Arduino
EM AÇÃO

Martin Evans
Joshua Noble
Jordan Hochenbaum

Novatec

Editor: Rubens Prates
Tradução: Camila Paduan
Revisão técnica: Rodrigo Stulzer
Revisão gramatical: Marta Almeida de Sá
Editoração eletrônica: Carolina Kuwabata

ISBN: 978-85-7522-373-4

Histórico de impressões:

Agosto/2013 Primeira edição

Novatec Editora Ltda.
Rua Luís Antônio dos Santos 110
02460-000 – São Paulo, SP – Brasil
Tel.: +55 11 2959-6529
Fax: +55 11 2950-8869
E-mail: novatec@novatec.com.br
Site: www.novatec.com.br
Twitter: twitter.com/novateceditora
Facebook: facebook.com/novatec
LinkedIn: linkedin.com/in/novatec
VC20130812

CAPÍTULO 1

Olá Arduino

Este capítulo aborda

- A história do Arduino
- Hardware do Arduino
- Configuração de hardware e software
- O primeiro LED intermitente

Para que o Arduino pode ser usado? As respostas são surpreendentemente diversificadas. O Arduino tem sido usado em uma grande variedade de projetos:

- Videogames, tais como Pong e Space Invaders, que trarão a alguns leitores lembranças de sua infância e introduzirão outros nos jogos que seus pais jogavam quando eram jovens, com gráficos monocromáticos e efeitos sonoros simples.
- Robôs seguidores de linha (line-following robots), que introduzem princípios de robótica, mas também são usados em fábricas e armazéns para entrega de componentes ao longo de trajetos predeterminados.
- Harpas de Luz que produzem música com um aceno de mãos, como as usadas internacionalmente pelo executor Little Boots.
- Controladores MIDI que controlam uma série de instrumentos.
- Robôs autobalanceados que imitam o Segway.

Todos esses são exemplos de projetos construídos com o uso do Arduino, um microcontrolador tão pequeno que cabe na palma de sua mão. Originalmente concebido para ser utilizado como uma ferramenta para projetos de computação por designers e estudantes de arte, o Arduino tem sido adotado como a ferramenta preferida das comunidades de desenvolvedores e fabricantes interessados na construção e prototipagem de seus próprios projetos.

Neste capítulo, veremos a história do Arduino e como ele se tornou a ferramenta que muitos fabricantes escolhem ao iniciar um novo projeto. Este histórico inclui suas origens no Interaction Design Institute de Ivrea e explica por que ele se tornou tão necessário. Abordaremos então os diferentes tipos de Arduinos disponíveis e as vantagens e desvantagens de cada um deles. Também veremos o que você precisa para começar: ferramentas, equipamentos e componentes eletrônicos sugeridos. Finalmente, completaremos este capítulo de abertura olhando para o ambiente de desenvolvimento integrado do Arduino (IDE) antes de fazer nosso primeiro projeto: um LED que acende e apaga alternadamente.

Vamos começar vendo de onde o Arduino vem.

1.1 Breve história do Arduino

O Arduino teve seu início no Interaction Design Institute na cidade de Ivrea, na Itália, em 2005. O professor Massimo Banzi procurava um meio barato de tornar mais fácil para os estudantes de design trabalhar com tecnologia. Ele discutiu seu problema com David Cuartielles, um pesquisador visitante da Universidade de Malmö, na Suécia, que estava procurando uma solução semelhante, e o Arduino nasceu. Os produtos existentes no Mercado eram caros e relativamente difíceis de usar. Banzi e Cuartielles decidiram desenvolver um microcontrolador que poderia ser utilizado pelos seus estudantes de arte e design em seus projetos. As principais exigências eram que fosse barato – o preço almejado não poderia ser mais do que o que um estudante gastaria se saísse para comer uma pizza – e que fosse uma plataforma que qualquer pessoa pudesse utilizar. David Cuartielles desenhou a placa, e um aluno de Massimo, David Mellis, programou o software para executar a placa. Massimo contratou um engenheiro local, Gianluca Martino, que também trabalhou no Design Institute ajudando alunos com seus projetos. Gianluca concordou em produzir uma tiragem inicial de duzentas placas.

A nova placa foi chamada Arduino em referência a um bar local frequentado por membros do corpo docente e alunos do instituto. As placas eram vendidas em forma de kit para que os alunos fizessem seus próprios projetos. A tiragem inicial foi rapidamente vendida, e mais unidades foram produzidas para manter a demanda. Designers e artistas de outras áreas ouviram falar do Arduino e quiserem usá-lo em seus projetos. Sua popularidade cresceu rapidamente quando o grande público percebeu que o Arduino era um sistema de fácil utilização, de baixo custo e que poderia ser usado em seus próprios projetos, bem como era

uma excelente introdução para programação de microcontroladores. O projeto original foi melhorado e novas versões foram introduzidas. As vendas dos Arduinos oficiais alcançaram agora a marca de 300 mil unidades, e eles são vendidos em todo o mundo por intermédio de uma série de distribuidores.

Existe agora um número de diferentes versões de placas de Arduino, então daremos uma olhada nelas na próxima seção.

1.2 Hardware Arduino

Temos até o momento uma série de versões do Arduino, todas baseadas em um microprocessador de 8 bits Atmel AVR reduced instruction set computer (RISC). A primeira placa foi baseada no ATmega8 rodando a uma velocidade de clock de 16 MHz com memória flash de 8 KB; mais tarde, placas tais como a Arduino NG plus e a Diecimila (nome italiano para 10.000) usava o ATmega168 com memória flash de 16 KB. As versões mais recentes do Arduino, Duemilanove e Uno, usam o ATmega328 com memória flash de 32 KB e podem comutar automaticamente entre USB e corrente contínua (DC). Para projetos que exigem mais Entrada/Saída e memória, há o Arduino Mega1280, com memória de 128 KB, ou o mais recente Arduino Mega2560, com memória de 256 KB.

