Introdução ao Arduino

Contato físico

O projeto Arduino é ideal para experiências com computação física. Demonstramos seu funcionamento com a construção de um timer simples.

por Dmitri Popov

Em seu website [1], o projeto Arduino se apresenta como uma "plataforma para protótipos eletrônicos de código aberto, baseada em hardware e software flexíveis e fáceis de usar", o que não soa muito animador. Mas essa minúscula e econômica plataforma lhe permite construir uma gama de projetos fantásticos – e não há exigência de habilidades com solda.

O projeto consiste basicamente de uma placa que pode ser programada para controlar diferentes dispositivos (motores, luzes etc.) conforme a entrada que ela recebe de sensores conectados. É possível conectar praticamente qualquer sensor à placa do Arduino – por exemplo, um resistor dependente de luz, um sensor de movimento, um termistor ou um sensor de pressão – e programar o microcontrolador da placa para executar várias ações com os controladores conectados aos pinos digitais de saída.

A programação é feita no IDE Arduino, que usa uma linguagem de programação baseada em Processing [2]. A linguagem em si é relativamente fácil de aprender, então mesmo que você não tenha nenhuma experiência com programação, pode aprender o básico rapidamente. A seção de referência no site do projeto contém uma visão detalhada da linguagem para quem quer iniciar, estando disponível também a versão

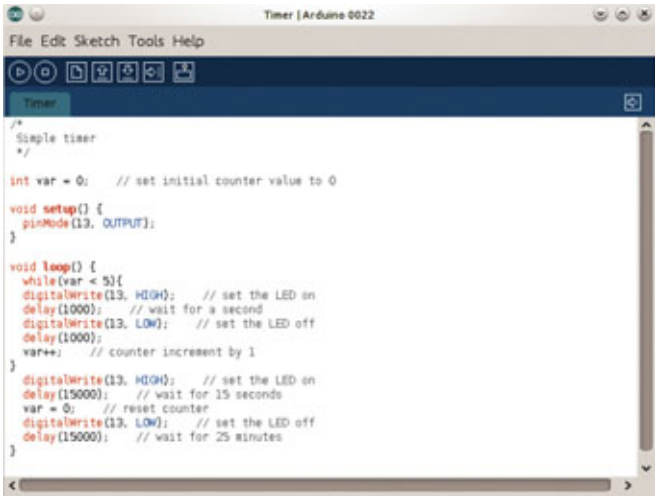

Figura 1 O IDE Arduino não é muito bonito, mas cumpre bem sua função.

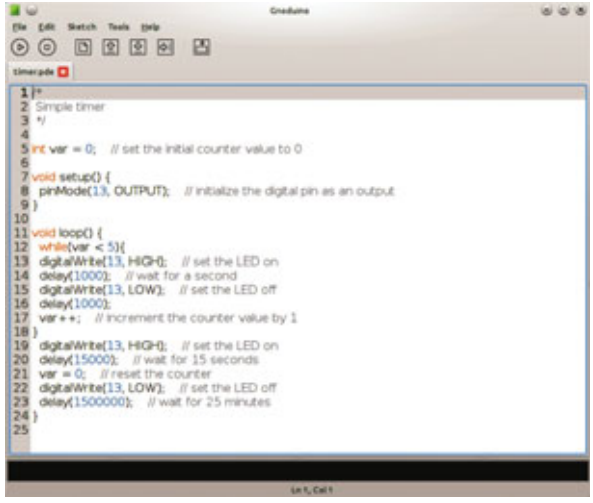

Figura 2 O Gnoduino é uma alternativa mais polida ao IDE padrão do Arduino.

