Aprenda a programar o Arduino – Parte I

[Arduino](http://amcollucci.com.br/category/eletronica/arduino/), [Engenharia](http://amcollucci.com.br/category/engenharia/), [Informática](http://amcollucci.com.br/category/informatica/)

por [Anthony Collucci](http://amcollucci.com.br/author/Anthony%20Collucci/)

Esse será o primeiro post de uma série onde pretendo ensinar a programar o Arduino. Esse tutorial leva em conta que você ja tenha instalado e configurado a ide do Arduino na sua máquina e que você já saiba ao menos o qué é o Arduino, algum conhecimento prévio de programação e bem vindo!

Primeiro programa no Arduino

Vamos fazer agora um [programa](http://amcollucci.com.br/2012/02/04/aprenda-a-programar-o-arduino-parte-i/) que não serve literalmente para nada!  Simplesmente vamos ver o que é obrigatório para o código poder roda veja abaixo:

*/\*                               Codigo Exemplo Numero 1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Aprenda a programar o Arduino – Primeiros PassosAutor: Anthony Collucci  
Data: 04/02/2012  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*void setup(){*

*/\* O setup é uma estrutura obrigatória, ele é utilizado para definirmos a  
função (i/o) dos pinos do Arduino, serve tambem para inicializarmos  
algumas bibliotecas.   \*/*

*}*

*void loop(){*

*/\* O loop tambem e uma estrutura obrigatória, ele executa o codigo  
entre {} em ciclos infinitos e repetitivos.  \*/*

*}*

No exemplo acima utilizamos as duas estruturas básicas Setup() e Loop() que são duas funções como veremos mais adiante. O texto entre “/\* \*/” é simplesmente um comentário ele não afeta em nada a execução do programa, é interessante que o comentário exista para facilitar a leitura e manutenção do código, mas deve -se evitar seu uso em excesso para não diminuir a legibilidade do código com muitas quebras. Isso aí e uma coisa que você vai [adquirindo](http://amcollucci.com.br/2012/02/04/aprenda-a-programar-o-arduino-parte-i/) com o tempo!!!

Quando você compilar deve receber um resultado como esse:

*Binary sketch size: 674 bytes (of a 126976 byte maximum)*

Esse resultado indica qual o tamanho do arquivo e que ele foi compilado com sucesso.

Repare que se você utilizar o seguinte codigo:

*void setup(){*

*}*

*void loop(){*

*}*

O tamanho do arquivo gerado será o mesmo indicando que os comentários são ignorados após a compilação. Agora atente para o codigo abaixo digite-o e compile:

*Void setup(){*

*}*

*Void loop(){*

*}*

O seguinte erro será exibido:

sketch\_feb04a:-1: error: ‘Void’ does not name a type  
sketch\_feb04a:0: error: ‘Void’ does not name a type  
sketch\_feb04a:1: error: ‘Void’ does not name a type  
sketch\_feb04a:5: error: ‘Void’ does not name a type

Isso indica que o “Void” não e um tipo válido para uma váriavel ou função, isso ocorre porque a linguagem usada no arduino e baseada no c/c++ que são linguagens case sensitive.

Começando a brincar com variáveis e comunicação serial

Em todo tutorial de Arduíno que encontram o primeiro exemplo é o do Led Blink, porém acho uma abordagem totalmente inútil, pois ela realmente não ensina nada… Vou começar por outro caminho. Vamos iniciar a comunicação serial e escrever o conteudo de uma váriavel na no console.

Digite esse codigo em sua IDE:

/\*  
Codigo Exemplo Numero 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Aprenda a programar o Arduino – Primeiros Passos

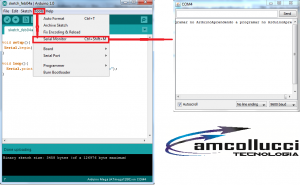
Autor: Anthony Collucci  
Data: 04/02/2012

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*/  
void setup(){  
//Inicia a comunicação serial a 9600 bauds  
Serial.begin(9600);

}

void loop(){  
// Envia a string “Aprendendo a programar no arduino” para a console  
Serial.print(“Aprendendo a programar no arduino”);  
}

Agora acesse a serial monitor conforme a figura abaixo:

[](http://amcollucci.com.br/wp-content/uploads/2012/02/abrindo-o-serial-monitor.png)

