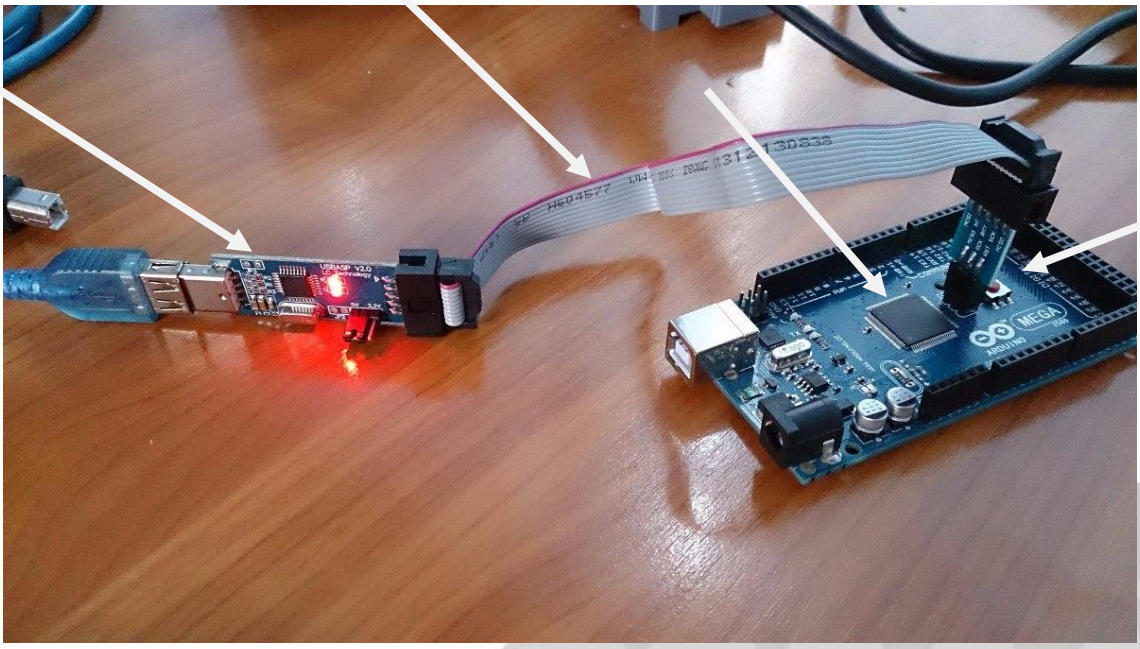
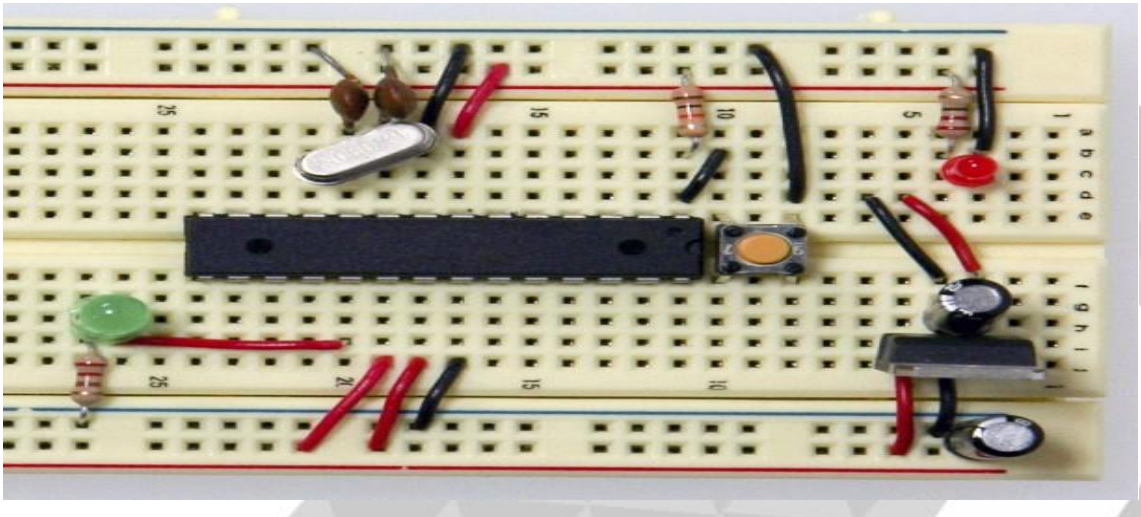
Como carregar sketchs, no microcontrolador ATmega16U2 do Arduino com gravador AVR USBasp quando ele não é reconhecido no PC.

