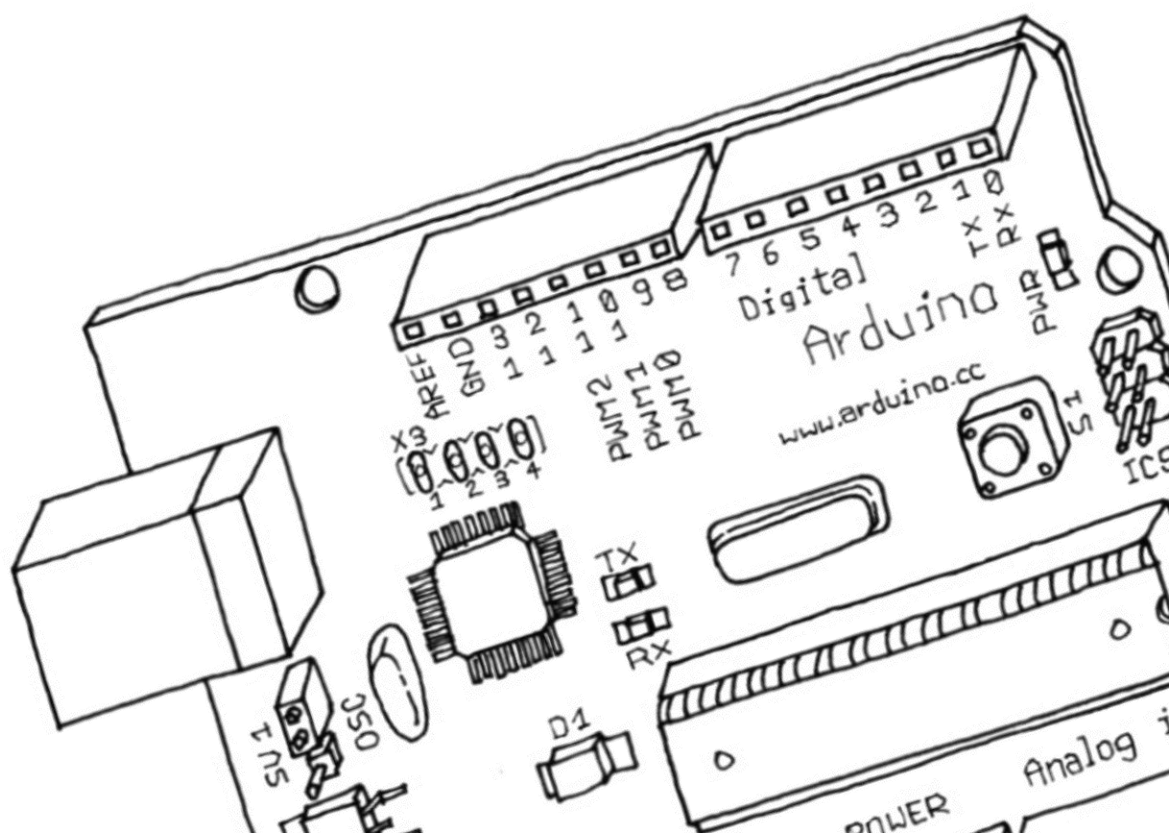




ARDUINO PARA INICIANTE

PROF. ANDERSON HARAYASHIKI MOREIRA



SUMÁRIO

Mensagem Importante	2
1. Introdução.....	3
1.1. Conhecendo seu Arduino Uno	4
1.2. O <i>Kit</i> Arduino para Iniciantes	6
1.3. Instalando o IDE do Arduino	10
1.4. Instalação do <i>driver</i> Arduino no Windows	11
1.5. Instalação do <i>driver</i> Arduino no Windows 8, sim é diferente!	14
1.6. Instalação no Linux	15
1.7. Instalação no Mac OS X	15
1.8. Selecionando sua placa Arduino.....	15
2. Chega de Blá Blá Blá, eu quero programar!!!	17
2.1. Piscando um LED é assim que se começa	17
2.2. Só piscar não dá! Eu que quero acender e apagar este LED.....	21
2.3. Meu dedo cansou, dá pra deixar ligado e desligar depois?	23
2.4. Esse botão está meio meia-boca, dá pra arrumar?	24
2.5. Ligado ou desligado, não dá pra se ter o meio-termo?	25
2.6. Criando um <i>Dimmer</i>	28
2.7. E o Arduino conversa com meu computador?	30
2.8. Usando o <i>display</i> LCD.....	31
2.9. Olha o fogo!!!	34
2.10. Fazendo música	38
2.11. Teremim, você sabe o que é isto?	39
3. O Projeto Final: O Piano de 4 Notas	44
4. Considerações Finais	46

MENSAGEM IMPORTANTE

Primeiramente gostaríamos de agradecer a você por adquirir o curso Arduino para Iniciantes da RoboCore®, prezamos pela qualidade de nossos produtos e serviços. Sempre estamos em busca da excelência em tudo que fazemos, portanto, críticas e sugestões podem ser enviadas à info@robocore.net que teremos grande satisfação em atender a todas as solicitações.

Este material destina-se única e exclusivamente a apoio didático do curso Arduino para Iniciantes da RoboCore®, qualquer reprodução e/ou distribuição sem o consentimento do autor ou da RoboCore® estará sujeita a punições segundo os Direitos Autorais.

Todos os experimentos apresentados nesta obra seguem procedimentos de segurança, não nos responsabilizamos por eventuais acidentes que venham acontecer com o leitor desta obra, é de responsabilidade do leitor seguir as instruções descritas neste material, bem como utilizar os equipamentos de segurança eventualmente necessários.

1. INTRODUÇÃO

O Arduino é uma plataforma de prototipagem eletrônica *open-source* que possui como objetivo a flexibilidade e facilidade de uso. É destinado a artistas, *designers*, hobbistas, estudantes, técnicos, engenheiros e qualquer pessoa interessada em criar objetos ou ambientes interativos, apesar de sua facilidade de uso o Arduino não está limitado a aplicações simples sendo possível utilizá-lo nas diversas e complexas soluções.

O microcontrolador presente na placa é programado usando a linguagem de programação Arduino (baseado em *Wiring*) utilizando apenas de um cabo USB para realizar a gravação, isto é de grande importância, uma vez que os computadores modernos dificilmente apresentam uma porta serial.

O ambiente de desenvolvimento Arduino IDE (*Integrated Development Environment*), baseado em *Processing*, pode ser baixado gratuitamente no *site* **<http://arduino.cc/en/Main/Software>** e trata-se de um ambiente multiplataforma, ou seja, pode ser executado em Linux, Mac OS X e Windows.

Você pode construir suas próprias placas do Arduino, uma vez que todo o projeto Arduino está sob a licença *open-source*. Todo o projeto e arquivos CAD podem ser baixados e você está livre para adaptá-los às suas necessidades, porém, se você quiser apenas utilizar uma das placas padrões, recomenda-se adquirir seu modelo e um representante autorizado, como a RoboCore® (<http://arduino.robocore.net/>).

Arduino está pronto para interagir com o mundo por meio de suas entradas e saídas onde é possível conectar os mais diversos sensores e atuadores. Projetos do Arduino podem ser *stand-alone*, ou seja, dependem apenas de sua placa Arduino, ou podem se comunicar com um *software* rodando em um computador, por exemplo, Flash, Processing, MaxMSP ou qualquer outra solução desenvolvida em um ambiente de programação desenvolvida por você.

Outra vantagem do Arduino é a comunidade de desenvolvedores que sempre está disposta a ajudar a todos que estejam com dificuldades, portanto, utilize os fóruns do Arduino e da RoboCore® para postar suas dúvidas e publicar seus projetos e progressos, afinal o maior objetivo do Arduino é despertar o fazedor que existe dentro de cada um de nós.

Existe uma infinidade de tipos de placas de Arduino e outros infinitos *Shields*, que são placas que podem ser empilhadas ao seu Arduino ampliando a capacidade desta fantástica placa, porém, neste material apenas abordaremos o Arduino Uno, modelo de referência e mais

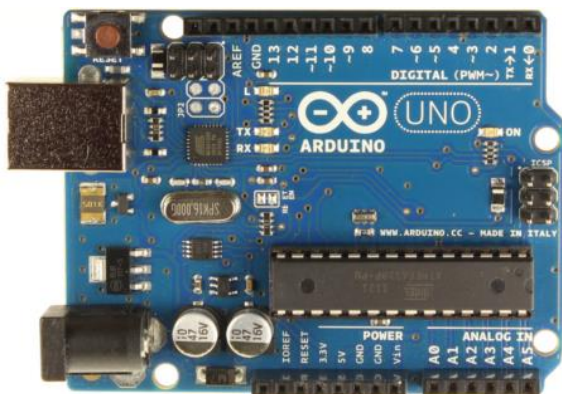
utilizado da família Arduino. Com essa placa é possível realizar projetos maravilhosos nos mais diversos segmentos da tecnologia, dê asas a sua imaginação, use sua criatividade e entre de cabeça no fantástico mundo do Arduino.

1.1. CONHECENDO SEU ARDUINO UNO

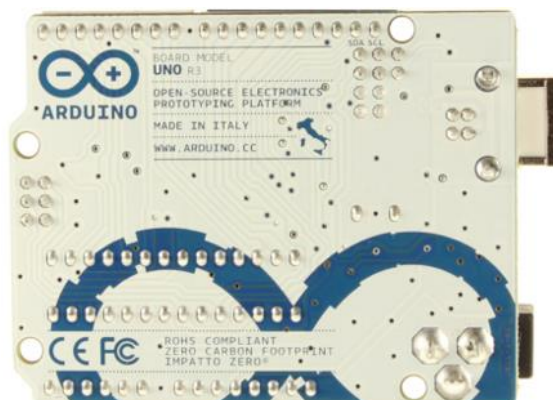
O Arduino Uno é uma placa microcontrolada baseada no ATmega328, possui 14 pinos que podem ser configurados como entradas ou saídas digitais, destes 14 pinos 6 podem ser usados como saídas PWM. O Arduino Uno também 6 entradas analógicas, um cristal de 16 MHz, conexão USB, um conector de alimentação e um botão de *reset*. A placa já vem pronta para ser programada, você pode alimentá-la pela própria porta USB de seu computador ou adquirir um adaptador AC/DC e liga-lo diretamente a uma tomada de sua residência.



ATENÇÃO: Fique atento à tensão máxima de alimentação de sua placa Arduino Uno, tensões de alimentação superiores a 20 V poderão queimar sua placa. Recomenda-se um tensão de alimentação entre 7 e 12 V.



Parte superior do Arduino Uno



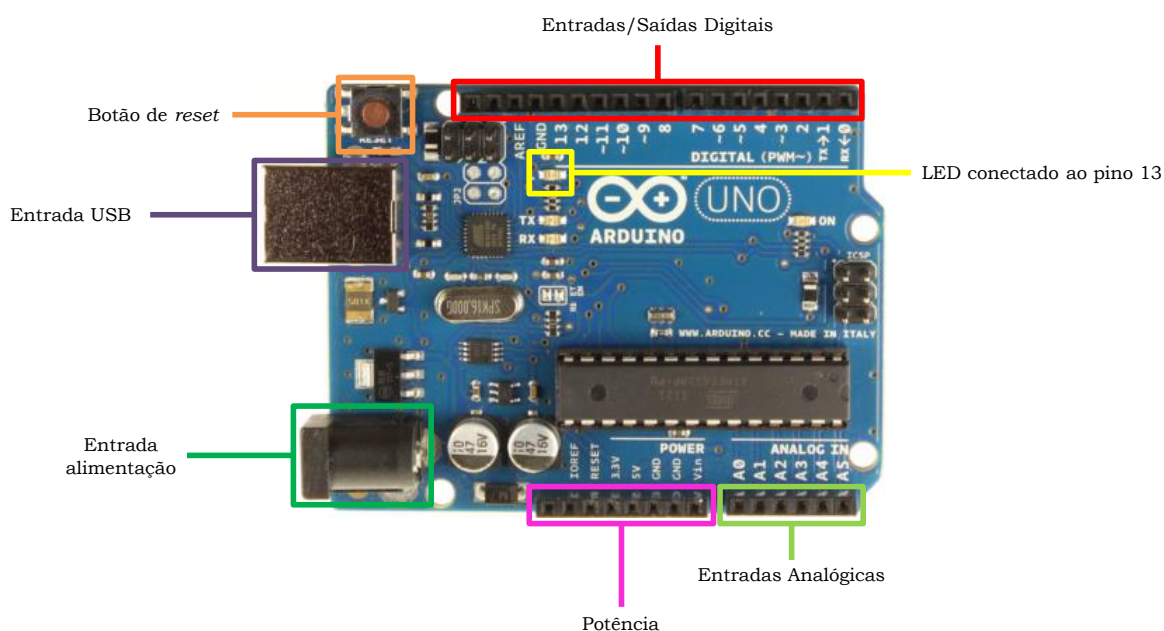
Parte inferior do Arduino Uno

O Uno difere-se do seu antecessor Arduino 2009 pelo fato de não utilizar mais o chip FTDI para realizar a conversão USB para Serial, em vez disso, é utilizado outro microcontrolador, o Atmega16U2, programado como um conversor USB-Serial. O nome Uno significa um em italiano (país de origem do Arduino) e foi utilizado para marcar o lançamento do Arduino 1.0 (IDE de programação do Arduino). O Uno e a versão 1.0 são as versões de referência do Arduino.