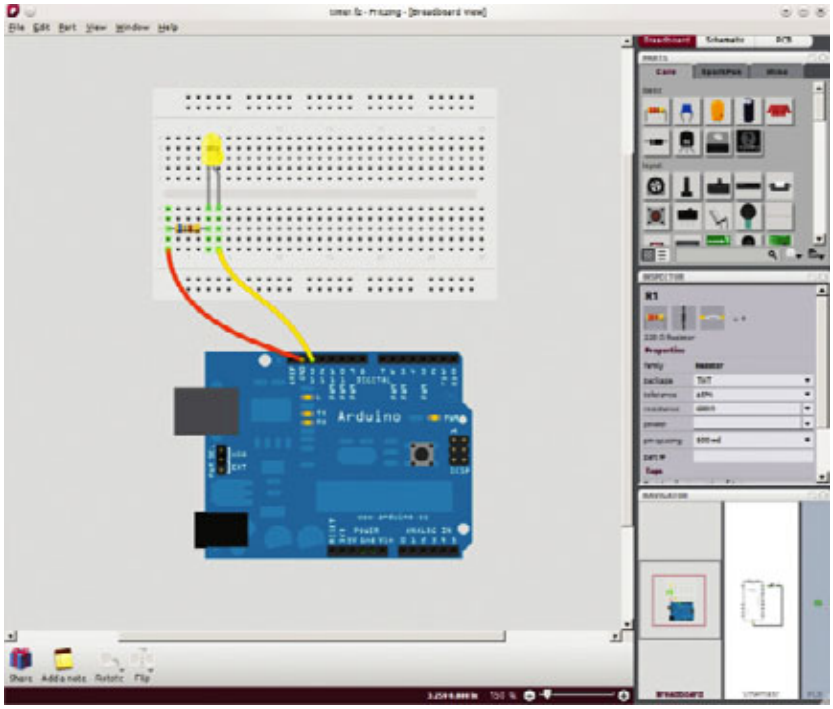

Figura 3 O Fritzing oferece um ambiente gráfico para criar projetos do Arduino.

livre do *Arduino Programming Notebook* [3] como guia rápido e prático.

Os programas Arduino são chamados de *sketches*. Você escreve, depura e faz o upload desses sketches na placa usando o IDE. O Arduino foi desenvolvido como uma forma de introduzir às pessoas o mundo da computação física (construir sistemas físicos interativos que podem perceber o mundo analógico e reagir a ele) mas o projeto é mais do que apenas uma ferramenta educacional. É possível usar a placa para todos os tipos de solução inteligente, desde um alarme ativado por movimentos até um disparador acionado por luz para sua câmera.

Você pode comprar diferentes modelos Arduino gastando a partir de 30 dólares, sendo desnecessário gastar todas suas economias para poder se iniciar no maravilhoso mundo da computação física. Você pode precisar acrescentar à sua lista de compras também, uma *protoboard* (placa com furos e conexões condutoras para montagem de circuitos elétricos experimentais), alguns cabos de jumper, alguns LEDs e resistores. Usando a protoboard, pode-se construir projetos Arduino sem precisar soldar componentes – uma solução perfeita para a experimentação.

Preparação

A primeira tarefa é instalar o ambiente de desenvolvimento para poder escrever sketches e fazer o upload no Arduino. Se estiver usando o Ubuntu ou derivados, instalar o programa necessário é simples:

```
sudo apt-get install arduino
```

Esse comando instala o IDE Arduino padrão, escrito em Java (**figura 1**). O IDE cumpre sua função, mas não se integra muito bem com o Gnome. Felizmente, o Gnoduino fornece uma alternativa polida e leve, projetada especificamente para esse ambiente de desktop (**figura 2**).

O Gnoduino exibe as janelas nativas do Gnome e tem melhor renderização de fontes, além do recurso útil de numerar linhas. O programa foi escrito em Python e, no Ubuntu, pode ser instalado a partir de um PPA dedicado [5], com os seguintes comandos:

```
sudo add-apt-repository
➥ ppa:pmjdebruijn/gnoduino-release
sudo apt-get update
sudo apt-get install gnoduino
```

Em seguida conecte a placa Arduino à sua máquina e você estará pronto para começar seu primeiro projeto. É possível criar todo tipo de dispositivos com o Arduino mas, para começar, é melhor algo simples. Então descreverei como construir um timer básico para treinarmos a técnica Pomodoro [6] de gerenciamento de tempo.

