ARDUINO – Um Guia Básico para Iniciantes

Conheça essa plataforma de prototipagem eletrônica

Um pouco de teoria

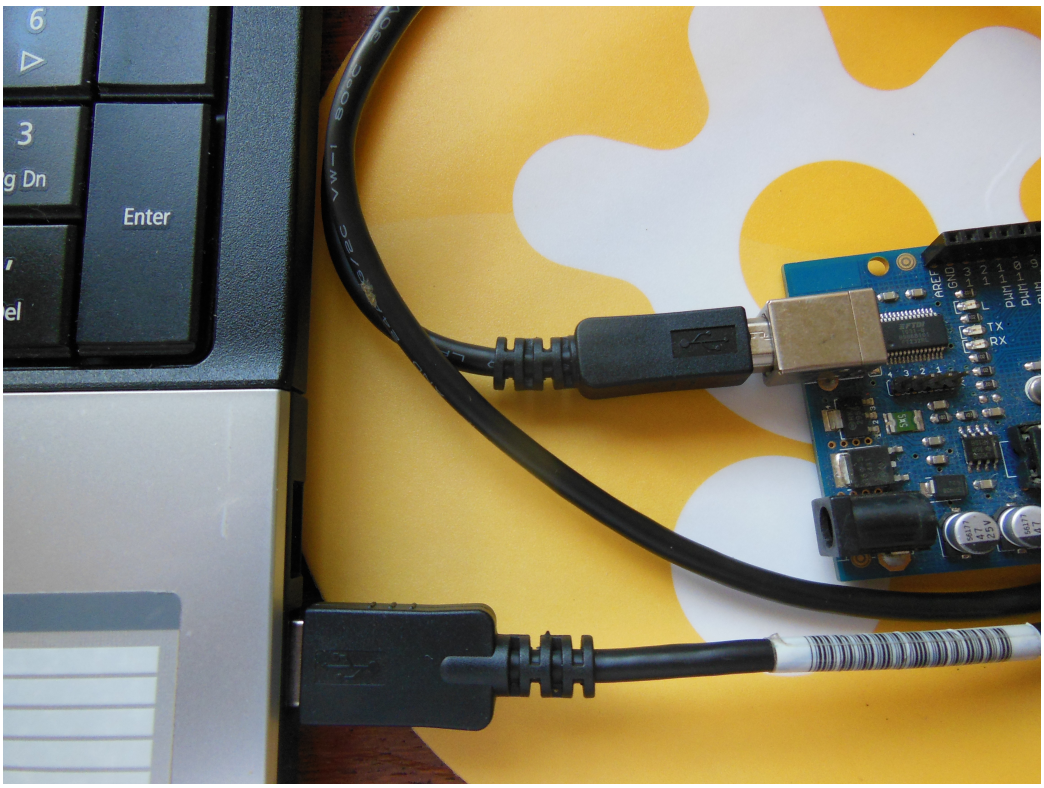
A convite do amigo de longa data Paulo Brites preparei esse *post* sobre o **Arduino**. Esse ano o Arduino está comemorando 10 anos de lançamento e continua atualizadíssimo como plataforma de prototipagem eletrônica. De uma idéia original de dois professores de Computação Física de uma escola de Artes Visuais na Itália, o Arduino conquistou o mundo inteiro e inspirou muitas outras plataformas de desenvolvimento de *hardware* como o **Raspberry Pi**, criado na Inglaterra há pouco mais de 2 anos. Mas, afinal, o que é esse tal de Arduino e para que serve?

Bom, vamos devagar e divagar. Pense no Arduino como uma pequenina placa eletrônica um pouco maior que um cartão de crédito onde estão montados um processador digital, um regulador de tensão de 5 volts, uma interface USB e alguns conectores onde estão as portas de entrada e saída de controle. O processador do Arduino é um microcontrolador de 8 bits, o ATmega328 da Atmel, uma fabricante americana de circuitos integrados.