Na tabela a seguir encontra-se um resumo com as principais características do Arduino Uno:

Microcontrolador	ATmega328
Tensão de operação	5V
Tensão de Entrada (recomendado)	7-12V
Tensão de Entrada (limites)	6-20V
Entradas/Saídas Digitais	14 (6 podem ser saídas PWM)
Entradas Analógicas	6
Máxima corrente nos pinos de entrada/saída	40 mA
Máxima corrente nos pinos de 3.3V	50 mA
Memória Flash	32 KB (ATmega328) onde 0.5 KB é usado com o bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidade de Clock	16 MHz

Na figura abaixo você encontra as partes principais que compõe sua placa Arduino Uno.



Agora que você já conhece as principais características da sua placa Arduino Uno vamos conhecer os demais componentes que você recebeu em seu *kit* que acompanha seu curso Arduino para Iniciantes da RoboCore®.

1.2. O Kit ARDUINO PARA INICIANTES

Ao longo do nosso curso iremos montar alguns circuitos eletrônicos que irão interagir com seu Arduino. Para montar os experimentos desse *kit* não é necessário nenhum tipo de curso anterior de eletrônica, porém, para ajudar você a identificar cada um dos componentes e deixar a compreensão do circuito um pouco mais fácil, iremos detalhar cada um dos componentes que compõe nosso exclusivo *kit* Arduino para Iniciantes.

Resistor:

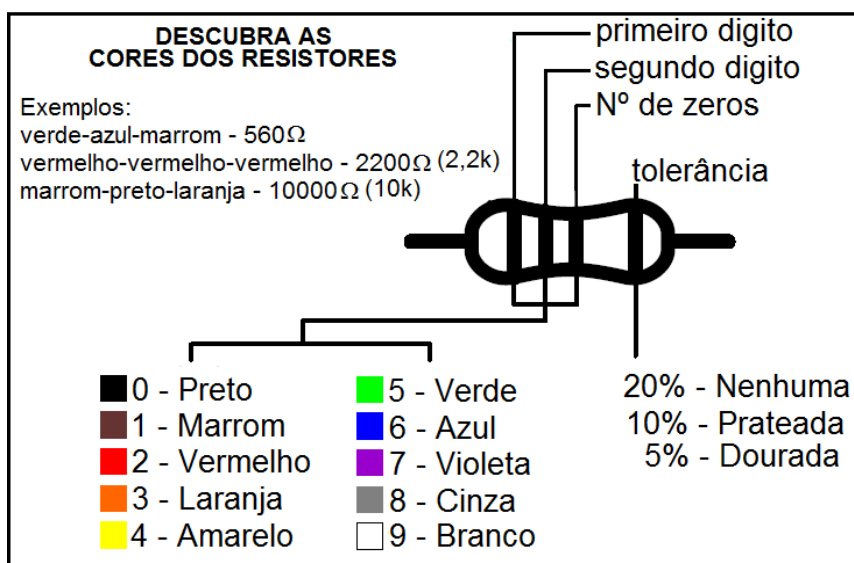


O que isto faz: Limita a corrente elétrica que passa pelo circuito. Para limitar mais ou menos corrente, o valor deste componente pode variar.

Número de pinos: 2 pinos de mesmo comprimento.

+ **Detalhes:** <http://pt.wikipedia.org/wiki/Resistor>

Para saber o valor de cada resistor, basta seguir o esquema abaixo:



Em nossos experimentos, usaremos apenas dois valores de resistores, o de 300Ω e o de 10kΩ. Você encontrará o símbolo deles nos esquemas elétricos. Para começar a se familiarizar, os símbolos são estes ao lado. Você consegue distinguir qual é o de 300Ω e qual é o de 10kΩ? Veja que eles são muito parecidos, porém você deve começar a leitura de cores pelo lado oposto ao dourado, que é responsável pela tolerância do resistor.

Buzzer:



O que isto faz: Quando uma corrente elétrica passa por ele, ele emite um som.

Número de pinos: 2 pinos (este componente tem polaridade, portanto fique atento na hora de ligá-lo).

+

Detalhes:

http://pt.wikipedia.org/wiki/Sensor_piezoel%C3%A9trico

Chave momentânea:



O que isto faz: Quando o botão é apertado, os contatos entre os terminais de cada lado são ligados entre si.

Número de pinos: 4 pinos (os 2 pinos de cada lado já estão em contato normalmente. Quando o botão é apertado os 4 entram em contato).

+ **Detalhes:** http://en.wikipedia.org/wiki/Push_button (em inglês)

Potenciômetro:



O que isto faz: Varia a resistência dos terminais conforme a haste superior é girada.

Número de pinos: 3 pinos (a resistência varia entre um dos pinos das extremidades com o pino do centro).

+ **Detalhes:** <http://pt.wikipedia.org/wiki/Potenci%C3%B4metro>

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).

Número de pinos: 2 pinos (um mais longo e outro mais curto).

+ **Detalhes:** http://pt.wikipedia.org/wiki/Diodo_emissor_de_luz

Sensor de luminosidade LDR:



O que isto faz: É uma resistência que varia de acordo com a luminosidade que incide sobre ele.

Número de pinos: 2 pinos de mesmo comprimento.

+ **Detalhes:** <http://pt.wikipedia.org/wiki/Ldr>

LED RGB:

O que isto faz: Pense em três LEDs de alto brilho: um vermelho, um verde e um azul. Agora, junte todos eles em um só. Pronto, isso é um LED RGB.

Número de pinos: 4 pinos, onde o maior deles é comum a todas as cores.

+ **Detalhes:** <http://pt.wikipedia.org/wiki/RGB>

Display de LCD:

O que isto faz: Mostra dados lidos pelo Arduino em letras e números, muito utilizado em diversos equipamentos eletrônicos. Este dispositivo mostra os dados que estão dentro do Arduino para os seres humanos de uma forma inteligível.

Número de pinos: 16 pinos. Iremos usar apenas 10 pinos.

+ **Detalhes:** <http://pt.wikipedia.org/wiki/LCD>

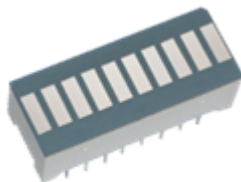
Sensor de temperatura LM35:

O que isto faz: É circuito integrado que mede a temperatura ambiente em °C (graus Celsius).

Número de pinos: 3 pinos do mesmo comprimento (cada pino tem uma função, portanto fique atento ao ligá-lo).

+ **Detalhes:**

<http://es.wikipedia.org/wiki/LM35> (em espanhol)

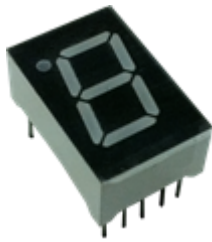
Barra gráfica de LEDs:

O que isto faz: Cada retângulo pequeno na barra gráfica é um LED. Serve para mostrar visualmente a intensidade de alguma grandeza.

Número de pinos: 20 pinos, usaremos todos - a cada 2 acendemos um LED.

+ **Detalhes:**

<http://goo.gl/GJXqER>

Display de 7 segmentos cátodo comum:

O que isto faz: Possui 7 LEDs em formato de número "8", com os quais é possível acender os números de 0 a 9. Ainda possui um LED indicador de ponto.

Número de pinos: 10 pinos.

+ Detalhes:

http://pt.wikipedia.org/wiki/Display_de_sete_segmentos

Circuito integrado 4511:

O que isto faz: Traduz um número em binário para o display de 7 segmentos, facilitando o uso do display e economizando portas do microcontrolador.

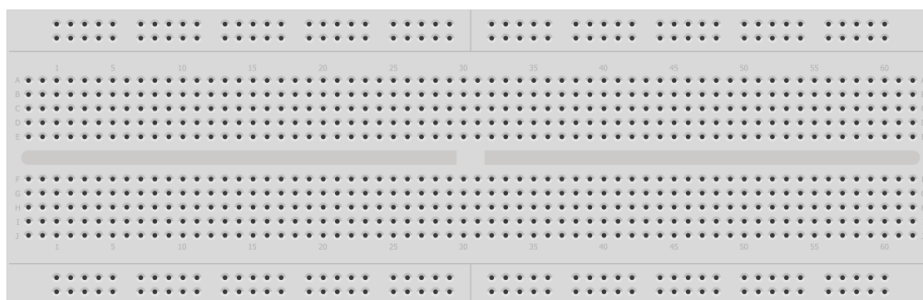
Número de pinos: 16 pinos.

+ Detalhes:

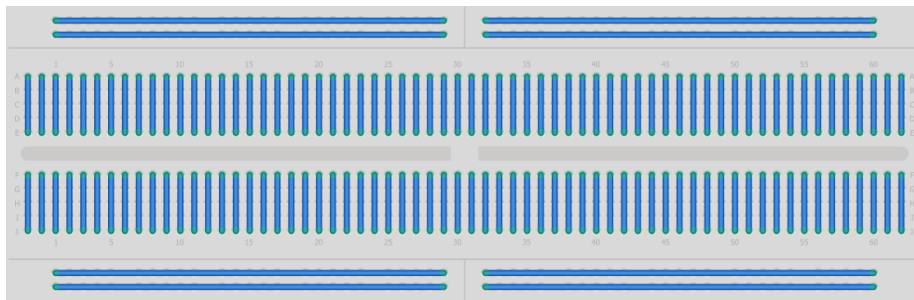
<http://goo.gl/h72GoC>

Cabo USB:

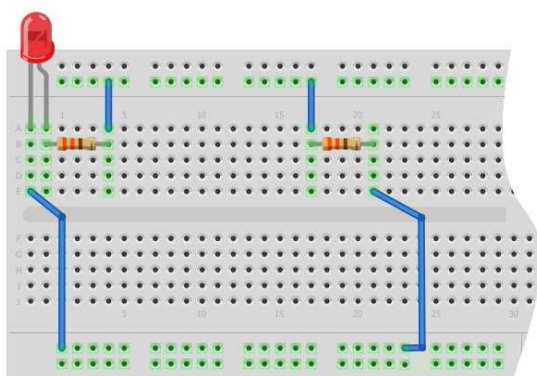
O que isto faz: Faz a comunicação entre o computador e o Arduino. A maioria dos Arduinos utiliza o cabo USB AB, porém, alguns Arduinos, como por exemplo o Leonardo, utilizam o cabo Micro USB.

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.

+ **Detalhes:** <http://pt.wikipedia.org/wiki/Protoboard>

Agora que já conhecemos todo o material que utilizaremos em nosso curso é hora de saber como conseguir e como instalar o ambiente de programação do Arduino (IDE).

1.3. INSTALANDO O IDE DO ARDUINO

Instalar não seria o termo mais apropriado para o ambiente de desenvolvimento do Arduino, por se tratar de um aplicativo desenvolvido em Java faz-se necessário apenas salvar em seu computador e executá-lo diretamente, usando desta característica o IDE do Arduino se caracteriza-se como uma interface multiplataforma, capaz de rodar nos mais diversos sistemas operacionais (Windows, Linux, Mac), sendo apenas necessário o *download* do ambiente compatível ao seu sistema operacional. O *download* pode ser feito na página <http://arduino.robocore.net/> e também existe uma cópia no CD que acompanha este kit, na pasta “\Ambiente de Desenvolvimento\” - para usar o ambiente de desenvolvimento que está no CD você DEVERÁ copiar para alguma local de seu disco rígido. Para abrir o programa, basta clicar duas vezes no ícone “**arduino.exe**”.