Olá pessoal vamos a mais um tutorial, e nele vamos ensinar como carregar sketchs no microcontrolador ATmega16U2 do Arduino com gravador AVR USBasp Quando não é reconhecido pela porta USB do PC.



Quando um projeto envolve eletrônica, seja ela analógica ou digital, em que é preciso automatizar algo ou torná-lo inteligente, nós precisamos utilizar microcontroladores, esses componentes são como o processador do nosso computador, porém não precisam de memória RAM externa e nem armazenamento para o programa, um microcontrolador já possui internamente memória para o programa, memória RAM para os dados, e uma unidade de processamento, porém obviamente possuem um desempenho muito inferior a um processador, até porque o seu propósito é outro. Hoje em dia os microcontroladores mais utilizados em projetos particulares de pessoas (não empresa) são os **PIC** da **Microchip** e os **Atmega** da **ATMEL**, e apesar de muitas pessoas acharem que o Arduino é um microcontrolador, ele não é, o Arduino é apenas uma plataforma de desenvolvimento que utiliza os microcontroladores da **ATMEL** (**Atmega**).

Para que um microcontrolador execute uma tarefa como por exemplo fazer uma leitura analógica, ativar um transistor que ativa um relê e acende uma lâmpada ou se comunicar com algum módulo ou dispositivo, é preciso que você programe-o, para isso, você pode utilizar linguagens de programação como Assembly, C, C++ e em alguns casos até mesmo Visual Basic, claro que a mais completa e recomendável é a **linguagem C**. Depois de programar o microcontrolador é preciso montar o seu circuito, só para alimentar o microcontrolador será preciso de um regulador de tensão, cristal oscilador, capacitor para desacoplamento, e uma fonte de alimentação, o problema é que novamente para pessoas que apenas tiveram uma idéia e não possuem muito conhecimento em eletrônica, pode ser meio complicado ter que montar todo o circuito do microcontrolador em uma breadboard (placa de ensaio) e ainda montar o circuito do seu projeto.



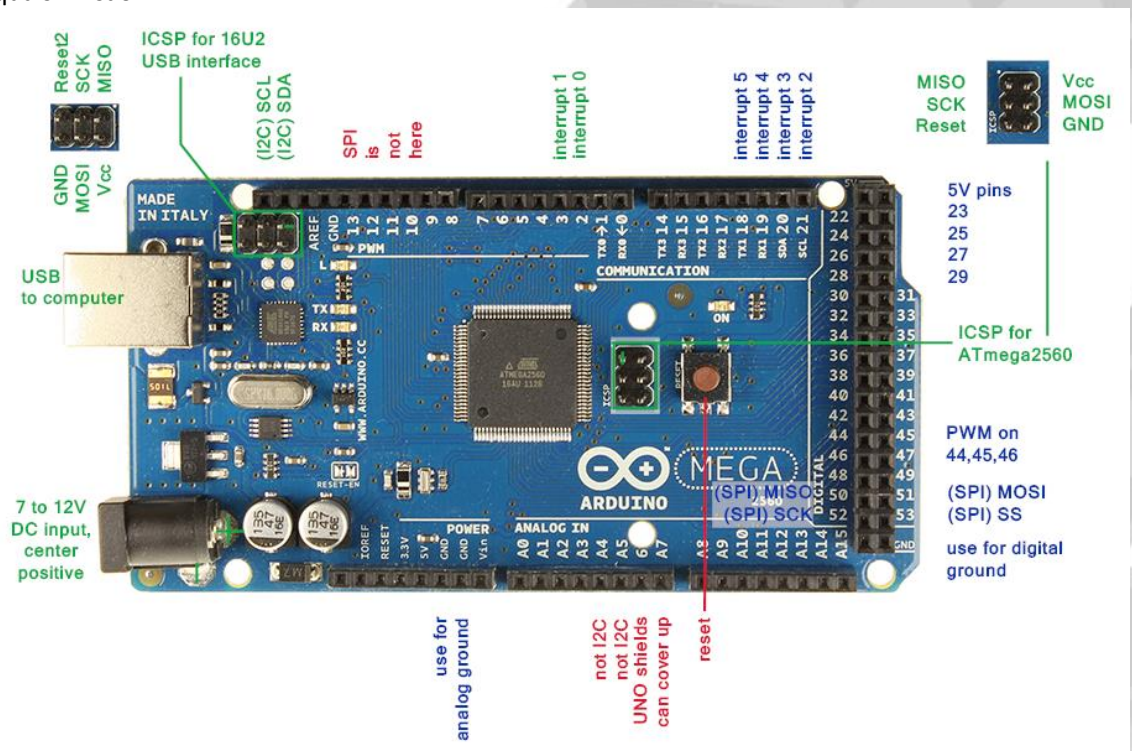
Pensando nisso o Arduino foi desenvolvido para facilitar a prototipagem, não só de técnicos em eletrônica e programadores, mas também de pessoas comuns, que querem automatizar seus projetos mas não possuem muito conhecimento na área. O Arduino é uma plataforma de desenvolvimento composta pelo Hardware (Arduino uno, mega uno, duemilanove, etc) e uma IDE (ambiente de desenvolvimento integrado) aonde você pode escrever o seu código em linguagem C/C++ e a própria IDE fará todo o processo de compilação e transferência do código do seu PC para o microcontrolador Atmega, o que torna todo o processo de programação e prototipagem muito mais rápido.