Mas, qual a diferença de um microcontrolador para um microprocessador? O pessoal da antiga deve lembrar bem dos nomes 8080, 6500 e Z-80, os microprocessadores que equipavam os primeiros computadores pessoais, como o TRS-80 (CP-500), o Apple-II, o Commodore-64 e o Sinclair ZX81. Aqueles chips tinham 40 pinos e processavam dados em barramentos de 8 bits (1 byte) e precisavam de vários outros circuitos integrados externos, como memórias RAM, *EPROM*, controladores de entrada e saída para teclado, *floppy-disk* e um circuito de RF para entrada em uma TV analógica que era usada como monitor. Internamente esses processadores eram somente uma CPU e um conjunto de registradores de 8 e 16 bits. Com o avanço da tecnologia esses microprocessadores evoluiram para sistemas muito densos com quase 500 pinos e barramentos de 32 e 64 bits, como os que equipam nossos atuais PCs, mas que ainda precisam de memórias e controladores externos para formar um computador completo. Você se lembra do Pentium e do Athlon?

Os microcontroladores são sistemas bem mais simples, operam ainda com 8 ou 16 bits (alguns já com 32 bits) mas que já vem internamente montados com memórias RAM, EPROM e *Flash*, e incorporam interfaces de entrada e saída com várias funções multiplexadas em muito menos pinos. Alguns microcontroladores tem somente 8 pinos! Por integrarem tudo numa só pastilha de silício, são chamados também de *computadores em um chip*. E como todo computador, os microcontroladores podem ser programados utilizando uma linguagem de programação padronizada.

O Arduino é um computador, já que vem montado, ou na linguagem corrente: embarcado, com um microcontrolador. Os primeiros modelos vinham com os microcontroladores ATmega8 e ATmega128 de 8 bits com 28 e 64 pinos, e interface serial RS-232 para ser conectada a um outro computador, um PC *desktop* ou *notebook*. Hoje todos os modelos já vem com interfaces seriais USB, I2C e SPI, entre outras. Em sua pequenina placa de circuito impresso são também montados dois conectores com as portas de entrada e saída digitais e um com as portas de entradas analógicas. A alimentação do Arduino de 5 volts é tomada diretamente da porta USB do computador pessoal ao qual ele está conectado.



Tudo bem, mas para que serve o Arduino? Basicamente o Arduino, como qualquer computador, pode detetar níveis lógicos ou coletar tensões variáveis de qualquer tipo de sensor em suas entradas, e a partir destas pode acionar alarmes, solenóides e motores. Bacana, mas é complicado programar o Arduino? Não, não é complicado escrever ordens na forma de textos para o Arduino executar. Esses textos, ou **códigos** como os programadores os chamam, são escritos segundo o padrão, chamado de **sintaxe**, da linguagem escolhida e armazenados numa das memórias internas do microcontrolador embarcado no Arduino, e são executadas uma a uma. Você se lembra de uma linguagem de programação bem antiga chamada *Basic*? Nela cada linha era numerada e continha uma ordem simples, como o comando *print* que mostrava na tela do computador uma mensagem simples escrita entre aspas. O Arduino tem uma linguagem própria também muito fácil e parecida com a consagrada linguagem C, esta criada nos anos 70. Mais adiante vamos propor experimentos com essa linguagem e programar o Arduino.

Tendo agora uma visão bem geral do que é e para que serve o Arduino, vamos por as mãos na massa? A primeira coisa que você, caro leitor, vai precisar é de um Arduino, claro. E também de computador pessoal. Qualquer um serve: um PC *desktop* ou *notebook* padrão **IBM** ou **Apple**. Não importa o sistema operacional do seu PC, qualquer um também serve: **Windows**, **Linux** ou **Mac OS-X**, o Arduino tem versões de *software* para todas essas plaformas. Por último, você vai precisar de um cabo USB para conectar o Arduino ao seu PC. Esse cabo tem que ter um conector USB tipo A macho numa ponta e uma conector USB macho tipo B na outra. Esse é mesmo cabo que você usa para conectar seu PC a sua impressora USB. Para nossas primeiras experiências qualquer modelo de Arduino serve. Hoje existe uma dezena de modelos; o mais comum, e mais barato, é o modelo *duemilanove*. O código escrito para um pode ser executado em qualquer outro modelo, com algumas poucas excessões, no caso dos novos modelos disponíveis mais potentes. Você não vai precisar de qualquer tipo de fonte de alimentação, como dissemos acima o Arduino é alimentado pela fonte do seu PC atraves do cabo USB.

