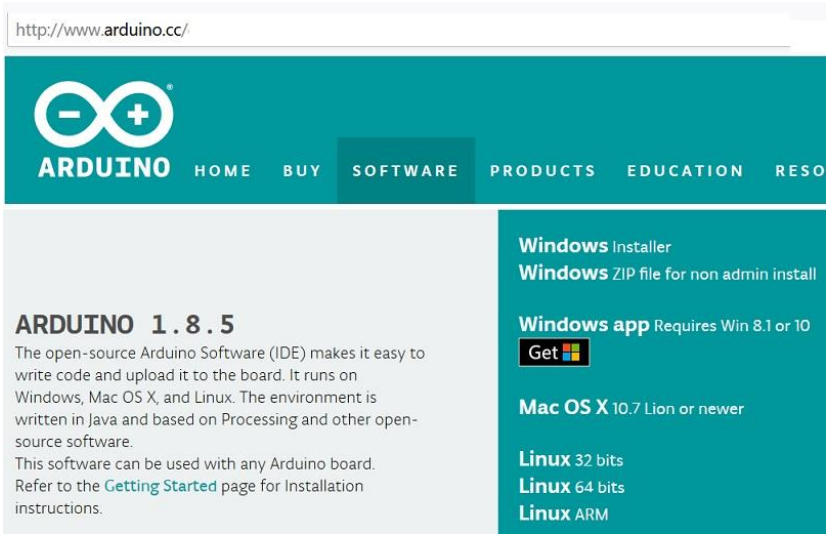
O comando digitalWrite(LED, HIGH); diz ao Arduino que desejamos "ligar" (parâmetro HIGH) o LED e o ponto-e-vírgula indica o fim da linha de comando. Na sequência, depois que o LED foi ligado, o programa diz, no comando delay(500); para fazer uma pausa de 500 milissegundos, isto é, meio segundo. Isso é importante pois logo em seguida vamos dar o comando de desligar o LED e se não pedirmos

para fazer uma pausa, não será possível a gente ver o LED aceso.

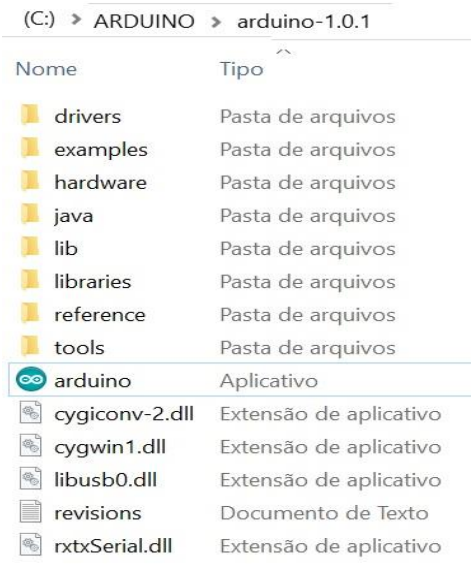
Os 4 comandos ficam dentro de um *laço de comando* (chamado LOOP, que se pronuncia LUPI) que começa com o comando void loop() { (com a chave que abre) e termina com o comando } (com a chave que fecha) que indica o final do loop. Dentro do loop o programa efetua uma volta ao primeiro comando do loop depois que executou o último comando e, assim, repete indefinidamente o ciclo, isto é, o LED fica acendendo e apagando contínuamente.

4 - Instalação do Editor ARDUINO no seu Notebook:

Você que está iniciando e não tem ainda o Editor ARDUINO instalado no seu computador, então entre no site da ARDUINO **http://arduino.cc/**, vá em SOFTWARE e baixe a última versão do Editor de acordo com o sistema operacional do seu computador (Android ou Ios). O site da Arduino apresenta a seguinte tela:

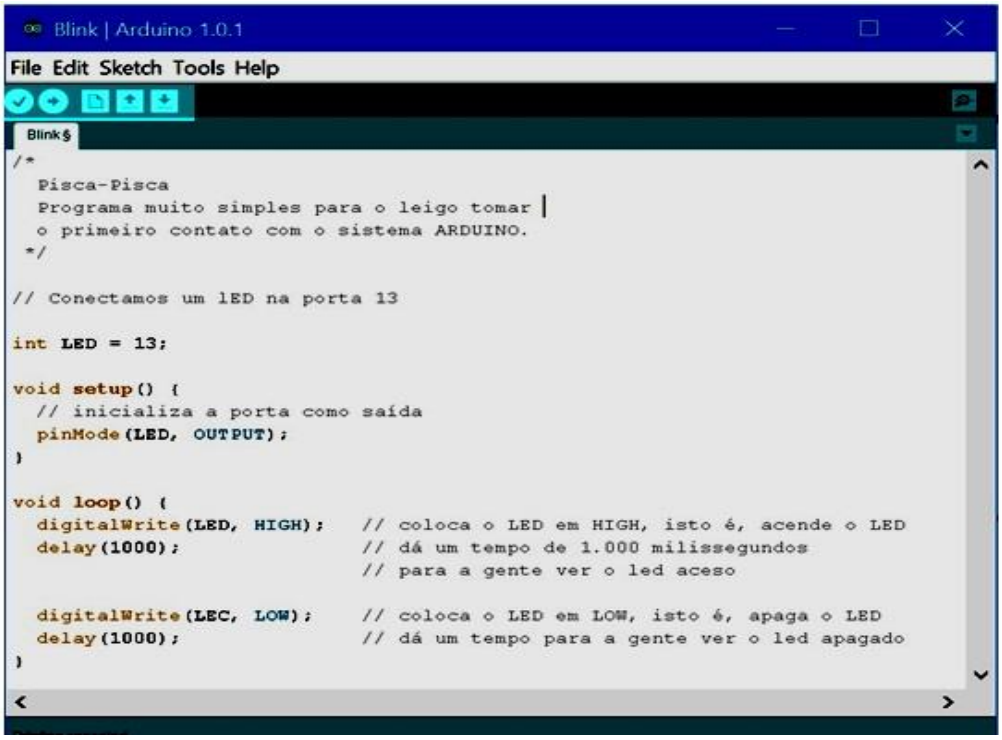


Sugiro criar uma pasta \ARDUINO e baixar o Editor Arduino lá: Depois de baixado, você verá uma lista de pastas e programas como a seguinte:



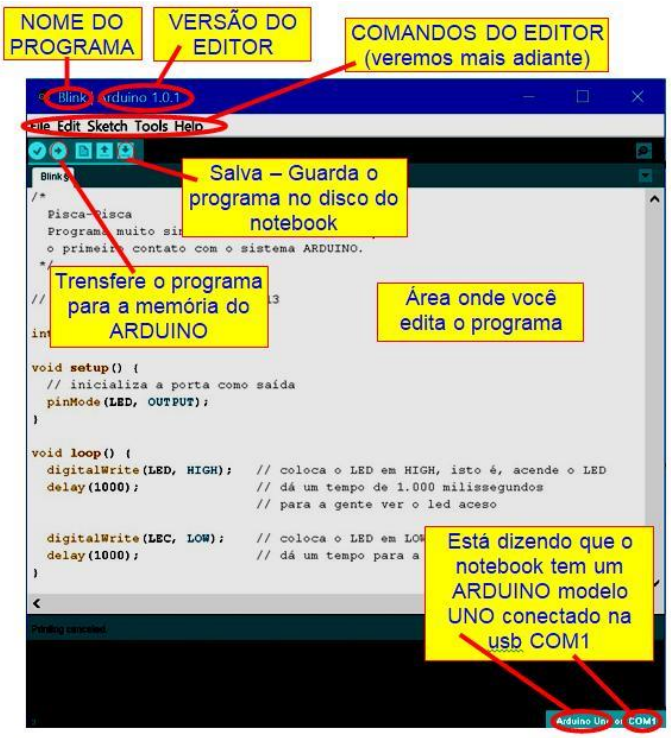
5 – Familiarizando-se com o Editor ARDUINO:

Ao clicar no Aplicativo arduino será apresentada a tela do Editor que tem a seguinte aparência:



Você, que é novo em Arduino, não vai entender esta tela e talvez ache até complicada. Mas não é.

Com calma, vamos ver cada uma das partes do Editor:

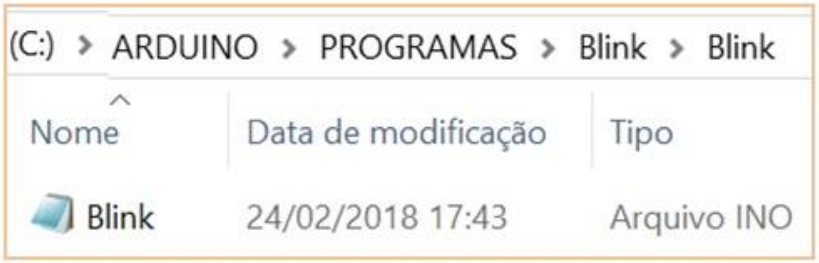


A parte mais importante do Editor ARDUINO é a área onde você edita o programa. Nela você digita o "programa" que, como veremos, não é nenhum "bicho-de-sete-cabeças". Você edita como se fosse um texto qualquer. Depois que terminar a digitação do programa você pode guardá-lo no disco rígido do seu computador.

NOME DO PROGRAMA:

O progrma precisa ter um nome. Este nome deve ser redigido conforme as regras do computador e não pode ter certos símbolos. O programa deve ser guardado numa pasta e esta pasta precisa ter, obrigatoriamente, o mesmo nome do programa.

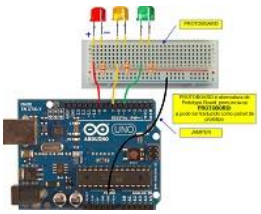
Eu sugiro criar uma pasta com o nome \PROGRAMAS dentro da pasta \ARDUINO.



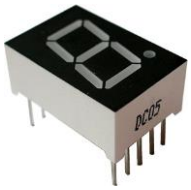
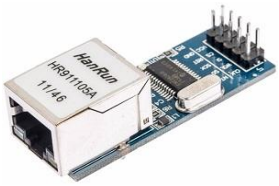
O Editor Arduino acrescenta, automaticamente, a terminação .INO ao programa.

6 - EXPERIÊNCIAS PRÁTICAS:

As tabelas seguintes apresentam as 15 experiências práticas para você se familiarizar com o funcionamento dos componentes. Realize, com atenção, cada uma das experiências, começando com as mais fáceis como o Piscar de um Led.

1- LED QUE PISCA	2- PISCA SEQUENCIAL	3- LED R.G.B.	4- Sendor LDR	5- Sensor Ultrassonic
				
Um início bem básico para você que está tomando contato pela primeira vez.	Usando um Protoboard para montar 3 LEDs que piscam um de cada vez sequencialmente.	LED que emite cores	Resistor que varia o valor da resistência em função da claridade.	Mede a distância até objetos

6- Sensor Reflexivo	7- LÊ C.R. da TV	8- Módulo Relé	9- Sensor PIR	10- Teclado 4X4
				
Mais sensível que o sensor ultrassônico, usa raio infravermelho para detectar objetos.	Recebe comando de um Controle Remoto (de TV, Ar Condicionado, etc.) e mostra na tela o código recebido.	Relé para comandar motores e outros aparelhos de alta potência.	Sensor de presença que detecta movimentos no ambiente.	Teclado de membrana que permite digitar dados.

11- Display de 1 dígito	12- Display de 32 dígitos	13- Comunicações INTERNET	14- Sensor Biométrico	15- RF-433
				
Display que mostra os algarismos e algumas letras.	Display que mostra em 2 linhas de 16 digitos cada.	Sensor INTERNET que permite ao ARDUINO falar com a WEB	Identifica a pessoa pela Impressão Digital.	Transmite/recebe dados por Rádio Frequência.

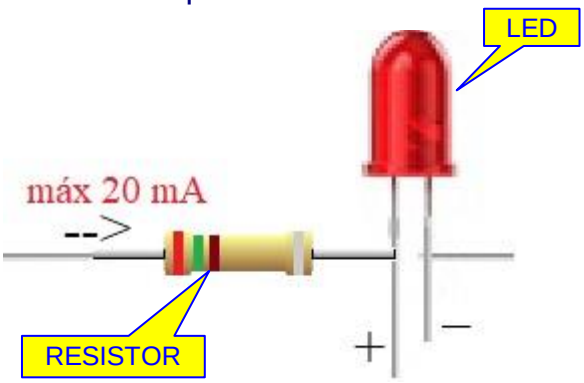
EXPERIÊNCIA Nº 1 LED Pisca-pisca

acessar com www.ebanataw.com.br/arduino/expLEDpisca.htm

O **LED** é um diodo que emite luz.

Basta fazer passar uma corrente que ele irá emitir uma luz. A cor da luz não é do LED pois a ciência não sabe, ainda, produzir luz com cores. A luz emitida pelo LED é branca e a cor é dada pelo *encapsulamento* (o plástico que recobre o led).

CUIDADO: A corrente que passamos pelo LED não pode ser maior que 20 miliamperes, de modo que você deve colocar um resistor em série com ele para limitar a corrente elétrica. Se você passar uma corrente maior que 20 miliamperes você vai queimar o led.



Sendo um diodo, o LED tem polaridade, isto é, uma das pernas é *polo positivo* e a outra é *polo negativo*. O polo positivo é a perna mais longa. O sentido da corrente deve ser do polo positivo para o polo negativo. Caso você ligue o LED ao contrário, isto é, entrando com a corrente pelo polo negativo, o LED não vai acender.

O valor do resistor vai depender do valor da voltagem.

Aplique a Lei de Ohm para calcular o valor do resitor. V = r.i isto é r = V / i

Para uma tensão de 5 Volts, r = 5 / 0,020 = 250 ohms

Para uma tensão de 3,5 Volts, r - 3,5 / 0,020 = 175 ohms

O valor do resistor não precisa ser exatamente estes. Colocando um resistor de, por exemplo 200 ohms, na voltagem de 5 Volts, o LED

acenderá mais brilhante e, ao contrário, com um resistor de 300 ohms o LED vai acender mais fraco.

Caso você não conheça o Código de Cores, veja convenção em
http://www.ebanataw.com.br/arduino/expresistor.htm

A PRIMEIRA EXPERIÊNCIA DE MONTAGEM COM O ARDUINO.

Nesta nossa *Primeira Montagem*, eu vou considerar que você é totalmente leigo em eletrônica. Então vou mostrar de forma bem detalhada, passo a passo, cada etapa desta Primeira Experiência de Montagem com o Arduino.

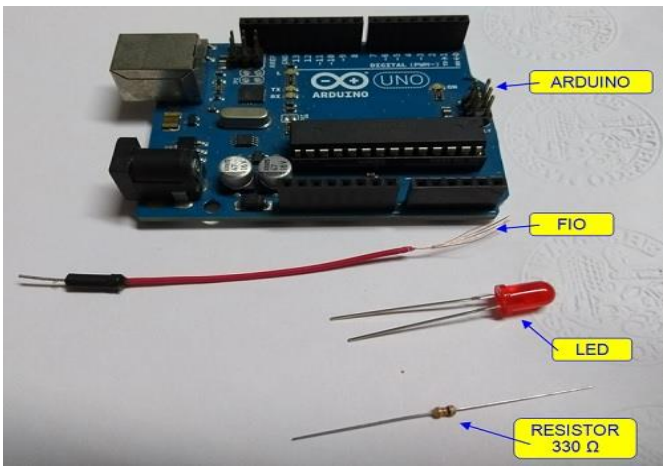
Peço especial paciência e atenção das pessoas portadoras de deficiências cognitivas e mais paciência e mais atenção ainda dos monitores e cuidadores dessas pessoas. (pessoas portadoras da síndrome de down, autismo, parkinson, idosos e outros)

Eu acredito e recomendo que, para as pessoas portadores dessas deficiências, o mergulho no Mundo Arduino é uma excelente ferramenta pois melhor compreende e mais resultados práticos extraem do Mundo Arduíno quem tem alto poder de concentração, de associação de idéias, de enxergar em multiplas dimensões e grande inventividade.

Grande capacidade de concentração não exclui as pessoas dispersivas ou inquietas pois na hiperatividade física (não pára quieta) a concentração caminha como se fosse num caminho paralelo e independente da atividade física. Aos monitores e cuidadores recomendo muita tolerância com eles pois as peças que compõem o Mundo Arduíno são, em geral, muito pequenas e de difícil manuseio. Alguns componentes como os Capacitores Eletrolíticos, os Transistores e os LEDs possuem polaridade, isto é, uma das pernas é polo positivo e a outra negativa e a montagem com as pernas trocadas resulta em circuito que não funciona.