As placas têm 14 pinos digitais, e cada um pode ser definido como entrada ou saída, e seis entradas analógicas. Além disso, seis dos pinos digitais podem ser programados para fornecer uma saída de modulação por largura de pulso (PWM). Diversos protocolos de comunicação estão disponíveis, incluindo serial, bus serial de interface periférica (SPI) e I2C/TWI. Incluídos em cada placa como recurso padrão estão um conector de programação serial in-circuit (ICSP) e um botão de reset.

NOTA: Placas especializadas chamadas shields podem expandir a funcionalidade básica do Arduino; elas podem ser empilhadas umas sobre as outras para adicionar ainda mais funcionalidade.

Agora veremos os modelos mais comuns de Arduinos disponíveis, começando com o Arduino Uno.

1.2.1 Arduino Uno

"O Jantar está Servido" era o título do blog que anunciava, em 25 de setembro de 2010, a chegada do Arduino Uno (uno significa um em italiano) e de seu irmão mais velho, o Mega2560. O Arduino Uno possui compatibilidade de pinos com os Arduinos anteriores, incluindo o Duemilanove e seu antecessor Diecimila.

A maior diferença entre o Uno e seus antecessores é a inclusão de um microcontrolador programado ATmega8U2 como um conversor USB-para-serial, substituindo o chipset FTDI obsoleto usado nas versões anteriores. O ATmega8U2 pode ser reprogramado para fazer o Arduino se parecer com outro dispositivo USB, tal como mouse, teclado ou joystick. Outra diferença é que ele possui uma tensão integrada de 3,3 V mais confiável, o que ajuda na estabilidade de algumas proteções que causavam problemas no passado. Veja o apêndice C para obter especificações técnicas completas.

A figura 1.1 mostra o layout da placa e os pinos no Arduino Uno.

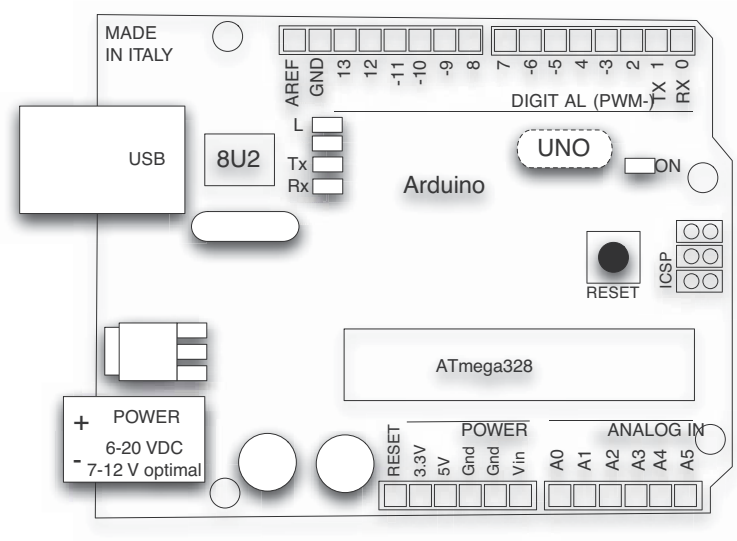

Figura 1.1 – Layout da placa e pinos do Arduino Uno.

O Arduino Uno é uma boa opção multiúso e é sua melhor aposta para uma placa de partida com fonte de alimentação autochaveada e tensão integrada de 3,3 V regulada.

1.2.2 Arduino Duemilanove

O Duemilanove (que significa 2009 em italiano) é uma das placas Arduino mais populares já produzidas, tendo substituído sua antecessora, a Arduino Diecimila. Todavia, por sua vez, foi substituído pelo Arduino Uno, mais novo e mais atual. O Duemilanove se caracteriza pela seleção de potência de comutação automática entre o externo e o USB e usa o processador ATmega328, embora os modelos anteriores a março de 2009 usassem o ATmega168. Seu layout de pinos e capacidades é idêntico ao Uno e usa o chipset FTDI para comunicação USB para serial.

Se você for comprar um novo Arduino, deve escolher o Arduino Uno. Se você já possui um Duemilanove, considere atualizar para o Uno caso você precise de uma tensão de 3,3 V mais estável ou queira fazer uma programação mais avançada com o ATmega8U2.

1.2.3 Arduino Ethernet

O Arduino Ethernet é uma versão de baixa potência do Arduino anunciada ao mesmo tempo em que o Uno. As principais diferenças entre uma versão e outra são que o Arduino Ethernet possui um conector RJ45 integrado para uma conexão Ethernet e um leitor de cartão micro SD. O Arduino Ethernet não possui um chip controlador USB para serial integrada, mas possui um conector de seis pinos que pode ser conectado a um cabo FTDI ou a uma porta serial USB para fornecer um link de comunicação para que a placa possa ser programada. Ela pode também ser alimentada por um módulo opcional Power over Ethernet (POE), o qual permite ao Arduino Ethernet retirar sua energia de um cabo Ethernet de par trançado Categoria 5 conectado.

O Arduino Ethernet é ideal para uso em monitoramento remoto e estações de registros de dados com leitor de cartão micro SD integrado e uma conexão com uma rede Ethernet com fio para alimentação.

1.2.4 Arduino Mega

O irmão mais velho da família Arduino, o Mega, usa um microprocessador de maior superfície de montagem. O ATmega1280, o Mega, foi atualizado ao mesmo tempo que o Uno, e o microprocessador usado agora é o ATmega2560. A nova versão possui memória flash de 256 KB, superior aos 128 KB do original.

O Mega fornece um aumento significativo na funcionalidade de entrada-saída em relação ao Arduino padrão; portanto, com o aumento da memória, ele é ideal para aqueles projetos maiores que controlam grandes quantidades de LEDs, possuem um grande número de entradas e saídas ou necessitam de mais de uma porta serial de hardware – o Arduino Mega possui quatro. As placas possuem 54 pinos digitais de entrada-saída, 14 dos quais podem fornecer saída analógica PWM, e 16 pinos de entrada analógica. A comunicação é feita com até quatro portas seriais de hardware. A comunicação SPI e o suporte para dispositivos I2C/TWI estão também disponíveis. A placa também inclui um conector ICSP e um botão de reset. Um ATmega8U2 substitui o chipset FTDI usado pelo seu antecessor e processa a comunicação serial USB.