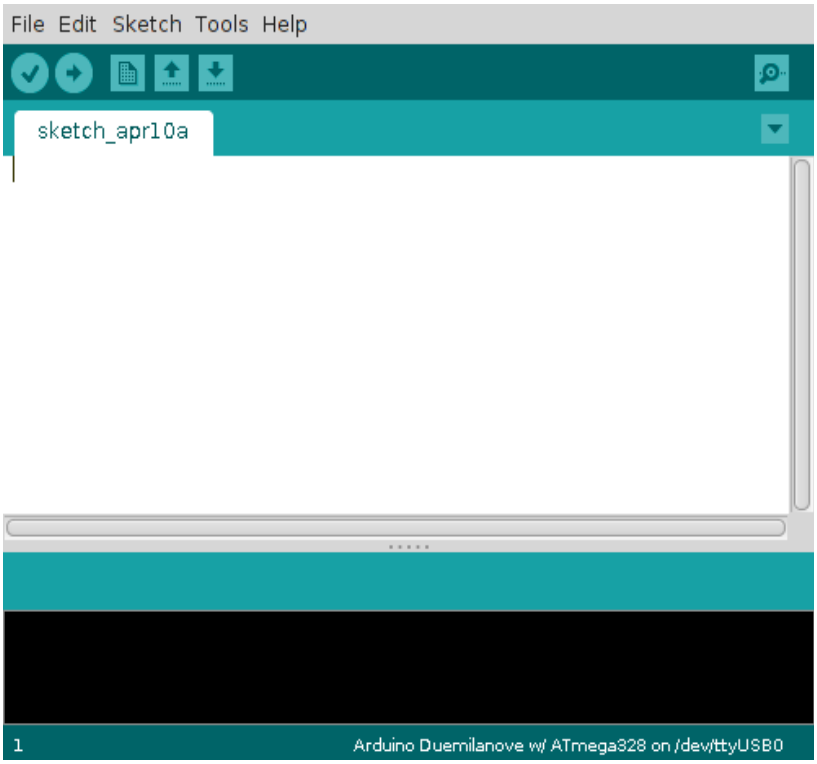
Um pouco de prática passo-a-passo

01 Download do software

O *software* para programação do Arduino pode ser livremente baixado da internet no endereço www.arduino.cc, o site oficial dessa plataforma. Nesse endereço *web*, clique na aba **Download** no topo da página e, a direita da nova página que abrir, selecione a versão para o sistema operacional do seu PC. Por ora, clique em **Just Download** e descarregue o arquivo compactado para sua área de trabalho (*Desktop*). Feche a página, se desejar.

02 Conexão Arduino-PC

Selecione uma porta USB no seu PC e conecte a ela o Arduino atraves do cabo USB de impressora. Repare que o LED verde PWR vai permanecer acesso, indicando que o Arduino está corretamente alimentado. Descompacte o arquivo baixado e repare que na sua área de trabalho é criada uma pasta com várias subpastas e dois outros arquivos, um executável e um de texto. Dê um duplo clique no arquivo executável para instalar e executar o aplicativo de programação do Arduino. Esse aplicativo é chamado de ambiente de desenvolvimento integrado, mais conhecido como IDE (*Integrated Development Environment*). A tela do IDE após sua execução é o da figura abaixo.