Bem, vamos à montagem da Primeira Experiência:

Veja o que vamos precisar para realizar esta primeira experiência:



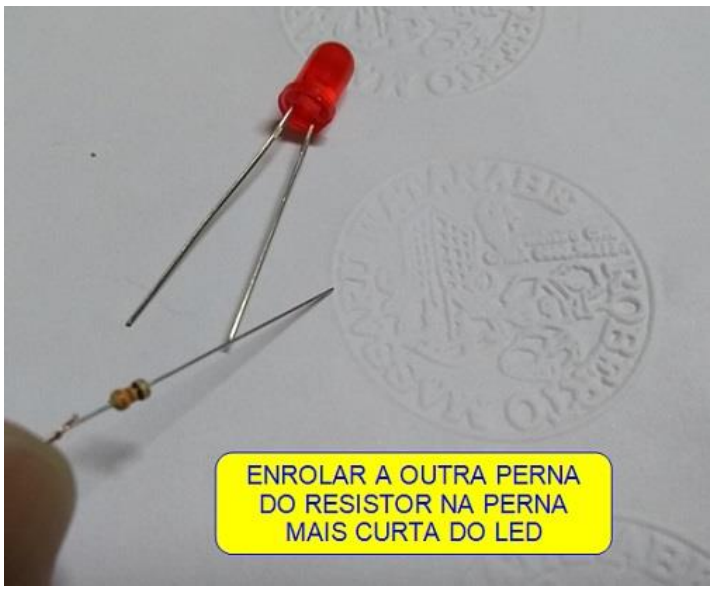
1º PASSO: Observe que a ponta do fio tem muitos filetes soltos e espalhados:

2º PASSO: Vamos enrolar a ponta do fio, com os dedos até que todos os filetes fiquem bem juntos:

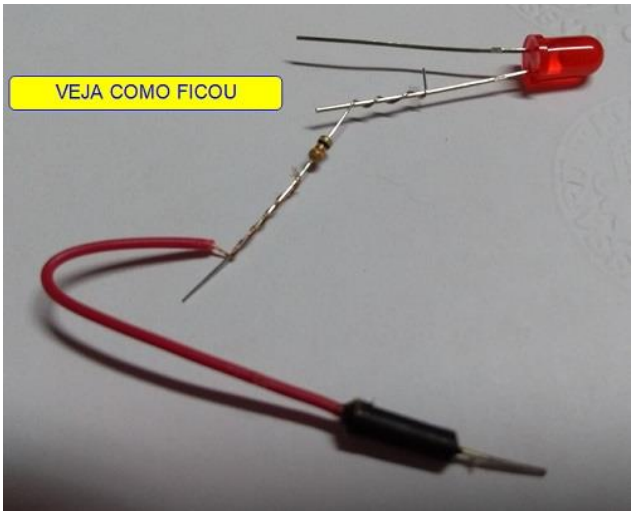
3º PASSO: Enrolar a ponta do fio em torno de uma das pernas do Resistor de 330 ohms. Pode ser qualquer uma das pernas, pois o resistor não possui polaridade:

4º PASSO: Enrolar bem apertadinho:

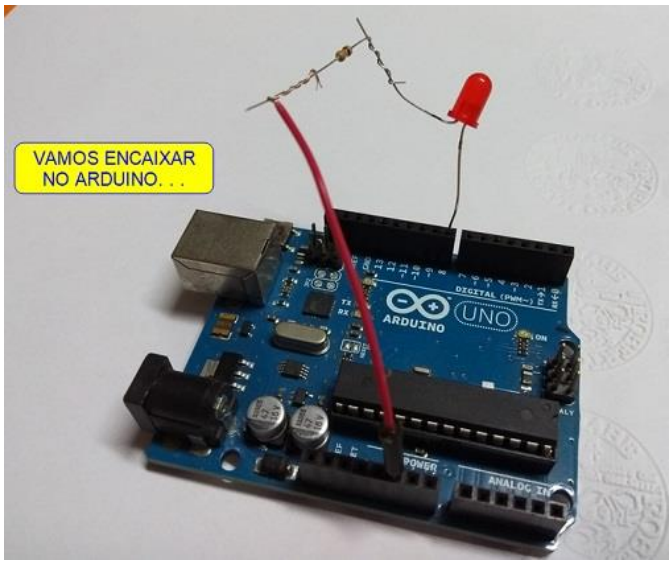
5º PASSO: Enrolar a outra perna do Resistor na *Perna Mais Curta* (polo negativo) do LED:



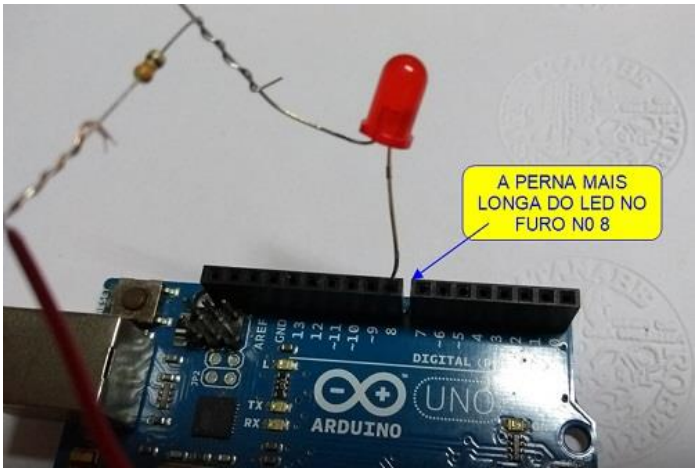
6º PASSO: Veja como ficou a montagem:



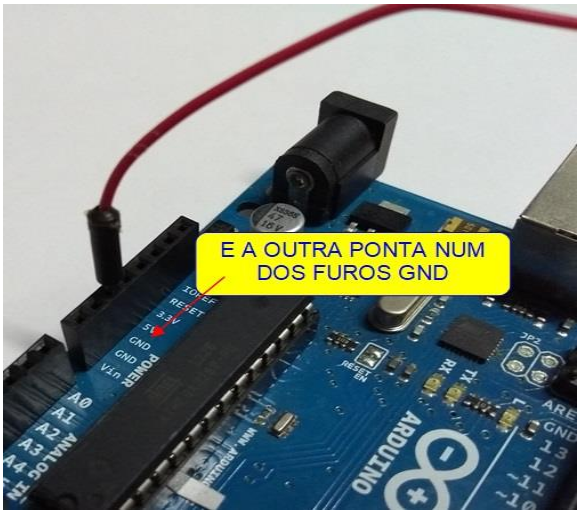
7º PASSO: Vamos encaixar no ARDUINO:



8º PASSO: A *Perna Mais Longa* (polo positivo) do LED num dos Furos Digitais do ARDUIDO, por exemplo, no Furo nº 8:



9º PASSO: A outra ponta num dos furos do GND:



Concluída a montagem física, mais conhecida como HARDWARE, passemos à montagem do SOFTWARE, que eu prefiro chamar de Programa.

O Programa que vai acionar e controlar o funcionamento dessa montagem é editado no Editor Arduíno que você deve já ter instalado no seu Notebook.

O programa propriamente dito é muito simples:

```
void setup() {
      pinMode(8, OUTPUT);
      }
void loop() {
     digitalWrite(8, HIGH);
     delay(1000);

     digitalWrite(8,LOW);
     delay(1000);
     }
```

O comando pinMode(8, OUTPUT) define para o Arduíno que a porta n^0 8 será uma porta de saída.

O comando digitalWrite(8,hIGH) ativa ou liga a porta 8, acendendo o LED que está nela.

O comando delay(1000) pede para o Arduíno parar e esperar passar 1.000 milissegundos, que é 1 segundo.

O comando digitalWrite(8, LOW) desativa ou desliga a porta 8, apagando o LED.

Depois, graças ao laço void loop(){ ... } o que está entre as chaves {} é repetido novamente, isto é, o Arduino volta a ligar o LED e assim por diante.

Digitado no Editor Arduino, o programa pode ser salvo no disco rígido do seu Notebook.

É sempre bom introduzir alguns comentários no programa pois, depois que você já fez muitos programas, pode não se lembrar bem do porque você introduziu determinados comandos nele. Os comentários devem ser colocados precedidos de certos sinais para que o Arduino não confunda com comandos.

Existem 2 símbolos que definem comentário e que o Arduino não deve levar em consideração:

Um desses símbolos é a barra dupla // que pode ser colocada numa linha:

```
// Feito por Roberto Watanabe em 08.03.2018
```

ou à frente de uma linha de comando:

```
digitalWrite(8, HIGH); // ativa a porta 8 (onde está o LED)
```

o outro é o par /* e */ que deve ser colocado em uma ou várias linhas e tudo o que estiver entre o primeiro par /* e o segundo par */ será considerado comentário e, portanto, ignorado pelo Arduino. Veja o exemplo:

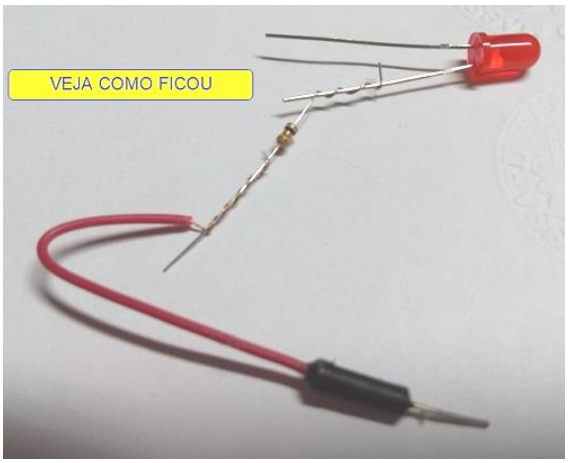
```
/*
  PiscaLED - Demonstra o funcionamento do Arduino.

  Roberto Massaru Watanabe
  R-0 28.01.2018
  R-1 30.01.2018 Introduzi os comentários.
*/
void setup() {
      pinMode(8, OUTPUT); // Define a porta 8 como porta de saída.
      }
void loop() {
     digitalWrite(8, HIGH); // Acende o LED.
     delay(1000);           // Pausa de 1 segundo.

     digitalWrite(8,LOW); // Apaga o LED.
     delay(1000);         // Pausa de 1 segundo.
     }
```

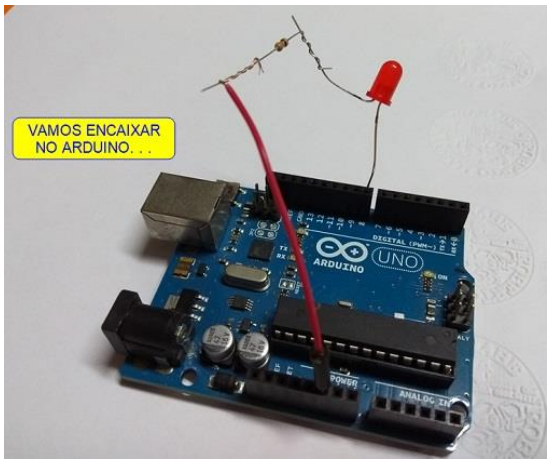
É sempre bom registrar a Revisão com números e data pois a gente costuma produzir muitos programas, às vezes muito semelhantes ou parecidos mas com um ou outro detalhe a mais ou a menos.

Por exemplo, para determinar a "proximidade" de objetos podemos fazê-lo com um Sensor Ultrassom, ou com um Transistor, ou com um Sensor Infravermelho e com muitos outros tipos de sensores. Cada um deles apresenta um certo grau de sensibilidade ou de acuidade ou de alcance. Então para aquela determinada aplicação a gente precisa testar cada um deles para ver qual deles atende melhor o controle desejado. Nestas situações, em vez de testar cada um deles num único programa, isto é, fazer as alterações no próprio é melhor produzir vários programas, um para cada sensor, e salvá-los com nomes diferentes. Obviamente, para saber depois que sensor o programa usa e quais os cuidados necessários será sempre oportuno escrever isso "dentro" do programa na forma de comentários.

NOTA IMPORTANTE:

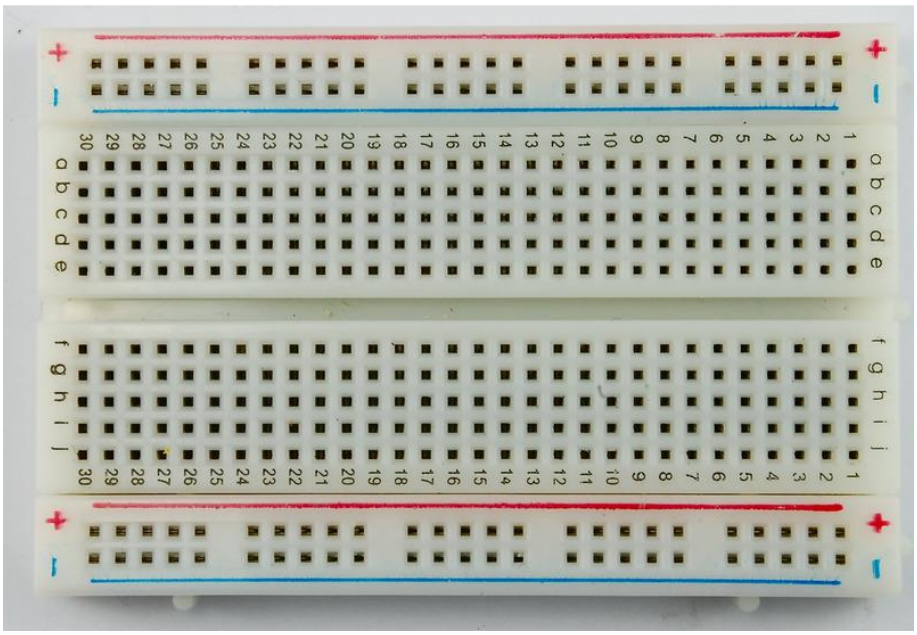
Para a ligação dos componentes na experiência n^0 1, anterior, você teve que enrolar pernas de componentes (fios) na perna de outro componente, como nas fotos:





Este tipo de ligação é muito precária pois pode ocorrer falhas no contato elétrico, a ligação pode escapar, e quando você desmontar a experiência, os componentes estarão com as pernas todas enroladas e tortas dificultando a montagem de outra experiência.

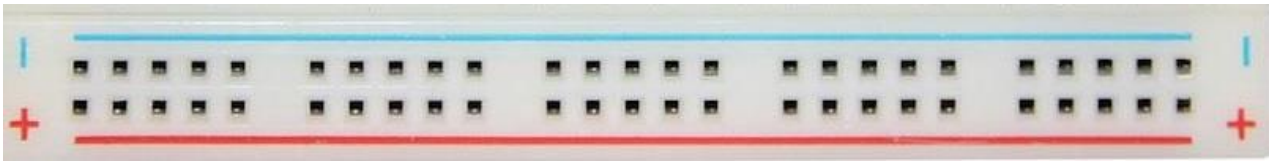
Para não danificar as pernas dos componentes, existe um componente muito prático denominado PROTOBOARD (pronuncia-se protobordi) que é uma plaquinha que mede 8,5X5,5 centímetros cheia de furos. Veja uma foto:



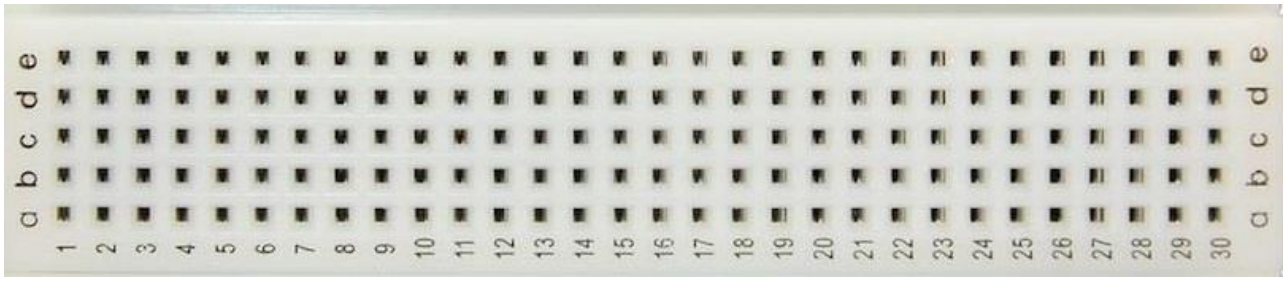
À primeira vista parece uma placa complicada mas vamos ver que existem ligações elétricas por baixo dos furos.

Os furos estão agrupados em 2 grupos:

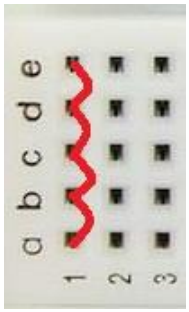
O grupo da borda:



E o grupo do meio:

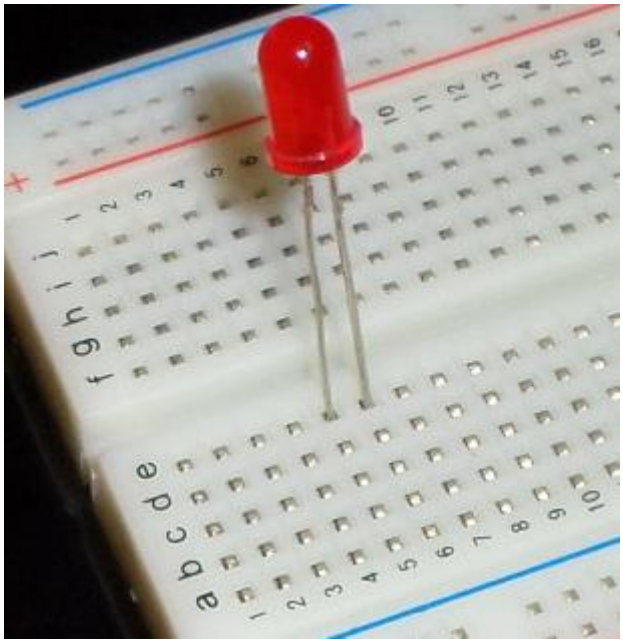


No grupo do meio, existe uma ligação elértrica entre os furos identificados com as letras a, b, c, d, e. Veja uma ilustração esquemática:

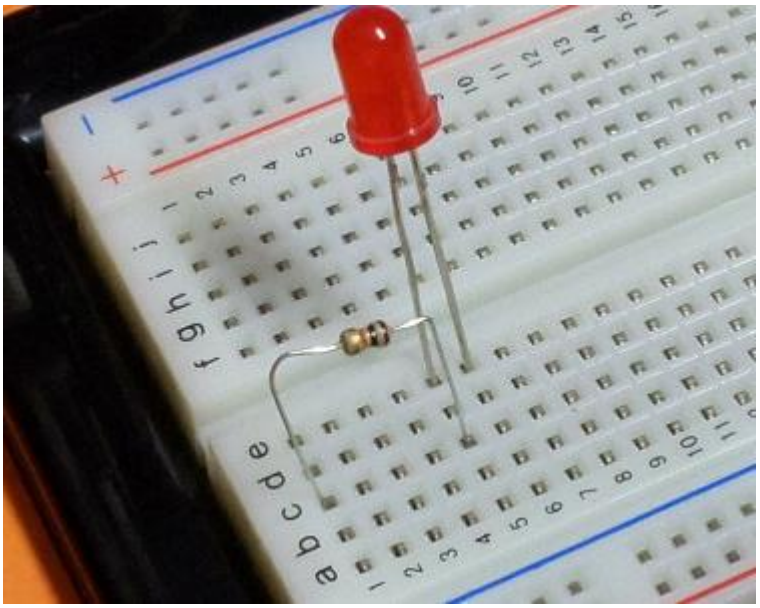


Então, quando queremos ligar 2 componentes,

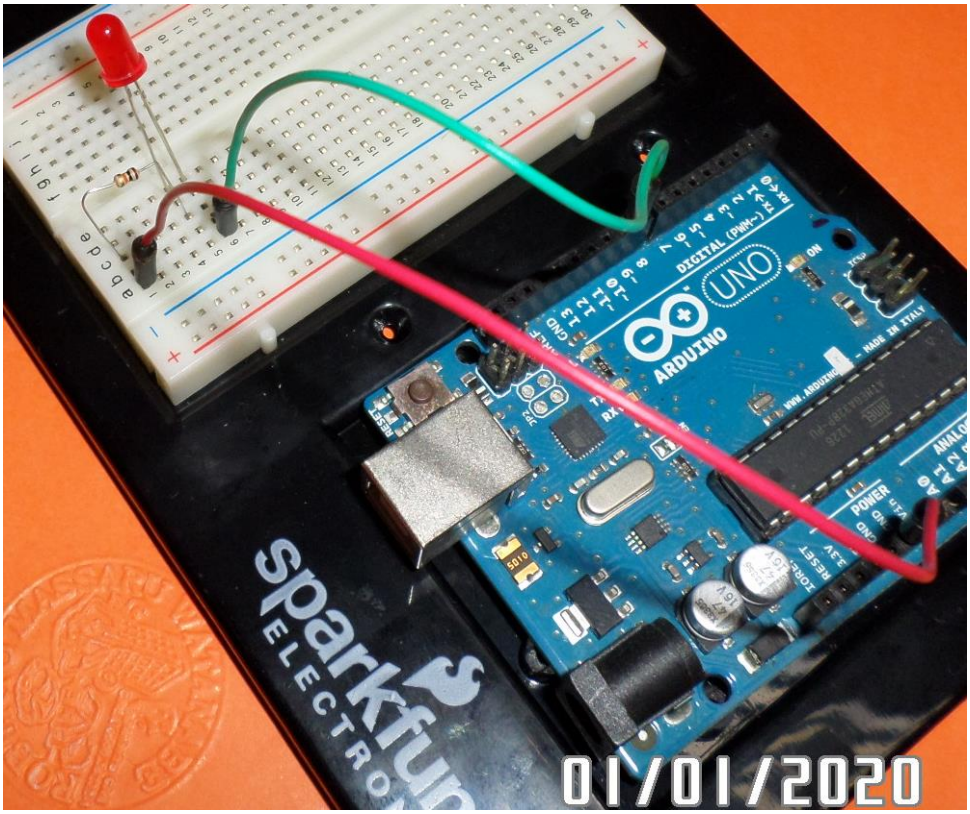
Basta enfiar as pernas de um deles nos furos do protoboard:



Por exemplo, a perna mais curta no furo e5 e a perna mais longa no furo e6. Em seguida enfiar as pernas do resistor nos furos c1 e c5. Observe que a perna do resistor enfiada no furo c5 está fazendo contato (por dentro do protoboard) com a perna mais curta do LED enfiada no futo e5.



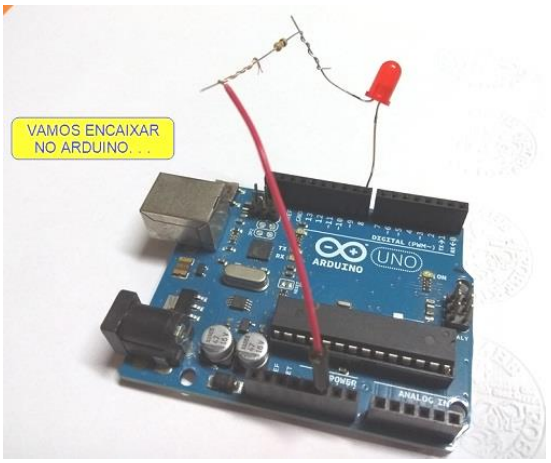
Dai, usando 2 jumpers, fazer a ligação do furo a1 com o GND no arduíno (fio vermelho) e a ligação do furo a6 com a porta 8 (fio verde) completando, assim, as conexões desta primeira experiência.

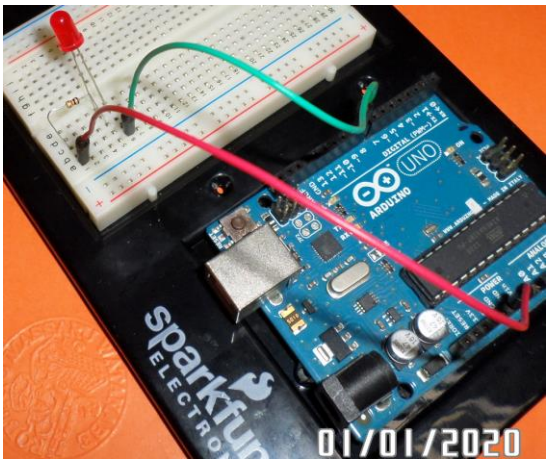


Observe que, além da placa do protoboard usamos uma placa de montagem da sparkfun que, por meio de parafusos e etiquetas autoadesivas mantém fixas as placas do Arduino e do protoboard.

Depois da experiência você pode guardar o conjunto todo numa caixa, para depois repetir ou para fazer demonstrações ou pode também desmanchar a montagem e guardar os componentes.

Compare, lado a lado, os 2 tipos de montagens:





As experiências seguintes serão todas montadas sobre a placa protoboard. Para mais detalhes sobre o protoboard, veja
http://www.ebanataw.com.br/arduino/protoboard.htm

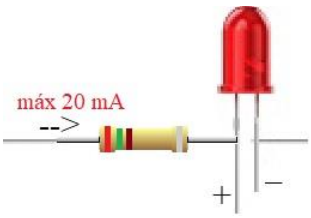
EXPERIÊNCIA Nº 2 LED – PISCA SEQUENCIAL

acessar com http://www.ebanataw.com.br/arduino/expled.htm

O **LED** é um diodo que emite luz.

Basta fazer passar uma corrente que ele irá emitir uma luiz. A cor da luz não é do LED pois não sabemos, ainda, produzir luz com cores. A luz emitida pelo LED é branca e a cor é dada pelo encapsulamento (o plástico que recobre o led).

CUIDADO: A corrente que passamos pelo LED não pode ser maior que 20 miliamperes, de modo que você deve colocar um resistor em série com ele para limitar a corrente elétrica.



Sendo um diodo, o LED tem polaridade, isto é, uma das pernas é polo positivo e a outra é polo negativo. O polo positivo é a perna mais longa. O sentido da corrente deve ser do polo positivo para o polo negativo. Caso você ligue o LED ao contrário, isto é, entrando com a corrente pelo polo negativo, o LED não vai acender.

O valor do resistor vai depender do valor da voltagem.

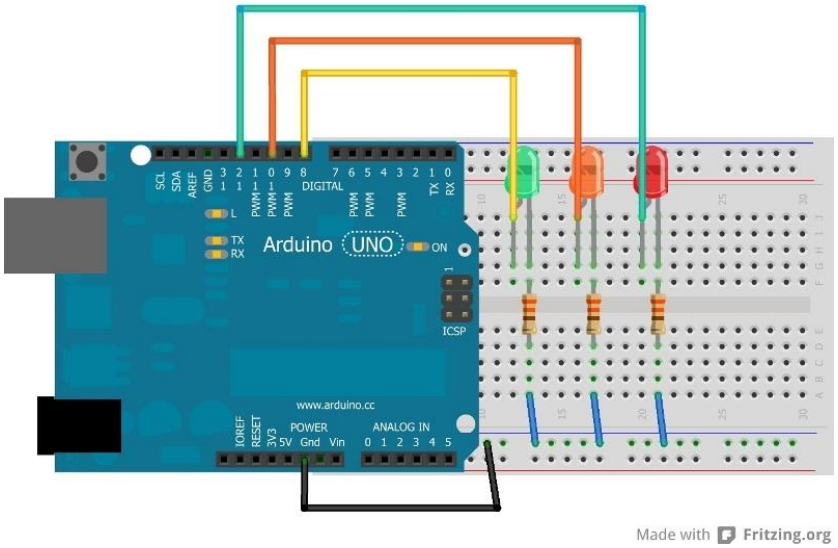
Aplique a Lei de Ohm para calcular o valor do resitor. V = r.i isto é r = V / i

Para uma tensão de 5 Volts, r = 5 / 0,020 = 250 ohms

Para uma tensão de 3,5 Volts, r - 3,5 / 0,020 = 175 ohms

O valor do resistor não precisa ser exatamente estes. Colocando um resistor de, por exemplo 200 ohms, na voltagem de 5 Volts, o LED acenderá mais brilhante e, ao contrário, com um resistor de 300 ohms o LED vai acender mais fraco.

Veja uma montagem com 3 LEDs que piscam sequencialmente, um de cada vez:



Veja o programa:

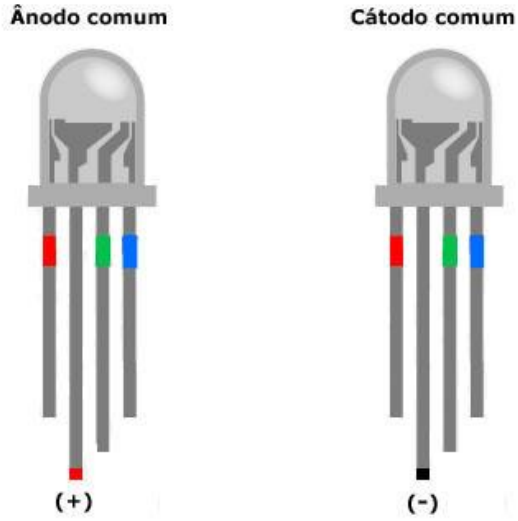
```
int led1 = 8;
int led2 = 10;
int led3 = 12;

void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
  }

void loop() {

  digitalWrite(led1, HIGH);   // acende o LED 1
  delay(100);                 // dá um tempo
  digitalWrite(led1, LOW);    // apaga o LED 1
  delay(100);                 // dá um tempo

  digitalWrite(led2, HIGH);   // acende o LED 2
  delay(100);                 // dá um tempo
  digitalWrite(led2, LOW);    // apaga o LED 2
  delay(100);                 // dá um tempo

  digitalWrite(led3, HIGH);   // acende o LED 3
  delay(100);                 // dá um tempo
  digitalWrite(led3, LOW);    // apaga o LED 3
  delay(100);                 // dá um tempo

  }
```

EXPERIÊNCIA Nº 3 LED R.G.B.

acessar com www.ebanataw.com.br/arduino/expledrgb.htm

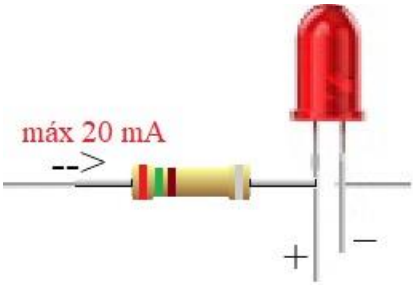
O **LED RGB** é um diodo que emite luz. Diferentemento do LED comum que tem apenas 2 pernas, o LED RGB tem 4 pernas.

Basta fazer passar uma corrente que ele irá emitir uma luiz. A cor da luz vai depende de qual das pernas está passando a corrente elétrica.



A perna mais comprida é a perna comum. As demais acionam a cor vermelha, verde e azul.

CUIDADO: A corrente que passamos pelo LED não pode ser maior que 20 miliamperes, de modo que você deve colocar um resistor em série com ele para limitar a corrente elétrica.



Sendo um diodo, o LED tem polaridade, isto é, uma das pernas é polo positivo e a outra é polo negativo. O polo positivo é a perna mais longa.

O sentido da corrente deve ser do polo positivo para o polo negativo. Caso você ligue o LED ao contrário, isto é, entrando com a corrente pelo polo negativo, o LED não vai acender.

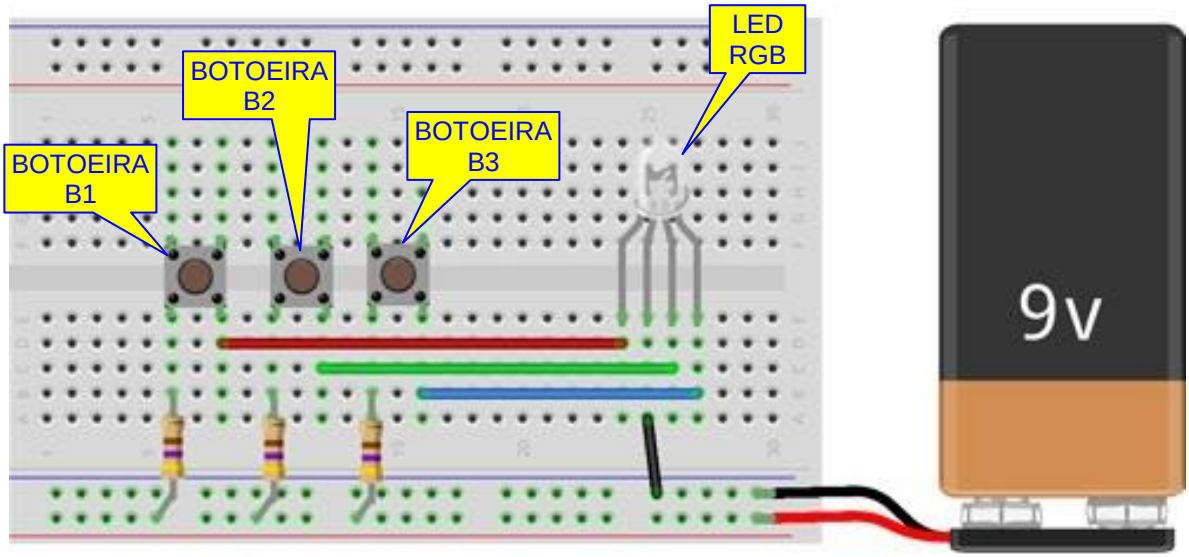
O valor do resistor vai depender do valor da voltagem.

Aplique a Lei de Ohm para calcular o valor do resitor. V = r.i isto é r = V / i

Para uma tensão de 9 Volts, r = 9 / 0,020 = 450 ohms

O valor do resistor não precisa ser exatamente estes. Colocando um resistor de, por exemplo 400 ohms, na voltagem de 9 Volts, o LED acenderá mais brilhante e, ao contrário, com um resistor de 500 ohms o LED vai acender mais fraco.

Veja uma montagem simples que nem precisa do ARDUINO para você ver como é o funcionamento do LED RGB. No caso,empreguei resistores de 470 ohms, 3 botoeiras e uma bateria de 9 Volts.

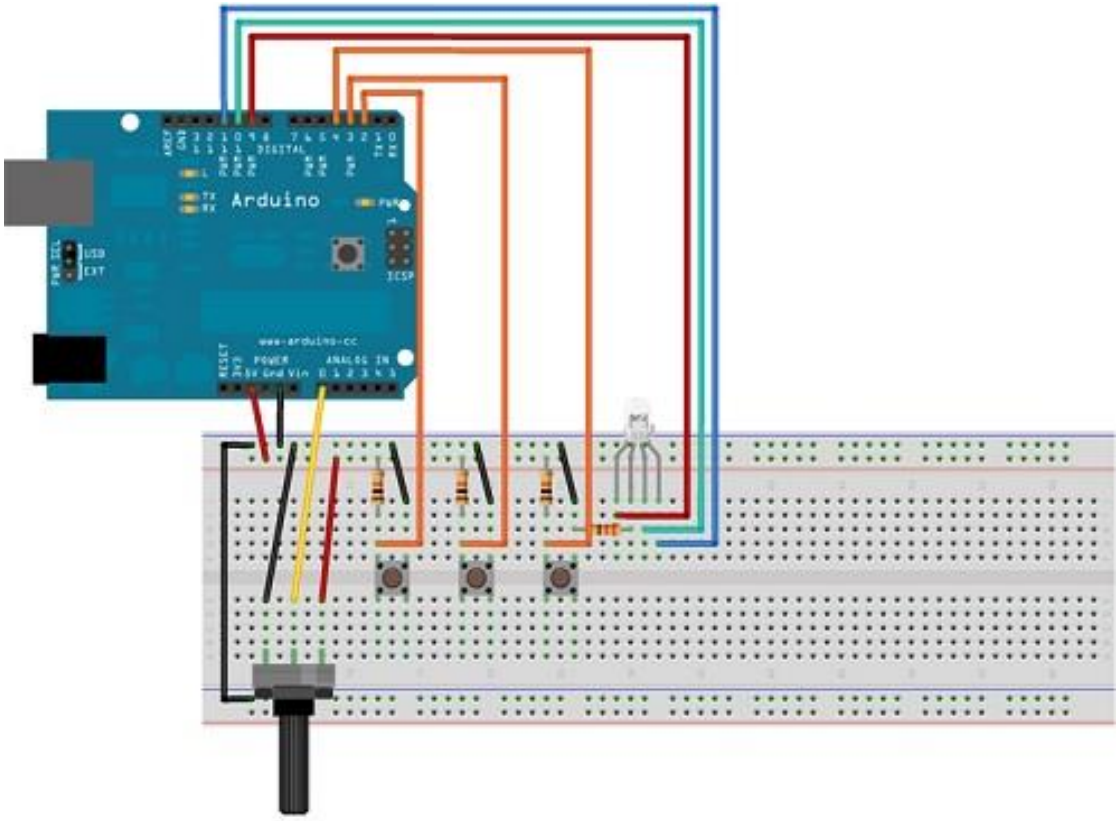


Pressionando individualmente as botoeiras B1, B2 ou B3 o LED vai acender na cor correspondente.