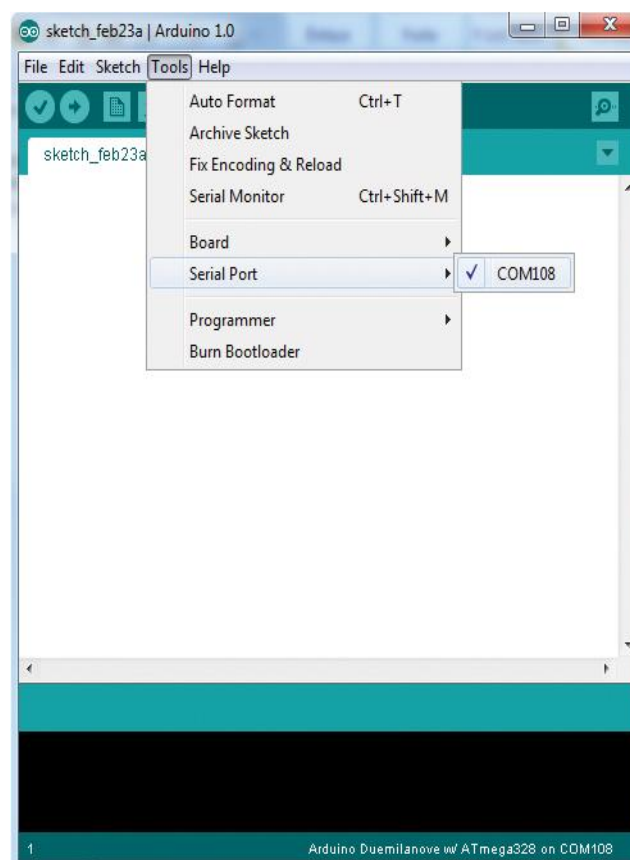


ATENÇÃO: Verifique a compatibilidade do seu sistema operacional com os programas disponíveis no CD.

1.4. INSTALAÇÃO DO DRIVER ARDUINO NO WINDOWS

Seu Arduino Uno é totalmente *Plug & Play*, uma vez rodando o ambiente de desenvolvimento, insira o cabo USB no Arduino e depois no computador. Seu computador deverá reconhecer automaticamente o Arduino e uma nova porta COM (no caso de sistema operacional Windows Vista, Windows 7 ou Linux). Caso o sistema operacional não reconheça a placa automaticamente, os *drivers* podem ser localizados na pasta “\arduino-1.0.1\drivers”.

Para selecionar esta nova porta COM onde o Arduino está localizado, abra o ambiente de desenvolvimento, então clique em **TOOLS > SERIAL PORT > COM X** (onde X é o número da porta que o Arduino foi instalado automaticamente). Na imagem a seguir temos um exemplo do que você deverá ver:



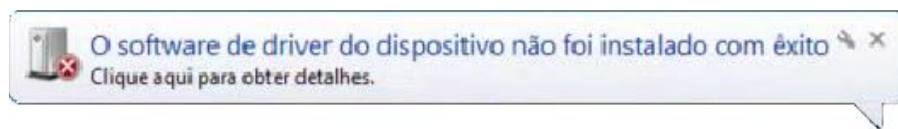
Note que o número da porta COM não é necessariamente 108 como na imagem acima. Cada computador poderá apresentar um número de porta diferente.


Seu Arduino não está sendo reconhecido pelo **Windows 7**?

Não se desespere seu Arduino não está com defeito, você não é a pessoa com mais azar na face de toda a Terra, veja abaixo a solução:

Por causa de fatores ligados a permissões do sistema, o Windows 7 algumas vezes impede que o *driver* seja instalado de uma determinada pasta, onde estão os *drivers* e ambiente de desenvolvimento do Arduino. Desta forma, temos que fazer com que o Windows “force” a instalação destes *drivers* de alguma forma. Siga os seguintes passos:

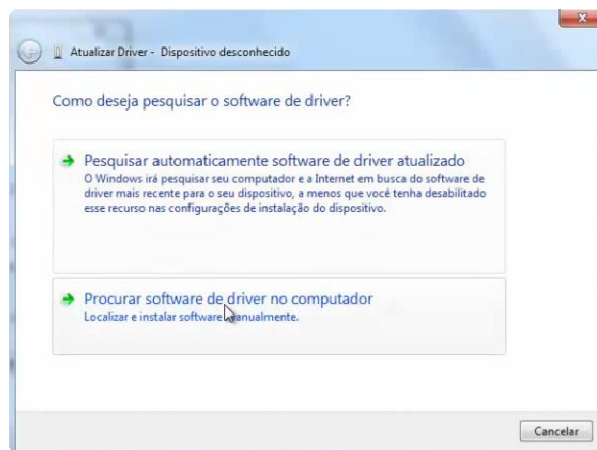
- 1) Conecte seu Arduino à porta USB de seu computador. Aguarde até aparecer a mensagem de erro de instalação de driver. A mensagem deve se parecer com a seguinte:



- 2) Feche esta mensagem. Clique em “Iniciar”  depois em “Dispositivo e Impressoras”. Você verá um dispositivo como “Não Especificado”, como mostra a figura abaixo:



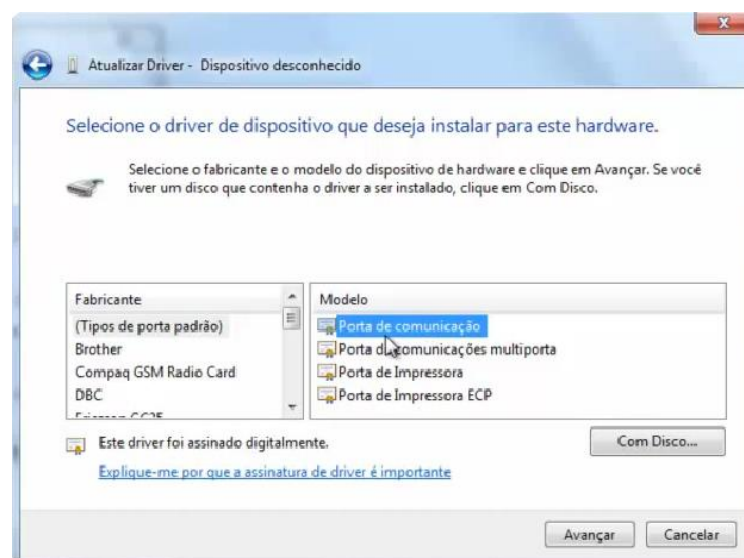
- 3) Clique com o botão direito do Mouse neste “Dispositivo Desconhecido” e depois em Propriedades;
- 4) Clique na aba “*Hardware*” e depois em “Propriedades”;
- 5) Na nova janela, clique no botão “Alterar Configurações”;
- 6) Clique agora em “Atualizar *Driver*...”;
- 7) Na janela que abrir, clique em “Procurar *Software* de *Driver* no Computador”;



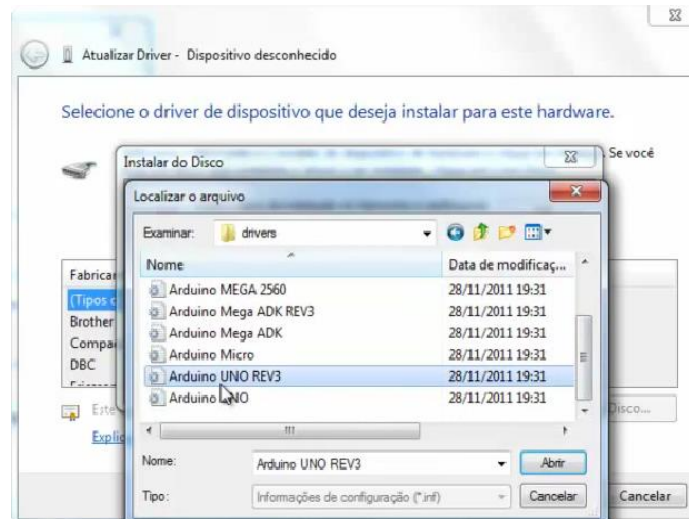
- 8) Neste ponto, não use a opção de seleção de diretório para escolher o *driver* do Arduino. Se você fizer isto, o Windows não irá permitir que o *driver* seja instalado, pois ele não tem permissão para carregar o *driver* da pasta em questão. Clique então em “PERMITIR QUE EU ESCOLHA EM UMA LISTA DE DRIVERS E DISPOSITIVOS NO COMPUTADOR”, como na figura a seguir:



- 9) Na janela que abrir, role a lista para baixo até encontrar “Portas (COM e LPT)” e clique em Avançar;
- 10) Na próxima janela, selecione “Porta de comunicação”, como na figura abaixo e clique em “Com Disco...”:



- 11) Na janela que abrir, faça a busca do driver pelo botão “Procurar”. Direcione esta busca para a pasta *DRIVERS*, do ambiente de desenvolvimento Arduino, e dentro dela clique em “ARDUINO UNO REV3”, caso esta seja sua placa Arduino, conforme a figura abaixo:



- 12) Clique em “Abrir”, então em “Ok” e depois em “Avançar”;
- 13) Clique em “Instalar este *software* mesmo assim”;
- 14) Pronto! Seu Arduino está instalado e pronto para ser usado! Agora, basta selecionar a porta serial do mesmo no ambiente de desenvolvimento Arduino e usá-lo.

1.5. INSTALAÇÃO DO DRIVER ARDUINO NO WINDOWS 8, SIM É DIFERENTE!

No Windows 8, a instalação de *drivers* não assinados é um pouco diferente. Para instalar o *driver* corretamente, siga o seguinte procedimento:



ATENÇÃO: Antes de prosseguir, certifique-se que o você salvou todos os documentos que estão abertos em sua máquina. Pois sua máquina será reiniciada.

- 1) Pressione as teclas "Windows" e "R" simultaneamente;
- 2) Copie e cole o seguinte comando: `shutdown.exe /r /o /f /t 00`
- 3) Selecione "*Troubleshoot*";

- 4) Selecione "*Startup Settings*";
- 5) Selecione "*Disable Driver Signature Enforcement*";
- 6) Instale novamente o *driver* do Arduino.

1.6. INSTALAÇÃO NO LINUX

A instalação dos *drivers* no Linux é muito fácil, basta entrar no terminal de comandos com a placa conectada ao computador, e digitar o seguinte:

Para distribuições Ubuntu:	<code>\$ sudo apt-get install arduino</code>
Para distribuições Fedora 17 ou posterior:	<code>\$ sudo yum install arduino</code>

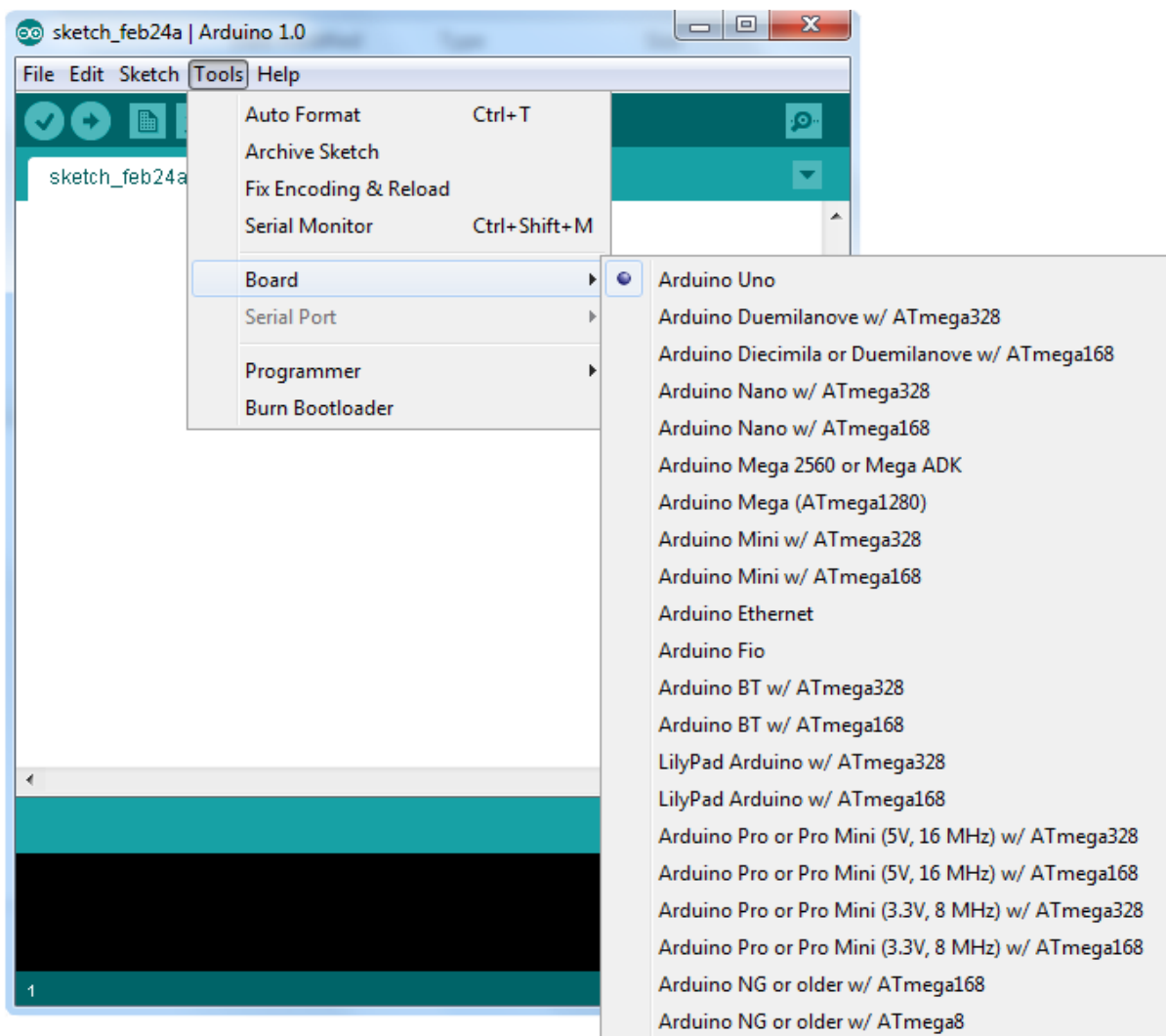
1.7. INSTALAÇÃO NO MAC OS X

Para fazer a instalação no Mac basta copiar o programa para sua pasta de aplicativos, como qualquer aplicativo para Mac. Ao conectar a placa ao computador via cabo USB, o LED *power* irá ligar na placa e a mesma será automaticamente reconhecida pelo computador. Entre nos aplicativos e abra o Arduino IDE. Em Tools > Serial Port procure sua placa Arduino. A descrição da porta de comunicação será algo como `/dev/tty.usbmodem` juntamente a algum número de identificação.