Hardware

Hoje em dia existem várias versões de placas **Arduino**, cada uma utiliza um microcontrolador Atmega diferente, no caso do Arduino Uno que é o Arduino que eu utilizarei nas próximas matérias e vídeo aulas, temos o **Atmega 328p** que possui **28 pinos** sendo **6 entradas analógicas** ADC (conversores analógicos digitais), **14 pinos de saída digital** sendo **6 deles com PWM** (pulse width modulation) aonde podemos utilizá-los como saída analógica (tensão entr 0 ~ 5V).

Como você pode ver, todo o circuito do **microcontrolador** Atmega já esta montado na placa, e há conectores laterais que correspondem a todos os pinos do Atmega, então ali temos os pinos analógicos e digitais e ainda os pinos de alimentação do **microcontrolador**
como **VCC**,**GND**, **AREF** e **RESET**, é claro que o **microcontrolador** já será alimentado através do circuito, esses conectores de alimentação apenas nos permite montar o nosso próprio circuito de alimentação, já os conectores correspondentes aos pinos ai sim deverão ser obrigatóriamente utilizados para que possamos ativar algo ou nos comunicarmos com outros componentes.

Caso você ache que o **Arduino Uno** tem poucos pinos, não se preocupe, você pode optar pelo **Arduino Mega Uno** com o microcontrolador SMD Atmega 2560, esse **Arduino** possui 54 pinos dos quais 14 são PWM.



IDE

Uma das vantagens do Arduino é que sua IDE é multi-plataforma, ou seja, pode ser instalada no **Windows**, **Linux** e **MAC OS** e por ser uma IDE dedicada ao **Arduino**, ela possui nativamente diversos exemplos de código e opções como por exemplo passar o **bootloader** do **Arduino** para um **microcontrolador** Atmega novo.
A linguagem adotada pelo Arduino é a linguagem C/C++, e caso você ache que essa linguagem é avançada e complicada, não se preocupe pois na verdade a linguagem C é estruturada e possui uma sintaxe limpa, clara e objetiva, e para facilitar ainda mais, o Arduino já tem diversas funções prontas, como por exemplo para leitura e escrita de dados analógicos e digitais, além de comunicação serial (PPM, I2C, TTL ...) e paralela e também tempo (delay), sendo assim não será preciso você se preocupar com o clock do microcontrolador e nem terá que implementar protocolos de comunicação serial ou paralela, resumindo, se você quer que o pino 13 fique positivo (5volts) basta você fazer:

```
void setup
pin Mode (13, OUTPUT); // Coloca o pino 13 em modo saída

digital Write (13, 1); // Deixa o pino 13 como positivo (5V)

void loop
```

Caso você já tenha conhecimento em programação para **microcontroladores**, você deve ter percebido que não tivemos que utilizar (diretamente) nenhum registrador e que está tudo bem simplificado.