Agora, veja uma montagem, também simples, e que usa o ARDUINO:

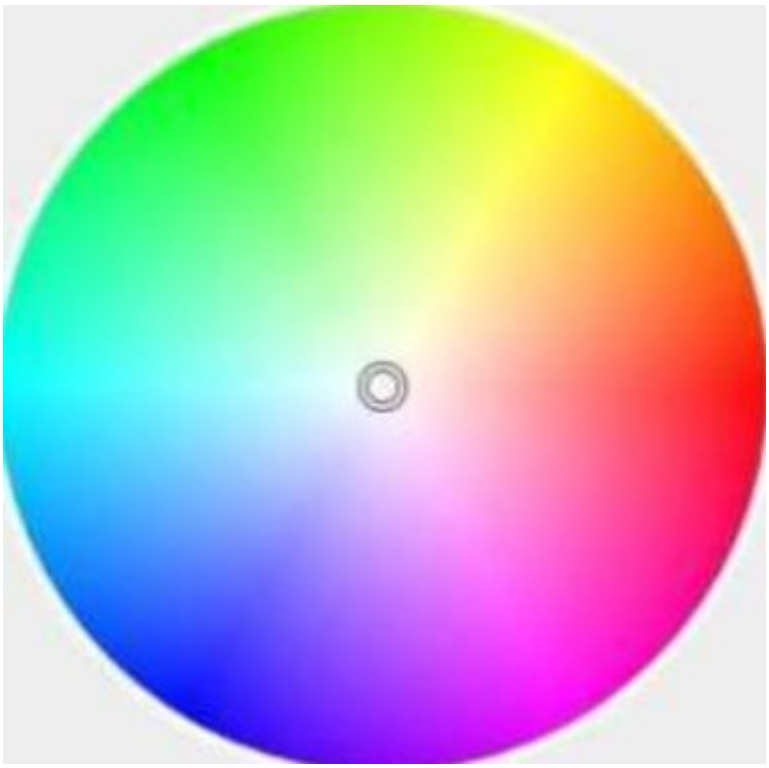
```
#define RED_RGB 9 //Define RED_RGB como 9
#define GREEN_RGB 10 //Define RED_RGB como 10
#define BLUE_RGB 11 //Define RED_RGB como 11
#define pot A0  //Define pot_RED como A0
#define RED_botao 2 //Define RED_botao como 2
#define GREEN_botao 3 //Define GREEN_botao como 3
#define BLUE_botao 4 //Define BLUE_botao como 4

int valor_RED; //Variável para armazenar o valor do vermelho
int valor_GREEN; //Variável para armazenar o valor do verde
int valor_BLUE; //Variável para armazenar o valor do azul
```

```
void setup()
{
pinMode(RED_botao, INPUT_PULLUP); //Configura o pino 2 como entrada e com o
resistor de pullup ativo
pinMode(GREEN_botao, INPUT_PULLUP); //Configura o pino 3 como entrada e com
o resistor de pullup ativo
pinMode(BLUE_botao, INPUT_PULLUP); //Configura o pino 4 como entrada e com o
resistor de pullup ativo
}

void loop()
{
while(!(digitalRead(RED_botao))) //Enquanto o botão do pino 2 estiver pressionado
{
valor_RED = map(analogRead(pot), 0, 1023, 0, 255); //Pega o valor do potenciômetro,
            //converte a escala de 0 a 1023 em 0 a 255, e armazena em
valor_RED
analogWrite(RED_RGB, valor_RED); //Gera o PWM no pino 9 (R - Vermelho), com o
valor convertido                 //do potenciômetro
}
//=====================================
while(!(digitalRead(GREEN_botao))) //Enquanto o botão do pino 3 estiver pressionado
{
valor_GREEN = map(analogRead(pot), 0, 1023, 0, 255); //Pega o valor do
potenciômetro,          //converte a escala de 0 a 1023 em 0 a 255, e armazena em
valor_GREEN

analogWrite(GREEN_RGB, valor_GREEN); //Gera o PWM no pino 10 (G - Verde),
com o valor convertido                  //do potenciômetro
}

while(!(digitalRead(BLUE_botao))) //Enquanto o botão do pino 4 estiver pressionado
{    valor_BLUE = map(analogRead(pot), 0, 1023, 0, 255);; //Pega o valor do
potenciômetro,          //converte a escala de 0 a 1023 em 0 a 255, e armazena em
valor_GREEN

analogWrite(BLUE_RGB, valor_BLUE); //Gera o PWM no pino 11 (B - Azul), com o
valor convertido                 //do potenciômetro
}

}
```



Acionando individualmente cada perna do LED RGB você verá o led acendendo em VERMELHO, VERDE e AZUL, isto é, Red, Green e Blue.

Acionando duas ou as três pernas com tensões que podem variar de 0 a 1023, você vai obter toda a gama de cores conforme o diagrama seguinte, dependendo da combinação de tensões nas pernas do led:



EXPERIÊNCIA Nº 4 Sensor de Luminosidade LDR

acessar com www.ebanataw.com.br/arduino/expLDR.htm

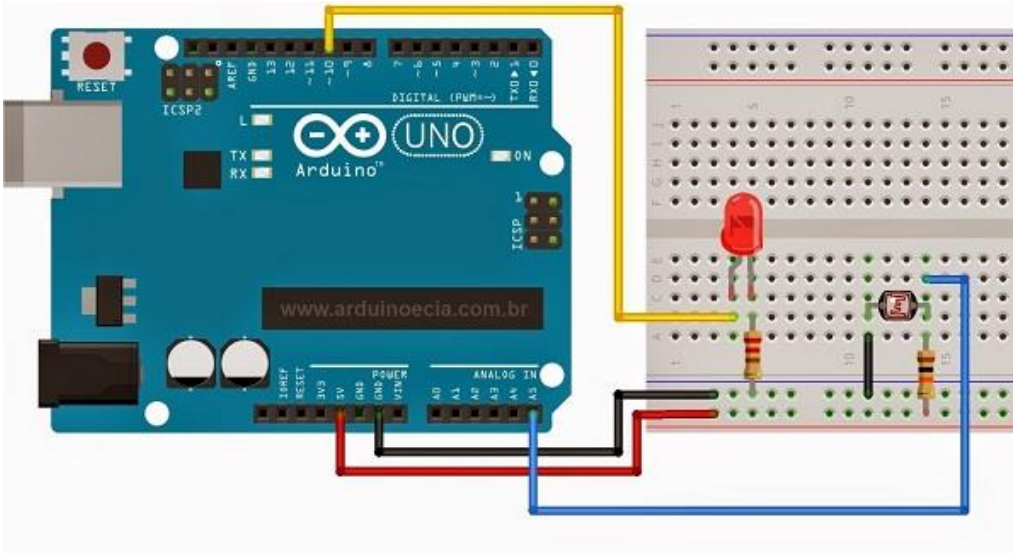
O LDR é um resistor que varia o valor da resistência elétrica em função da claridade. LDR significa Light Dependent Resistor.

Aquele dispositivo que liga a luz ao anoitecer possui como principal componente, um LDR:

Uma aplicação simples: Um sensor LDR exposto no meio ambiente capta a luz e varia seu valor (entre 0 a 1024). O valor será Zero quando o ambiente estiver totalmente escuro e 1024 quando o ambiente estiver totalmente claro.

O programa lê o valor do LDR na *portaLDR* que está conectado na porta analógica A5:

```
int estado = analogReas(portaLDR)
```



O programa para este circuito lê o valor da porta analógica (que deve estar na faixa de 0 a 1024), verificando se o valor é maior do que 800 (LDR encoberto) e consequentemente acendendo o led.

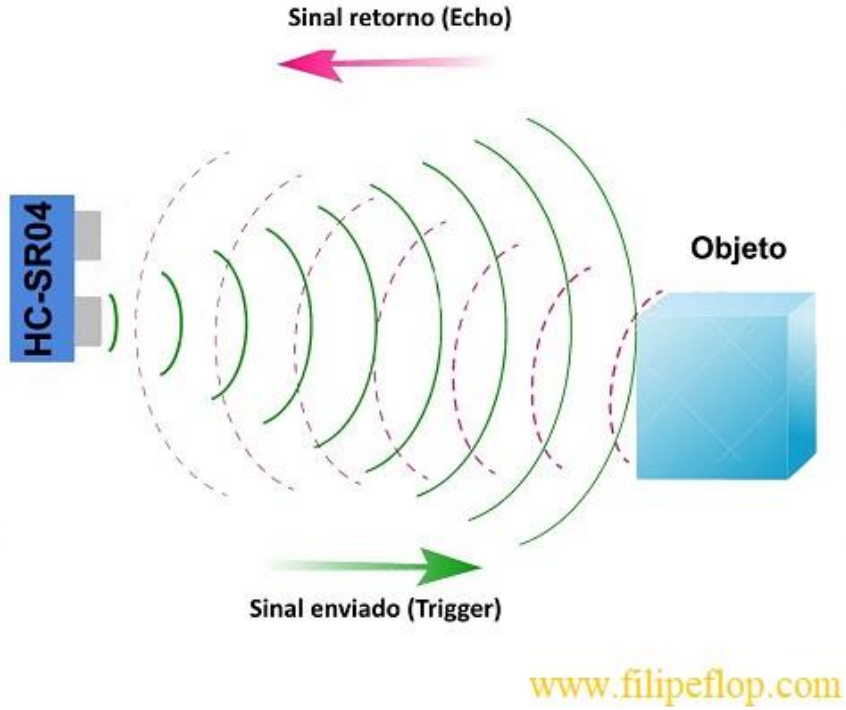
Usei no meu circuito um resistor de 330 ohms para o led e de 10K para o LDR. Caso o seu circuito não funcione adequadamente, ajuste esses valores :

```
// Programa : LDR - Sensor de Iluminação
// Autor : Arduino e Cia

int portaLed = 10; //Porta a ser utilizada para ligar o led
int portaLDR = A5; //Porta analógica utilizada pelo LDR

void setup()
{
 pinMode(portaLed, OUTPUT); //Define a porta do Led como saída
}

void loop()
{
 int estado = analogRead(portaLDR);  //Lê o valor fornecido pelo LDR

 // Caso o valor lido na porta analógica seja maior do que
 // 800, acende o LED
 // Ajuste o valor abaixo de acordo com o seu circuito
 if (estado > 800)
 {
  digitalWrite(portaLed, HIGH);
 }
 else  //Caso contrário, apaga o led
 {
  digitalWrite(portaLed, LOW);
 }
}
```

EXPERIÊNCIA Nº 5 Sensor Ultrassônico HC-SR-04

acessar com www.ebanataw.com.br/arduino/expultrassonico.htm

O sensor HC-SR-04 é feito de duas partes: uma que emite um raio ultrassônico (invisível e inaudível), que propaga pelo ar, e ao encontrar um objeto, reflete nele e volta. Ao voltar, a outra parte, esta sim um sensor, capta o raio refletido e fornece ao ARDUINO o tempo que o raio levou entre sair do emissor e ser captado de volta pelo receptor.



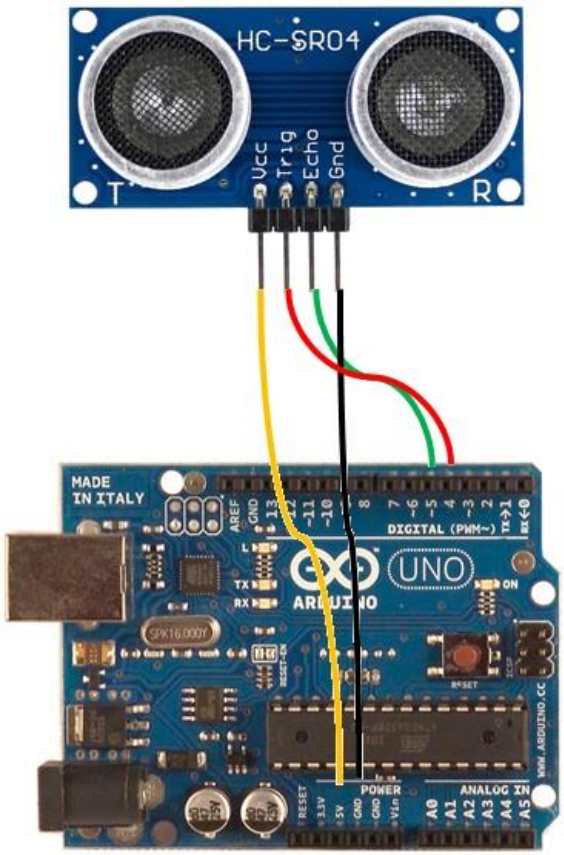
A distância entre o sensor HC-SR-04 e o objeto pode ser calculado pela fórmula:

Distância = [Tempo ECHO em nível alto * Velocidade do Som] / 2

Velocidade (média) do som no ar em CNTP = 340 metros por segundo.

A velocidade do som poder ser considerada igual a 340 m/s, logo o resultado é obtido em metros se considerado o tempo em segundos. Na fórmula a divisão por 2 deve-se ao foto que a onda é enviada e rebatida, logo ela percorre 2 vezes a distância procurada.

Veja uma montagem prática:



Material:

Sensor Ultrassonico HC-SR-04

Alcance de 2 cm a 4 metros com precisão de 3 mm.

Distância = (tempo em Nivel Alto X Velocidade do Som) / 2.

Protoboard

Jumpers

Placa Arduino

Notebook

ALTERNATIVA USANDO A BIBLIOTECA Ultrasonic.h

```
/* ================================
* \watanabe\Demo04-Distancia
*
* Circuito que mede distancias com
* o Sensor de Ultrassom HC-SR-04.
*
* Roberto Massaru Watanabe
* R-0
* R-1 03/02/2018
```

```
* ================================
*/

#include <Ultrasonic.h>

#define PINO_TRG 4
#define PINO_ECHO 5

Ultrasonic Ultrasonic(PINO_TRG, PINO_ECHO);

void setup() {
Serial.begin(9600);
Serial.println("Lendo Dados Recebidos ...");
}

void loop() {

float cmMsec;

long microsec = Ultrasonic.timing();

cmMsec = Ultrasonic.convert(microsec, Ultrasonic::CM);

Serial.print("Distância em cm: ");
Serial.print(cmMsec);

delay(1000);

}
```

ALTERNATIVA NÃO USANDO A BIBLIOTECA Ultrasonic.h

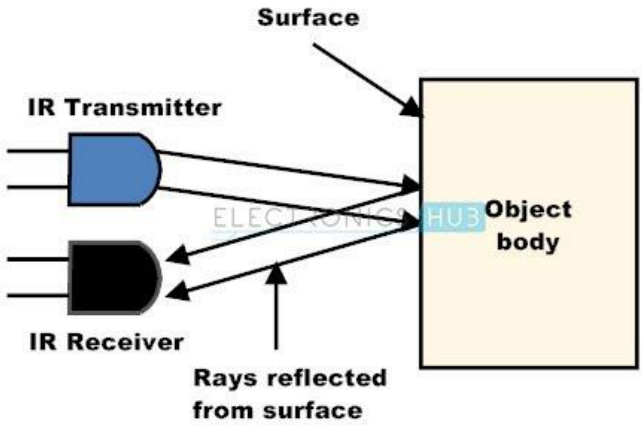
```
/* ================================
* \watanabe\Demo04a-Distancia
*
* Circuito que mede distâncias com
* o Sensor de Ultrassom HC-SR-04.
* Alternavia SEM USAR a Biblioteca Ultrasonic.
*
* Roberto Massaru Watanabe
* R-0
* R-1 04/02/2018
* ================================
*/

const int Gatilho = 4;
const int Receptor = 5;

void setup() {
pinMode(Gatilho, OUTPUT);
pinMode(Receptor, INPUT);

Serial.begin(9600);
```

```
Serial.println("Lendo Dados Recebidos ...");
}

unsigned long ping(){
digitalWrite(Gatilho, HIGH);
delayMicroseconds(10);
digitalWrite(Gatilho, LOW);
return pulseIn(Receptor, HIGH);
}

void loop() {

int range = ping() / 64;
delay(50);

Serial.print("Distância = ");
Serial.print(range);
Serial.print(" cm.\n");
delay(1000);

}
```

EXPERIÊNCIA Nº 6 Sensor Reflexivo TCRT-5000

acessar com www.ebanataw.com.br/arduino/expReflexivo.htm

O sensor TCRT-5000 é conhecido como Ssnsor Reflexivo pois sua operação é caracterizada pela emissão de um raio Infravermelho (lado azul) que, batendo um objeto, reflete de volta e é captado por um fototransistor (lado preto).



O seguinte exercício mostra bem o funcionamento do Sensor Reflexivo:

Os resistores são: de 330 ohms para o led infravermelho e de 10k para o fototransistor.

PROGRAMA:

```
// Programa : Teste sensor óptico reflexivo
// Autor : Arduino e Cia
int objeto = 0;
void setup()
{
  pinMode(7, INPUT); //Pino ligado ao coletor do fototransistor
  Serial.begin(9600);
}

void loop()
{
  objeto = digitalRead(7);
  if (objeto == 0)
  {
    Serial.println("Objeto : Detectado");
  }
  else
  {
    Serial.println("Objeto : Ausente !");
  }
}
```

EXPERIÊNCIA Nº 7 LÊ CONTROLE REMOTO

acessar com www.ebanataw.com.br/arduino/expLeControleRemoto.htm

O sensor VS-1838 é um componente que recebe o raio infravermelho emitido por qualquer Controle Remoto baseado na luz infravermelha.

Os sensores emitem o raio infravermelho na forma "codificada" contendo o código da tecla que foi pressionada no controle. Os aparelhos comandados por controle remoto possuem, internamente, uma tabela de códigos e funções e essa tabela é diferente de um fabricante para outro. É por isso que um controle remoto da TV de uma certa marca não serve para a TV de outra marca.