03 Configurando o IDE

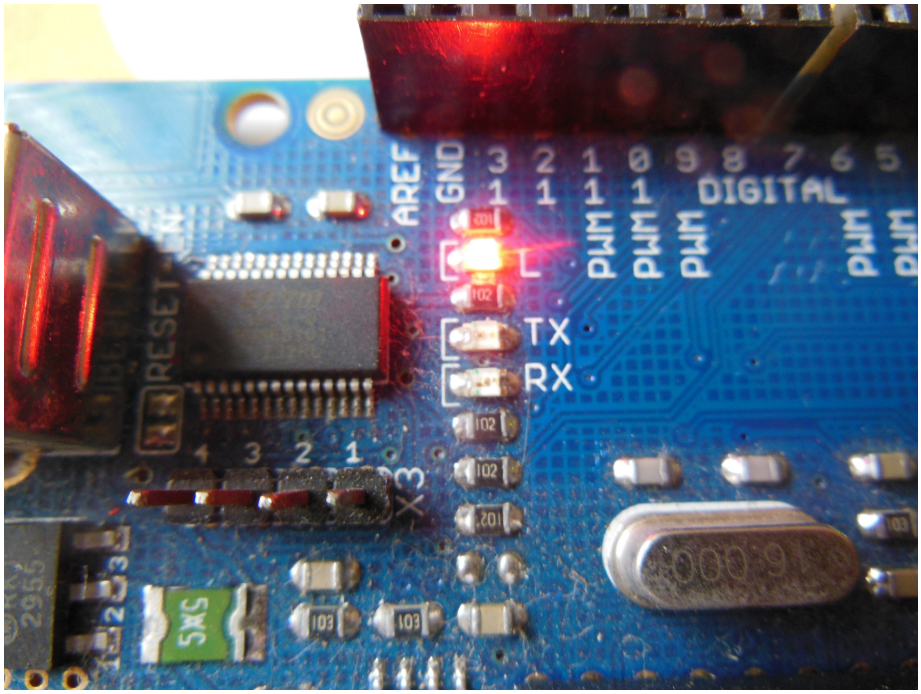
No menu de opções no topo do IDE clique em **Tools>Board** e selecione o modelo de Arduino que você conectou ao seu PC. Para os experimentos desse *post* estamos usando o Arduino *Duemilanove*. Observe na barra inferior do IDE se o aplicativo reconheceu e já selecionou uma porta USB para se comunicar com o Arduino, se não vá em **Tools>Serial Port** e selecione uma porta porta serial disponível.

04 Primeiro experimento: Hello World!

Normalmente o Arduino já vem com um LED vermelho conectado a sua porta digital 13. Nesse primeiro experimento vamos fazer com que esse LED vermelho pisque numa frequência de 1 pulso por segundo. Para isso vamos aproveitar um programa já pronto que vem com alguns outros dentro do IDE do Arduino. No topo da tela do IDE, clique no botão **Open**, aquele com uma seta para cima, e selecione **01.Basics>Blink** para carregar na área de programação do aplicativo o código desse programa *Blink.*

05 Fazendo o upload do código no Arduino

Para fazer o LED no pino digital 13 piscar, é necessário fazer o upload do programa *Blink* na memória Flash do microcontrolador do Arduino. Clique no botão **Upload**, o com uma seta apontada para a direita. Observe que no Arduino os LEDs TX e RX piscam rapidamente, indicando o estabelecimento da comunicação entre este e o seu PC. Por fim, observe que o LED vermelho do pino 13 fica 1 segundo aceso e outro segundo apagado. Parabéns, seu primeiro programa no Arduino funcionou!



06 Entendendo o código carregado

Vamos agora entender o que faz esse LED vermelho ficar piscando numa frequência de 1Hz.
O código carregado, mais conhecido como **Sketch**, no Arduino tem as seguintes linhas de instruções:

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.
   This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Logo no topo do *sketch*, o texto multilinhas entre **/*** e ***/** não tem qualquer efeito no programa, são somente linhas de comentários com informações sobre o programa e podem ser eliminadas. Da mesma forma, qualquer texto depois de duas barras (//) são igualmente comentários, de uma só linha, e podem também ser eliminadas.