Abrindo o Serial Monitor- Opening the Serial Monitor

O resultado como se segue e uma sucessão de impressões da string “Aprendendo a programar no arduino”. Vimos que iniciamos a comunicação serial através do comando *serial.begin(9600)*que colocamos dentro do*setup(){} .*O  valor de 9600 pode variar dependendo da sua aplicação veja mais sobre velocidade da porta serial nesse post:

<http://www.tech-faq.com/difference-between-bit-rate-and-baud-rate.html>

O comando *Serial.print(“string aqui”)*que faz com que a informação seja impressa no serial monitor. Adiante veremos outros exemplos do uso do Serial.print, mas por ora quero que reparem que todo comando que não constroí um bloco (não tem “chaves” como o setup e o loop) ou então que não e uma diretiva de pré-processador (#include, #define que veremos adiante tambem)  termina com ponto e vírgula, isso indica o fim do comando.

Digite agora o código abaixo, e veremos um exemplo de uso de váriavel:

/\*  
Codigo Exemplo Numero 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Aprenda a programar o Arduino – Primeiros Passos

Autor: Anthony Collucci  
Data: 04/02/2012

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*/

/\* Podemos declarar uma variavel por linha \*/  
int a;  
int b;

/\* Ou podemos declarar diversas variaveis por linha \*/  
int c,d;

void setup(){  
//Inicia a comunicação serial a 9600 bauds  
Serial.begin(9600);  
a = 2;  
b = 3;  
}

void loop(){  
// Observe o \n ele e um escape de new line, ou seja muda a linha  
Serial.print (“Aprendendo a programar no Arduino \n”);  
/\* Veja como e facil fazer operações com o arduino, as variaveis  
c e d receberao os valores da soma e multiplicação de a e b.\*/  
c=a+b;  
d=a\*b;  
/\* depois de efetuado o calculo imprimimos o valor na console \*/  
Serial.print (“O valor da soma de a e b é: “);  
Serial.print (c);  
Serial.print (“\n”);  
Serial.print (“O valor da multiplicação de a e b é: “);  
Serial.print (d);  
Serial.print (“\n”);  
delay (1000);  
}

Agora começamos a complicar um pouquinho mais o nosso código =]. Mas o que fizemos e muito simples criamos 4 variaveis a,b,c,d que são do tipo [int](http://arduino.cc/en/Reference/Int" \o "Tipo Int" \t "_blank) ou seja podem receber [números inteiros](http://www.brasilescola.com/matematica/numeros-inteiros.htm). Depois dentro do setup iniciamos a comunicação serial e atribuimos valores para as variaveis a e b. Dentro do loop, nós realizamos a operação de soma de “a” e “b” e colocamos o resultado na variável “c” depois realizamos a operação de multiplicação da váriavel “a” e “b” e colocamos na variavel “d”, por ultimo usamos o Serial.print e imprimimos o resultado na console. O comando [delay (1000)](http://arduino.cc/en/Reference/Delay" \o "Referência do Comando delay" \t "_blank) faz com que haja um espera de um segundo entre uma repetição e outra.

Boas práticas…

Vamos agora dar uma olhada em alguns trechos do código e torna-los mais “profissional”. Nesse post só vou tratar da diminuição do tamanho do programa realizando o tratamento de variáveis e tornando assim o codigo mais limpo rápido e menor.

No exemplo 3 a IDE nos retorna a seguinte [mensagem](http://amcollucci.com.br/2012/02/04/aprenda-a-programar-o-arduino-parte-i/):

Binary sketch size: 4326 bytes (of a 126976 byte maximum)

Agora vamos analisar alguns trechos do código e realizar alterações.

As seguintes linhas:

/\* Podemos declarar uma variavel por linha \*/  
int a;  
int b;

/\* Ou podemos declarar diversas variaveis por linha \*/  
int c,d;

serão alteradas para

/\* Declaração de constantes \*/

const unsigned int a = 2;  
const unsigned int b = 3;

/\* Declaração de variáveis. \*/

int c,d;

e as seguintes linhas serão deletadas:

a = 2;  
b = 3;

Por quê realizamos essas alterações? Simples, você deve ter notado que os valores de “a” e “b” nesse código são constantes ou seja não se alteram durante a execução do programa, então não deixamos um espaço de memória alocada para uma variável do tipo int, simplesmente declaramos diretamente seu valor. O comando “const” declara ela como constante. E como no exemplo o sinal dos operadores não influencia ja que são ambos positivos, nos suprimimos ele com o “unsigned”.