1.8. SELECIONANDO SUA PLACA ARDUINO

No IDE do Arduino é possível programar todos os modelos de placas da família Arduino, atualmente existem 27 modelos diferentes que podem ser programados por esse mesmo ambiente, portanto para salvar códigos em sua placa Arduino, você precisa primeiramente selecionar corretamente qual placa está usando.

Para realizar a configuração do tipo de placa utilizada basta ir ao menu **TOOLS** e depois **BOARD**, selecionando o modelo compatível a sua placa Arduino. Em nosso curso, como já sabemos, utilizaremos o modelo Arduino Uno, modelo este selecionado na figura apresentada a seguir.



ATENÇÃO: Caso você não esteja utilizando a placa Arduino UNO, selecione a placa correta.



DICA: Nos experimentos abaixo procure sempre utilizar o ambiente de desenvolvimento (IDE) Arduino maximizado.



DICA: Para tornar o envio de códigos do computador para a placa Arduino mais rápido, desconecte os periféricos de seu computador, como dispositivos de *Bluetooth*, etc.

serve apenas para o programador colocar informações que podem ajudar a ele ou a outras pessoas, em um futuro, entender o que esse programa ou linha de código faz. Existem duas formas de se escrever um comentário em um programa do Arduino, a linha de comentário ou o bloco de comentário.

Toda linha de comentário é iniciada por `//` tudo que está localizado a direita do `//` é interpretado como comentário até que seja pulado de linha (também conhecido como *enter*). O bloco de comentário por sua vez é delimitado por `/*` e `*/`, portanto tudo que está entre este `/*` e `*/` é um comentário, por exemplo:

```
/* Isto tudo é um comentário!!!  
   Olha eu posso até pular linha!!!  
   Fim do comentário. */
```



DICA: Sempre que possível utilize comentários, pode parecer chato e uma perda de tempo escrever comentários em um programa que talvez somente nós iremos utilizar, não parece chato é realmente chato, mas não é perda de tempo. Muitas vezes fazemos um programa na esperança de nunca mais fazer qualquer tipo de alteração, mas sempre existe aquele dia em que precisamos editar alguma coisinha no programa e daí nos deparamos com aquela bagunça e não sabemos nem por onde começar. Moral da história, normalmente começamos um novo programa do zero, perdendo mais tempo do que o que seria necessário para fazer os benditos comentários, portanto, **COMENTE!!!**

P.S.: Esta é uma das minhas experiências de vida, não comentar e ter que refazer o serviço, experiência a não ser seguida.

Agora que já sabemos o que um comentário é e sabemos que é bom utilizá-los, vamos entender nosso primeiro programa.

Lembrando nossa segunda linha do programa:

```
int led = 13; // Pino no qual o LED está conectado, no caso 13
```

Esta linha é uma declaração de uma variável. Variáveis como o próprio nome já diz é algo que pode variar (Ah vá!), ou seja, damos esse nome a um endereço de memória que guarda em seu conteúdo uma informação que pode ser alterada a qualquer momento do programa. Em nosso caso a declaração de variáveis serve para falar para o Arduino em qual pino ou porta estará ligado o LED.

Um conceito importante a se saber sobre variáveis é se ela é global ou local. Uma variável global pode ser vista e editada dentro de toda e qualquer função do programa (daqui a pouco eu explico o que é uma função!), já uma variável local, apenas pode ser vista e/ou modificada dentro da função onde ela foi declarada. Para simplificar nossas vidas vou me restringir apenas ao universo Arduino.

Toda variável que for declarada fora de uma função, por exemplo, **setup()**, **loop()**, *etc* são variáveis globais. Variáveis declaradas dentro de funções são variáveis locais.



DICA: Tenha o hábito de colocar como primeira coisa de seu programa a declaração de variáveis globais, logo abaixo do comentário inicial de seu programa, é melhor para a organização do mesmo.

Nosso próximo passo é entender as funções, o programa do Arduino é composto de duas partes obrigatórias as funções **setup()** e **loop()**. Funções são blocos de programa delimitados por “{” e “}”.

A função **setup()** e todo seu conteúdo é executado apenas uma vez quando o Arduino é iniciado, entenda essa função como um conjunto de instruções enviadas ao Arduino para que ele se configure adequadamente para fazer os comandos que virão na sequência.

A função **loop()** é a função principal do Arduino que será executada de forma repetitiva, todos os comandos pertencentes a esta função serão executados em um *loop* infinito. Tudo que você queira que seu Arduino faça ponha dentro desta função.

No nosso programa dentro da função **setup()** apenas configuramos o pino do LED (pino 13) como saída, uma vez que um pino digital pode ser tanto entrada como saída. Essa configuração é feita por meio do comando **pinMode**, como pode ser visto:

```
// A rotina setup é executada uma vez quando pressionado o reset:
void setup() {
  pinMode(led, OUTPUT); // inicializa o pino do LED como saída.
}
```

Lembre-se que led é uma variável que criamos e que é igual a 13. Caso o pino a ser configurado for uma entrada troque **OUTPUT** por **INPUT**.

O restante do programa está todo dentro da função **loop()** e é composto apenas por dois comandos o **digitalWrite** e o **delay**.

```
// A rotina loop é executada infinitas vezes até o Arduino ser desligado
void loop() {
    digitalWrite(led, HIGH); // Liga o LED (HIGH é a tensão colocada no pino, no caso 5V)
    delay(1000);             // Espera 1 s ou 1000 ms
    digitalWrite(led, LOW);  // Desliga o LED (LOW equivale a 0V ou GND)
    delay(1000);             // Espera 1 s
}
```

O comando **digitalWrite** liga a saída do Arduino caso se use **HIGH** ou desliga se **LOW** for usado, novamente é necessário informar qual pino vai ser ligado ou desligado. O comando **delay** faz o Arduino aguardar o tempo em milissegundos [ms] especificado, só para lembrar 1000 ms é igual a 1 s.

Bem é isso, terminamos nosso primeiro programa. Como a função **loop()** é executada repetidamente, nosso programa irá ficar piscando um LED ligado ao pino 13 até que o Arduino seja desligado.

A melhor parte deste programa é que ele não precisa de nenhum circuito eletrônico adicional para ser testado, você lembra que nossa placa do Arduino já possui um LED conectado ao pino 13, portanto é só fazer o *upload* do programa e verificar o resultado.

Para fazer o *upload*, primeiramente precisamos compilar o programa para só depois 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 esse aqui →



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*. Espere então o *upload* ser completado e pronto. Você deverá ver o led da placa piscando com intervalos de 1 segundo.

O botão é esse outro aqui →





DICA: Altere os tempos do `delay` e verifique que você altera a velocidade em LED pisca, se você deixar o tempo muito pequeno parecerá que o LED está aceso direto.

Você provavelmente alterou os tempos de forma igual nos dois comandos `delay`, tente colocar dois valores diferentes e bem pequenos, por exemplo, 1 e 5, que equivale a 1 ms com o LED ligado e 5 ms com o LED desligado, faça o *upload* e veja o que aconteceu de diferente.

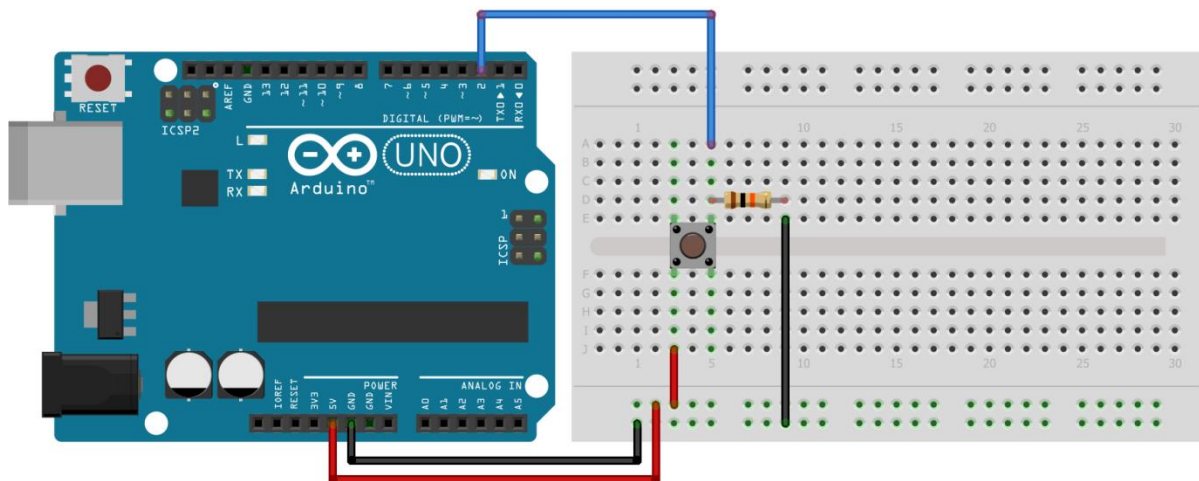


DICA: Essa modificação que você acabou de fazer está diretamente relacionada a um recurso do Arduino chamado PWM, *Pulse-Width Modulation* que veremos na sequência do nosso curso.

2.2. SÓ PISCAR NÃO DÁ! EU QUE QUERO ACENDER E APAGAR ESTE LED

Nosso primeiro programa foi válido como aprendizado, porém, acredito que você deve ter se cansado dele após poucos minutos. Sabendo disso vamos aprender mais algumas coisinhas para incrementar nossos projetos. Nesse exemplo iremos aprender a usar um botão no Arduino.

Primeiramente monte o circuito abaixo:



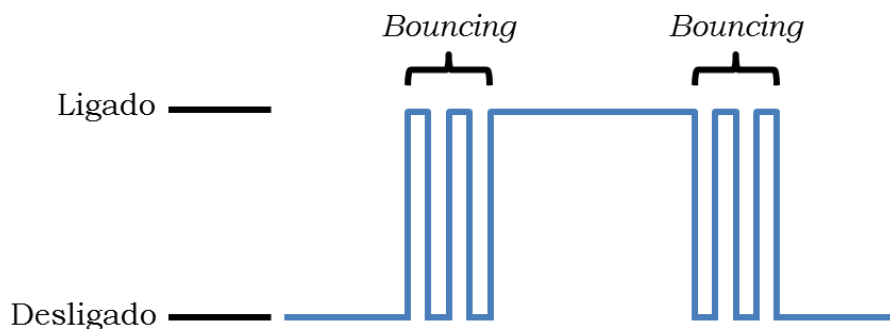
O programa deste exemplo está na sequência, já aumentou um pouco comparando com nosso primeiro programa.

invertemos o estado dessa variável, se está aceso apagamos se está apagado acendemos. Ao final utilizamos essa memória para comandar a saída ligada ao LED.

Simple, não é? Então faça logo esse *upload* e teste seu novo programa.