A primeira instrução do programa está na linha ***int led = 13;***. Essa instrução é a declaração da variável usada no programa, e dá o nome *led* ao pino 13 do Arduino; o prefixo *int* indica que *led* é uma variável inteira. Repare que o comando deve sempre terminar com um ponto-e-vírgula, como na linguagem de programação C.

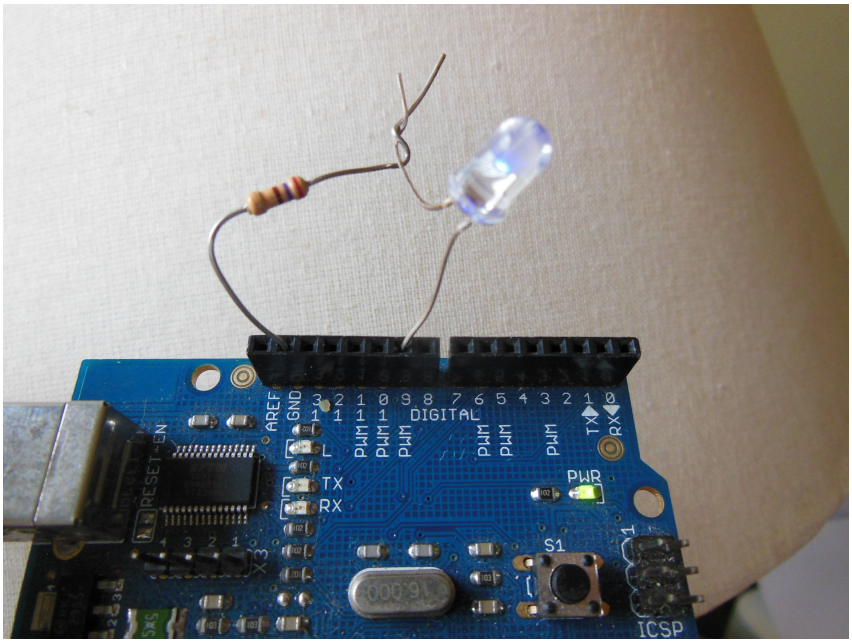
A seguir vem as duas funções obrigatórias em todo programa na linguagem do Arduino: as funções **void setup()** e **void loop()**. A primeira serve para configurar os pinos e portas de comunicação do Arduino; aqui o pino 13 (batizado de *led*) é configurado como saída digital com a função *pinMode(led, OUTPUT)*.
A função *void loop()* encapsula os comandos que deverão ser executados continuamente pelo microcontrolador do Arduino; aqui o pino 13 recebe uma tensão de 5 volts durante 1 segundo, com as funções *digitalWrite(led, HIGH)* e *delay(1000)*. Depois esse pino 13 recebe 0 volt por mais 1 segundo, com as funções *digitalWrite(led, LOW)* e *delay(1000)*. Esse ciclo se repete infinitamente, fazendo o LED piscar até que o Arduino seja resetado. Repare que os comandos dessas duas funções obrigatórias devem estar sempre dentro das chaves **{** … **}**.

07 Modificando o programa

Experimente modificar o tempo em que o LED vermelho no pino 13 do seu Arduino fica aceso e o tempo em que ele fica apagado mudando os valores entre os parênteses nas duas funções **delay()**. Mude um desses valores para 500 (milissegundos) e o outro para 250, por exemplo. Carregue novamente o *sketch* no Arduino clicando no botão *upload* e observe o LED. Em menos de um minuto você conseguiu mudar a frequência de operação do seu oscilador montado com o Arduino. Se esse circuito não fosse programável, você teria provavelmente que ajustar mecanicamente um potenciômetro, ou trocar um resistor ou um capacitor montado na placa.