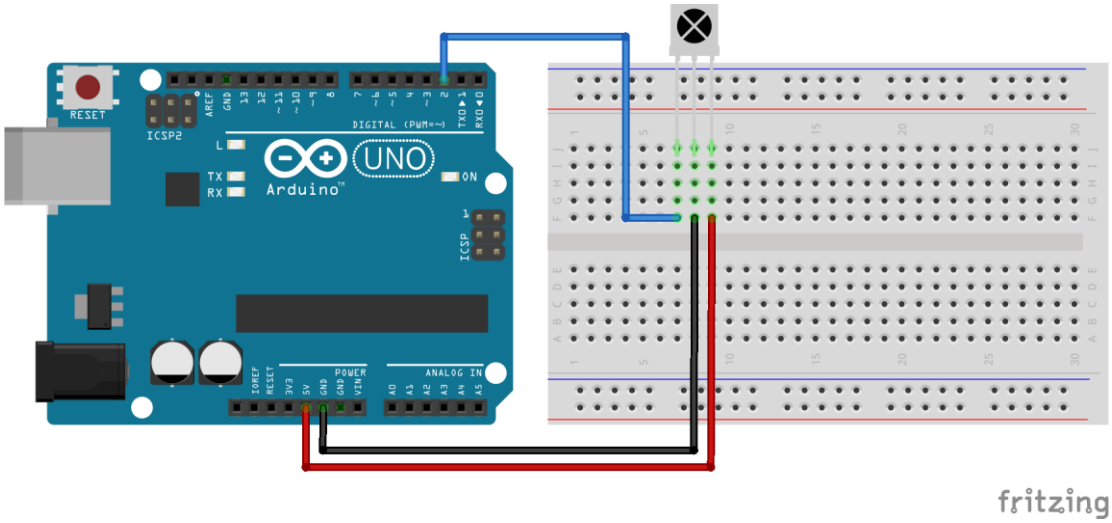
Quando você pressiona uma tecla no controle remoto, ele consulta uma tabela interna e emite um raio (invisível) que é "codificado", isto é, carrega o código da tecla que foi pressionada. O aparelho de TV, ao receber o raio, interpreta o código trazido pelo raio e, consultando a tabela interna, descobre qual é a função desejada pelo usuário.

Nesta experiência você poderá descobrir quais são os códigos de cada

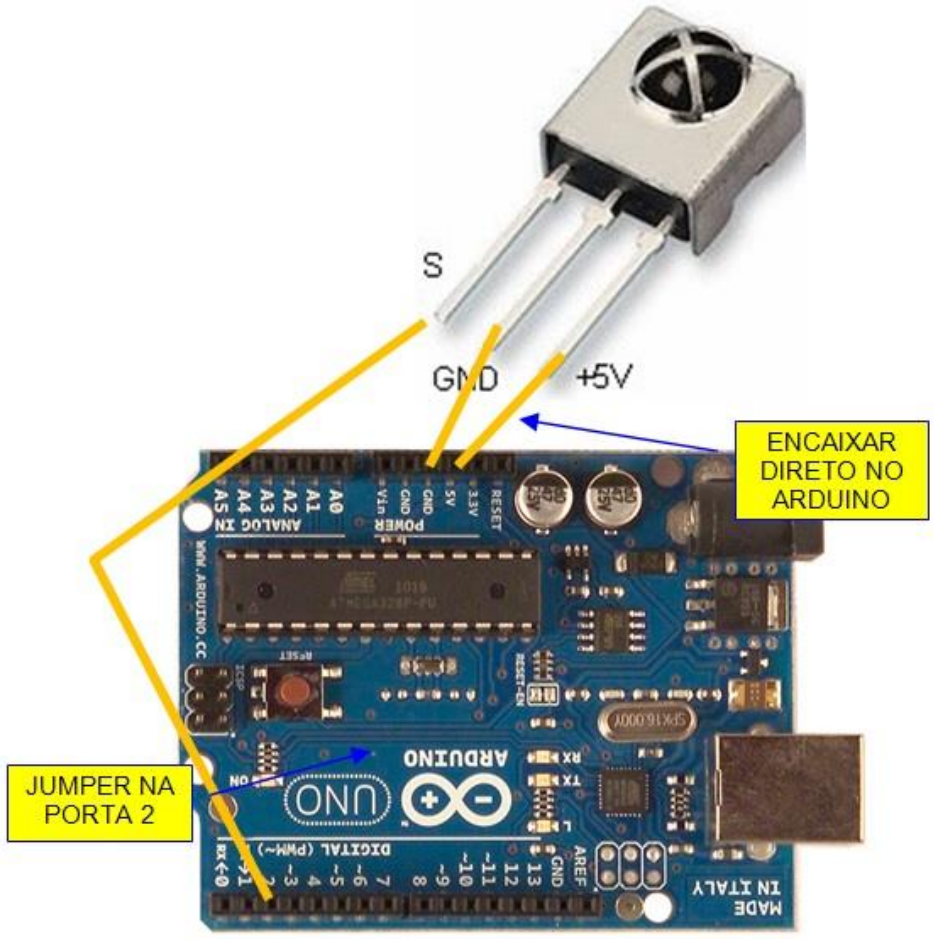
uma das teclas do seu controle remoto.

Para realizar a experiência, você vai precisar ter o ARDUINO, o sensor VS-1838 e um controle remoto qualquer que pode ser o da TV, do CD-playere, do Ar Condicionado ou qualquer outro controle remoto.

Monte o circuito:



Se você não tem a placa Protoboard, então monte direto:



Carregue ou digite a partir da listagem abaixo (com Ctrl-C e Ctrl-V) no

Editor Arduino:

```
 /*==================
* \WATANABE\DEMO-05-LeCR da TV
* Circuito que recebe comando do Controle Remoto
* e mostra no Monitor o código da tecla pressionada no CR
*
* Usa o Sensor VS-1838B com a perna de sinal
* conectado na porta 2 do Arduino
*==========================
* Roberto Massaru Watanabe
* R-0 31/01/2018
*/
#include <IRremote.h>
int RECV_PIN = 2;
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn();
}
void loop(){
  if (irrecv.decode(&results)){
    Serial.println(results.value,HEX);
    irrecv.resume();
    }
}
```

Com o Controle Remoto apontado para o VS-1838, apertando qualquer tecla será mostrado no monitor o código da tecla pressionada.

Imaginem as aplicações que você pode inventar. Ligar e desligar coisas como lâmpadas, TV, rádios, fornos, motores, bombas, etc. tudo de longe e sem sair do lugar. Você tem a alternativa de colocar um sensor VS-1838 em cada aparelho e tem também a alternativa de num único VS-1838 comandar com um Motor Shield todos os aparelhos, definindo e associando cada tecla do controle remoto a cada aparelho.

Vamos colocar a imaginação para trabalhar.

EXPERIÊNCIA Nº 8 Módulo RELÉ com 2 canais

acessar com www.ebanataw.com.br/arduino/modulorele.htm

Este Módulo Relé 5V com 2 canais é a alternativa perfeita pra quem busca um módulo compacto e de qualidade para projetos com Arduino e outros controladores.

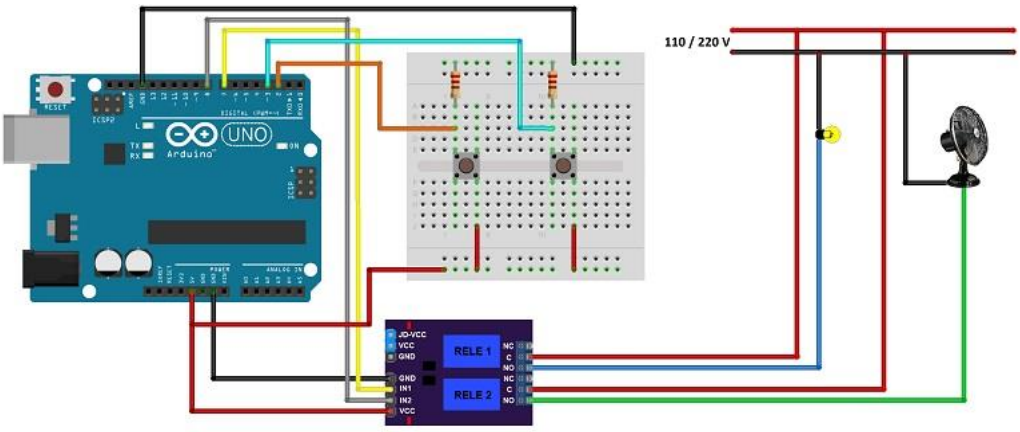
Com este módulo você consegue fazer acionamento de cargas de 200V AC, como lâmpadas, equipamentos eletrônicos, motores, ou usá-lo para fazer um isolamento entre um circuito e outro. Em dúvida de como usar? Confira o blog instrutivo AQUI.

O módulo é equipado com transistores, conectores, leds, diodos e relés de alta qualidade. Cada canal possui um LED para indicar o estado da saída do relé. Se precisar de mais relés 5v, veja o Modulo Relé 5v com 4 Canais.

Especificações:

– Modelo: SRD-05VDC-SL-C (Datasheet)
– Tensão de operação: 5VDC
– Permite controlar cargas de 220V AC
– Corrente típica de operação: 15~20mA
– LED indicador de status
– Pinagem: Normal Aberto, Normal Fechado e Comum
– Tensão de saída: (30 VDC a 10A) ou (250VAC a 10A)
– Furos de 3mm para fixação nas extremidades da placa
– Tempo de resposta: 5~10ms
– Dimensões: 51 x 38 x 20mm
– Peso: 30g

O controle do relé 1 é feito pela porta 7 do Arduino Uno, e o relé 2 é controlado pela porta 8. As duas portas são definidas como saídas e alternam os estados LOW e HIGH, lembrando que o estado baixo (LOW), é que aciona o relé:



O botão da esquerda aciona o relé 1, que por sua vez está ligado à uma lâmpada. Já o botão da direita controla o relé 2, que no nosso exemplo liga e desliga um ventilador, mas pode ser substituído por qualquer aparelho de sua preferência, como uma cafeteira ou um motor , desde que a corrente exigida não ultrapasse 10 A

Utilizamos novamente as portas 7 e 8, e vamos apenas alterar o programa, para que seja feita a leitura dos botões e o respectivo acionamento dos relés. Como estamos utilizando push-buttons, a cada acionamento o estado do relé será invertido, ligando ou desligando o dispositivo.

```
//Programa : Teste Módulo Rele Arduino - Botoes
//Autor : FILIPEFLOP
//Porta ligada ao pino IN1 do modulo
int porta_rele1 = 7;
//Porta ligada ao pino IN2 do modulo
int porta_rele2 = 8;
//Porta ligada ao botao 1
int porta_botao1 = 2;
//Porta ligada ao botao 2
```

```
int porta_botao2 = 3;
//Armazena o estado do rele - 0 (LOW) ou 1 (HIGH)
int estadorele1 = 1;
int estadorele2 = 1;
//Armazena o valor lido dos botoes
int leitura1 = 0;
int leitura2 = 0;

void setup()
{
 //Define pinos para o rele como saida
 pinMode(porta_rele1, OUTPUT);
 pinMode(porta_rele2, OUTPUT);
 //Define pinos dos botoes como entrada
 pinMode(porta_botao1, INPUT);
 pinMode(porta_botao2, INPUT);
 //Estado inicial dos reles - desligados
 digitalWrite(porta_rele1, HIGH);
 digitalWrite(porta_rele2, HIGH);
}
void loop()
{
 //Verifica o acionamento do botao 1
 leitura1 = digitalRead(porta_botao1);
 if (leitura1 != 0)
 {
  while(digitalRead(porta_botao1) != 0)
  {
   delay(100);
  }
  //Inverte o estado da porta
  estadorele1 = !estadorele1;
  //Comandos para o rele 1
  digitalWrite(porta_rele1, estadorele1);
 }

 //Verifica o acionamento do botao 2
 leitura2 = digitalRead(porta_botao2);
 if (leitura2 != 0)
 {
  while(digitalRead(porta_botao2) != 0)
  {
   delay(100);
  }
  //Inverte o estado da porta
  estadorele2 = !estadorele2;
  //Comandos para o rele 2
  digitalWrite(porta_rele2, estadorele2);
 }
}
```

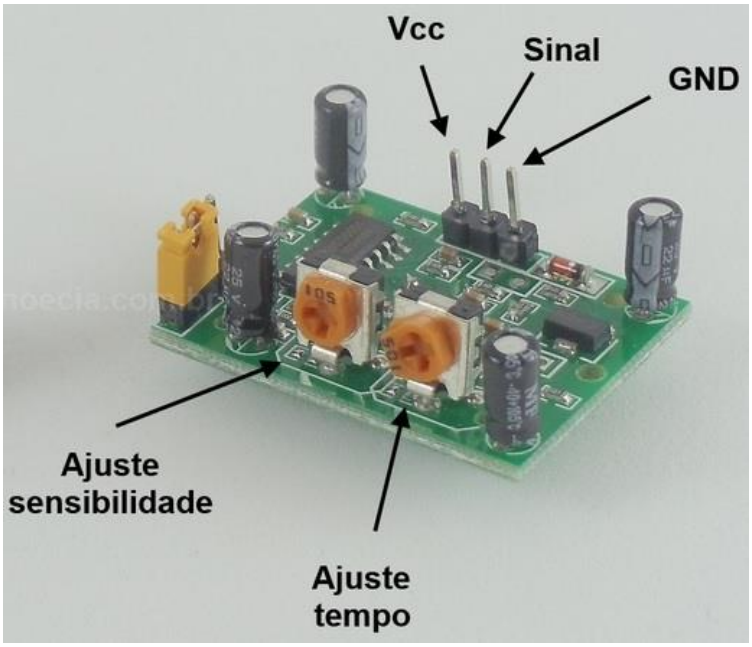
EXPERIÊNCIA Nº 9 SENSOR DE PRESENÇA **P.I.R.**

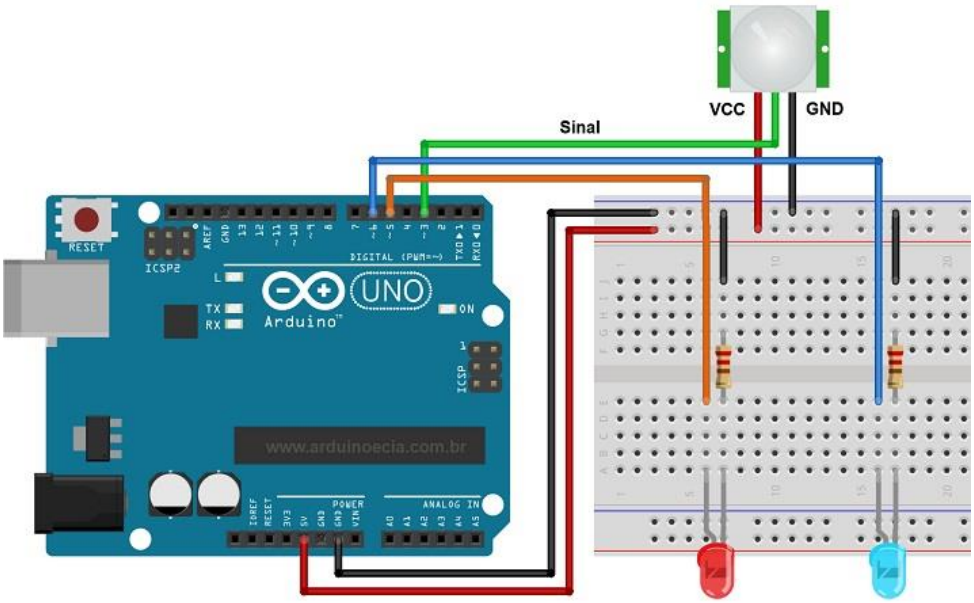
acessar com www.ebanataw.com.br/arduino/expPIR.htm

O sensor de presença tipo PIR (*Passive Infrared Sensor*, ou Sensor Infravermelho Passivo) serve para detectar movimentos no ambiente. Instalado dentro de uma capa protetora, uma *lente fresnel* que melhora a sensibilidade do sensor.

Ele fica o tempo todo "vendo" os objetos e quando ocorre uma variação no sinal infravermelho a saída é acionada por um determinado tempo. A lente fresnel tem a função de "ampliar" o campo de visão do sensor, condensando a luz.

Na parte de baixo encontramos os 2 pinos de alimentação (4,5 a 20 Volts) e sinal, os potenciômetros para ajuste da sensibilidade e tempo de acionamento da saída, e também o jumper que controla o modo de operação do trigger (gatilho).





Atenção quando for fazer as conexões pois alguns módulos apresentam os pinos Vcc e GND invertidos. Na dúvida, consulte o datasheet do módulo ou verifique as indicações na placa.

Nesse módulo, praticamente não há necessidade de programação no Arduino, já que a saída é colocada em HIGH (ALTO), quando um objeto for detectado, e permanece assim pelo tempo que configurarmos no potenciômetro. Basta então definirmos o que será feito com as saídas do Arduino, isto é, se vai acionar algum alarme.

No loop do programa, o valor lido da porta 3 (ligada ao pino de sinal do sensor), é constantemente checado, e caso ocorra movimentação em frente ao sensor, o led vermelho ligado à porta 5 é acionado. Caso contrário, é o led azul ligado à porta 6 que permanece acionado.

```
// Programa : Sensor de presenca com modulo PIR
// Autor : Arduino e Cia

int pinoledverm = 5; //Pino ligado ao led vermelho
int pinoledazul = 6; //Pino ligado ao led azul
int pinopir = 3;  //Pino ligado ao sensor PIR
int acionamento;  //Variavel para guardar valor do sensor

void setup()
{
  pinMode(pinoledverm, OUTPUT); //Define pino como saida
  pinMode(pinoledazul, OUTPUT); //Define pino como saida
  pinMode(pinopir, INPUT);   //Define pino sensor como entrada

}
```

```
void loop()
{
 acionamento = digitalRead(pinopir); //Le o valor do sensor PIR
 if (acionamento == LOW)  //Sem movimento, mantem led azul ligado
 {
   digitalWrite(pinoledverm, LOW);
   digitalWrite(pinoledazul, HIGH);
 }
 else  //Caso seja detectado um movimento, aciona o led vermelho
 {
   digitalWrite(pinoledverm, HIGH);
   digitalWrite(pinoledazul, LOW);
 }
}
```

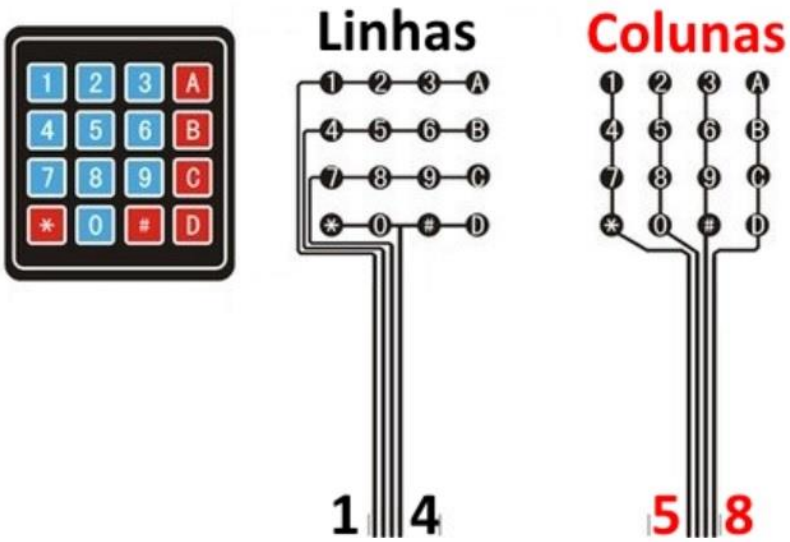
EXPERIÊNCIA Nº 10 TECLADO 4X4

acessar com www.ebanataw.com.br/arduino/expteclado.htm

O Teclado Matricial 4×4 é um componente do *Mundo Arduino* utilizado para entrada de dados.