2.4. ESSE BOTÃO ESTÁ MEIO MEIA-BOCA, DÁ PRA ARRUMAR?

Você deve ter percebido que esse nosso programa parece não funcionar direito às vezes, e talvez já esteja duvidando da qualidade desse curso. Calma eu consigo me redimir. O problema que está acontecendo em nosso programa anterior já é bem conhecido no fantástico mundo da eletrônica, este fenômeno é chamado de *Bouncing* e está representado na figura abaixo.



O *Bouncing* ocorre toda a vez que se altera o estado de um botão, isto se deve ao fato de irregularidades nas superfícies metálicas localizadas na parte interna do botão, e não adianta querer trocar de botão todos são assim. Felizmente o *bouncing* geralmente dura de 10 a 50 ms, portanto ensinarei pra você como implementar um *Debouncing* para evitar esse inconveniente. Nosso *Debouncing* está implementado no código abaixo.

```

/*****
**
**          CURSO ARDUINO PARA INICIANTES - ROBOCORE
**
**  Exemplo 04: Usando um botão com debouncing
**
**
*****/

int led = 13; // Pino no qual o LED está conectado, no caso 13
int botao = 2; // Pino no qual o botão está conectado, no caso 2
int estado = 0; // Variável para guardar o estado do botão (0 = Desligado e 1 = Ligado)
int estado_led = 0; // Variável para guardar o estado do led (0 = Desligado e 1 = Ligado)

// A rotina setup é executada uma vez quando pressionado o reset:
void setup() {
  pinMode(led, OUTPUT); // inicializa o pino do LED como saída.
  pinMode(botao, INPUT); // inicializa o pino do botão como entrada.
}

// A rotina loop é executada infinitas vezes até o Arduino ser desligado

```

```
void loop() {
    estado = digitalRead(botao); // Lê a entrada do botão

    if(estado == HIGH) {        // Se o botão for apertado
        delay(50);              // Espera 50 ms (necessário para o debouncing)
        estado = digitalRead(botao); // Lê a entrada do botão novamente
        while(digitalRead(botao)) // Só libera o programa após soltar o botão
        {
            delay(1);
        }
        if(estado == HIGH) {    // Se o botão realmente for apertado faz a sequência normal
            if(estado_led == HIGH) { // O estado do LED é aceso?
                estado_led = LOW;    // Manda apagar
            }
            else {                // O estado do LED é apagado?
                estado_led = HIGH;   // Manda acender
            }
        }
    }

    if(estado_led == HIGH) {    // Se o estado do LED for aceso
        digitalWrite(led, HIGH); // Liga o LED
    }
    else {                      // Se o estado do LED for apagado
        digitalWrite(led, LOW);  // Desliga o LED
    }
}
```

Faça o *upload* e veja como melhorou! Se o resultado ainda não ficou a seu agrado altere o tempo do *delay*.

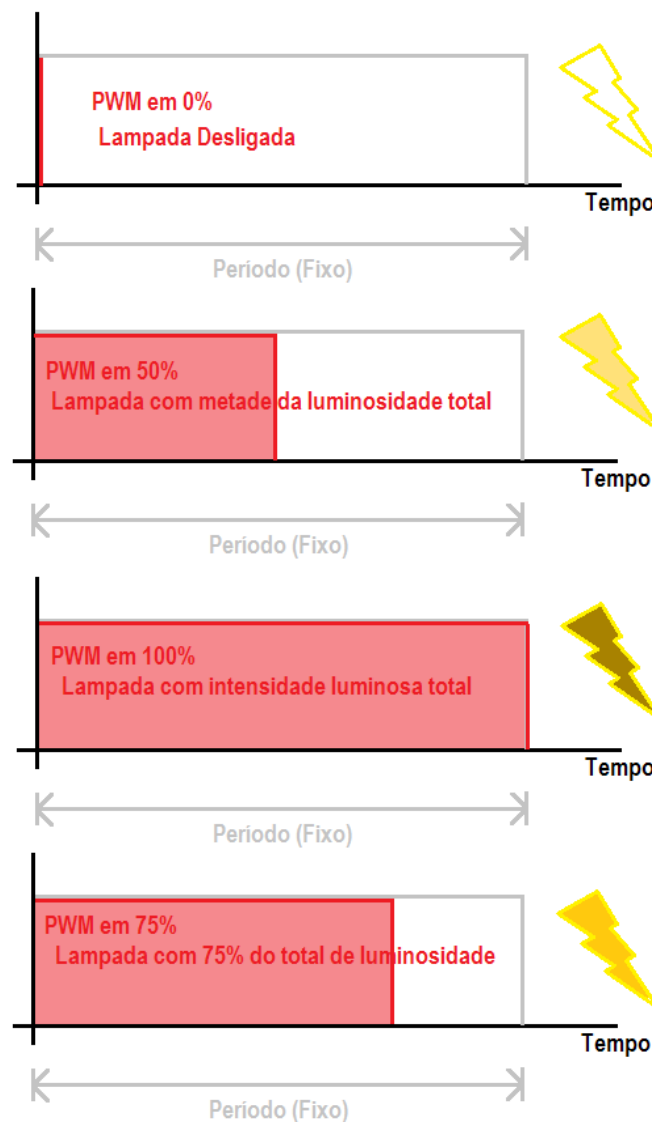
Muito bem, nesse momento já dominamos a estrutura e os comandos básicos do Arduino, mas você já deve estar pensando até esse momento eu só aprendi a fazer o interruptor mais caro que eu já vi, certo? Então vamos aprender os outros comandos do Arduino.

2.5. LIGADO OU DESLIGADO, NÃO DÁ PRA SE TER O MEIO-TERMO?

Legal já sabemos ligar e desligar um LED, mas seria mais legal se conseguíssemos acendê-lo de forma proporcional, sabe pra fazer aquele ambiente a meia-luz dá pra fazer isso? A resposta é: Claro que sim, mas pra isso precisaremos usar a saída analógica do Arduino, que ao invés de colocar 0 ou 5 V em sua saída permite valores intermediários, como por exemplo, 3.9 V.

As saídas analógicas do Arduino também estão localizadas nos pinos de entrada/saída digital, porém, as saídas analógicas são apenas as que possuem um ~ ao lado de seu número. No Arduino Uno as saídas analógicas são os pinos 3, 5, 6, 9, 10 e 11, totalizando 6 saídas analógicas.

Para gerar as saídas analógicas o Arduino utiliza um recurso que já havia mencionado anteriormente o PWM, a figura abaixo explica o conceito por trás deste recurso.

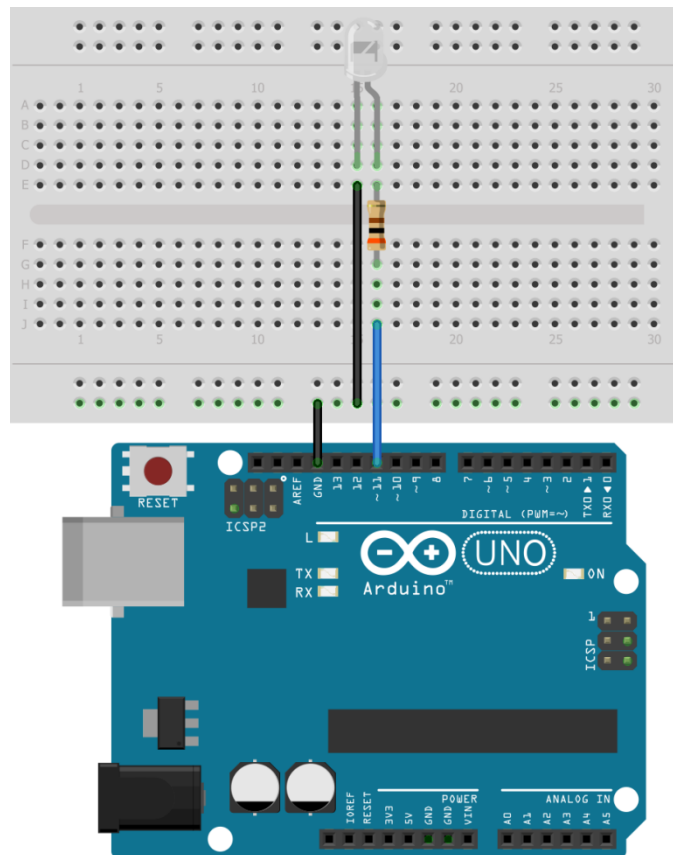


O conceito é simples, tendo apenas dois estados ligado e desligado o PWM consiste em deixar a saída parte do tempo ligada parte do tempo desligada, porém, em período de tempo muito pequeno para que nossos olhos não consigam perceber que desligamos a saída. O período total em que esse processo de ligar e desligar acontece é fixo, e é a razão entre o tempo ligado e o tempo desligado que varia a intensidade da saída.

Para você entender melhor vamos considerar que nosso período é de 10 ms se utilizarmos o PWM em 50%, ou seja, 5 ms ligado e 5 ms desligado, veremos a lâmpada, ilustrada pelo raio amarelo, acesa com metade da intensidade luminosa que teríamos se deixássemos a lâmpada ligada os 10 ms. Para um tempo de 2 ms ligado e 8 ms desligado, teríamos um

PWM em 20%, ou seja, a lâmpada pareceria estar acesa com 20% da sua intensidade máxima.

Vamos montar um circuito para testar essa funcionalidade do Arduino, agora precisamos trocar a porta do LED, uma vez que o pino 13 não é PWM. Já que teremos que usar um LED a mais no circuito, vamos aproveitar para usar um LED de alto-brilho, aquele transparente do seu *kit*, você verá como ele é bem mais bonito.



O programa é bem simples e está apresentado aqui embaixo:

```

/*****
**
**          CURSO ARDUINO BÁSICO - ROBOCORE
**
**  Exemplo 05: LED com efeito Fading
**
**
**
*****/

int led = 11;    // Pino no qual o LED está conectado, no caso 11
int saida = 0;   // Valor que será colocado no PWM, varia de 0 a 255

void setup() {
  // não precisamos colocar nada no setup, para uma saída analógica não precisa configurar.
}

```

```
void loop() {  
  // Fade in do valor mínimo ao máximo em incrementos de 5.  
  for(saida = 0 ; saida <= 255; saida +=5) {  
    analogWrite(led, saida);  
    delay(30);    // Espera 30 ms para se ver o efeito  
  }  
  
  // Fade out do valor máximo ao mínimo em incrementos de 5.  
  for(saida = 255 ; saida >= 0; saida -=5) {  
    analogWrite(led, saida);  
    delay(30);    // Espera 30 ms para se ver o efeito  
  }  
}
```

A primeira grande novidade em nosso programa é a estrutura **for**, que faz parte do conjunto de estruturas de repetição presentes nas linguagens de programação. Encare o **for** como uma estrutura que executará o conteúdo englobado pelo conjunto de “{” e “}” uma certa quantidade de vezes. No nosso caso iniciamos a variável *saida* com 0 (*saida* = 0) e aumentamos seu valor de 5 em 5 (*saida* +=5) até que ela atinja 255 (*saida* <= 255), só assim a estrutura **for** e seu conteúdo parará de ser executada. Esta mesma estrutura é executada mais uma vez, porém, decrementando a variável ao invés de incrementar.

Outra novidade é o próprio comando para se utilizar a saída analógica, **analogWrite**, o funcionamento é bem parecido com o comando **digitalWrite**, primeiro se informa o pino da saída (no nosso caso 11) e depois informa-se a intensidade da saída (qualquer valor inteiro de 0 a 255).

Faça o *upload* desse programa em sua placa Arduino agora mesmo e veja o LED acender e apagar gradativamente.

2.6. CRIANDO UM DIMMER

Difícilmente iremos utilizar uma lâmpada funcionando do mesmo jeito do exemplo anterior, é muito mais inteligente utilizar um *dimmer*. Um *dimmer* nada mais é que um potenciômetro usado para variar a intensidade da luminosidade da lâmpada.

Utilizaremos o potenciômetro como um sensor analógico, a medida que alterarmos sua resistência, variaremos a intensidade luminosa do LED.

O circuito que devemos montar nesse exemplo é bem parecido com o anterior, sendo apenas necessário adicionar o potenciômetro a uma das 6 entradas analógicas do Arduino, como pode ser observado no circuito abaixo.