Quando voce compilar vair obter o seguinte resultado:

Binary sketch size: 4274 bytes (of a 126976 byte maximum)

Economizamos 52 bytes  vamos prosseguir. No exemplo abaixo ja fazemos algumas alterações substanciais no código:

/\*  
Codigo Exemplo Numero 3.1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Aprenda a programar o Arduino – Primeiros Passos

Autor: Anthony Collucci  
Data: 04/02/2012

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*/

/\* declaramos as constantes \*/  
const unsigned int a = 2;  
const unsigned int b = 3;

/\*declarmos a variavel resultado\*/  
unsigned int res;

void setup(){  
//Inicia a comunicação serial a 9600 bauds  
Serial.begin(9600);

}

void loop(){  
// Observe o \n ele e um escape de new line, ou seja muda a linha  
Serial.print (“Aprendendo a programar no Arduino \n”);  
/\* Veja como e facil fazer operações com o arduino, as variaveis  
c e d receberao os valores da soma e multiplicação de a e b.\*/

//imprimimos primeiro o valor da soma  
res=a+b;  
Serial.print (“O valor da soma de a e b é: “);  
Serial.print (res);  
Serial.print (“\n”);  
//depois o valor da multiplicação  
res=a\*b;  
Serial.print (“O valor da multiplicação de a e b é: “);  
Serial.print (res);  
Serial.print (“\n”);  
delay (1000);  
}

Reduzir o numero de váriaveis não reduziu o código, mas aplicar o “unsigned” foi efetivo. A [vantagem](http://amcollucci.com.br/2012/02/04/aprenda-a-programar-o-arduino-parte-i/) de diminuir as váriaveis e que seu codigo usa menos memória ram. Agora o resultado foi esse:

Binary sketch size: 4172 bytes (of a 126976 byte maximum)

Acabamos de economizar 154 bytes.

Por ultimo reduzindo o [escopo das variaveis](http://www.mtm.ufsc.br/~azeredo/cursoC/aulas/c760.html) conseguimos reduzir ainda mais o tamanho do código:

Retiramos as linhas abaixo que estavam acima do setup(){}

/\* declaramos as constantes \*/  
const unsigned int a = 2;  
const unsigned int b = 3;

/\*declarmos a variavel resultado\*/  
unsigned int res;

E recolocamos dentro do loop:

void loop(){  
/\* declaramos as constantes \*/  
const unsigned int a = 2;  
const unsigned int b = 3;

/\*declarmos a variavel resultado\*/  
unsigned int res;

O novo resultado foi:

Binary sketch size: 4140 bytes (of a 126976 byte maximum)

Ou seja conseguimos 182 bytes de redução no código. Essa técnica é bem criticada por muitos programadores, já houve um caso que quase fui crucificado aqui no blog, e o assunto foi tão polêmico que preferi retirar a matéria. Mas ela é uma técnica muito válida quando temos um projeto complexo a ser feito no Arduino, normalmente em projetos complexos temos de incluir várias bibliotecas, que por si só ja impactam o tamanho do programa, não precisamos então agravar isso criando programas ruins!

Bom em breve postarei a continuação desse post. Espero que ele tenha sido útil!

*Att,*

*Anthony Collucci*

Aprenda a programar o Arduino – Parte 2

[Arduino](http://amcollucci.com.br/category/eletronica/arduino/), [Eletrônica](http://amcollucci.com.br/category/eletronica/), [Engenharia](http://amcollucci.com.br/category/engenharia/), [Linguagem C](http://amcollucci.com.br/category/informatica/linguagem-c/)

por [Anthony Collucci](http://amcollucci.com.br/author/Anthony%20Collucci/)

Quando estava montando um braço mecânico na faculdade precisei de usar algumas estruturas que não**rodassem em loop**, me deparei com a pergunta como faço para executa-las fora do loop do Arduino sendo que ele trata essa função como se fosse uma main() **(por ser obrigatória)**  do C (dummies abaixo eu explico [melhor](http://amcollucci.com.br/2012/02/12/aprenda-a-programar-o-arduino-parte-2/)ok ;) )