Possui 16 teclas dispostas em 4 linhas x 4 colunas, e um conector de 8 pinos para ligação com o Arduino.

Internamente são 16 teclas push-buttons tipo membrana dispostos na configuração abaixo em um formato keypad. Conforme a tecla é pressionada, é feita a conexão entre a linha e a coluna correspondentes. Se pressionarmos a tecla A no teclado matricial, será feita a conexão entre os pinos 1 (linha 1) e 8 (coluna 4), se pressionarmos a tecla 7, será feita uma conexão entre os pinos 3 (linha 3) e 5 (coluna 1), e assim por diante:



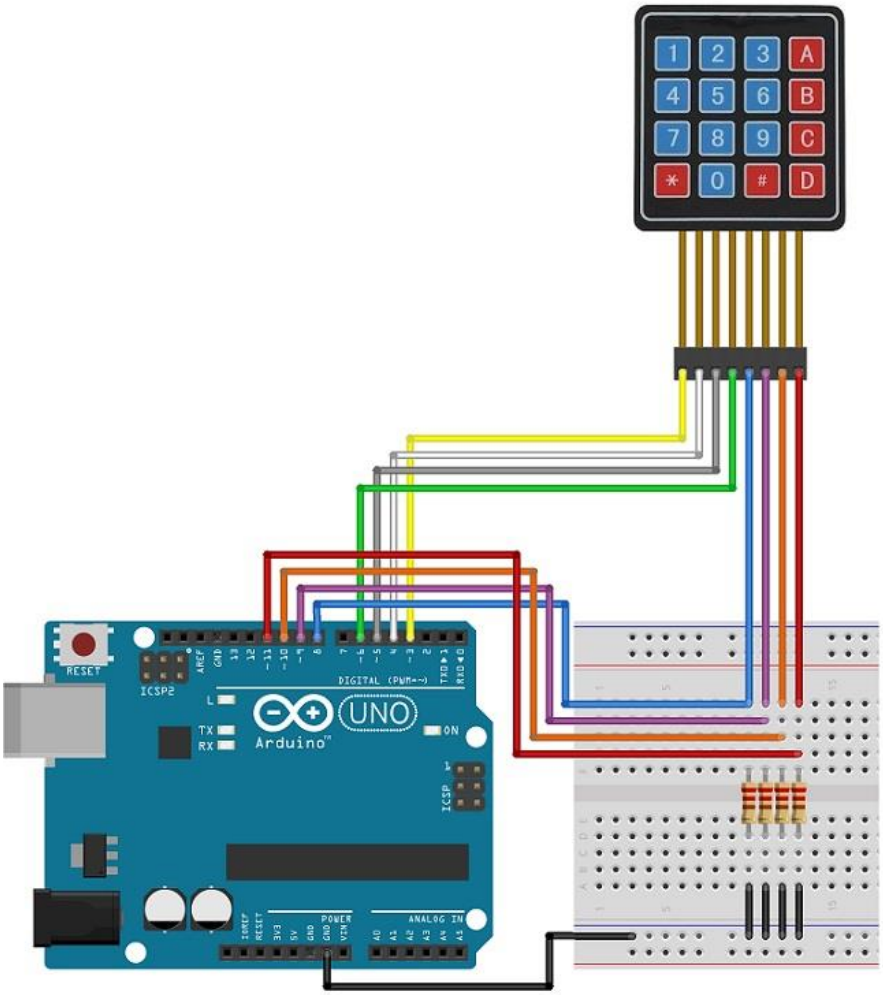
CONEXÕES:

Vamos utilizar 8 portas do Arduino para ligação ao teclado matricial, sendo 4 para as linhas, e 4 para as colunas.

Os pinos das *linhas* deverão ser configurados como OUTPUT (Saída), e os pinos das **colunas** como INPUT (Entrada).

Nos pinos referente às colunas, vamos utilizar 4 resistores pull-down,

mantendo-as em nível baixo quando não houver acionamento das teclas:



PROGRAMA:

No programa, primeiro definimos todos os pinos das *linhas* como saída (pinos 3, 4, 5 e 6), e os pinos das *colunas* como entrada (pinos 8,9,10 e 11).

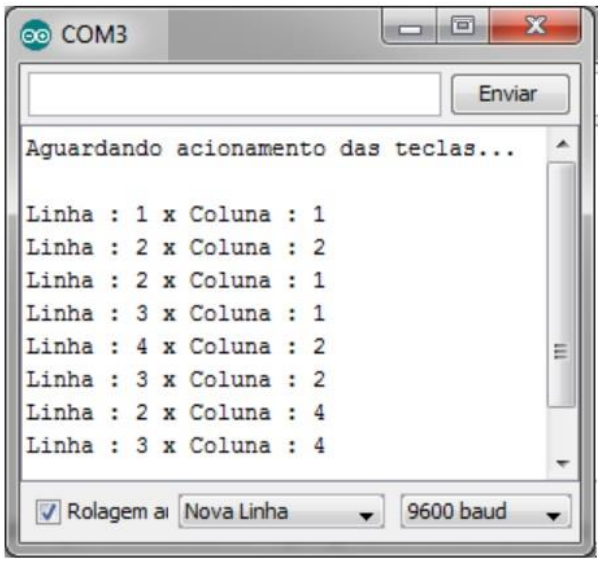
Um loop se encarrega de colocar cada pino de saída (linhas) em estado alto (HIGH), e verificar se alguma tecla foi pressionada, por meio de um comando **IF** para cada coluna. Caso isso aconteça, é gerada uma saída no serial monitor com a informação correspondente à qual tecla foi pressionada no teclado matricial:

```
//Programa : Teste teclado matricial 4x4
//Autor : FILIPEFLOP

void setup()
{
  //Pinos ligados aos pinos 1, 2, 3 e 4 do teclado - Linhas
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
```

```
  //Pinos ligados aos pinos 5, 6, 7 e 8 do teclado - Colunas
  pinMode(8, INPUT);
  pinMode(9, INPUT);
  pinMode(10, INPUT);
  pinMode(11, INPUT);

  Serial.begin(9600);
  Serial.println("Aguardando acionamento das teclas...");
  Serial.println();
}

void loop()
{
   for (int ti = 3; ti<7; ti++)
   {
   //Alterna o estado dos pinos das linhas
   digitalWrite(3, LOW);
   digitalWrite(4, LOW);
   digitalWrite(5, LOW);
   digitalWrite(6, LOW);
   digitalWrite(ti, HIGH);
   //Verifica se alguma tecla da coluna 1 foi pressionada
   if (digitalRead(8) == HIGH)
   {
    imprime_linha_coluna(ti-2, 1);
    while(digitalRead(8) == HIGH){}
   }

   //Verifica se alguma tecla da coluna 2 foi pressionada
   if (digitalRead(9) == HIGH)
   {
    imprime_linha_coluna(ti-2, 2);
    while(digitalRead(9) == HIGH){};
   }

   //Verifica se alguma tecla da coluna 3 foi pressionada
   if (digitalRead(10) == HIGH)
   {
    imprime_linha_coluna(ti-2, 3);
    while(digitalRead(10) == HIGH){}
   }

   //Verifica se alguma tecla da coluna 4 foi pressionada
   if (digitalRead(11) == HIGH)
   {
    imprime_linha_coluna(ti-2, 4);
    while(digitalRead(11) == HIGH){}
   }
  }
  delay(10);
}
```

```
void imprime_linha_coluna(int x, int y)
{
      Serial.print("Linha : ");
      Serial.print(x);
      Serial.print(" x Coluna : ");
      Serial.print(y);
      delay(10);
      Serial.println();
}
```

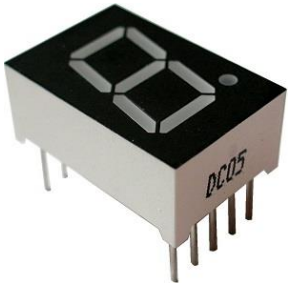
É um programa muito simples e a única coisa que ele faz é mostrar a linha e coluna da tecla acionada. Veja um exemplo:



Para mostrar a letra pressinada, você deve montar uma tabela interna relacionando LinhaXColuna com a letra ou algarismo.

EXPERIÊNCIA Nº 11 DISPLAY de 1 DÍGITO

acessar com www.ebanataw.com.br/arduino/expdisplay1digito.htm



O display de 1 dígito é um componente que empregamos em projetos onde é necessário apresentar informações de forma visual.

Este display possui apenas um dígito, sendo que cada um dos 7 segmentos pode ser acionado de forma independente, formando o caractere desejado.

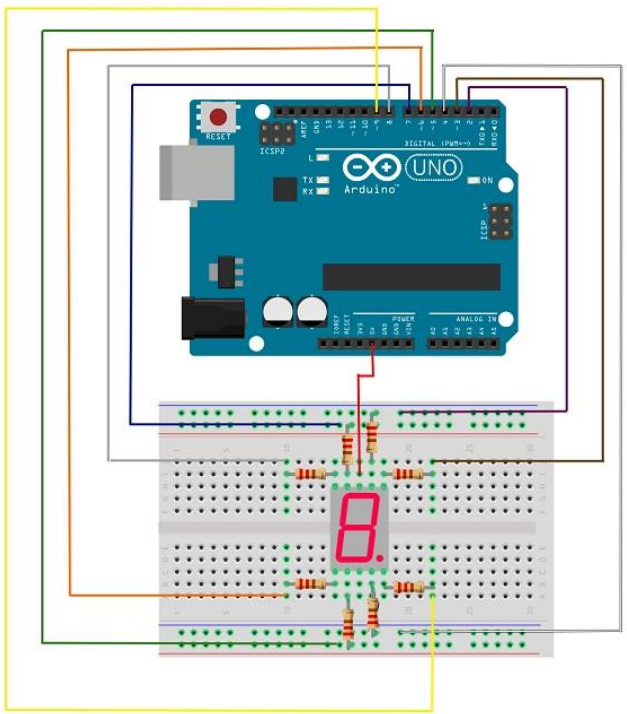
O display de 7 segmentos pode ter a configuração anodo comum ou catodo comum.

Se o display possuir configuração anodo comum, o pino de alimentação deverá ser conectado ao VCC da plataforma embarcada e os demais pinos do display terão configuração catodo comum.

Caso o display tenha configuração catodo comum, o pino de alimentação deverá ser conectado ao GND da plataforma embarcada e os demais pinos do display terão configuração anodo comum. Para esta prática utilizei o display com configuração anodo comum.

CONEXÕES:

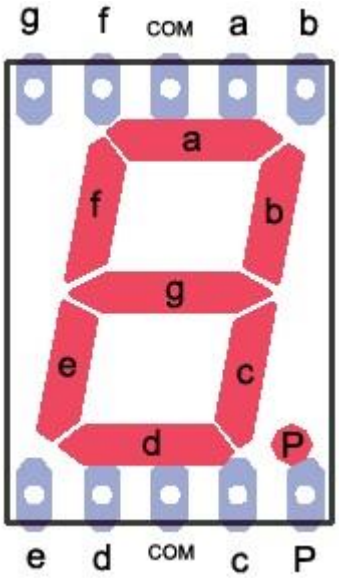
-



PROGRAMA:

```
byte seven_seg_digits[16][7] = { { 1,1,1,1,1,1,0 },  //DIGITO 0
                                 { 0,1,1,0,0,0,0 },  //DIGITO 1
                                 { 1,1,0,1,1,0,1 },  //DIGITO 2
                                 { 1,1,1,1,0,0,1 },  //DIGITO 3
                                 { 0,1,1,0,0,1,1 },  //DIGITO 4
                                 { 1,0,1,1,0,1,1 },  //DIGITO 5
                                 { 1,0,1,1,1,1,1 },  //DIGITO 6
                                 { 1,1,1,0,0,0,0 },  //DIGITO 7
                                 { 1,1,1,1,1,1,1 },  //DIGITO 8
                                 { 1,1,1,0,0,1,1 },  //DIGITO 9
                                 { 1,1,1,0,1,1,1 },  //DIGITO A
                                 { 0,0,1,1,1,1,1 },  //DIGITO B
                                 { 1,0,0,1,1,1,0 },  //DIGITO C
                                 { 0,1,1,1,1,0,1 },  //DIGITO D
                                 { 1,0,0,1,1,1,1 },  //DIGITO E
                                 { 1,0,0,0,1,1,1 }   //DIGITO F
                                 };
void setup(){
  pinMode(2, OUTPUT); //PINO 2 -> SEGMENTO A
  pinMode(3, OUTPUT); //PINO 3 -> SEGMENTO B
  pinMode(4, OUTPUT); //PINO 4 -> SEGMENTO C
  pinMode(5, OUTPUT); //PINO 5 -> SEGMENTO D
  pinMode(6, OUTPUT); //PINO 6 -> SEGMENTO E
  pinMode(7, OUTPUT); //PINO 7 -> SEGMENTO F
  pinMode(8, OUTPUT); //PINO 8 -> SEGMENTO G
  pinMode(9, OUTPUT); //PINO 9 -> SEGMENTO PONTO
  writePonto(0);
}
void writePonto(byte dot){  //FUNÇÃO QUE ACIONA O PONTO NO DISPLAY
  digitalWrite(9, dot);
}
void sevenSegWrite(byte digit){ //FUNCÇÃO QUE ACIONA O DISPLAY
  byte pin = 2;

  for (byte segCount = 0; segCount < 7; ++segCount){ //PARA "segCount" IGUAL
A 0, ENQUANTO "segCount" MENOR QUE 7, INCREMENTA "segCount"
    digitalWrite(pin, seven_seg_digits[digit][segCount]); //PERCORRE O ARRAY
E LIGA OS SEGMENTOS CORRESPONDENTES AO DIGITO
    ++pin; //INCREMENTA "pin"
  }
    writePonto(1); //LIGA O PONTO DO DISPLAY
    delay(100); //INTERVALO DE 100 MILISEGUNDOS
    writePonto(0); //DESLIGA O PONTO DO DISPLAY
}
//MÉTODO RESPONSÁVEL PELA CONTAGEM DE 0 A 9 E CONTAGEM DE
"A" ATÉ "F" (NA CONTAGEM HEXADECIMAL "A"=10 / "B"=11 / "C"=12 /
"D"=13 / "E"=14 / "F"=15)
void loop() {
  for (byte count = 0; count < 16; count++){ //PARA "count" IGUAL A 0,
ENQUANTO "count" MENOR QUE 16, INCREMENTA "count"
```

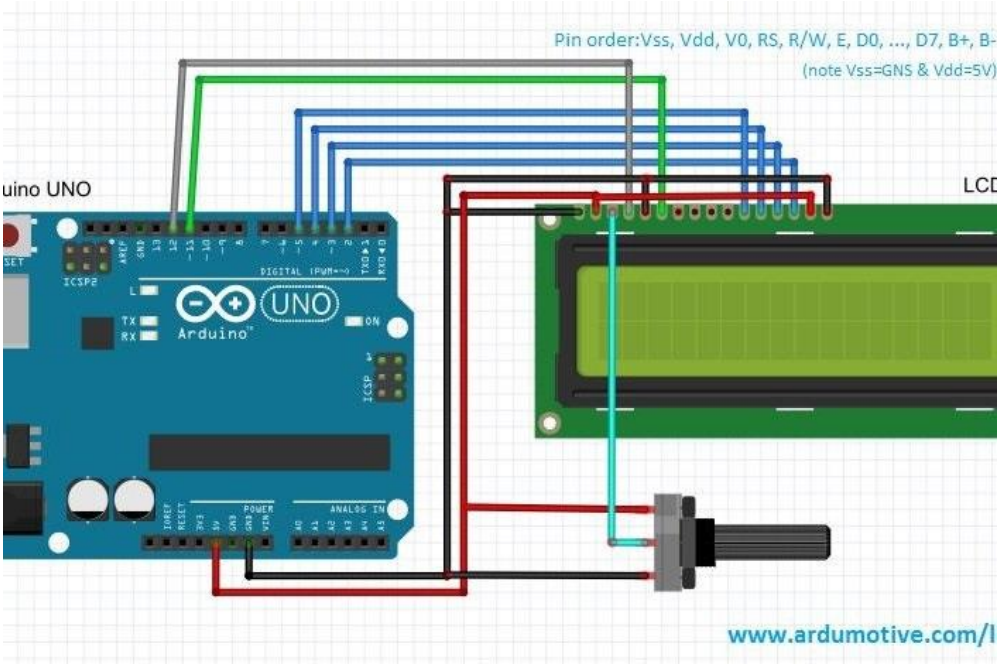
```
    delay(500); //INTERVALO DE 500 MILISEGUNDOS
    sevenSegWrite(count); //FAZ A CONTAGEM
  }
  delay(4000); //INTERVALO DE 4 SEGUNDOS
}
```