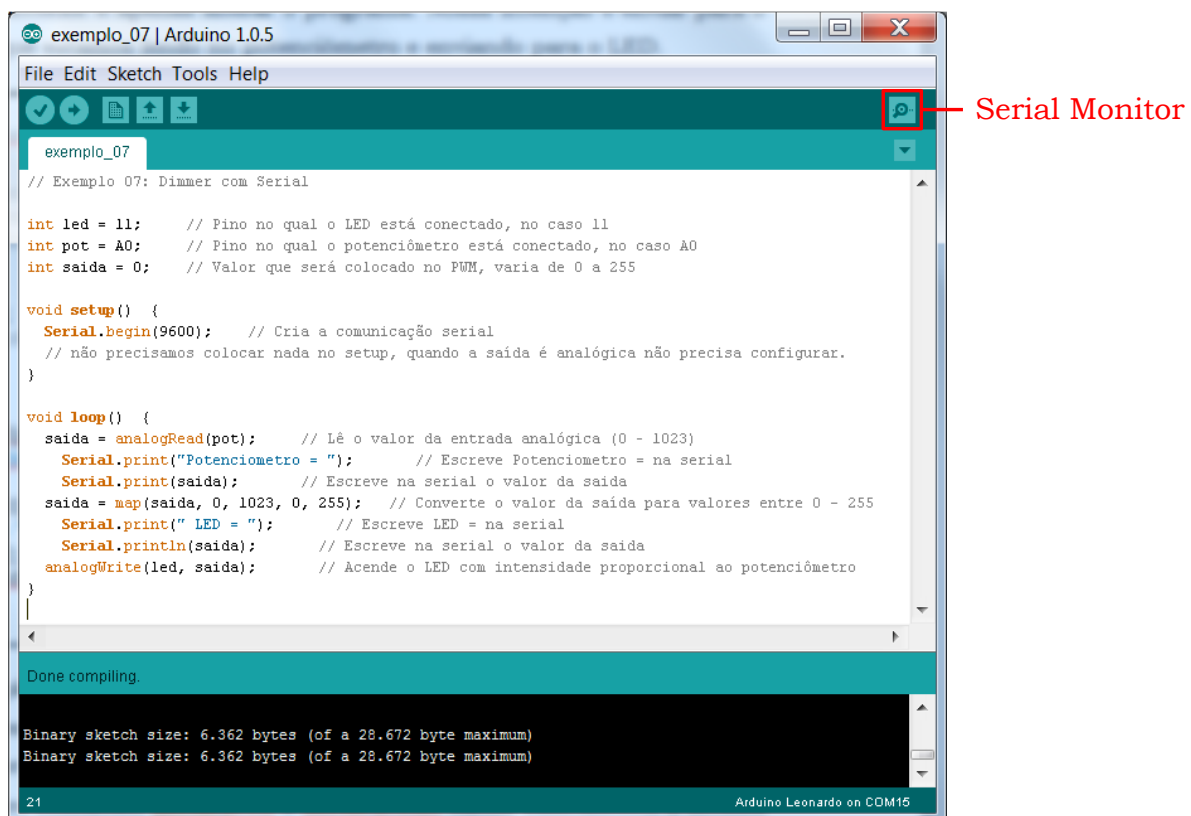


De novidade neste nosso programa temos apenas o comando `analogRead` e o comando `map`. O `analogRead` é responsável por fazer a leitura da entrada analógica, neste comando é

serial, o 9600 é o valor de *baudrate* que a velocidade que são enviados os dados pela serial. O segundo e o terceiro comando, **Serial.print** e **Serial.println** fazem basicamente a mesma coisa, enviar o dado pela serial, a diferença é que o **Serial.println** pula linha ao final do último dado enviado.

Podem ser enviados dados (variáveis) e textos pela serial, para enviar uma variável basta colocar o nome da variável dentro dos parênteses. Para se enviar um texto pela serial deve-se coloca-lo entre aspas duplas (“ ”), recomenda-se não utilizar acentuação nos textos.

Faça o *upload* do programa em seu Arduino, para visualizar os dados enviados pela serial abra o **Serial Monitor** no IDE do Arduino.



2.8. USANDO O DISPLAY LCD

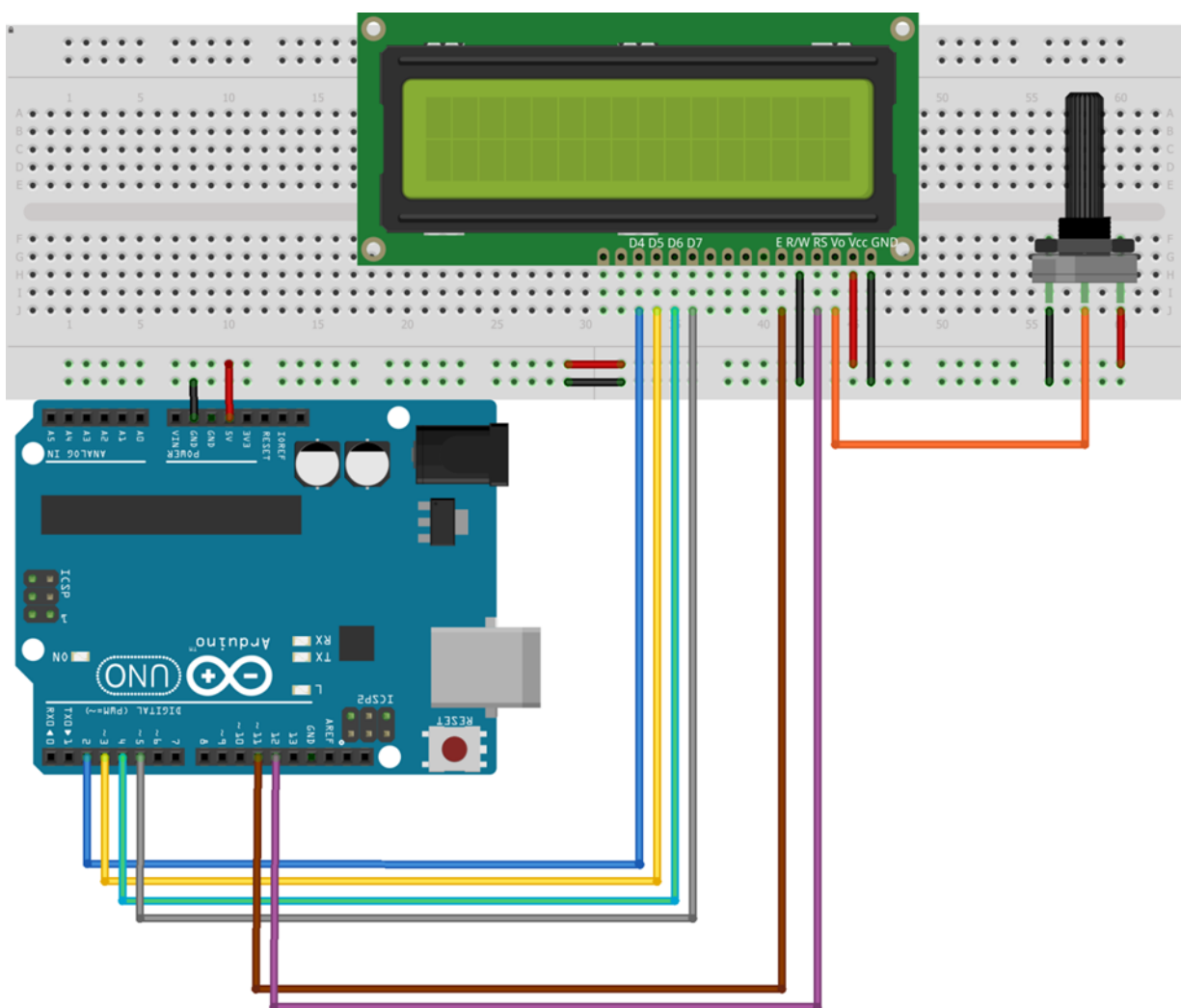
Você pode ter ficado empolgado com o fato do Arduino poder conversar com seu computador, mas provavelmente deve ter ficado se perguntando “Toda a vez que eu quiser saber o que o Arduino está fazendo eu terei que conectá-lo ao meu computador?”. A resposta é não.

Um recurso bastante interessante e útil em projetos no mundo do Arduino são os *displays* de LCD. Baratos e de fácil utilização, esses tipos de *display* são capazes de imprimir

informações ao usuário de maneira similar à comunicação serial com o computador, com a vantagem de não necessitar de um computador.

Existem diversos tipos e tamanhos de *displays* LCD, o que utilizaremos é o 16x2. Este *display* apresenta 2 linhas e 16 colunas, por isso o nome (sugestivo não?). Independente do tamanho os pinos de configuração e comunicação desses *displays* são geralmente iguais.

Os *displays* LCDs são controlados por meio de uma comunicação paralela. Esta por sua vez pode ser configurada de duas maneiras: 4 bits ou 8 bits. O modo de 4 bits requer sete pinos de I/O do Arduino, enquanto que o modo de 8 bits requer 11 pinos. Para a exibição de textos, você pode fazer quase tudo, no modo de 4 bits, em nosso exemplo controlaremos um *display* LCD 16x2 no modo de 4 bits, afinal de contas sempre é bom economizar alguns pinos do Arduino. Monte o circuito abaixo, atentando-se nos pinos do *display* LCD, para podermos iniciar nosso exemplo.



Para imprimir um dado no *display* LCD basta utilizar a função `lcd.print(dado a ser impresso)`, lembrando que ser for imprimir um texto este deve estar entre “ ”. Não perca mais tempo faça o *upload* desse programa em seu Arduino e vamos ver o tempo que o Arduino está ligado impresso no *display* LCD.

```

/*****
**
**          CURSO ARDUINO PARA INICIANTES - ROBOCORE          **
**
** Exemplo 08: Usando o display LCD                      **
**
*****/

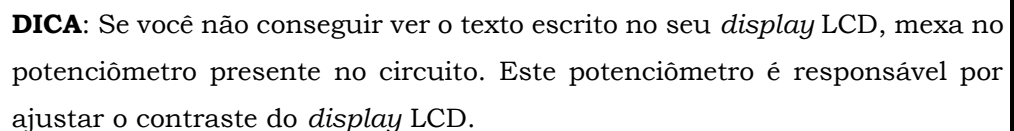
#include <LiquidCrystal.h> // Inclui a biblioteca do Display LCD

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // Inicializa os pinos do display LCD

void setup() {
    lcd.begin(16, 2); // Define o tamanho do display LCD (16 colunas e 2 linhas)
    lcd.print("Alo Arduino"); // Imprime o texto no display LCD
}

void loop() {
    lcd.setCursor(0, 1); // Posiciona o cursor na coluna 0 e linha 1 (A linha 1 é a segunda
linha)
    lcd.print(millis()/1000); // Imprime os segundos que se passaram desde que o Arduino foi
ligado
}

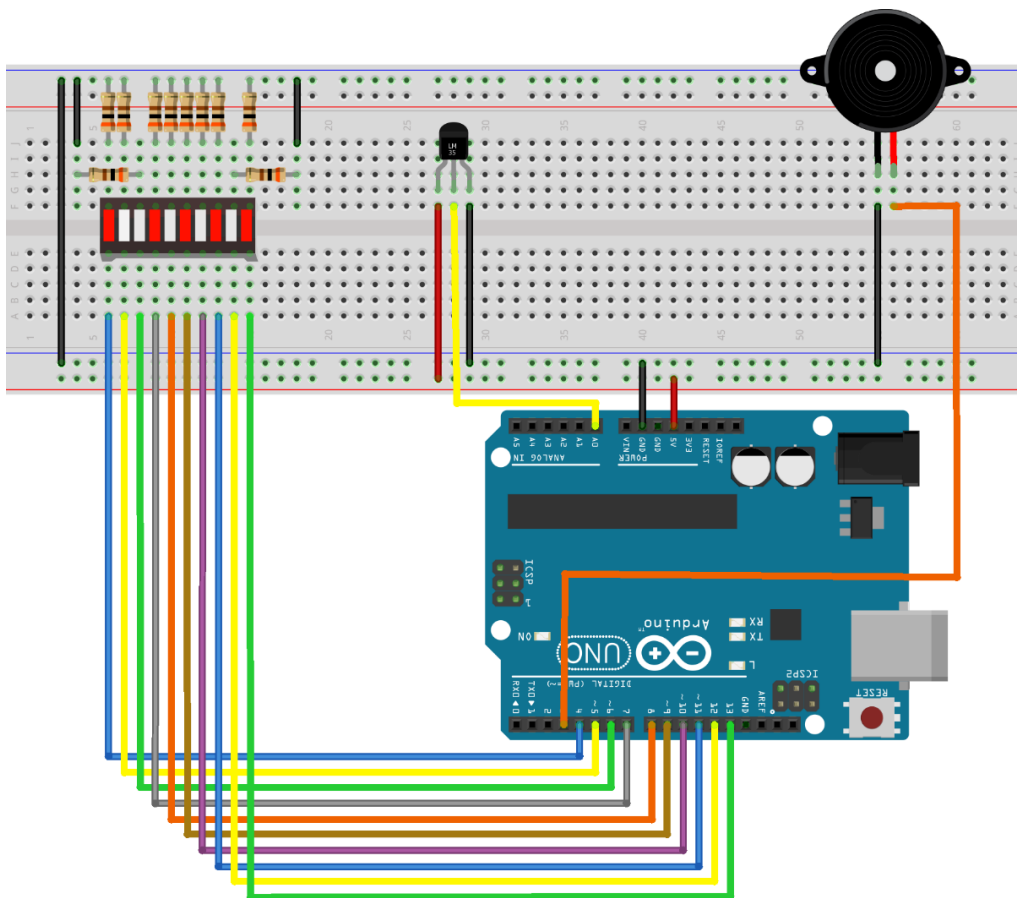
```



2.9. OLHA O FOGO!!!

Até agora não utilizamos nosso Arduino para ler o valor vindo de um sensor analógico. Neste experimento além de medirmos a temperatura ambiente, por meio de um sensor de temperatura, criaremos uma indicação luminosa com a barra gráfica de LEDs e um sinal de alarme utilizando um *buzzer*.