No Arduino o [programa](http://amcollucci.com.br/2012/02/12/aprenda-a-programar-o-arduino-parte-2/) começa a ser executado pela função Setup(). Depois de concluído ele procura a função loop() e inicia sua execução, o que eu fiz e vou mostrar nesse post e como interromper o fluxo e apontar uma função que você criou para ser executada. É extremamente simples, pode até parecer boçal… Mas eu passei algumas boas horas para ter a epifania, **pois ao imaginar o codigo do arduíno ele parece tratar a setup como se fosse uma função para configurar o mesmo, mas na verdade dentro do proprio loop, você pode setar um pino, dar um atach numa interrupção etc…**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Observe que não deixamos de executar o loop como bem observado pelo bubulindo no forum do Arduino, como vocês podem conferir a discussão no link abaixo:**

[*http://arduino.cc/forum/index.php/topic,93283.0.html*](http://arduino.cc/forum/index.php/topic,93283.0.html)

*\* Trechos em negrito inseridos para corrigir erro na publicação original*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Então vejamos o seguinte código:

void setup(){

Serial.begin(9600);

naoqueroloop();

}

void loop(){

}

void naoqueroloop(){

Serial.print(“Estou fora do loop”);

}

Se você executou corretamente quando abrir o Serial monitor obterá a seguinte tela.

[](http://amcollucci.com.br/wp-content/uploads/2012/02/saida-serial-monitor.png)

Saída Serial Monitor Arduíno

Sem o meu logo é claro… Agora veja como o programa seria executado se não houvessemos criado essa função:

void setup(){

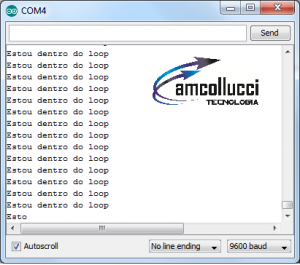
Serial.begin(9600);

}

void loop(){  
Serial.print(“Estou dentro do loop \n”);

}

O código acima gera a seguinte saída:

[](http://amcollucci.com.br/wp-content/uploads/2012/02/Saida-Serial-monitor-loop.png)

Saida Serial monitor loop()

Bom já que estamos aqui não vou gastar o post apenas explicando isso. Vamos falar um pouco tambem sobre funções. Funções servem para agrupar comandos que juntos fazem alguma tarefa.

Veja a definição segundo o Site do curso de C da UFSC :  “Uma função é um bloco de código de programa que pode ser usado diversas vezes em sua execução. O uso de funções permite que o programa fique mais legível, mais bem estruturado. Um programa em C consiste, no fundo, de várias funções colocadas juntas. ”

O corpo de uma função e composto pelo tipo de retorno, nome da funçao, parametros, chaves de inicio e fim da função, e comandos..

tipo nome\_funcao (parametros) {

comando\_1;

comando\_2;

}

Funçoes podem ter o tipo void, int, float e qualquer outro que uma váriavel possa receber, nos exemplos anteriores vimos vários exemplos de funçoes do tipo “**void**” , esse tipo de função não retorna nenhum valor, ou seja apos ser chamada ela cumpre sua função e não devolve nenhum resultado para o programa.

Para esclarecer melhor vejamos uma função do tipo int.

void setup(){

Serial.begin(9600);

}

void loop(){

int numero1=8;  
int numero2=10;  
int resultado=soma(numero1,numero2);  
Serial.print(resultado);  
Serial.print(“\n”);  
delay(1000);  
}

int soma(int a, int b){  
int c;  
c=a+b;  
return c;  
}

No programinha acima, temos uma função chamada soma(), essa função e do tipo int, e tem dois parametros, a e b, do tipo inteiro ou em [inglês](http://amcollucci.com.br/2012/02/12/aprenda-a-programar-o-arduino-parte-2/) integer. Dentro dessa função foram declaradas 3 variaveis a,b,c duas delas tem seu valor obtido através da passagem de parametros *soma(numero1,numero2)* nesse momento o valor de *a* e *b*passam a ser os mesmos de *numero1* e *numero2*, logo abaixo a variável c recebe o valor da soma de *a* e *b*esse valor é retornado atraves do comando *return c;* Já no loop() a variável resultado recebe o valor de retornado pela função *soma()*.

Até agora tá tranquilo não é?