O Mega opera com a maioria dos shields disponíveis, mas é uma boa ideia verificar se um shield será compatível com seu Mega antes de comprá-lo. Compre o Mega quando você realmente necessitar utilizar pinos adicionais de entrada-saída e precisar de maior capacidade de memória. Veja o apêndice C para especificações técnicas completas.

A figura 1.2 mostra os pinos e o layout da placa.

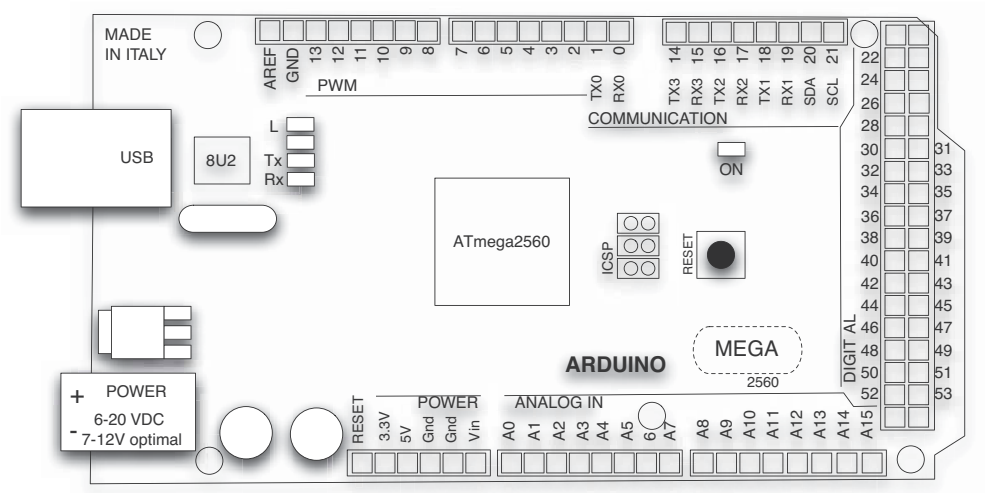

Figura 1.2 – Os pinos e o layout do Arduino Mega; note os pinos adicionais de entrada-saída e as portas seriais extras comparadas com o Arduino Uno.

Agora vamos dar uma olhada em algumas opções mais especializadas de Arduino.

1.2.5 Outras placas Arduino

O Arduino original gerou um grande número de variações que compactou o projeto de maneiras diferentes, geralmente em resposta a uma necessidade. Vamos dar uma olhada em dois deles: o Lily-Pad e o Nano.

Lilypad Arduino

Desenvolvido pela SparkFun Electronics e pela Leah Buechley, o LilyPad Arduino é ótimo para projetos têxteis e em esteiras industriais. Ele foi projetado com grandes blocos de conexão que podem ser costurados ao tecido, e há uma gama de acessórios costuráveis disponíveis, incluindo sensores de luz, campainhas, LEDs tricolores, sensores de temperatura, kits de costura e acelerômetros. Sua versão de baixa potência é ainda lavável; apenas não se esqueça de tirar suas baterias antes de molhá-lo.

A principal diferença entre o LilyPad e outros Arduinos é uma velocidade de processamento menor de 8 MHz, em contrapartida à normal de 16 MHz. Uma coisa a se observar: a tensão de entrada não deve exceder 5,5 V. Veja na figura 1.3 uma foto do Lily-Pad Arduino.

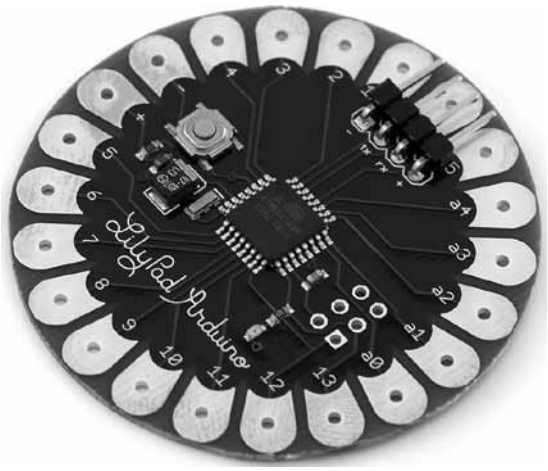

Figura 1.3 – O LilyPad Arduino é adequado para se costurar ao tecido, e há uma gama de acessórios costuráveis disponível.

Arduino Nano

Se o seu projeto tem espaço limitado, o Arduino Nano é a melhor escolha. Projetado e produzido pela Gravitech, a versão 3.0 do Nano (com o processador ATmega328) tem uma mini-USB integrada, um formato compacto para uso em placas de testes.

O Nano possui funcionalidade similar à do Duemilanove, mas tem dois pinos adicionais de entrada analógica. A alimentação para a placa é fornecida por USB ou por dois pinos separados: o pino 30 pode receber uma tensão desregulada entre 6 V e 20 V, ou o pino 27 pode receber uma tensão regulada de 5,5 V. A placa seleciona a tensão que for mais alta.

O tamanho reduzido da placa faz com que seja ideal para projetos com espaço limitado.

1.2.6 Ataque dos clones

Desde o início, o Arduino foi concebido como um hardware de código aberto (open-source). Usuários eram livres para acessar o projeto, baixar os arquivos computer-aided design (CAD) publicados, produzir e vender hardware baseados neles. Isso levou à produção de um número de cópias do Arduino, com vários fabricantes de clones usando a especificação original para fazer suas próprias alterações.

O nome Arduino é uma marca registrada que impede que derivados utilizem seu nome em seus produtos, a menos que a permissão seja dada pela equipe do Arduino.