Nesse projeto só precisamos de três componentes: um LED, um resistor (entre 200 e 680 ohms) e dois cabos de jumper. O resistor é opcional, sendo usado para limitar a corrente e evitar danos ao LED.

Esquematização

Antes de começar a montar os componentes na matriz de contato e cabeá-los, é uma boa ideia desenhar um esquema para ser usado como referência. Embora você possa usar caneta e papel para fazê-lo, há uma ferramenta melhor chamada Fritzing [7]. Esse programa oferece um ambiente gráfico completo para projetos Arduino: você pode usá-lo para desenhar layouts na matriz de contato, esquematização ou circuitos impressos.

Usar o Fritzing em um projeto tão simples pode parecer exagero, mas assim que seus projetos Arduino forem ficando mais avançados, você vai começar a apreciar as capacidades e recursos desse programa. O Fritzing não exige uma instalação tradicional. Para obter a última versão, acesse o site do projeto, descompacte o arquivo baixado e execute o script em Bash `Fritzing.sh` no diretório correspondente para iniciar o aplicativo.

A visão *Breadboard* no Fritzing permite criar designs na protoboard com o posicionamento de componentes e seu cabeamento (**figura 3**). A coleção principal de componentes

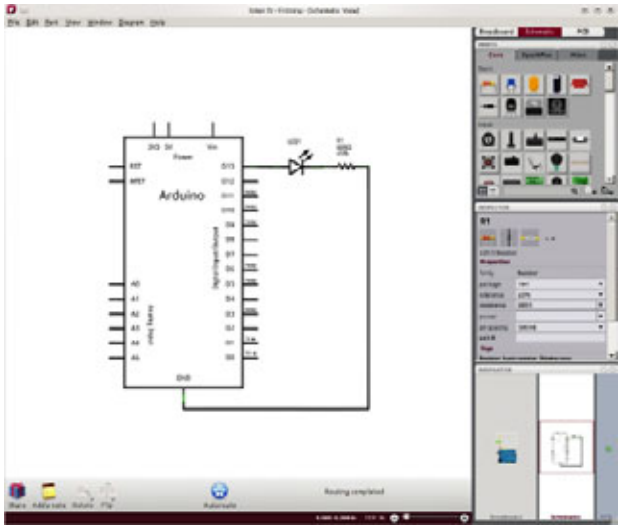

Figura 4 O Fritzing também gera esquemas de cabeamento baseados no projeto da protoboard.

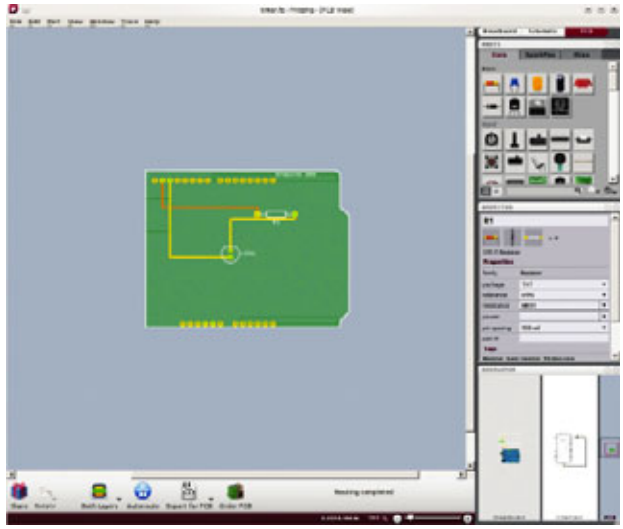

Figura 5 Com o Fritz você pode projetar esquemas de circuitos impressos.

na paleta *Parts* inclui praticamente todas as partes essenciais: de resistores a switches, de LEDs a diversos tipos de sensor. O Fritzing também permite a importação de coleções de componentes de terceiros. Para fazê-la, selecione *File/Parts Bin/Open* e escolha a coleção que deseja. Você pode até desenhar suas próprias partes. A página *Creating Custom Parts* [8] do website tem bastante material sobre isso.