Então dá uma olhada nesse programa e tentem entender, tem muita coisa que eu não expliquei ainda mas é bom pra dar uma exercitada, e tambem para quem já tem algum conhecimento, o programa a seguir imprimi no serial monitor o triângulo de pascal.

/\*  
Codigo Exemplo Triangulo de Pascal

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Aprenda a programar o Arduino – Primeiros Passos

Autor: Anthony Collucci  
Data: 12/02/2012

Esse algoritmo não foi otimizado  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*/

/\* sendo k o numero da coluna e n o numero da linha\*/  
int k,n;

/\*  
criaremos uma matriz de 20 por 20 para armazenar os valores  
o que nos dará 400 numeros na memoria!!!!  
\*/

int numero [19][19];

void setup(){

Serial.begin(9600);  
zera\_variaveis();  
pascal();  
}

void loop(){  
}

void pascal (){

/\*inia o calculo pela linha 1 coluna 1\*/  
k=n=1;  
numero [n][k]=1;  
Serial.print(numero [n][k]);  
Serial.print(“\n”);

for (n=2;n<19;n++){  
for (k=1;k<=n;k++){  
numero [n][k] = efetua\_calculo();  
Serial.print(numero [n][k]);  
Serial.print(“  “);  
delay(100);  
}  
Serial.print(“\n”);  
}

}

int efetua\_calculo(){  
int resultado;

if (numero[n-1][k]==0){  
resultado = numero [n-1][k-1];  
return resultado;  
}  
if (numero[n-1][k-1]==0){  
resultado = numero [n-1][k];  
return resultado;  
}

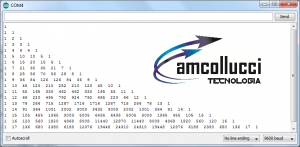
if (numero[n-1][k-1]!=0 && numero[n-1][k]!=0){  
resultado = numero [n-1][k-1] + numero [n-1][k];  
return resultado;  
}  
return 0;  
}

void zera\_variaveis(){

/\* Fui obrigado a [criar](http://amcollucci.com.br/2012/02/12/aprenda-a-programar-o-arduino-parte-2/) essa funçao pois ao criar as variaveis  
elas vem com um valor aleatorio, lembrando que a matriz começa  
no indice 0 mas estamos facilitando as coisas

\*/  
for (n=1;n<19;n++){  
for (k=0;k<19;k++){  
numero[n][k]=0;  
}  
}  
}

Aqui você vê a saída deste programa:

[](http://amcollucci.com.br/wp-content/uploads/2012/02/pascal.png)

Triângulo de Pascal - Arduíno

Fico por aqui até a próxima!

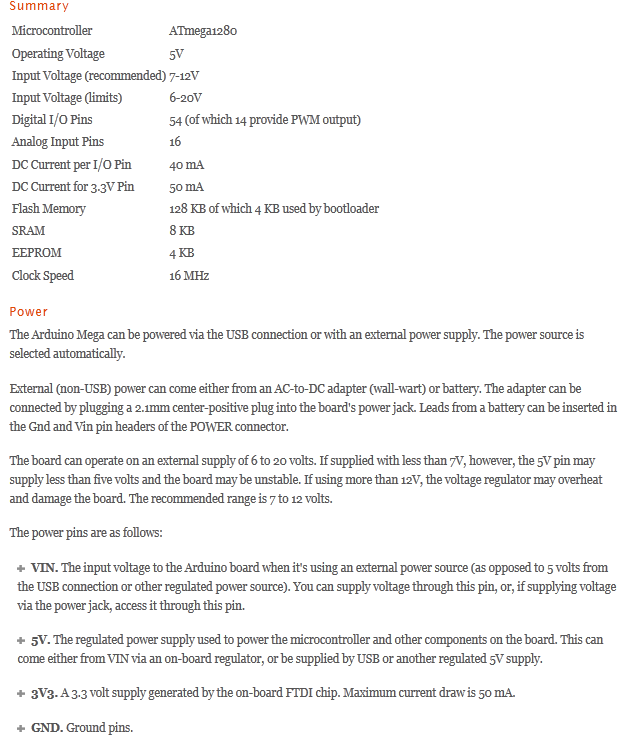
Aprenda a programar o Arduino – Parte 3

[Arduino](http://amcollucci.com.br/category/eletronica/arduino/), [Eletrônica](http://amcollucci.com.br/category/eletronica/), [Engenharia](http://amcollucci.com.br/category/engenharia/), [Informática](http://amcollucci.com.br/category/informatica/), [Linguagem C](http://amcollucci.com.br/category/informatica/linguagem-c/)

por [Anthony Collucci](http://amcollucci.com.br/author/Anthony%20Collucci/)

Estamos chegando à terceira Parte desse tutorial, hoje começaremos a trabalhar com o arduino ligado a algum[hardware](http://amcollucci.com.br/2012/02/18/aprenda-a-programar-o-arduino-parte-3/) e advinha … vamos piscar leds….