EXPERIÊNCIA Nº 12 DISPLAY 32 DÍGITOs

acessar com www.ebanataw.com.br/arduino/expdisplay32digitos.htm

CONEXÕES:



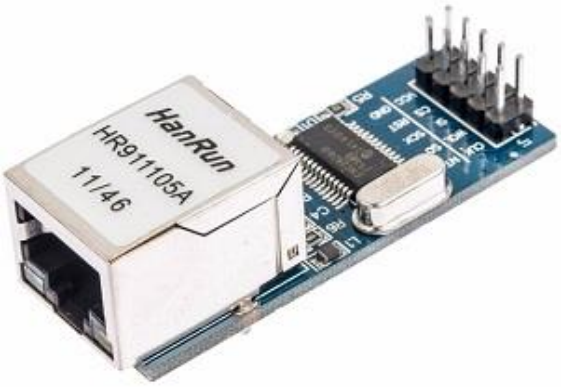
PROGRAMA:

The lcd.begin(16,2) command set up the LCD number of columns and rows. For example, if you have an LCD with 20 columns and 4 rows (20x4) you will have to change this to lcd.begin(20x4).
The lcd.print("--message--") command print a message to first column and row of lcd display. The "message" must have maximum length equal to lcd columns number. For example, for 16 columns display max length is equal with 16 and for 20 columns display max length is equal with 20.
The lcd.setCursor(0,1) command will set cursor to first column of second row. If you have an LCD 20x4 and you want to print a message to column five and third row you have to use: lcd.setCursor(4,2).

```
/* Arduino Tutorial: Learn how to use an LCD 16x2 screen
More info: http://www.ardumotive.com/how-to-use-an-lcd-dislpay-en.html */

//Include LCD library
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
// set up the LCD's number of columns and rows:
lcd.begin(16, 2);
// Print a message to the LCD.
lcd.print("Hello World!");
}

void loop() {
// set the cursor to column 0, line 1
// (note: line 1 is the second row, since counting begins with 0):
lcd.setCursor(0, 1);
//Print a message to second line of LCD
lcd.print("Codebender");
}
```

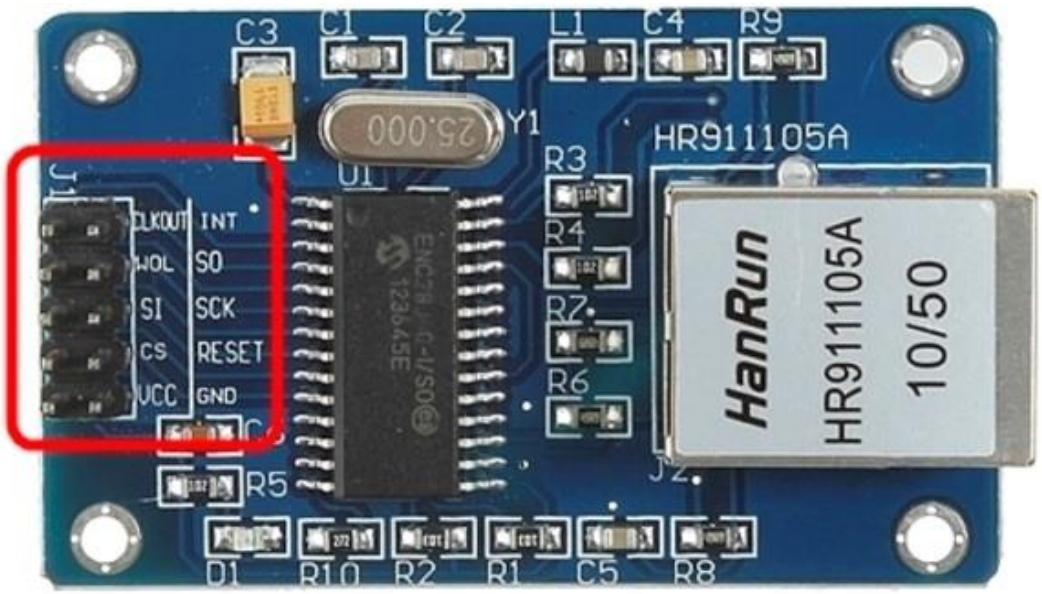
EXPERIÊNCIA Nº 13 SENSOR INTERNET

acessar com www.ebanataw.com.br/arduino/expinternet.htm



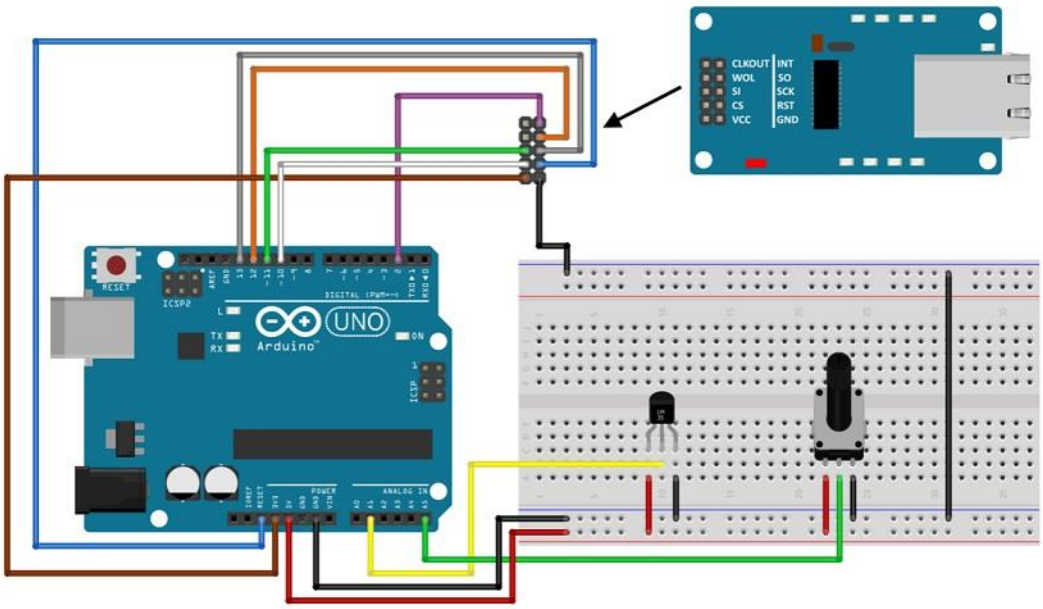
O Módulo Ethernet ENC28J60, é um sensor que pemite ao Arduino comunicar-se via INTERNET.

CONEXÕES:

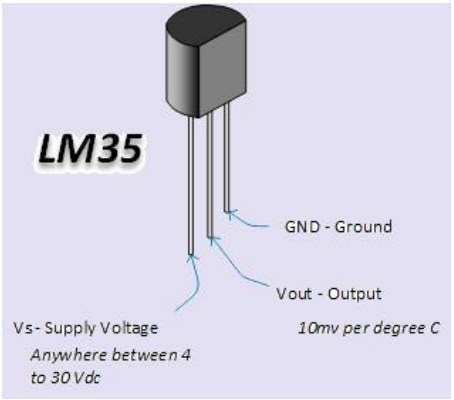


A comunicação do módulo sensor Internet com o Arduino é feita através de 10 pinos que têm os seguintes significados:

	MÓDULO INTERNET	ARDUINO
1	CLKOUT	não usado
2	WOL	não usado
3	SI	D11
4	CS	D10
5	VCC	3V3
6	INT	D2
7	SO	D12
8	SCK	D13
9	RESET	RESET
10	GND	GND



Como componente de entrada da montagem, vamo empregar um Sensor de Temperatura LM-35. Mais detalhes sobre o funcionamento deste sensor de temperatura em www.ebanataw.com.br/arduino/catalogosensores.htm).



PROGRAMA:

Específico para o módulo sensor Ethernet ENC28J60.

Para enviar os dados do sensor e do potenciômetro para o Arduino, vamos utilizar duas bibliotecas, a ETHER_28J60 (download ETHER_28J60) e a etherShield (download ethershield). Descompacte os dois arquivos e coloque as pastas correspondentes dentro da pasta LIBRARIES da IDE do seu Arduino.

Essas bibliotecas são bem práticas pois permitem a criação de páginas html utilizando poucos comandos, como você pode ver no código abaixo.

A linha com o comando static uint8_t ip[4] = {192, 168, 1, 199} é a responsável por determinar o endereço IP da placa na sua rede. Adapte essa linha às especificações da sua rede local e conecte o módulo ao seu roteador ou hub utilizando um cabo de rede com conector padrão RJ45.

```
//Programa : Modulo ethernet ENC28J60 com LM35 e potenciometro
//Autor : FILIPEFLOP
#include "etherShield.h"
```

```
#include "ETHER_28J60.h"
//Define o MAC da placa de rede. Nao eh necessario alterar
static uint8_t mac[6] = {0x54, 0x55, 0x58, 0x10, 0x00, 0x24};
//Define o endereco IP da sua placa
static uint8_t ip[4] = {192, 168, 1, 199};
static uint16_t port = 80;
ETHER_28J60 e;
int pin = 1;       // Pino analogico para ligacao do LM35
int tempc = 0;  // Variavel que armazena a temperatura em Celsius

// Variáveis para temperatura máxima e mínima:
int maxtemp = -100,mintemp = 100;
int i;

void setup()
   {
    //Inicializa a placa com as configuracoes fornecidas
    e.setup(mac, ip, port);
   }
void loop()
{
 //Calcula o valor da temperatura:
 tempc = ( 5.0 * analogRead(pin) * 100.0) / 1024.0;

 //Armazena a temperatura máxima na variavel maxtemp:
 if(tempc > maxtemp) {maxtemp = tempc;}
 //Armazena a temperatura máxima na variavel mintemp

 if(tempc < mintemp) {mintemp = tempc;
}
    if (e.serviceRequest())
    {
      e.print("<H1>FILIPEFLOP - Teste ENC28J60</H1><br/>");
      e.print("<b>Valor potenciometro porta analogica 5 : ");
      e.print(analogRead(A5));
      e.print("<br/><br/>");
      e.print("Valor temperatura : ");
      e.print(tempc);
      e.print("&deg<br/><br/>");
      e.print("Temperatura minima : ");
      e.print(mintemp);
      e.print("&deg<br/>");
      e.print("Temperatura maxima : ");
      e.print(maxtemp);
      e.print("&deg<b/>");
      e.respond();
     }
 delay(100);
}
```

Teste o programa abrindo um browser e digitando na barra de endereços o **IP** que você configurou para a sua placa. Será aberta uma tela parecida com essa:

EXPERIÊNCIA Nº 14 SENSOR BIOMÉTRICO DE IMPRESSÃO DIGITAL

Acessar com : www.ebanataw.com.br/arduino/expbiometrico.htm

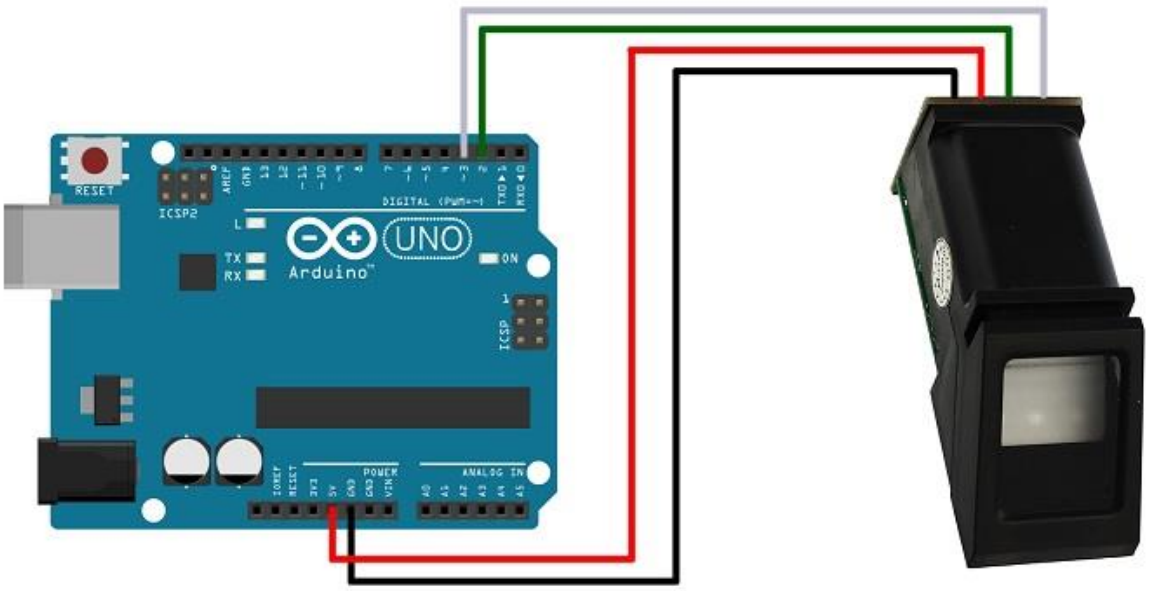
O sensor de impressão digital é um sensor óptico de impressão digital, que "lê" a impressão do dedo, compara com as impressões guardadas na sua memória e indica para o Arduíno se a impressão lida é igual a uma das impressões guardadas na memória.

A capacidade de armazenamento na memória é de cerca de 120 impressões.

Um LED acende indicando o estado de leitura.

Constituído por seis fios, o Leitor Biométrico utiliza somente quatro destes para realizar o seu funcionamento, no qual temos um fio que corresponde ao GND, VCC, TX e RX, observe a relação de cores:

- Preto: GND;
- Branco: RX;
- Verde: TX;
- Vermelho: VCC.



O funcionamento do sensor Biométrico é feito em 2 etapas, uma para o cadastramento das impressões e outro para a identificação.

Antes de darmos início à programação do Leitor Biométrico, entre no link abaixo e realize o Download da biblioteca necessária para o desenvolvimento da montagem.

LINK PARA DOWNLOAD:
(http://blog.usinainfo.com.br/wp-content/uploads/2016/07/Sensor-Biométrico.zip)

Feito o download, descompacte a pasta "Sensor Biométrico" na pasta exemplos do Arduino e a pasta "Sensor Biométrico Library-master" na pasta Library do Arduino.

1° PASSO

Neste primeiro momento vamos realizar a conexão do nosso equipamento junto ao Arduino para realizar a leitura e o cadastramento das digitais, processo o qual será melhor detalhado no transcorrer do projeto em um passo-a-passo ilustrativo e de fácil compreensão.

OBS: O pino Amarela e Azul contido no cabo do produto não será utilizado em nosso projeto.

PROGRAMA-1 ENROL.INO:

```
// Programa para teste de funcionamento do Leitor Biometrico
// Traduzido e adaptado por Usinainfo

#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>

uint8_t id;

uint8_t getFingerprintEnroll();

SoftwareSerial mySerial(2, 3); // Pino 2 como entrada do sensor (fio Verde)
                               // Pino 3 como saída do sendor (fio Branco)
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

void setup()  {
  Serial.begin(9600);
  Serial.println("Procurando Leitor Biometrico ...");

  finger.begin(57600);   // Define a taxa de dados para a porta serial do sensor

  if (finger.verifyPassword()) {
    Serial.println("Leitor Biometrico encontrado");
  }
  else {
    Serial.println("Leitor Biometrico nao encontrado");
    while (1);
  }
}
```

```
uint8_t readnumber(void) {
  uint8_t num = 0;
  boolean validnum = false;
  while (1) {
    while (! Serial.available());
    char c = Serial.read();
    if (isdigit(c)) {
      num *= 10;
      num += c - '0';
      validnum = true;
    } else if (validnum) {
      return num;
    }
  }
}

void loop()   {
  Serial.println("Pronto para Cadastrar Dados! Indique o ID# o qual deve ser
salvo");
  id = readnumber();
  Serial.print("Inscrever ID#");
  Serial.println(id);

  while (!  getFingerprintEnroll() );
}

uint8_t getFingerprintEnroll() {

  int p = -1;
  Serial.print("Esperando digital para inscrever #");
  Serial.println(id);
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
      case FINGERPRINT_OK:
        Serial.println("Imagem Capturada");
        break;
      case FINGERPRINT_NOFINGER:
        Serial.println(".");
        break;
      case FINGERPRINT_PACKETRECIEVEERR:
        Serial.println("Erro ao se Comunicar");
        break;
      case FINGERPRINT_IMAGEFAIL:
        Serial.println("Erro ao Capturar Imagem");
        break;
      default:
        Serial.println("Erro Desconhecido");
        break;
    }
  }
```

```
// OK success!

p = finger.image2Tz(1);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Imagem convertida");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Imagem muito Confusa");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Erro ao se Comunicar");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Impossivel encontrar caracteristicas da digital");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Impossível encontrar características da digital");
    return p;
  default:
    Serial.println("Erro Desconhecido");
    return p;
}

Serial.println("Retire o dedo");
delay(2000);
p = 0;
while (p != FINGERPRINT_NOFINGER) {
  p = finger.getImage();
}
Serial.print("ID ");
Serial.println(id);
p = -1;
Serial.println("Coloque o mesmo dedo novamente");
while (p != FINGERPRINT_OK) {
  p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Imagem Capturada");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.print(".");
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Erro ao se Comunicar");
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Erro ao Capturar Imagem");
      break;
    default:
      Serial.println("Erro Desconhecido");
```

```
      break;
    }
  }

  // OK success!

  p = finger.image2Tz(2);
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Imagem Convertida");
      break;
    case FINGERPRINT_IMAGEMESS:
      Serial.println("Imagem muito Confusa");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Erro ao se comunicar");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Nao foi possível encontrar características da impressao
digital");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      Serial.println("Nao foi possível encontrar características da impressao
digital");
      return p;
    default:
      Serial.println("Erro Desconhecido");
      return p;
  }

  // OK converted!
  Serial.print("Criando cadastro para #");
  Serial.println(id);

  p = finger.createModel();
  if (p == FINGERPRINT_OK) {
    Serial.println("Digital Combinada");
  }
  else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Erro ao se Comunicar");
    return p;
  }
  else if (p == FINGERPRINT_ENROLLMISMATCH) {
    Serial.println("Digital nao corresponde");
    return p;
  }
  else {
    Serial.println("Erro Desconhecido");
    return p;
  }

  Serial.print("ID ");
```

```
  Serial.println(id);
  p = finger.storeModel(id);
  if (p == FINGERPRINT_OK) {
    Serial.println("Armazenado!");
  }
  else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Erro ao se Comunicar");
    return p;
  }
  else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("Impossível Armazenar Dados");
    return p;
  }
  else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Erro ao salvar na memoria");
    return p;
  }
  else {
    Serial.println("Erro Desconhecido");
    return p;
  }
}
```