Seeeduino (Sim, 3 E'S)

Se você gosta da cor vermelha, esta é a sua placa. Projetada e produzida por Seeed Studio, em Shenzhen, na China, o Seeeduino é baseado no projeto da Diecimila, uma das placas iniciais do Arduino, e pode ser comprado com um microprocessador ATmega168 ou ATmega328. Utiliza componentes SMD discretos e tem uma cor vermelha característica.

A placa é compatível com o layout de pinos e dimensões da Diecimila. As melhorias incluem a detecção automática entre alimentação via USB e externa e melhores fontes de alimentação integradas.

Seeeduino Film

O Seeeduino Film traz um conceito de utilização diferente daquele de arquitetura baseada em tecido do LilyPad. Este clone Arduino flexível, o qual pode também ser usado em projetos de registro de dados, tem uma montagem em superfície ATmega168 em uma placa de circuito impresso flexível. Em vez de shields, a expansão é ativada por meio do que os fabricantes chamam de frames. Um frame foi produzido até o momento consistindo de um barômetro, memória flash de 32 MB e um acelerômetro de três eixos, que deve ser mais do que suficiente para atingir o objetivo.

BoArduino

O BoArduino é uma pequena placa similar ao Nano 3.0, mas disponível apenas em forma de kit, portanto são requeridas habilidades para solda. Produzido pela Adafruit Industries, o BoArduino é projetado para ser ligado diretamente em uma placa de teste sem solda. O kit está disponível em duas versões, uma com USB e outra com uma conexão serial na qual um cabo adicional é requerido. Ele usa o ATmega328.

Sippino

O Sippino é um clone miniatura da SpikenzieLabs, compatível com Arduino, vendido em forma de kit, e assim como o BoArduino requer conhecimentos de solda. O Sippino utiliza o ATmega328, mas também pode utilizar o ATmega168. Todos os pinos digitais e analógicos de entrada e saída são trazidos para fora em uma única linha para que possam ser conectados diretamente a uma placa de testes sem solda. Um cabo serial FTDI-USB é necessário para programar a placa.

Ebay

Um grande número de placas clone é vendido no eBay, muitas dessas placas são cópias da Duemilanove. Aqui estão algumas coisas a se considerar ao adquirir qualquer clone: tenha certeza de que ele possui um microprocessador ATmega328 e que os conectores são adequados para adicionar shields.

O primeiro Arduino que compramos no eBay possuía conectores machos ao invés de fêmeas, o que tornou difícil adicionar shields. Também tivemos de comprar alguns jumpers especiais para conectar à placa de testes. Isso foi suficiente para que iniciássemos, mas é melhor evitar tais enganos e verificar se você está comprando o que você realmente deseja.

1.2.7 Obtendo um Arduino

Se você está procurando um Arduino, recomendamos que comece com o Uno, com sua conexão USB superior e uma melhor fonte de alimentação regulada integrada.

O Arduino Uno está disponível em diversas lojas online. As três mais importantes nos Estados Unidos são SparkFun Electronics (*www.sparkfun.com*), Adafruit Industries (*http://adafruit.com*) e Maker Shed (*http://makershed.com/*). No Reino Unido, há a SK Pang Electronics (*http://skpang.co.uk*) e a oomlout (*http://oomlout.co.uk*). Uma lista completa dos distribuidores ao redor do mundo está disponível no website principal do Arduino (*http://Arduino.cc/en/Main/Buy*).

Se você já tem um Arduino, pode iniciar a conexão ao seu sistema e criar seu ambiente de trabalho.

1.3 Configurando seu ambiente de trabalho

Quando você receber um Arduino novinho em folha, provavelmente ficará ansioso para começar. Esta seção deve ajudá-lo a acabar com essa ansiedade. Assim que conectar seu Arduino ao seu computador pela primeira vez, você aprenderá o que é necessário para configurar o seu ambiente de trabalho.

Para iniciar, você precisará de um Arduino. Como mencionado na seção anterior, um Duemilanove ou um Uno é uma boa opção. Você também precisará de um cabo USB para conectar o Arduino ao seu computador.

1.3.1 Software para Arduino

Neste momento, seu Arduino é somente uma placa com alguns componentes eletrônicos. Para que ela faça algum trabalho útil, você deve dar-lhe instruções, e é por isso que você precisa do IDE do Arduino. O IDE do Arduino fornece tudo o que é necessário para programá-lo, incluindo vários exemplos de programas ou sketchs que demonstram como conectá-lo e comunicar-se com alguns dispositivos comuns, tais como LEDs, LCDs e alguns sensores.

Você ficará satisfeito de saber que, assim como o hardware, o software para o Arduino é de código aberto e pode ser baixado gratuitamente de *http://Arduino.cc/en/Main/Software*. Tenha certeza de baixar a versão correta para seu sistema. As versões do IDE estão disponíveis para Windows, Mac OS X e Linux. Para instruções completas de cada plataforma, veja o apêndice A.

É importante familiarizar-se com o IDE, pois é onde você escreverá todo o seu código. No mundo do Arduino, um bloco de códigos é chamado de sketch. Um sketch fornece ao Arduino uma lista de instruções e o Arduino esboça a sua ideia. O IDE ajuda a mascarar muito da complexidade do Arduino, tornando-o mais fácil de desenvolver projetos.

NOTA: O termo *sketch* vem de Processing, uma linguagem frequentemente ensinada a estudantes de design e artes, e na qual o Arduino IDE é baseado. Aqueles que já são familiarizados com programação devem pensar em um sketch como sendo um programa de software.

1.3.2 Configuração básica de hardware

A placa Arduino é conectada ao seu computador via USB. O cabo USB fornece os 5 V necessários para alimentar o Arduino e proporciona energia suficiente para acender um par de LEDs, permitindo alguma experimentação básica.