Void setup e **void loop**. Essas duas funções são obrigatórias no **Arduino**, a função **void setup** não retorna nada e nem recebe nenhum argumento, essa função é utilizada para configuração, no exemplo acima configuramos o pino **13** para **saída** de dados (**OUTPUT**). Já a função **void loop** também não retorna nada e nem recebe nenhum argumento, quando rodamos um programa (firmware) no **microcontrolador**, é de nosso interesse que esse programa fique rodando para sempre em um loop infinito, então para isso devemos escrever as rotinas de nossa firmware dentro da função loop, pois ela será repetida infinitamente.

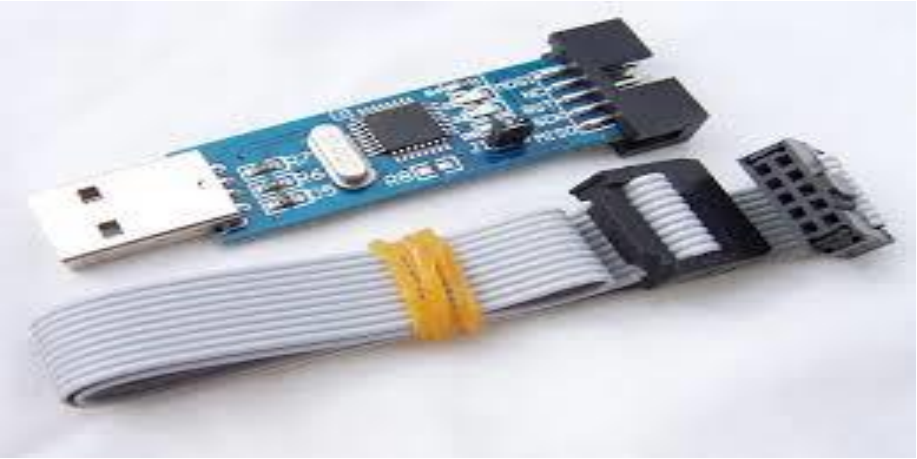
Vantagems

Uma das vantagems do **Arduino Uno**, é o seu baixo custo comparado a outras pataformas de desenvolvimento, e a **IDE** é gratuita e **open source**. você estará aprendendo sobre uma nova área com infinitas possibilidades.
Podemos apenas clicar em um botão e transferir o código para um circuito que já esta montado em uma placa de circuito impresso. Em questão de **protótipo** o **Arduino** é sim uma ótima opção.
Nós também podemos utilizar circuitos eletrônicos para nos ajudar em tarefas **hacking,** como monitoramento, segurança de computadores, módulos de Firewall ou servidores pequenos, em fim, novamente tudo vai depender da nossa criatividade.

O que é gravador AVR USBasp

USBasp é um gravador para controladores **Atmel AVR** com uma conexão **USB** construída em sua placa. É constituído de um **ATMega** e outros simples componentes eletrônicos. O gravador apenas utiliza de um

driver **USB** para comunicação, não sendo necessário um controlador **USB** especial.