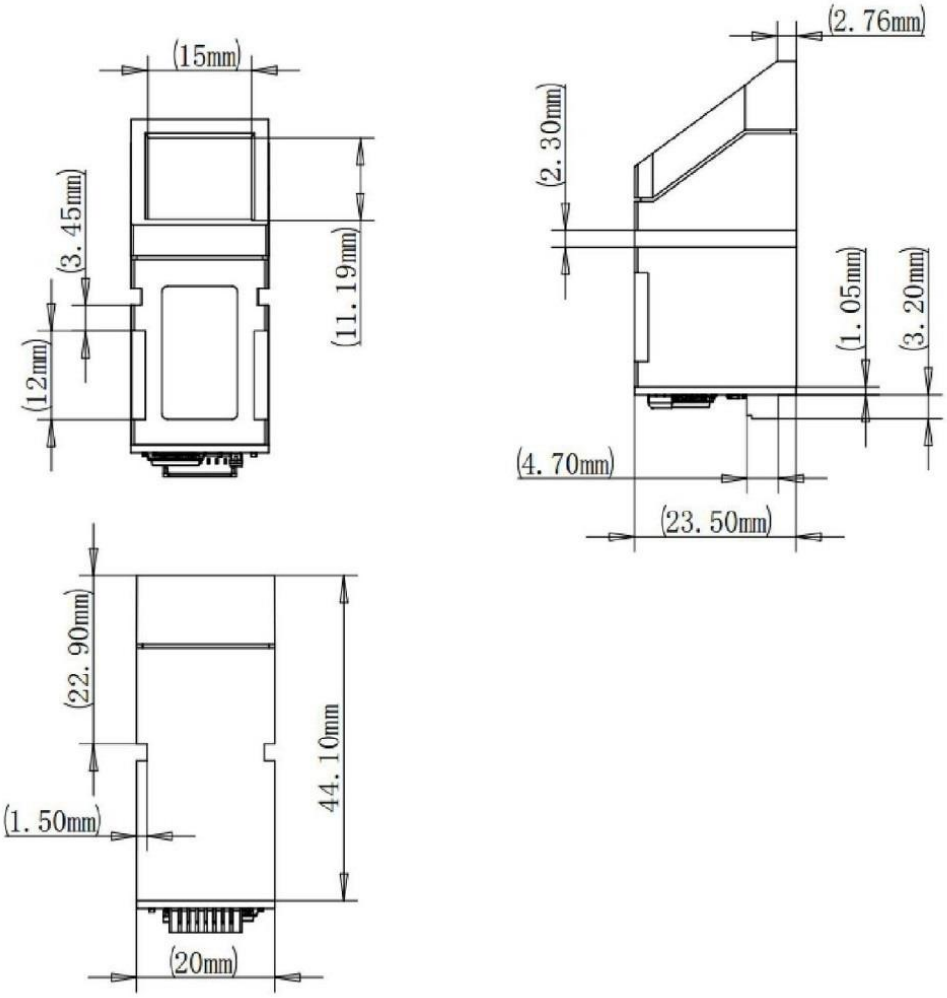
PROGRAMA-2 FINGERPRINT.INO:

```
// Programa para teste de funcionamento do Leitor Biometrico
// Traduzido e adaptado por Usinainfo

#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>

int getFingerprintIDez();

SoftwareSerial mySerial(2, 3);
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

void setup()  {
  Serial.begin(9600);
  Serial.println("Iniciando Leitor Biometrico");
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);

  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Leitor Biometrico Encontrado");
  } else {
    Serial.println("Leitor Biometrico nao encontrada");
    while (1);
  }
  Serial.println("Esperando Dedo para Verificar");
}
```

```
void loop()
{
  getFingerprintIDez();
  digitalWrite(12, HIGH);
  delay(50);
}

uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Imagem Capturada");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println("Dedo nao Localizado");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Erro ao se comunicar");
      return p;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Erro ao Capturar");
      return p;
    default:
      Serial.println("Erro desconhecido");
      return p;
  }

  p = finger.image2Tz();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Imagem Convertida");
      break;
    case FINGERPRINT_IMAGEMESS:
      Serial.println("Imagem muito confusa");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Erro ao se comunicar");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Impossivel localizar Digital");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      Serial.println("Impossivel Localizar Digital");
      return p;
    default:
      Serial.println("Erro Desconhecido");
      return p;
  }

  p = finger.fingerFastSearch();
  if (p == FINGERPRINT_OK) {
```

```
  Serial.println("Digital Encontrada");
 } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Erro ao se comunicar");
  return p;
 } else if (p == FINGERPRINT_NOTFOUND) {
  Serial.println("Digital Desconhecida");
  return p;
 } else {
  Serial.println("Erro Desconhecido");
  return p;
 }

 Serial.print("ID # Encontrado");
 Serial.print(finger.fingerID);
 Serial.print(" com precisao de ");
 Serial.println(finger.confidence);
}

int getFingerprintIDez() {
 uint8_t p = finger.getImage();
 if (p != FINGERPRINT_OK)  return -1;

 p = finger.image2Tz();
 if (p != FINGERPRINT_OK)  return -1;

 p = finger.fingerFastSearch();
 if (p != FINGERPRINT_OK)  return -1;

 digitalWrite(12, LOW);
 digitalWrite(11, HIGH);
 delay(1000);
 digitalWrite(11, LOW);
 delay(1000);
 digitalWrite(12, HIGH);
 Serial.print("ID # Encontrado");
 Serial.print(finger.fingerID);
 Serial.print(" com precisao de ");
 Serial.println(finger.confidence);
 return finger.fingerID;
}
```

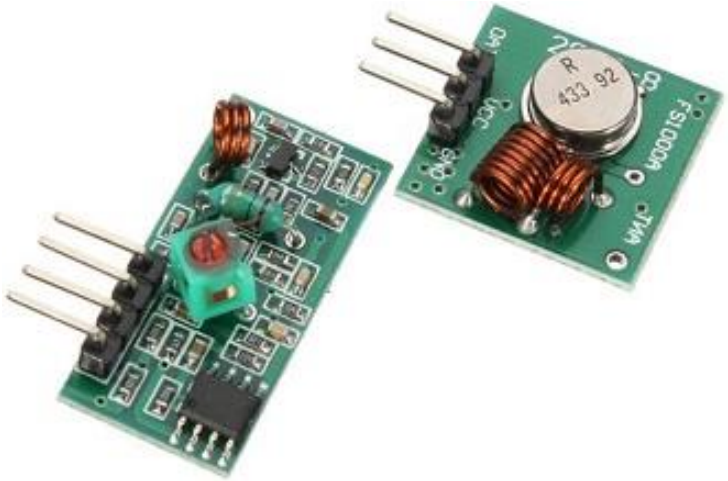
DIMENSÕES DO SENSOR PARA VOCÊ FAZER O SEU PROJETO:



Veja dois exemplos práticos de porta cuja fechadura é controlada por um Sensor Biométrico de Impressão Digital.

EXPERIÊNCIA Nº 15 RECEPTOR E TRANSMISSOR DE RF

acessar com www.ebanataw.com.br/arduino/expRF.htm



Componentes básicos para comunicação via ***RF*** (rádio frequência), está presente em sistemas de alarmes, controle remoto, aquisição de dados e robótica em geral.

Os módulos **RF-433 mHz** alcançam até 200 metros sem obstáculos com modulação AM e frequência de trabalho de 433mHz.

Pode também ser usado para comunicação entre Arduinos, enviando sinais de um Arduino para outro. O módulo transmissor funciona com alimentação entre 3,5Volts e 12Volts.

####### ESPECIFICAÇÕES TRANSMISSOR #####

Modelo: MX-FS-03V
Alcance: 20-200 metros (conforme voltagem)
Tensão de operação: 3,5 a 12Volts
Modo de operação: AM (Modulação em Amplitude)
Taxa de transferência: 4K BITS/segundo
Potência de transmissão: 10 mW
Frequência de transmissão: 433 mHz
Pinagem: Dados-VCC-GND (Esq.->Dir.)
Dimensões: 19 x 19mm

ESPECIFICAÇÕES RECEPTOR

Modelo: MX-05V
Tensão de operação: 5 Volts DC
Corrente de operação: 4 mA
Frequência de recepção: 433 mHz
Sensibilidade: -105 dB
Dimensões: 30 x 14 x 7 mm

MATERIAIS NECESSÁRIOS PARA A MONTAGEM:

2 placas Arduino Uno

1 Módulo RF 433 MHz Transmissor

1 Módulo RF 433 MHz Receptor

1 Push button

1 Led

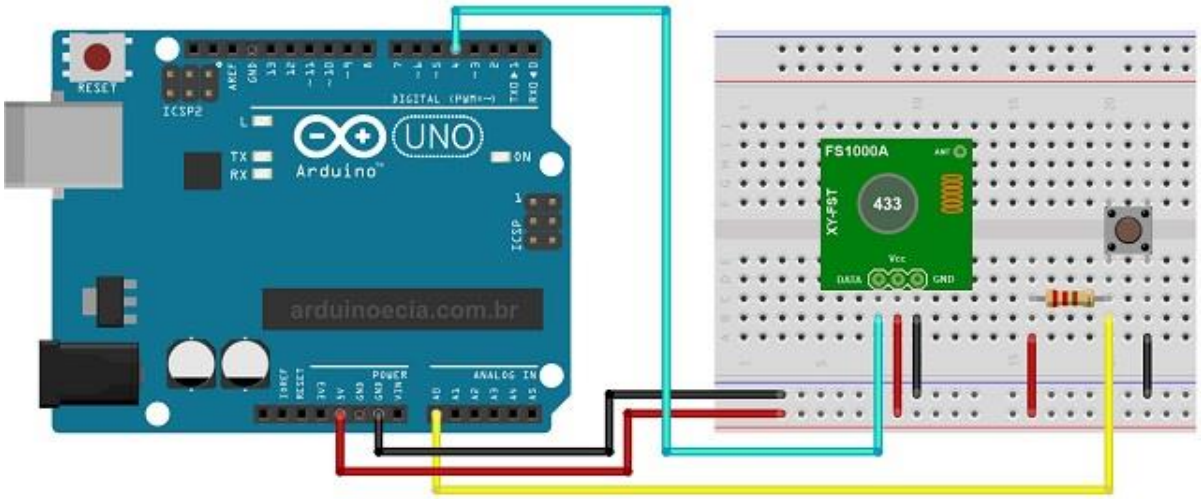
1 resistor de 220 ohms (resistor pull-up do push-button)

1 resistor de 470 ohms (para o led)

Circuito Arduino Uno - Transmissor

No circuito transmissor o pino DATA do módulo RF será conectado ao pino digital 4 do Arduino. O push-button vai no pino analógico A0, e a alimentação do circuito é de 5 Volts:

RF 433 MHz - Circuito Transmissor

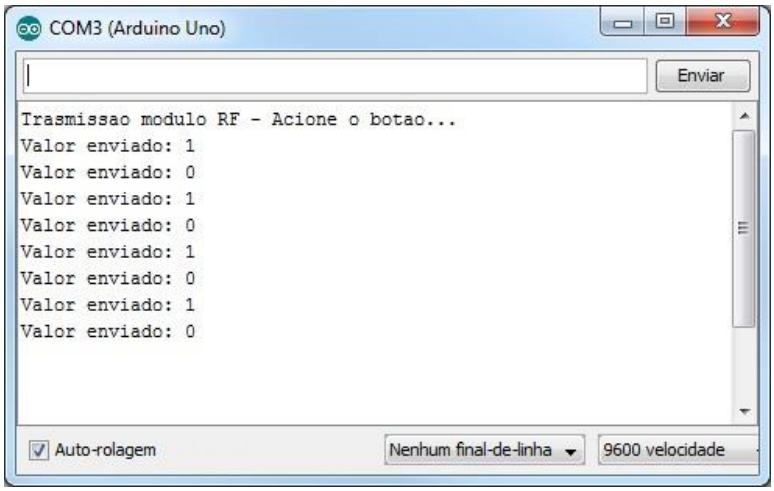


Antes de carregar o programa, faça o download da biblioteca **VirtualWire**, neste link(www.airspayce.com/mikem/arduino/VirtualWire/VirtualWire-1.20.zip). Faça o download, descompacte o arquivo e coloque a pasta VirtualWire dentro da pasta LIBRARIES da IDE do Arduino.

Carregue o programa abaixo no Arduino que será o TRANSMISSOR. Em caso de problemas com a recepção, tente alterar a velocidade de comunicação alterando a linha vw_setup(5000) colocando um valor menor, lembrando que o valor deve ser o mesmo tanto programa do transmissor como do receptor.
O programa verifica se o botão foi pressionado, e nesse caso inverte o valor da variável estado (0 ou 1) para string, enviando esse valor para o módulo transmissor RF.

```
//Programa : Comunicacao com Modulo RF 433 - Emissor
                    //Autor : Arduino e Cia

                    #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);
                     }
                    }
```

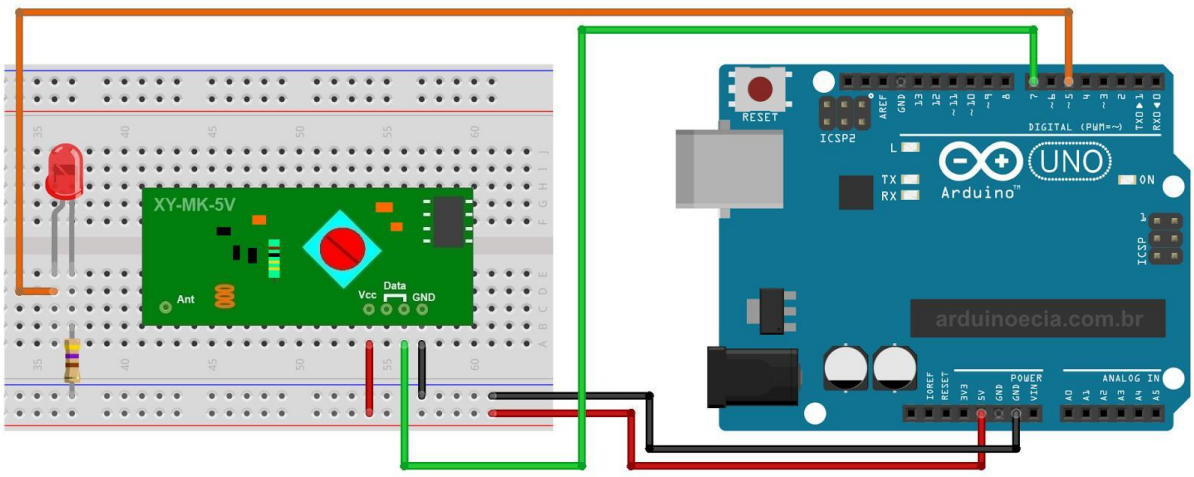
Verifique o funcionamento do circuito abrindo o serial monitor e checando se o estado do botão está sendo alterado:



Circuito Arduino Uno - Receptor

A alimentação do módulo receptor também é feita por meio do pino 5V do Arduino. O receptor possui 2 pinos de dados, e qualquer um deles pode ser utilizado para ligação ao Arduino, na porta 7. O led que vai acender ou apagar conforme os comandos enviados pelo Arduino Transmissor está ligado à porta 5:

RF 433 MHz - Circuito Receptor

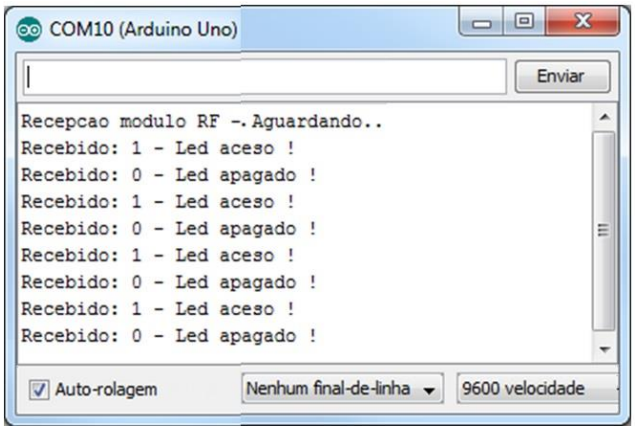


O ideal é alimentar o Arduino com o módulo receptor utilizando uma bateria ou fonte externa, para testar o alcance do módulo.
O programa do receptor recebe do transmissor o valor 0 ou 1 em formato string, converte esse valor para int (inteiro) e aciona a porta 5, ligando ou desligando o led.

```
//Programa : Comunicacao com Modulo RF 433 - Receptor
//Autor : Arduino e Cia
```

```
#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 !");
                }
     }
}
```

Você também pode acompanhar o resultado da recepção desse circuito no serial monitor, que vai ter informações sobre o valor recebido e o estado do led:



O conteúdo do presente tutorial se encontra disponível para acesso online no site www.ebanataw.com.br/arduino/ganheiumkit.htm ou pelo QR-CODE ao lado:

ARDUINO

MEU PRIMEIRO KIT

ASSINATURA DIGITAL

A presente apostila recebe Assinatura Digital com Certificação Digital de acordo com as disposições normativas da ICP-Brasil – Infraestrutura de Chaves Públicas Brasileira, instituída pela Medida Provisória Nº 2200-2 de 24/08/2001.

A assinatura válida está incorporada no quadro ao lado deste arquivo PDF. Se o quadro ao lado se apresentar em branco é por que o conteúdo foi violado e, portanto, não possui mais o aval do autor Roberto Massaru Watanabe.