1.3.3 Sua caixa de ferramentas do Arduino

Aqui está uma lista de compras que recomendamos para alguém que está começando a utilizar o Arduino:

- Arduino (Uno ou Duemilanove)
- Miniplaca de testes (breadboard) e jumpers (usados para montar pequenos circuitos)
- Seleção de LEDs
- Seleção de resistores
- Bateria de 9 V
- Conector da bateria
- Fotorresistor
- Pequeno motor de corrente contínua ou servomotor
- Piezo buzzer (um tipo de alto-falante pequeno, como aqueles encontrados em cartões musicais)
- Potenciômetro (um tipo de resistor variável)

Os projetos típicos que você pode desenvolver com esses componentes incluem LEDs intermitentes, semáforos miniatura, uma campainha musical e um interruptor ativado por luz.

Se você se sentir um pouco mais aventureiro, pode adicionar os seguintes componentes:

- GPS Adafruit e shield de registro de dados para gravação de dados de sensores, hora e posição
- Shield Adafruit Wave para reprodução de áudio a partir de um cartão de memória SD para efeitos especiais
- Shield motor para acionar alguns motores, possivelmente o início de movimento de robôs

Um kit de componentes básicos, incluindo um Arduino e uma seleção de componentes, pode ser adquirido em diversos revendedores que oferecem desconto quando você compra um kit.

Agora que seu ambiente de trabalho foi configurado, está na hora de escrever seu primeiro sketch e executar o hardware equivalente a "Hello World."

1.4 Faça algo acontecer!

Antes de você sair correndo para aproveitar todos estes componentes emocionantes, tudo o que você precisa para o seu primeiro exemplo é um Arduino, pois todos eles possuem um LED integrado conectado ao pino digital 13. Para este primeiro exemplo, você fará um LED acender e apagar repetidamente.

NOTA: Se você quiser se aventurar mais, nós também incluímos instruções para adicionar um LED externo na seção 1.4.3.

1.4.1 Seu primeiro LED intermitente

Os LEDs estão disponíveis em uma gama de cores, mas o LED conectado ao pino 13 do Arduino é normalmente verde. O LED acende quando uma corrente é aplicada a ele, então você pode usar o pino 13 como uma chave. Quando você a ligar, ela acenderá o LED, e quando você a desligar, ela apagará o LED.

Vamos começar escrevendo o sketch.

1.4.2 Sketch para fazer um LED piscar

Inicie o IDE do Arduino e copie o código seguinte. Pode parecer um pouco difícil em um primeiro momento, mas não se preocupe. Nós vamos entrar em mais detalhes sobre o que tudo isso significa mais adiante neste capítulo.

Listagem 1.1 – Código necessário para fazer um LED piscar

```
void setup() {
   pinMode(13, OUTPUT);
}
```

```
void loop() {
   digitalWrite(13, HIGH);
   delay(1000);
   digitalWrite(13, LOW);
   delay(1000);
}
```

O código é simples. Você está atribuindo o pino digital 13 como uma saída e, em seguida, você entra em um loop no qual algum código muda o pino 13 para `HIGH` ou `LOW` durante um segundo. O valor do atraso é dado em milissegundos, então mil milissegundos lhe dá um atraso de tempo de um segundo.

NOTA: Certifique-se de copiar o código exatamente como mostrado. Cuidado com o ponto e vírgula (;) ao final de algumas linhas e o uso correto das letras maiúsculas. No que diz respeito ao Arduino, `digitalwrite` não é o mesmo que `digitalWrite`.

1.4.3 Conectando tudo

Se você conectar seu Arduino ao seu computador por meio do cabo USB, este sketch controlará o LED integrado próximo ao pino 13.

Você pode também controlar um LED externo conectando-o entre o pino 13 e o GND. A conexão é mostrada na figura 1.4.

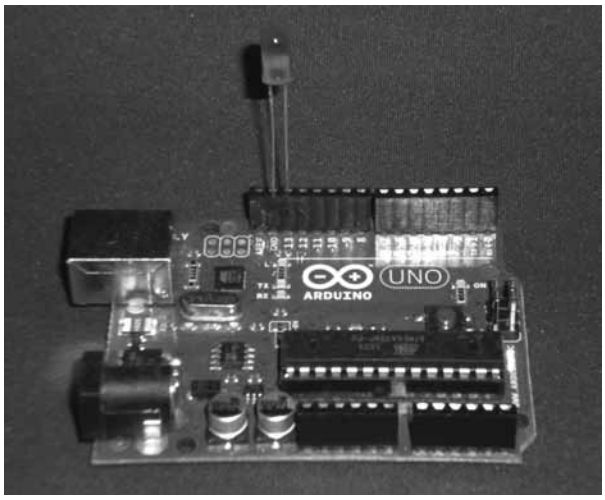

Figura 1.4 – LED inserido entre o pino 13 e o GND. Note que a perna menor é conectada ao GND.

Note que o LED deve estar conectado de forma correta – a perna mais curta é o catodo ou –, e a mais comprida é o anodo ou +, então empurre a perna mais longa dentro do pino 13 e a mais curta dentro do GND. Se você desconhece alguns termos da eletrônica, uma boa cartilha pode ser encontrada em *http://electronics-club.info/components.htm*.

NOTA: Um resistor limitador de corrente pode ser necessário para prevenir a queima do LED, e veremos isso no capítulo 2. Por enquanto, vamos apenas usar o LED integrado existente. Uma vez que o LED foi inserido, você pode passar para a próxima seção para testar sketch.

1.4.4 Carregando e testando

Hora de ver se o seu sketch funciona! Primeiro, conecte o Arduino ao seu computador por meio do cabo USB. Você agora tem alguns ajustes a fazer entre o software e o Arduino.

Então você precisa definir o tipo de placa. Selecione **Tools > Board** e então selecione o tipo de Arduino que você está usando. Veja a figura 1.5.