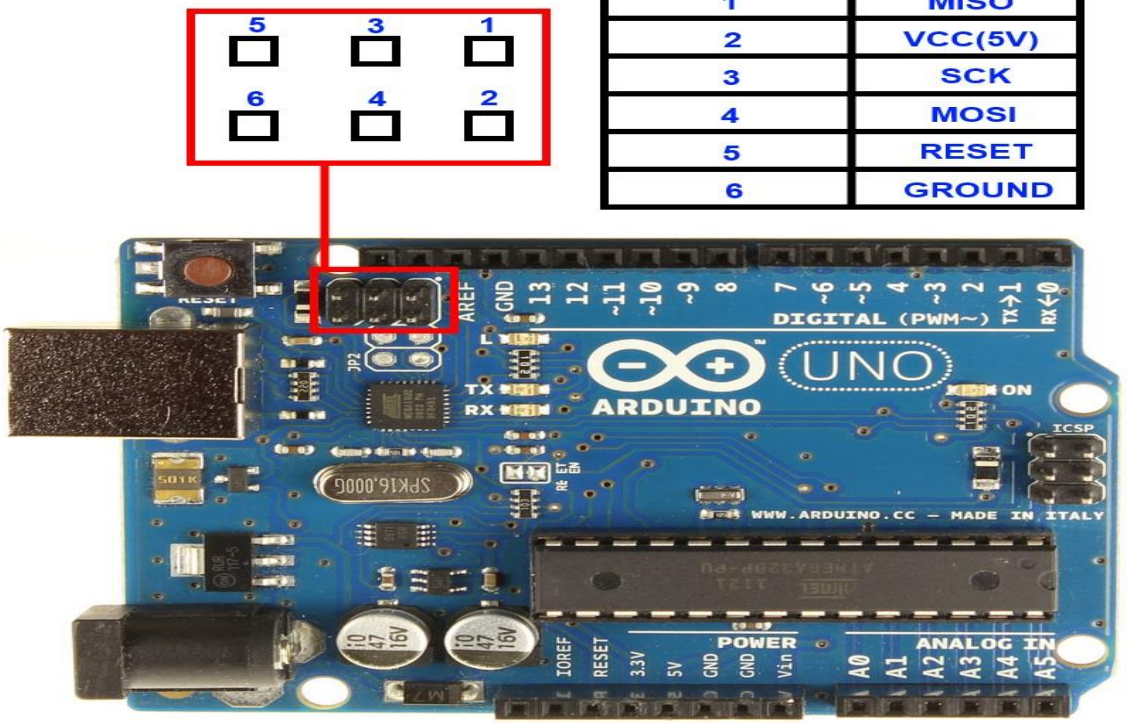
Quando o **Arduino** não é mais reconhecido pelo seu PC mas ele liga quando recebe tensão? Aqui ensinaremos a fazer uma gravação de sketch usando um gravador **AVR USBasp**, que tem preço bem acessível.

Imagine você, utilizando seu bom e velho **Arduino Uno** em algum projeto qualquer, e, de repente, ele não funciona mais. Apesar de seu led ON estar aceso, você pluga em seu computador, não faz mais o "barulhinho" quando pluga o USB.

Bem, isto aconteceu comigo no ano passado, quando tentei fazer o **bootloader** em um **Arduino** Pro Mini através de um **Arduino Uno**. Depois de muita procura, descobri que o chip **microcontrolador** responsável pela comunicação do **Arduino UNO** com o computador, chamado ATmega16U2 (vide figura 1) não funcionava mais:

Existe um gravador chamado **AVR USBasp** que realiza a gravação através dos pinos **ICSP** do **Arduino**, estes conjuntos com 6 pinos que estão destacados na figura 2:

Os pinos à direita são responsáveis pela gravação no **ATmega328**, **microcontrolador** principal do **Arduino UNO**. Já os pinos à esquerda e acima são referentes ao **ATmega16U2**. As especificações dos pinos estão localizados na figura 3:

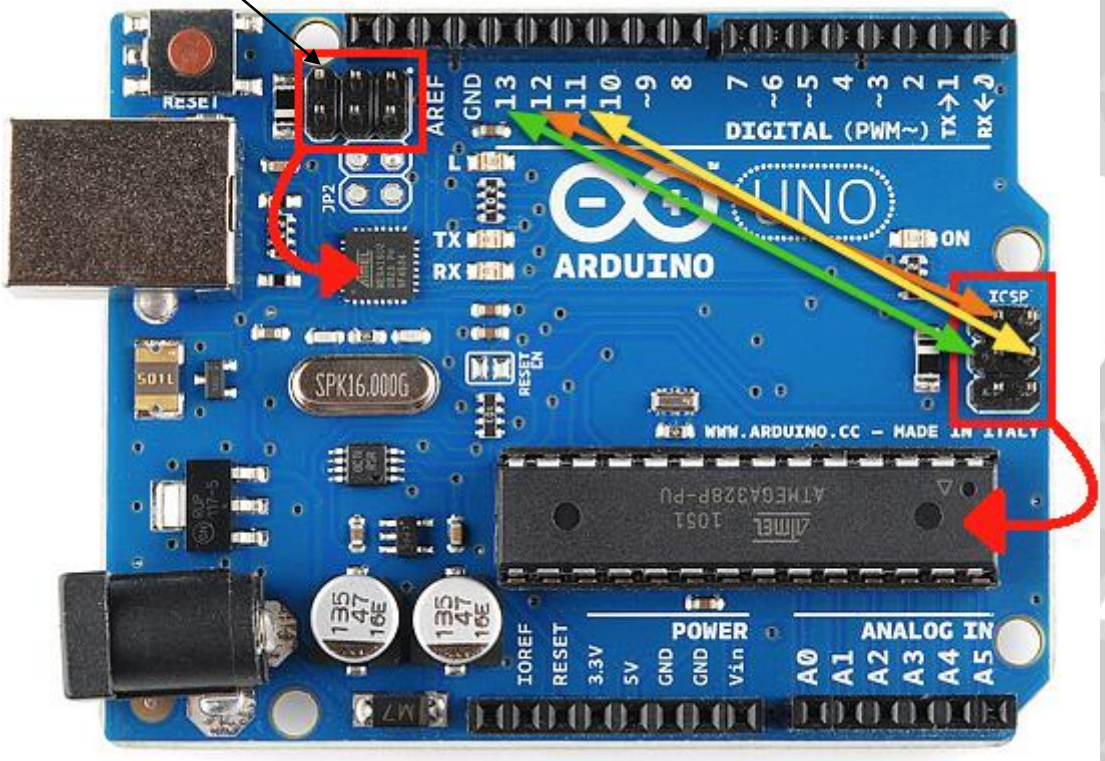


ICSP header(USB interface)

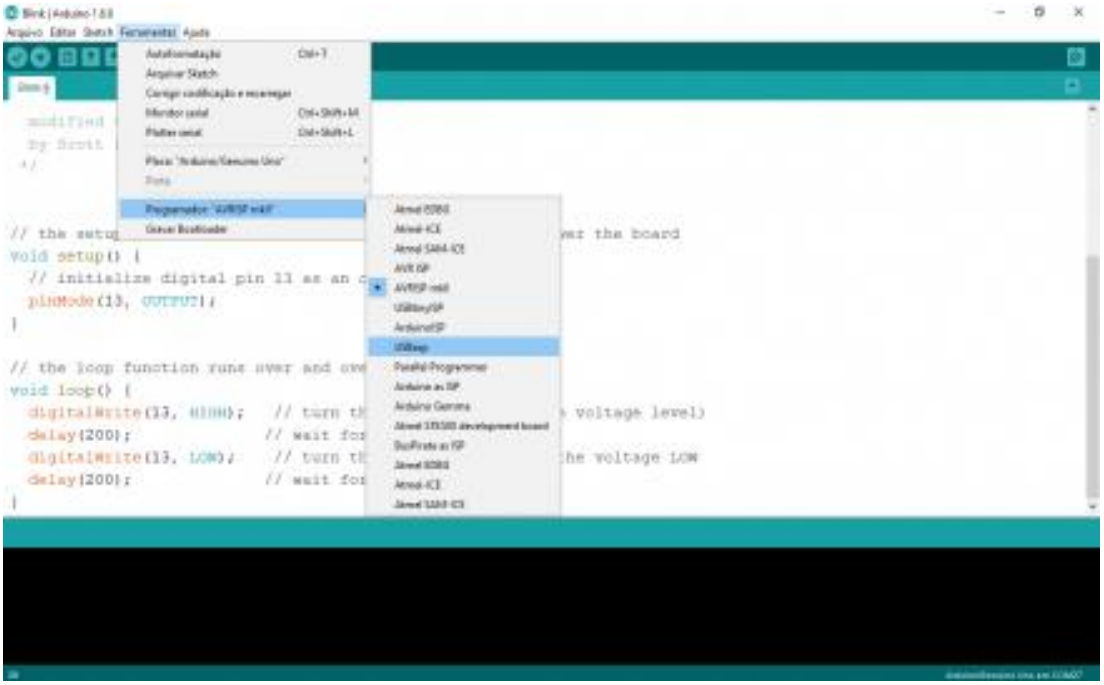
Name	Function
1	MISO
2	VCC(5V)
3	SCK
4	MOSI
5	RESET
6	GROUND