Assim que posicionar um componente na protoboard, você pode modificar suas propriedades na paleta *Inspector*. É possível, por exemplo, especificar tolerância, resistência e espaçamento de pinos para resistores, além de cor e tamanho, para LEDs. Para conectar os componentes com cabos de jumper, clique com o botão direito no socket desejado da matriz, depois arraste o mouse até o socket de destino, enquanto segura o botão direito. Depois é possível mudar a cor do cabo adicionado na paleta *Inspector* para melhorar a leitura do desenho. Por padrão, o Fritzing usa cabos retos, que podem ser muito limitadores em estruturas mais complexas. Mas é possível curvar os cabos pressionando a tecla [**Ctrl**], clicando e arrastando.

Enquanto você trabalha no design da matriz, o Fritzing silenciosamente gera um esquema de cabeamento para seu projeto, que pode ser visto ao mudarmos para a visão *Schematic* (**figura 4**). Todas as conexões entre componentes são consideradas ligações potenciais, sendo exibidas como linhas finas; você precisa usar o mouse para cabear os componentes de fato. Fazer isso manualmente pode ser um desafio, especialmente se estiver trabalhando com um desenho complexo. Felizmente, o Fritzing tem uma solução elegante: o recurso *Autoroute* pode cabear automaticamente os componentes.

Os resultados podem às vezes ser decepcionantes, mas na maioria das vezes esse recurso faz um bom trabalho. Além disso, você pode melhorar esse resultado manualmente, se necessário. O Fritzing também permite que você exporte o esquema de cabeamento para diversos formatos. É possível salvar o esquema como uma imagem com *File/Export/As Image*, que oferece as opções PNG, SVG e PDF, entre outros. Também é possí-

Listagem 1: Sketch para um timer

```
01 int var = 0;
02
03 void setup() {
04   pinMode(13, OUTPUT);
05 }
06
07 void loop() {
08   while(var < 5){
09     digitalWrite(13, HIGH);
10     delay(1000);
11     digitalWrite(13, LOW);
12     delay(1000);
13     var++;
14   }
15   digitalWrite(13, HIGH);
16   delay(15000);
17   var = 0;
18   digitalWrite(13, LOW);
19   delay(1500000);
20 }
```

Listagem 2: Sketch aperfeiçoado para um timer

```
01 int var = 0;
02
03 void setup() {
04   pinMode(9, OUTPUT);
05   pinMode(13, OUTPUT);
06 }
07
08 void loop() {
09   while(var < 5){
10     digitalWrite(13, HIGH);
11     delay(1000);
12     digitalWrite(13, LOW);
13     delay(1000);
14     var++;
15   }
16   digitalWrite(9, HIGH);
17   delay(15000);
18   var = 0;
19   digitalWrite(9, LOW);
20   delay(1500000);
21 }
```

vel usar *File/Export/List of parts (Bill of Materials)* para gerar uma lista de compras bem formatada, com todos os componentes usados nos projetos.

Se isso não for suficiente, ainda é possível usar o Fritz para desenhar um circuito impresso, portanto você pode transformar seu protótipo de matriz de contato em um produto de fato ou em um Arduino shield (**figura 5**). Melhor ainda, o serviço Fritzing Fab [9] pode produzir circuitos impressos, prontos para uso, baseados em seu design. Por enquanto, contudo, usaremos o Fritzing para criar um projeto de protoboard para o timer, como o da **figura 3**. Depois você pode usar o desenho como guia para cabear os componentes na matriz real e conectá-los ao Arduino para completar o projeto de hardware.

Sketch

O próximo passo é escrever um sketch para o timer. Basicamente, você precisa programar o Arduino para ligar o LED a cada 25 minutos, sinalizando que é hora de um intervalo. Para deixar as coisas um pouco mais interessantes, você pode fazer o LED piscar cinco vezes, para atrair a atenção, antes de ligar. Para começar, inicie o Arduino IDE ou o Gnoduino e use o sketch da **listagem 1**.