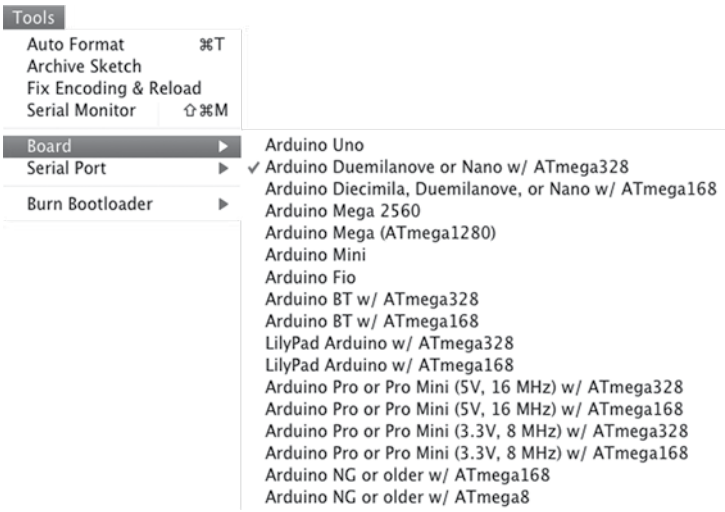

Figura 1.5 – Neste exemplo, o Duemilanove foi selecionado, mas você pode ver que há uma grande lista a escolher.

Depois, você precisa definir a porta serial, pois o USB enxerga o Arduino como uma conexão serial. Selecione **Tools > Serial Port** e então escolha sua porta serial (veja a figura 1.6). Em um sistema usando Mac OS X para um Arduino Uno, a porta será identificada como algo parecido com `/dev/tty.usbmodem`; para placas mais antigas como a Duemilanove ou Diecimila, será algo como `/dev/tty.usbserial`. Em um sistema Windows, a porta será identificada como COM3.

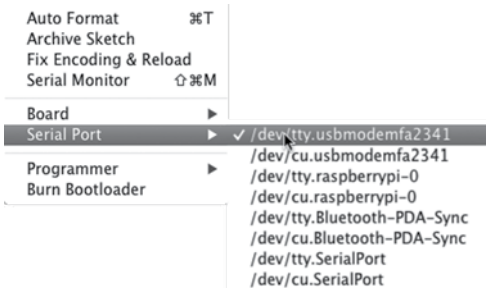

Figura 1.6 – Selecione a placa serial correta da lista.

NOTA: A figura 1.6 mostra a seleção para um sistema usando Mac OS X. A porta em um sistema Windows é diferente e será algo parecido com COM3.

O próximo passo é clicar no botão Upload no IDE. Veja a figura 1.7.

Figura 1.7 – Clique no botão Upload para carregar o sketch para o Arduino.

Espere alguns segundos, e então o LED deve começar a piscar em uma taxa de cerca de uma vez por segundo.

NOTA: O Arduino retém o programa em sua memória mesmo se ele for desligado até que você carregue outro sketch.

É sempre emocionante quando você vê o primeiro LED piscando e sabe que tudo está funcionando corretamente, mas isso não é tudo o que você pode fazer com seu Arduino. Você agora obterá uma visão mais detalhada do IDE e fará um tour na tela principal do editor de códigos.

1.5 Passeando pelo IDE

Como informado anteriormente, o IDE é baseado na linguagem Processing, a qual foi projetada para facilitar a utilização e a aprendizagem. O IDE fornece tudo o que você precisa para escrever e carregar sketches (programas) no Arduino.

1.5.1 O editor principal

Quando o IDE é carregado pela primeira vez, ele abre um sketch em branco; para esse sketch é dado automaticamente um nome temporário com uma data de referência. O nome pode ser alterado depois para algo mais apropriado quando você salvar o sketch.

A figura 1.8 mostra o IDE que contém um sketch, com anotações para os vários botões e as janelas.

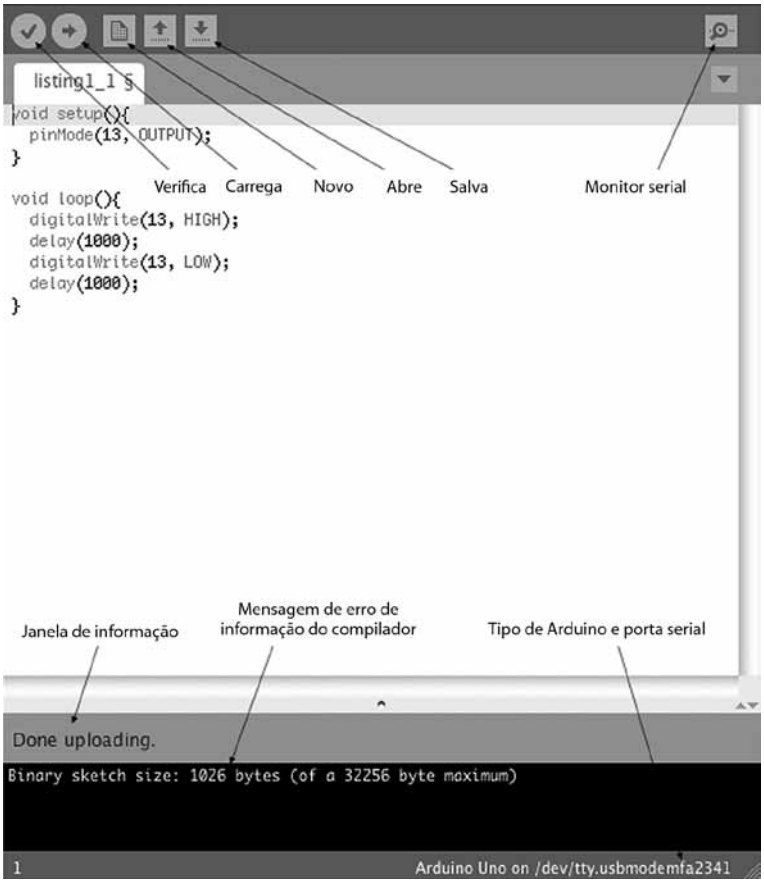

Figura 1.8 – Sketch típico com os botões e as áreas da tela rotulada.