Para quem já está familiarizado com módulos tipo **SPI** vai ter familiaridade com estes pinos.
Agora vamos para a parte prática. O gravador que utilizei é este, disponível aqui:

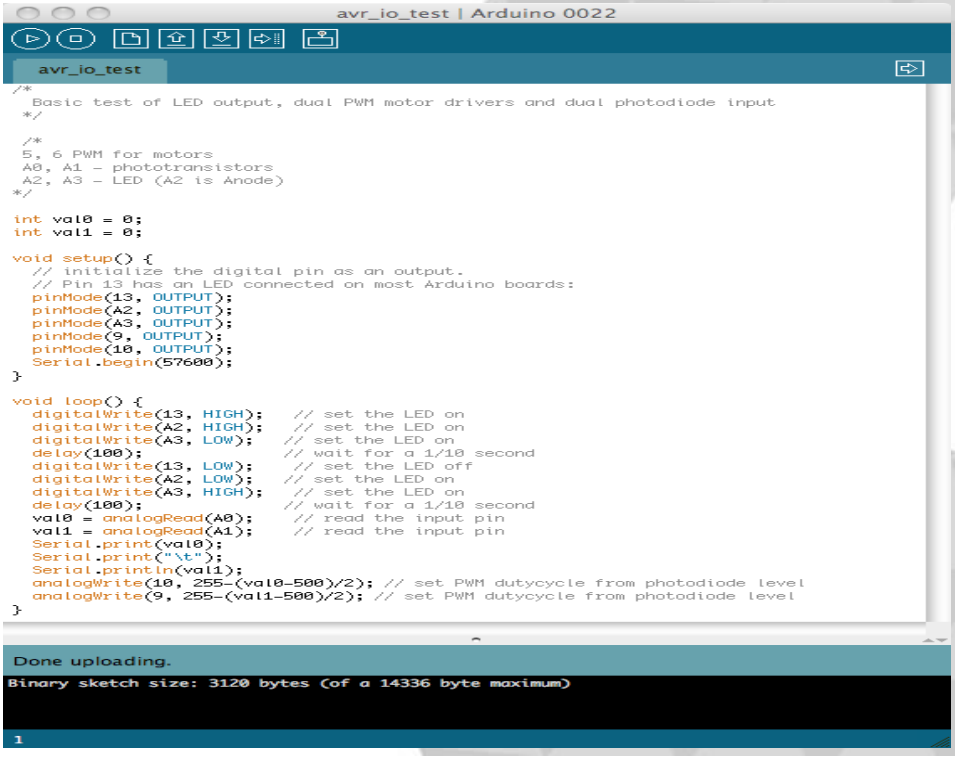
Caso esteja usando **Windows**, é preciso instalar o **driver do gravador**, disponível **aqui**. Em ambiente Linux não é necessária instalação, apenas permissão para o dispositivo USB, caso você não seja root. Após instalar e plugar seu dispositivo no computador, faça as ligações do gravador para o Arduino:



Agora inicialize a **IDE Arduino** e selecione no menu **Ferramentas** - **Programador** - **USBasp**, conforme a próxima imagem:



Na hora de realizar o **upload**, não faça pelo método comum. Selecione **Sketch** - Carregar usando **Prograd**or, conforme a figura 7:



Se tudo der certo, teremos carregado um **blink** com intervalo de **200 milissegundos**. Se não der, pode ocorrer alguns destes problemas:
Se estiver usando **Linux**, pode haver o problema de não carregar porque o dispositivo **USB** não está autorizado. Faça as alterações pelo **root** do Terminal;
No ambiente **Windows**, verifique se o gravador **USB** foi corretamente instalado. Caso haja algum problema, você deve realizar a instalação através do Gerenciador de dispositivos.