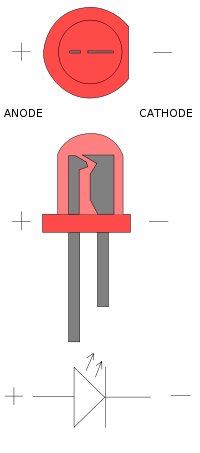
Antes de ligarmos qualquer coisa no Arduíno é interessante sabermos um pouco sobre suas características elétricas, aqui vai o datasheet do Arduino Mega 1280 para outros modelos  basta procurar no [http://arduino.cc](http://arduino.cc/).

[](http://amcollucci.com.br/wp-content/uploads/2012/02/datasheet.png)

Então sabemos que o máximo de corrente que o arduino pode suportar em cada pino é de cerca de 40mA.  Os datasheets que encontrei informam que para uma voltagem de 5v os leds de 5mm  trabalham em média com 40mA.

**Não necessariamente a porta do seu arduíno irá queimar se você ligar por exemplo 2 leds em uma mesma porta mas isso não e recomendavel a não ser que você queira correr riscos**.

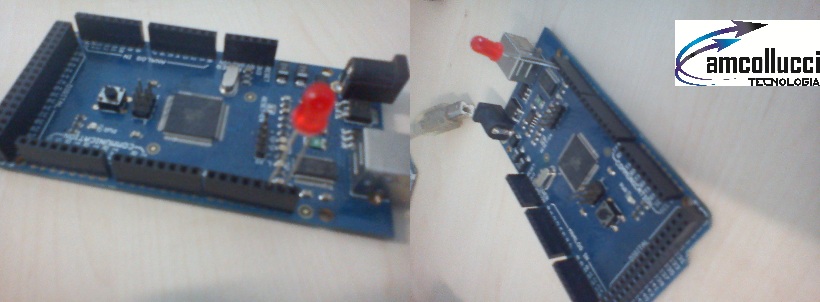
O Led e um componente polarizado portanto não deve ser ligado invertido pois pode queimar. Veja a figura abaixo extraída da [wikipedia](http://pt.wikipedia.org/wiki/Diodo_emissor_de_luz" \o "Diodo - Wikipedia" \t "_blank) ensinando a polarizar o Led.

[](http://amcollucci.com.br/wp-content/uploads/2012/02/200px-+-_of_Led.svg_.png)

Anodo e Catodo do LED

Portanto o Anodo, com a “perna maior” tem de ir numa porta de IO e o Catodo, deve ir numa porta de GND. (Veremos que isso não é obrigatório mais adiante…).

A montagem é bem simples e você pode conferir na [foto](http://amcollucci.com.br/2012/02/18/aprenda-a-programar-o-arduino-parte-3/) abaixo

[](http://amcollucci.com.br/wp-content/uploads/2012/02/arduino-blink.jpg)

Arduino Blink

**O Catodo, deve ser ligado ao pino GND e o Anodo ao Pino 13**

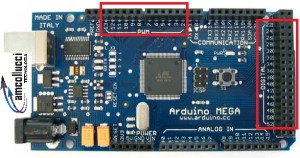
Então lets coding!!!!

void setup()  
{  
pinMode(13, OUTPUT);  
digitalWrite(13, HIGH);  
}

void loop()  
{  
}

Acabamos de ligar o led.

O comando **pinMode**recebe um valor inteiro, que deve ser igual ao numero da porta (estão impressos no circuito do arduino), e uma flag que informa se ele e de saída (Output), ou entrada (Input), no Arduino Mega 1280

[](http://amcollucci.com.br/wp-content/uploads/2012/02/DSC00004.jpg)

Portas PWM e Digital

as portas PWM**\*** e Digital utilizam o comando digitalWrite que tambem recebe dois parametros o primeiro e um numero inteiro que é o numero da porta e o segundo e se ela esta alta ou baixa (0 ou 1).