A barra de tarefas ao longo do topo do editor principal tem as seguintes funções:

- **Verify** – Verifica sketches em busca de erros. Os erros são apresentados na parte inferior da tela.
- **New** – Abre um novo sketch.
- **Open** – Abre uma lista de sketches salvos previamente e exemplos de sketches.
- **Save** – Salva o sketch e solicita um nome se esse for o primeiro a ser salvo.
- **Upload** – Verifica o código em busca de erros antes de carregar o sketch para o Arduino.
- **Serial monitor** – Abre o monitor serial em uma nova janela (veja figura 1.9 na próxima seção).

No parte inferior da tela principal existem duas janelas. A primeira fornece informação de status e comentários; a segunda fornece informações quando você está verificando e carregando sketches. Qualquer erro de código também será reportado aqui.

O editor de código adicionalmente verifica a combinação das chaves, {}, usadas para designar blocos de código, executa o realce de sintaxe e automaticamente endenta o seu código para facilitar a leitura.

1.5.2 Monitor serial

O monitor serial mencionado na seção anterior monitora os dados entre o Arduino e o sistema do computador central através do cabo USB conectado. O Arduino pode enviar informações utilizando código e pode também recebê-lo. Você pode ver isso na figura 1.9.

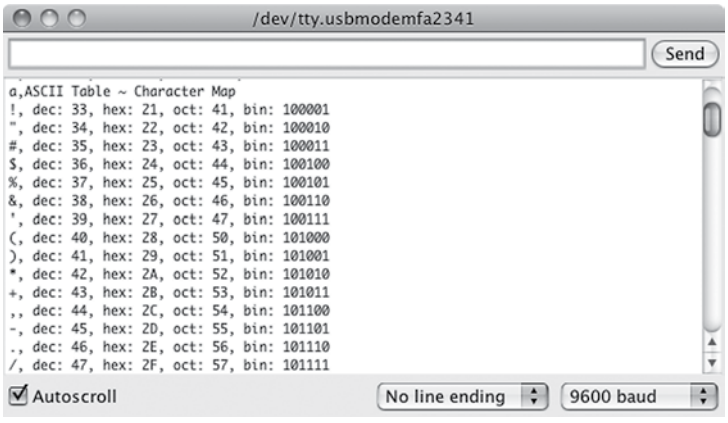

Figura 1.9 – Monitor serial mostrando a saída de um Arduino imprimindo uma tabela ASCII.

A parte superior da janela do monitor serial é usada para enviar dados ao Arduino. Você pode, por exemplo, usá-la para enviar comandos de controle ao Arduino, para girar um servomotor a um número variável de graus ou para abrir ou fechar um interruptor. A parte principal da janela mostra os dados de saída do Arduino. Isso pode ser usado para verificar a saída de um GPS ou para executar outro monitoramento de sinal.

O motor serial é muito útil para depuração do código ao ligar o Arduino ao sistema do computador que executa o software que então interage de alguma maneira com o Arduino; você pode usar o monitor serial para verificar se o Arduino está gerando os dados corretamente no formato esperado. No monitor serial, você também pode definir a taxa de transmissão usada para a comunicação, a rolagem automática de texto e a forma de término de linha que é anexada aos dados enviados ao Arduino.

1.5.3 Erros de captura

Agora vamos retornar ao editor principal. A área principal da tela é o editor de código onde você digita seu código. Uma vez que terminou de introduzir seu código, você tem a opção de verificar ou carregar seu sketch ao Arduino.

Quaisquer erros de código são reportados na parte inferior da janela. Na figura 1.10, introduzimos um erro omitindo ponto e vírgula (;) ao final de uma das linhas do código.

Os detalhes do erro são fornecidos, bem como a linha onde o erro ocorre. O verificador de código fornece informações suficientes para apontá-lo na direção correta, se não mostrar exatamente o que está errado. Como você pode ver na figura 1.10, o verificador de código identificou corretamente o ; que falta e onde o erro ocorreu.

1.5.4 Processo

O que o IDE realmente faz com o seu código? Quando você pressiona o botão "carregar", ele verifica o código procurando erros e executa algumas traduções mínimas para converter o sketch para um C++ válido. O código é então compilado, o que significa que é convertido numa forma que pode ser entendido pelo Arduino. O arquivo resultante é então combinado com as bibliotecas padrão do Arduino antes de ser carregado no seu hardware.

Agora que você fez um tour pelo IDE, é hora de ter uma noção melhor dos sketches do Arduino.

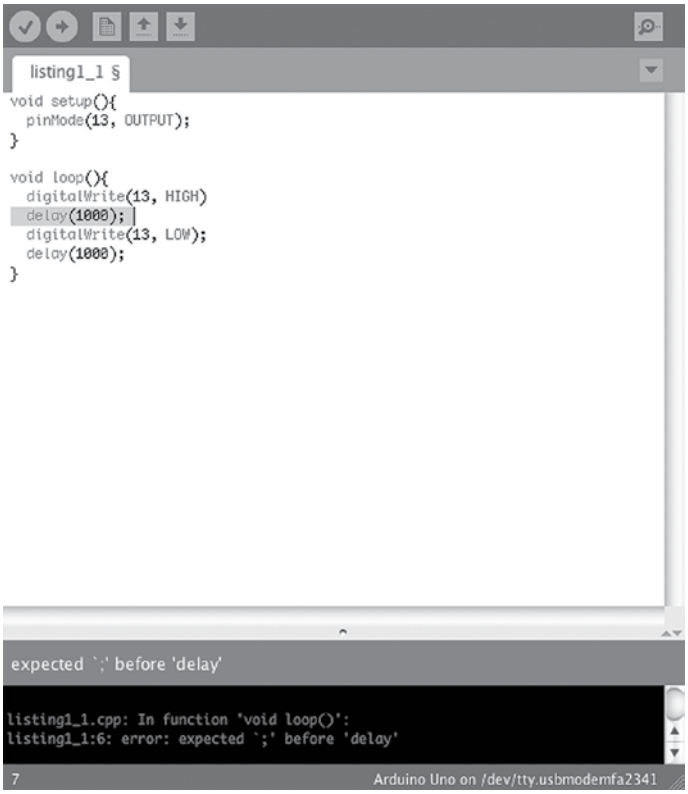

Figura 1.10 – O editor de código reporta um erro que introduzimos no código. O verificador de código indica em qual linha ele acha que o erro ocorreu, bem como o que ele espera.