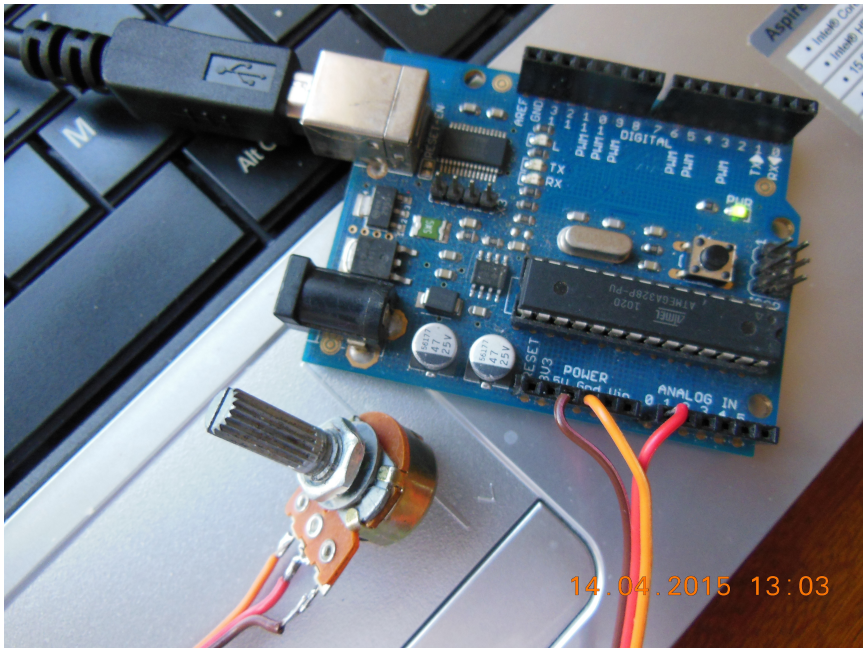
08 Inserindo um LED e um resistor externo

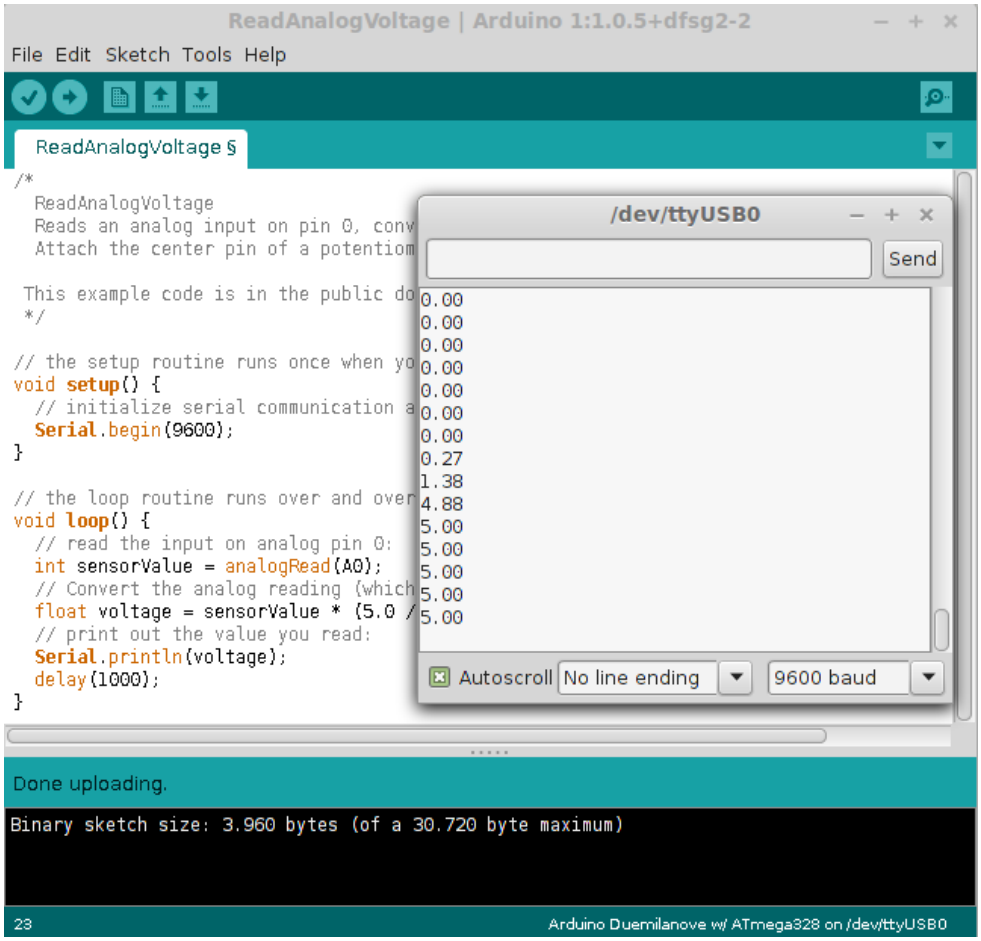
Nesse experimento vamos utilizar uma outra porta digital do Arduino e nela conectar um LED externo em série com um resistor limitador de corrente. Na porta digital 9 encaixe o terminal de um LED correspondente ao anodo, aquele com o maior comprimento. Conecte agora um dos terminais de um resistor de 270 ohms de baixa potência no terra (GND) das barras de portas digitais, o penúltimo à esquerda da segunda barra. Por fim, una os dois terminais livres desses componentes, simplesmente torcendo-os com os dedos. Agora, no IDE do Arduino clique no botão **Open** e selecione **01.Basics > Fade**, para carregar o *sketch Fade*. Esse programa gera digitalmente no pino 9 uma tensão analógica que varia de 0 a 5 volts, fazendo com que o brilho do LED vá aumentando devagar até atingir seu brilho máximo, quando então vai diminuindo até apagar; esse o ciclo se repete infinitamente. Confirme o funcionamento do circuito deixando mais lento o processo mudando na última linha do programa o valor na função *delay()* de 30 para 1000 (milissegundos) e meça com um voltímetro digital essa tensão variável entre os pinos 9 e terra.