O que esse sketch faz? Como qualquer programa ou script, sua primeira tarefa é definir as variáveis a serem usadas. Nesse caso, a declaração `int var = 0` define a variável `var`, que atuará como um contador, com um valor de `0` (zero). Além disso, qualquer sketch Arduino deve conter dois blocos. A função `setup()` contém todo o código a ser executado uma vez no início do programa, enquanto a função `loop()` contém o programa em si, que é executado em repetição.

"Por que isso?", você pode se perguntar. Porque, diferentemente de um computador comum, o Arduino não pode executar vários programas ao mesmo tempo e os programas não podem terminar. Basicamente, quando você liga a placa Arduino ela executa o código, parando apenas quando você desliga. Neste caso, a função `setup()` inicializa o pino digital 13 como uma saída, usando a declaração `pinMode(13, OUTPUT)`. Isso permite que o sketch controle o LED conectado ao pino 13.

A função `loop()` contém o programa em si, que pode ser dividido, a grosso modo, em duas partes. A primeira é o laço `while` que liga o LED (a declaração `digitalWrite(13, HIGH)`), espera um segundo (`delay (1000)`), desliga o LED (`digitalWrite(13, LOW)`), espera outro segundo e então incrementa o contador em `1` (`var++`). O loop executa enquanto o valor de `var` é menor que 5 (ou seja, o LED pisca cinco vezes). Quando `var` chega a 5, a segunda parte do programa entra em ação, ligando o LED, esperando 15 segundos, desligando o LED e esperando 25 minutos.

Para verificar erros no sketch, use o botão *Verify* no IDE Arduino. Se tudo correr bem, você deve ver a mensagem *Binary sketch size: 1086 bytes (of a 32256 byte maximum)* (o tamanho pode variar); do contrário, surgirão alertas indicando os erros. Finalmente, pressione *Upload* para enviar o sketch para o Arduino. A placa começará a executar o código imediatamente. Parabéns, seu primeiro hardware Arduino funciona!

Etapas posteriores

Embora esse timer Arduino seja bem simples, você pode aperfeiçoá-lo. Por exemplo, pode adicionar um segundo LED no pino 9 e reescrever o sketch para que o primeiro LED pisque antes de o segundo ser ligado (**listagem 2**). Em vez de usar um segundo LED, você pode conectar um alto-falante e programar o Arduino para disparar um tom ou melodia simples. Resumindo, há muitas opções para refinar e melhorar esse simples timer. E depois que dominar o básico, você pode trabalhar em projetos mais complexos e empolgantes. ■

Mais informações

[1] Arduino: http://arduino.cc/

[2] Linguagem de programação Processing: http://processing.org/

[3] Arduino Programming Notebook: http://www.lulu.com/product/file-download/arduino-programming-notebook/3524028

[4] Gnoduino: http://gnome.eu.org/evo/index.php/Gnoduino

[5] PPA Gnoduino: http://launchpad.net/~pmjdebruijn/+archive/gnoduino-release

[6] Técnica Pomodoro: http://www.pomodorotechnique.com/

[7] Fritzing: http://fritzing.org/

[8] Partes customizadas no Fritzing: http://fritzing.org/learning/tutorials/creating-custom-parts

[9] Fritzing Fab: http://fab.fritzing.org/fritzing-fab

O autor

Dmitri Popov é formado em letras (idioma russo) e lingüística computacional; há vários anos trabalha como tradutor técnico e colaborador free-lancer. Já publicou mais de 500 artigos sobre software de produtividade, computação móvel, aplicativos web e outros tópicos relacionados à informática. Seus artigos já apareceram em sites e revistas da Dinamarca, Inglaterra, EUA, Alemanha, Rússia e, agora, do Brasil.

Gostou do artigo?

Queremos ouvir sua opinião.
Fale conosco em
cartas@linuxmagazine.com.br

Este artigo no nosso site:
http://lnm.com.br/article/6109