Esse [programa](http://amcollucci.com.br/2012/02/18/aprenda-a-programar-o-arduino-parte-3/) tambem pode ser reescrito da seguinte forma:

void setup()  
{  
pinMode(13, OUTPUT);  
digitalWrite(13, 1);  
}

void loop()  
{  
}

Podemos passar a informação da porta por uma variavel:

int pino = 13;  
void setup() {  
pinMode(pino, OUTPUT);  
digitalWrite(pino, HIGH);  
}

void loop()  
{  
}

E claro sempre podemos otimizar o código eliminando o sinal e colocando como uma constante alterando a linha:

int pino = 13;

Para:

const unsigned int pino = 13;

Agora vamos brincar criando um objeto chamado led.

class led  {  
private:  
int pino;  
int estado;  
public:  
void desliga\_led();  
void liga\_led();  
void set\_pino(int);  
};

void led::set\_pino(int i) {  
pino = i;  
pinMode(pino, OUTPUT);  
}

void led::desliga\_led(){  
estado=0;  
digitalWrite(pino,estado);  
}  
void led::liga\_led(){  
estado=1;  
digitalWrite(pino,estado);  
}

void setup() {  
led l1;  
l1.set\_pino(13);  
l1.liga\_led();  
}

void loop(){ }

**Aos não iniciados em programação orientada a objetos eu sugiro uma pausa pro café tirar os sapatos, relaxar e ler o seguinte link:**

<http://www.dca.fee.unicamp.br/cursos/POOCPP/node3.html>

Vamos começar a analisar o código:

  class led  {  
private:  
int pino;  
int estado;  
public:  
void desliga\_led();  
void liga\_led();  
void set\_pino(int);  
};

O trecho destacado em cima, define como a classe vai ser. Quais são os seus metodos e variaveis. Nele vemos duas distinçoes claras **private:** e **public: ,**tudo que estiver dentro de private, só pode ser alterado ou manipulado por algo que esteja dentro da classe. Tudo que esta dentro de public pode ser acessado externamente.

No nosso exemplo declaramos as variáveis **pino** e **estado** como variáveis privadas, ou seja só podem ser acessadas por métodos que estiverem dentro da classe.

Já na divisão **public** declaramos os metodos **desliga\_led()** **liga\_led()** e **set\_pino()** mas não programamos essas ffunçoes veja que até o momento não haviamos programado de fato essas funções.

void led::set\_pino(int i) {  
pino = i;  
pinMode(pino, OUTPUT);  
}

No trecho acima vimos a função set\_pino, o leitor poderá perceber, que estamos configurando o **pino i** como saída, esse **i** vai ser passado outra parte do código quando chamarmos a função.As outras funções seguem um padrão semelhante.

O que fizemos na verdade é [criar](http://amcollucci.com.br/2012/02/18/aprenda-a-programar-o-arduino-parte-3/) um método para cada tarefa. O que a príncipio parece muito difícil e de pouca utilidade visto que o mesmo código somente aumentou de tamanho e não de utilidade, mas veremos adiante o quanto isso pode ser útil. **Mas atente que ao utilizar esse tipo de programação seu código ocupara mais espaço em físico.**

**Criando nossa própria biblioteca para um LED**

*P\*\*\* o cara vai ensinar a criar uma biblioteca, e ainda nem explicou direito como funciona um if!? No Arduíno a programação e voltada para interação com o ambiente. Simplesmente com seus comandos básicos isso se torna muito dificil, já existem diversas bibliotecas criadas para o Arduíno, elas extedem as funções dele para  trabalhar com cartões SD, Motores, Blue Tooth, Sim Cards.. ou seja uma infinidade de coisas. Porém muita gente ao dar****#include****não sabe nem o que esta fazendo, na parte 4 deste tutorial eu pretendo explicar o funcionamento de um cartão de memória e criarmos algumas aplicações interessantes com ele, portando e interessante que o leitor entenda como funciona uma biblioteca de funçoes, até mesmo para localizar seus métodos sem conhecer ela a fundo ou ter de recorrer a alguma documentação.*

**Então let’s go.**

Eu utilizei o Notepad++ mas você pode usar o que quiser, notepad, wordpad, geany, devc++, eclipse c,vi… Ou qualquer outro editor.