1.6 Anatomia de um sketch

Um sketch típico consiste de duas partes ou rotinas: a primeira é a rotina de inicialização chamada setup, e a segunda é a rotina chamada loop, que geralmente contém o corpo principal do código. Vamos dar uma olhada mais detalhada nessas duas rotinas a seguir.

1.6.1 Uma rotina chamada setup

Quando você deseja sair para uma corrida, existem coisas que deve fazer antes de iniciar: colocar seus tênis de corrida, pegar uma garrafa de água e alongar-se. Acontece o mesmo com o Arduino. Ele deve ser preparado ou configurado antes que você possa trabalhar com ele.

Esta configuração está contida dentro de uma rotina de inicialização ou função adequadamente chamada `setup` (veja a listagem a seguir). Coisas típicas que você deve fazer no setup incluem a inicialização dos pinos digitais – definindo-os como `INPUT` ou `OUTPUT` – e definição da taxa de transmissão para a comunicação serial.

Listagem 1.2 – A função setup

```
void setup() {
   pinMode(13,OUTPUT);
   Serial.begin(9600);
}
```

O código de setup na listagem 1.2 configura o pino 13 como uma saída e configura a comunicação serial a uma taxa de transmissão de 9.600. O void na frente do setup apenas significa que a função não retorna um valor.

Mesmo que você não tenha algo a configurar, a rotina é ainda requerida ou um erro será gerado ao verificar ou carregar um sketch. Apenas digite uma função vazia com um comentário no código:

```
   void setup() {
   // nada para configurar
   {
```

Agora vamos olhar para a outra função necessária, `loop`.

1.6.2 O loop infinito

Quando você vai para uma corrida, corre do começo até o fim (no entanto você define o fim). É o mesmo com um Arduino; ele roda continuamente em uma rotina de loop ou função chamada `loop` até que alguma condição seja satisfeita ou o Arduino seja desligado. A listagem seguinte mostra o loop para o LED intermitente da listagem 1.1.

Listagem 1.3 – Um exemplo de uma função de loop que pisca um LED aceso e apagado

```
void loop() {
   digitalWrite(13, HIGH);
   delay(1000);
   digitalWrite(13,LOW);
   delay(1000);
}
```

Neste caso, o Arduino fica em loop repetidamente, acendendo o LED por um segundo e apagando por um segundo, e continua até que o Arduino seja desligado.

Agora que você conhece os princípios básicos de como escrever um sketch, é hora de fechar o capítulo com um lembrete importante.

1.7 Comentando o código

Você escreveu um grande pedaço de código e está realmente orgulhoso. Agora imagine que, seis meses depois, alguém mais está lendo seu trabalho e encontra o seu sketch, mas não consegue descobrir o que ele faz ou como funciona. Uma simples descrição ajudaria imensamente. Aqui é onde o comentário do seu código se torna inestimável.

Existem duas maneiras de colocar comentários em um sketch: como uma única linha ou como um bloco. Uma única linha tem barras duplas (//) ao início da linha. Isso diz ao compilador que aquele conteúdo é apenas um comentário e pode ser ignorado. Se você deseja adicionar um bloco de códigos como um comentário, inicie o bloco com/* e termine com */.

Ambos os métodos são demonstrados aqui:

```
// Este é um comentário de linha única
/* E este é um bloco de comentário em
algumas linhas
*/
```

Onde você deve colocar os comentários? Cada sketch deve possuir um bloco de comentários no topo ou um cabeçalho do sketch, dando uma descrição do que o sketch faz, indicando quem o escreveu, a data e o número da versão. A próxima listagem mostra um exemplo de cabeçalho.

Listagem 1.4 – Exemplo de código de cabeçalho

```
/*
Código para piscar um LED
Autor: Martin Evans
Data de criação: 1 de Setembro de 2009
Versão 1.0
*/
```

Comentários de linha única espalhados por todo o sketch permitirão que você veja rapidamente o que as partes individuais do código fazem. Você não precisa comentar cada parte do código, somente lugares onde acha que ajudaria alguém a entender o código no futuro. Provavelmente é melhor ter muitos comentários do que poucos. A listagem seguinte mostra alguns comentários de códigos típicos.

Listagem 1.5 – Exemplo de comentários de códigos

```
void setup() {
   Serial.begin(9600);
   // imprime o título com quebra de linha ao final
   Serial.println("ASCII Table ~ Character Map");
}
// primeiro ASCIIcharacter visível '! é o número 33:
int thisByte = 33;
/* você pode também escrever caracteres ASCII em aspas simples.
por exemplo. '!' é o mesmo que 33, então você pode também usar este:
int thisByte = '!; */
```

Estudamos o editor de código e o IDE, vimos como um sketch é formado com a função `setup` e `loop` e discutimos a importância dos comentários de códigos.

1.8 Resumo

Este foi um capítulo abrangente, apresentando muitos fundamentos. Começamos aprendendo um pouco da história do Arduino e seu início no Interaction Design Institute na Itália. Vimos o layout dos pinos e os principais componentes do Arduino Uno e Mega. Vislumbramos algumas outras versões do Arduino, incluindo o Lily-Pad e o Seeeduino Film e o que eles oferecem. Você configurou um ambiente de trabalho e escreveu seu primeiro sketch, começando a ver seu Arduino ganhar vida.

Vimos em detalhes o software IDE do Arduino e os componentes de um sketch, com as rotinas `setup` e `loop`, o uso do monitor serial e também a importância de comentar seu código.

O próximo capítulo é um tutorial que abrange o desenvolvimento gradual de um projeto e os passos envolvidos para completá-lo.