O sensor utilizado é o LM35, um sensor semicondutor, que se assemelha a um transistor, e que ao ler uma variação de temperatura, oferece uma variação de tensão em seus terminais. Para realizar este experimento, deve-se realizar a montagem do circuito a seguir:



O programa para nosso alarme é bem simples, quase todos os conceitos para se fazer esse experimento você já possui, só é preciso aprender como o *buzzer* funciona. O *buzzer* é acionado em uma frequência definida, correspondente a uma nota musical. Pode parecer difícil fazer isso, porém, basta utilizar a função `tone`(Pino do *Buzzer*, Frequência da Nota, Duração em ms). A barra gráfica de LEDs pode parecer diferente, mas ela funciona exatamente igual aos LEDs que já utilizamos nos últimos experimentos. Para se ler o valor proporcional à temperatura do ambiente iremos utilizar a função de leitura analógica `analogRead`(PinoSensor).

```

/*****
**
**          CURSO ARDUINO PARA INICIANTEs - ROBOCORE
**
**
** Exemplo 09: Alarme de temperatura
**
*****/

int PinoSensor = A0; // Pino no qual o sensor de temperatura está conectado, no caso A0
int Buzzer = 3;      // Pino no qual o buzzer está conectado, no caso 2
int ValorSensor = 0; // Variável para guardar a leitura do sensor de temperatura
float temperatura = 0; // Variável para guardar o valor convertido do sensor para um
                        // valor de temperatura
int SetPoint = 40;   // Set point de temperatura, acima desse valor dispara o alarme
int led1 = 4;        // Pino no qual o primeiro LED da barra gráfica será ligado, no caso 3
int led2 = 5;        // Pino no qual o segundo LED da barra gráfica será ligado, no caso 4
int led3 = 6;        // Pino no qual o terceiro LED da barra gráfica será ligado, no caso 5
int led4 = 7;        // Pino no qual o quarto LED da barra gráfica será ligado, no caso 6
int led5 = 8;        // Pino no qual o quinto LED da barra gráfica será ligado, no caso 7
int led6 = 9;        // Pino no qual o sexto LED da barra gráfica será ligado, no caso 8
int led7 = 10;       // Pino no qual o sétimo LED da barra gráfica será ligado, no caso 9
int led8 = 11;       // Pino no qual o oitavo LED da barra gráfica será ligado, no caso 10
int led9 = 12;       // Pino no qual o nono LED da barra gráfica será ligado, no caso 11
int led10 = 13;      // Pino no qual o décimo LED da barra gráfica será ligado, no caso 12

void setup() {
  pinMode(led1, OUTPUT); // Configura o pino do LED 1 como saída
  pinMode(led2, OUTPUT); // Configura o pino do LED 2 como saída
  pinMode(led3, OUTPUT); // Configura o pino do LED 3 como saída
  pinMode(led4, OUTPUT); // Configura o pino do LED 4 como saída
  pinMode(led5, OUTPUT); // Configura o pino do LED 5 como saída
  pinMode(led6, OUTPUT); // Configura o pino do LED 6 como saída
  pinMode(led7, OUTPUT); // Configura o pino do LED 7 como saída
  pinMode(led8, OUTPUT); // Configura o pino do LED 8 como saída
  pinMode(led9, OUTPUT); // Configura o pino do LED 9 como saída
  pinMode(led10, OUTPUT); // Configura o pino do LED 10 como saída
  pinMode(Buzzer, OUTPUT); // Configura o pino do buzzer como saída
  Serial.begin(9600); // Cria a comunicação serial
}

void loop() {
  ValorSensor = analogRead(PinoSensor); // Lê o valor do sensor de temperatura
  temperatura = 5.0*ValorSensor*100/1024; // Converte o valor lido do sensor em um valor
  // de temperatura
  Serial.print("Valor do Sensor = "); // Escreve Valor do Sensor = na serial
  Serial.print(ValorSensor); // Escreve o valor do sensor na serial
  Serial.print(" Valor da Temperatura = "); // Escreve Valor do Sensor = na serial
  Serial.println(temperatura); // Escreve o valor do sensor na serial

  if (temperatura > SetPoint) {
    tone(Buzzer, 1000, 250); // O valor lido é maior que o set point, aciona o alarme
  }
  else {
    noTone(Buzzer); // O valor lido é menor que o set point, não aciona o alarme
  }
  if (temperatura <= 20) { // A temperatura é menor que 20, apaga todos os LEDs
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
  }
}

```



```
digitalWrite(led3, LOW);
digitalWrite(led4, LOW);
digitalWrite(led5, LOW);
digitalWrite(led6, LOW);
digitalWrite(led7, LOW);
digitalWrite(led8, LOW);
digitalWrite(led9, LOW);
digitalWrite(led10, LOW);
}

if (temperatura > 20) {      // A temperatura é maior que 20, aciona o LED 01 e 02
    digitalWrite(led1, HIGH);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
    digitalWrite(led5, LOW);
    digitalWrite(led6, LOW);
    digitalWrite(led7, LOW);
    digitalWrite(led8, LOW);
    digitalWrite(led9, LOW);
    digitalWrite(led10, LOW);
}

if (temperatura > 25) {      // A temperatura é maior que 25, aciona o LED 03 e 04
    digitalWrite(led1, HIGH);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
    digitalWrite(led5, LOW);
    digitalWrite(led6, LOW);
    digitalWrite(led7, LOW);
    digitalWrite(led8, LOW);
    digitalWrite(led9, LOW);
    digitalWrite(led10, LOW);
}

if (temperatura > 30) {      // A temperatura é maior que 30, aciona o LED 05 e 06
    digitalWrite(led1, HIGH);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
    digitalWrite(led5, HIGH);
    digitalWrite(led6, HIGH);
    digitalWrite(led7, LOW);
    digitalWrite(led8, LOW);
    digitalWrite(led9, LOW);
    digitalWrite(led10, LOW);
}

if (temperatura > 35) {      // A temperatura é maior que 35, aciona o LED 07 e 08
    digitalWrite(led1, HIGH);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
    digitalWrite(led5, HIGH);
    digitalWrite(led6, HIGH);
    digitalWrite(led7, HIGH);
    digitalWrite(led8, HIGH);
    digitalWrite(led9, LOW);
    digitalWrite(led10, LOW);
}

if (temperatura > 40) {      // A temperatura é maior que 40, aciona o LED 09 e 10
    digitalWrite(led1, HIGH);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
    digitalWrite(led5, HIGH);
    digitalWrite(led6, HIGH);
    digitalWrite(led7, HIGH);
    digitalWrite(led8, HIGH);
    digitalWrite(led9, HIGH);
    digitalWrite(led10, HIGH);
}
}
```



DICA: Abra o *Serial Monitor* para verificar o valor correspondente à temperatura ambiente. Caso o valor esteja muito menor que o valor de 900 do *SetPoint*, altere seu valor.

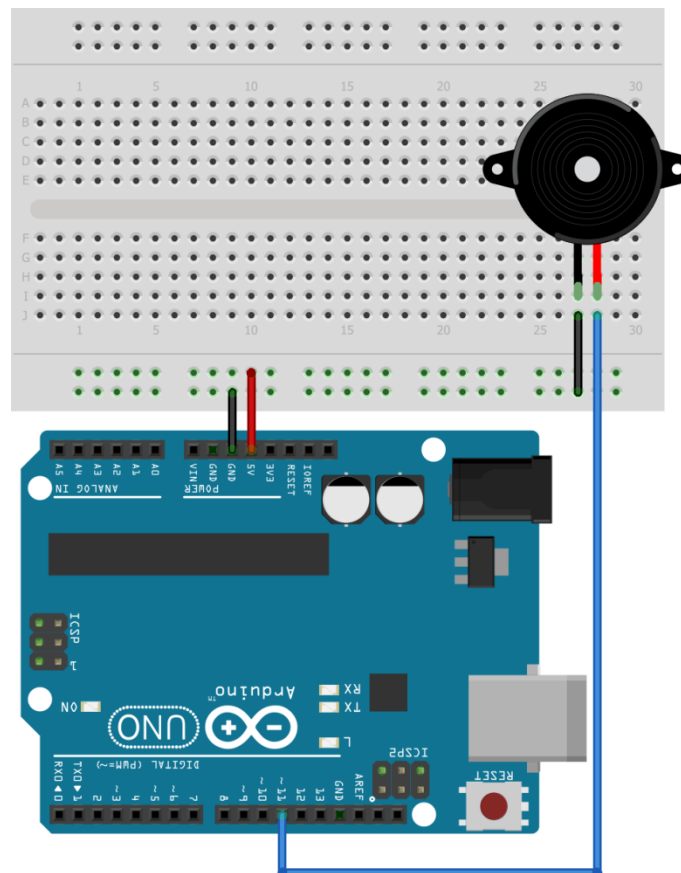


DICA: Para aumentar a temperatura lida pelo sensor basta segurar o sensor. Quanto maior a temperatura maior o valor lido pelo Arduino.

2.10. FAZENDO MÚSICA

Você sempre vontade de tocar algum instrumento musical, mas nunca teve o dom (para não dizer coordenação motora)? Se a sua resposta foi sim seus problemas se acabaram, agora você pode programar seu Arduino para tocar uma música para você. Neste experimento iremos aprender a utilizar uma função do Arduino que é capaz de produzir notas musicais.

Primeiramente monte o circuito abaixo, ele é bem simples é só retirar a parte do sensor de temperatura do experimento anterior.



O funcionamento do *buzzer* é exatamente igual ao experimento anterior, apenas para facilitar o entendimento do programa, em seu início foram definidas as frequências das notas que são utilizadas na música a ser tocada pelo Arduino (para isso foi utilizado o comando `#define`, por exemplo, `#define NOTE_C4 262`). Para uma nota musical ficar em cima da outra entre cada uma das notas foi utilizado um `delay` referente a pausa entre cada nota.

Não perca mais um minuto, faça o *upload* desse programa em seu Arduino e vamos escutar uma musiquinha.

```

/*****
**
**          CURSO ARDUINO PARA INICIANTES - ROBOCORE
**
** Exemplo 10: Tocando uma melodia
**
**
*****/

#define NOTE_C4  262
#define NOTE_G3  196
#define NOTE_A3  220
#define NOTE_B3  247

int Buzzer = 11; // Pino no qual o buzzer está conectado, no caso 11

void setup() {
  // tone(Pino, Frequência, Duração)
  tone(Buzzer,NOTE_C4,250); // Toca a nota de acordo com os parâmetros
  delay(325); // Espera o tempo da pausa
  noTone(Buzzer); // Para de tocar a nota, prepara para próxima nota
  tone(Buzzer,NOTE_G3,125); // Toca a nota de acordo com os parâmetros
  delay(160); // Espera o tempo da pausa
  noTone(Buzzer); // Para de tocar a nota, prepara para próxima nota
  tone(Buzzer,NOTE_G3,125); // Toca a nota de acordo com os parâmetros
  delay(160); // Espera o tempo da pausa
  noTone(Buzzer); // Para de tocar a nota, prepara para próxima nota
  tone(Buzzer,NOTE_A3,250); // Toca a nota de acordo com os parâmetros
  delay(325); // Espera o tempo da pausa
  noTone(Buzzer); // Para de tocar a nota, prepara para próxima nota
  tone(Buzzer,NOTE_G3,250); // Toca a nota de acordo com os parâmetros
  delay(325); // Espera o tempo da pausa
  noTone(Buzzer); // Para de tocar a nota, prepara para próxima nota
  tone(Buzzer,0,250); // Toca a nota de acordo com os parâmetros
  delay(325); // Espera o tempo da pausa
  noTone(Buzzer); // Para de tocar a nota, prepara para próxima nota
  tone(Buzzer,NOTE_B3,250); // Toca a nota de acordo com os parâmetros
  delay(325); // Espera o tempo da pausa
  noTone(Buzzer); // Para de tocar a nota, prepara para próxima nota
  tone(Buzzer,NOTE_C4,250); // Toca a nota de acordo com os parâmetros
  delay(325); // Espera o tempo da pausa
  noTone(Buzzer); // Para de tocar a nota, prepara para próxima nota
}

void loop() {
  // A melodia apenas é executada uma vez no setup
}

```



DICA: Acesse <http://www.robocore.net/upload/notas.txt> para conseguir a frequência de todas as notas musicais.