Primeiro devemos criar um arquivo de header que no nosso caso vai se chamar amcled.h esse arquivo deve conter o codigo abaixo

#ifndef amcled\_h  
#define amcled\_h

#include “Arduino.h”

class led  
{  
private:  
int pino;  
int estado;  
public:  
void desliga\_led();  
void liga\_led();  
void set\_pino(int);  
void blk(int);  
};  
#endif

Depois devemos criar um arquivo chamado amcled.cpp

#include “Arduino.h”  
#include “amcled.h”

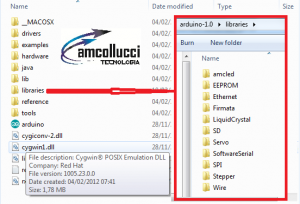
void led::set\_pino(int i)  
{  
pino = i;  
pinMode(pino, OUTPUT);  
}

void led::desliga\_led(){  
estado=0;  
digitalWrite(pino,estado);  
}  
void led::liga\_led(){  
estado=1;  
digitalWrite(pino,estado);  
}  
void led::blk(int time){  
liga\_led();  
delay(time);  
desliga\_led();  
delay(time);  
}

Você precisa agora criar um arquivo que vai destacar os comandos que você criou ao digita-los (colocar eles em laranja na IDE do arduino). esse arquivo deve se chamar keywords.txt

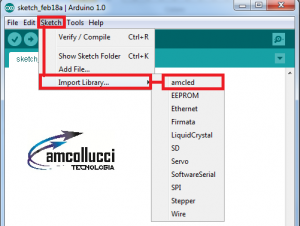
desliga\_led KEYWORD1  
liga\_led    KEYWORD2  
set\_pino    KEYWORD3  
blk            KEYWORD4

Agora você deve criar uma pasta chamada amcled dentro da pasta libraries do Arduino conforme abaixo, e colar dentro dela todos os arquivos que criamos:

[](http://amcollucci.com.br/wp-content/uploads/2012/02/library-1.png)

Biblioteca de Led

Depois disso só precisamos abrie e fechar a IDE do arduíno e já poderemos usar a nossa biblioteca.

[](http://amcollucci.com.br/wp-content/uploads/2012/02/library-2.png)

Biblioteca amcled pronta para o uso

Agora vamos reescrever o último programa que fizemos utilizando a nossa biblioteca:

#include <amcled.h>  
void setup() {  
led l1;  
l1.set\_pino(13);  
l1.liga\_led();  
}

void loop(){ }

Ainda pode parecer que o código ficou maior que o primeiro mas vamos prosseguir para o exemplo do led Blink do site do Aduino disponivel em:

 /\*  
\* Created 1 June 2005  
\* copyleft 2005 DojoDave <http://www.0j0.org>  
\* http://arduino.berlios.de  
\*  
\* based on an orginal by H. Barragan for the Wiring i/o board  
\*/

int ledPin = 13;

void setup()  
{  
pinMode(ledPin, OUTPUT);  
}

void loop()  
{  
digitalWrite(ledPin, HIGH);  
delay(1000);  
digitalWrite(ledPin, LOW);  
delay(1000);  
}

O programa acima faz com que o led pisque a cada 1s, pois ele alterna o estado do pino 13 utilizando digitalWrite. Veja eles reescrito com a nossa lib.

/\*  
\* Criado em 18 de Fevereiro de 2012  
\* copyleft 2012 – Copie e distribua =]  
\* http://amcollucci.com.br  
\*  
\* based on an orginal by H. Barragan for the Wiring i/o board  
\*/  
#include <amcled.h>

led l1;  
void setup()  
{  
l1.set\_pino(13);  
}

void loop()  
{  
l1.liga\_led();  
delay(1000);  
l1.desliga\_led();  
delay(1000);  
}

Nada muito diferente não é?

Então veja o seguinte codigo: =]

#include <amcled.h>

led l1;  
void setup(){  
l1.set\_pino(13);  
}  
void loop(){  
l1.blk(1000);  
}

Bom acho que consegui apresentar com sucesso o Led Blink com delay, da maneira mais completa que pude. Durante os proximos tutoriais pretendo avançar ainda mais em alguns itens que toquei durante essa parte.

Então até a próxima

**\***Obs.:Os pinos PWM possuem ainda outras características.