09 Medindo tensões analógicas com o Arduino

Nesse último experimento vamos fazer um voltímetro digital DC para baixas tensões com o Arduino. Primeiramente, solde 3 pequenos pedaços de fios nos 3 terminais de um potenciômetro de qualquer valor entre 1Kohms e 100Kohms. Insira o fio soldado ao terminal central do potenciômetro no pino analógico **A0** da barra de entradas analógicas do Arduino, marcada como *ANALOG IN*. Conecte um dos outros dois fios do potenciômetro no ponto de 5 volts e o outro no terra (GND), na barra *POWER* do Arduino. Agora clique no botão *Open* e selecione **01.Basics** > **ReadAnalogVoltage**. Acrescente dentro da função *loop()*, depois de *Serial.println(voltage);* a seguinte função de temporização: **delay(1000);** e faça o *upload* do programa. Por fim, clique no botão **Serial Monitor** do Arduino, bem à direita da barra de botões do IDE. Veja as tensões lidas pela porta analógica A0 a cada segundo. Gire o cursor do potenciômetro e observe que os valores medidos variam conforme a nova posição, de 0 a 5 volts.





Conclusão

Bem, aqui terminamos essa pequena introdução prática ao Arduino. Espero que tenhamos alcançado nosso intento, esclarecer muitas das dúvidas dos leitores novatos nessa formidável plataforma de desenvolvimento eletrônica. Se conseguimos, nos propomos a escrever num futuro próximo um novo *post* com novos e fáceis experimentos. Por último, quero deixar um brinde para aqueles que desejarem se aprofundar na linguagem do Arduino: em minha página na internet www.ordemnatural.com.br o leitor poderá baixar gratuitamente a **Cartilha de Programação em C para o Arduino**, um livreto de 20 páginas no formato pdf. Lá também o leitor poderá adquirir meu livro digital **Experimentos com o Arduino - edição 2.0** no formato *kindle/mobi*.

João Alexandre da Silveira – ordemnatural@outlook.com – www.ordemnatural.com.br