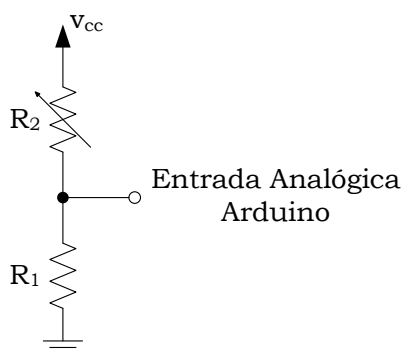
2.11. TEREMIM, VOCÊ SABE O QUE É ISTO?

Você estar aí se perguntado que diabos é um Teremim, certo? O teremim é um dos primeiros instrumentos musicais completamente eletrônicos, controlado sem qualquer contato físico pelo músico. Na imagem abaixo você pode ver a cara desse inusitado instrumento musical.

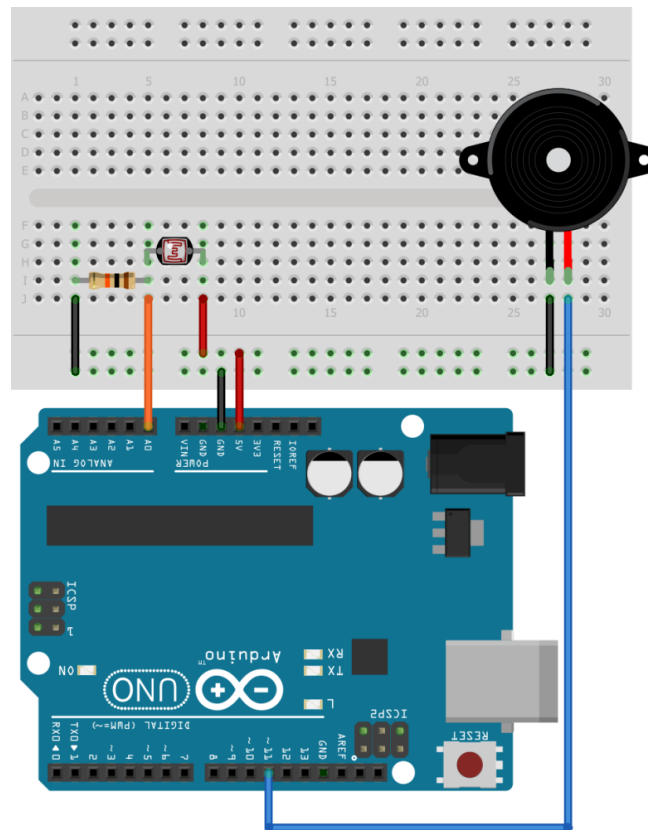


Não iremos construir um teremim verdadeiro, para isso precisaríamos de conhecimentos mais avançados em eletrônica. Nosso experimento consiste em criar um dispositivo que imita o funcionamento de um teremim. Para criar nosso teremim caseiro iremos utilizar o nosso *buzzer* em conjunto com nosso LDR.

Primeiramente é importante entender o circuito que utilizaremos neste experimento, a configuração utilizada, é chamada de divisor de tensão, poderá ser usada em diversas ocasiões onde queremos ler um sinal de um sensor analógico.



O sensor de luminosidade (LDR) presente no kit, utilizado para medir a quantidade de luz em certo ambiente, possui o comportamento de variar o valor de sua resistência proporcionalmente à variação da grandeza medida. Variando o valor da resistência a tensão lida pelo Arduino também irá variar. Monte o circuito abaixo para podermos criar nosso Teremim.



Na minha opinião, este é o programa mais complexo de todos os nossos experimentos, a ideia dele é reunir todos os conceitos aprendidos até o momento. Usamos rotinas de repetição, decisão além de leituras de sinais analógicos, rotinas de tempo e de criação de notas musicais. Dê uma olhada geral nele.

```

/*****
**
**          CURSO ARDUINO PARA INICIANTES - ROBOCORE
**
** Exemplo 11: Teremim
**
**
**
*****/

int Valor_sensor; // Variável para armazenar a intensidade de luz do LDR
int Sensor_Min = 1023; // Variável para calibrar o menor valor do sensor
int Sensor_Max = 0; // Variável para calibrar o maior valor do sensor
int Buzzer = 11; // Pino no qual o buzzer está conectado, no caso 11

int ledPin = 13; // Pino no qual o LED está conectado, no caso 13

void setup() {
  pinMode(ledPin, OUTPUT); // Inicializa o pino do LED como saída.
  digitalWrite(ledPin, HIGH); // Liga a saída do LED

  // Rotina de calibração do LDR (Tempo de duração: 5 segundos)
  while (millis() < 5000) {
    // Grava o maior valor lido no LDR
    Valor_sensor = analogRead(A0);
    if (Valor_sensor > Sensor_Max) {

```

```
        Sensor_Max = Valor_sensor;
    }
    // Grava o menor valor lido no LDR
    if (Valor_sensor < Sensor_Min) {
        Sensor_Min = Valor_sensor;
    }
}
// Desliga o LED para indicar fim da rotina de calibração do LDR
digitalWrite(ledPin, LOW);
}

void loop() {
    Valor_sensor = analogRead(A0); // Lê o valor do LDR

    // Converte o valor do sensor na faixa de frequência das notas musicais
    int nota = map(Valor_sensor, Sensor_Min, Sensor_Max, 50, 4000);
    tone(Buzzer, nota, 20); // Toca a nota por 20 ms
    delay(10); // Espera 10 ms
}
```

Uma das diferenças desse programa é que executaremos algumas tarefas dentro do **setup**, para ser mais específico será executada uma rotina de calibração do nosso teremim dentro do **setup**. Essa calibração consiste em determinar o maior e o menor valor lido pelo Arduino referente à luminosidade sentida pelo LDR. Utilizaremos a luminosidade como uma medida indireta da posição da mão da pessoa que estará tocando o teremim, quanto mais perto a mão mais escuro, quanto mais longe mais claro.

Toda essa rotina de calibração dura 5 segundos e é feita dentro do laço de repetição **while** do **setup**.

```
digitalWrite(ledPin, HIGH); // Liga a saída do LED

// Rotina de calibração do LDR (Tempo de duração: 5 segundos)
while (millis() < 5000) {
    // Grava o maior valor lido no LDR
    Valor_sensor = analogRead(A0);
    if (Valor_sensor > Sensor_Max) {
        Sensor_Max = Valor_sensor;
    }
    // Grava o menor valor lido no LDR
    if (Valor_sensor < Sensor_Min) {
        Sensor_Min = Valor_sensor;
    }
}
// Desliga o LED para indicar fim da rotina de calibração do LDR
digitalWrite(ledPin, LOW);
```

O LED do pino 13 é usado para indicar que a rotina de calibração está ativa, LED aceso rotina ativa, LED apagado rotina finalizada. Durante a calibração espera-se que a pessoa movimente a mão, que irá utilizar para tocar o teremim, para cima e para baixo em cima do LDR. Com isso o Arduino irá registrar a maior e a menor posição da sua mão e ajustará a faixa de valores possíveis melhorando a escala da leitura.

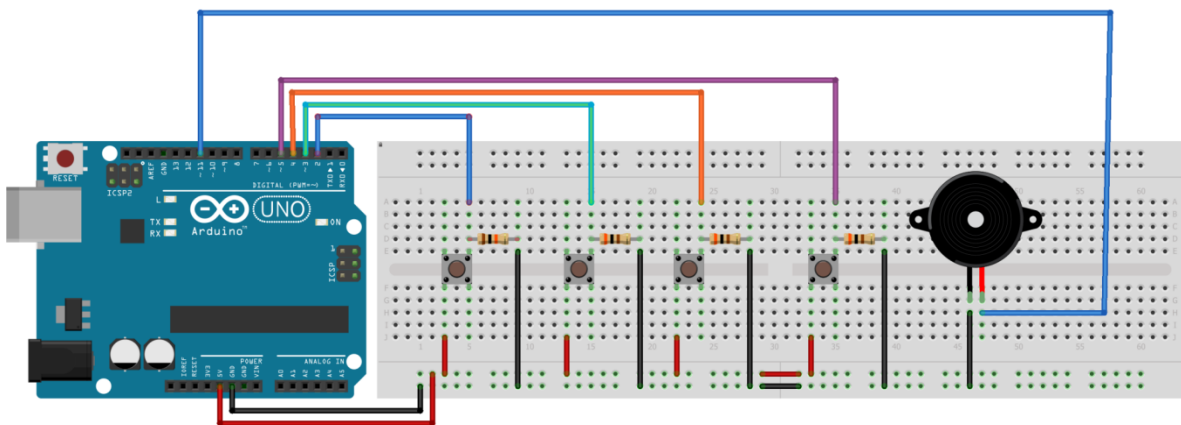
Passado a rotina de calibração ficou fácil, no loop apenas temos a função `int nota = map(Valor_sensor, Sensor_Min, Sensor_Max, 50, 4000)`, que transformará a posição da mão em uma faixa de frequências entre 50 e 4000, e a função `tone(Buzzer, nota, 20)` que tocará a frequência da nota proporcional a posição da mão.

Faça o *upload* desse programa e desperte o músico reprimido dentro de você.

3. O PROJETO FINAL: O PIANO DE 4 NOTAS

Está chegando ao fim a saga do seu treinamento inicial no mundo do Arduino, nesse momento você já deve possuir todos os conhecimentos básicos que são necessários para entender, fazer e criar o seu próprio projeto utilizando esta fantástica ferramenta. Como projeto de graduação no nosso curso eu te proponho criar um piano de 4 notas.

Logo abaixo eu coloco para você o circuito do nosso projeto, porém, gostaria que nesse momento você tentasse montar o circuito sem bisbilhotar o circuito abaixo. Caso você trave em alguma parte do circuito volte em nossos experimentos anteriores e tente rever o conceito que você está com dificuldade.



A mesma ideia serve para o programa, ele é simples, mas utiliza boa parte dos comandos básicos que trabalhamos nesse curso. Acredito que você conseguirá entendê-lo sem maiores problemas, mas a ideia de criar o programa sem “colar” do programa pronto abaixo continua valendo. Tente criar seu próprio programa com suas próprias características (lembra que eu disse que cada um tem seu jeito próprio de programar?).

Lembre-se de configurar os pinos de entrada e saída (use a dica de criar variáveis para isso), confira se os pinos que você colocou no programa são os mesmos que estão montados no circuito. Não existe uma só maneira de fazer este projeto, fique a vontade para dar seu toque pessoal, assim você poderá dizer que esse projeto é de fato seu.

Lembre-se se você travar em algum ponto consulte os experimentos anteriores porque neles estão todos os comandos básicos para se fazer qualquer projeto básico utilizando um Arduino.

www.robocore.net

4. CONSIDERAÇÕES FINAIS

Bem, nosso curso chegou ao fim, mas não chegaram ao fim os infinitos projetos que agora você pode desenvolver. Com esses mesmos componentes presentes no seu *kit* você pode criar, por exemplo, uma lâmpada que acende automaticamente quando ficar escuro, utilizando o LDR e o LED. Outra ideia que você pode por em prática, também utilizando o LDR, é criar uma distorção para seu piano do projeto final.

Com o sensor de temperatura e o *display* LCD você pode criar um termômetro para seu quarto. Projetos que usam LEDs como os do *kit* você encontrará aos montes na *internet*. Já aproveitando que falei de *internet* não deixe de acessar o *site* da RoboCore® (www.robocore.net), use e abuse do nosso fórum. Nele existem duas seções que você pode tirar grande proveito, a seção Arduino e a seção de Tutoriais, seja para tirar dúvidas seja para compartilhar com outras pessoas seus progressos e projetos.

Espero que este curso tenha sido proveitoso para você e desejo muita sorte neste fantástico mundo que você acabou de conhecer. Explore todas as possibilidades dele, seja um **Arduinizador do Mundo**.

Arduinize o Mundo
RoboCore®: Curso Arduino para Iniciantes
www.robocore.net