



PROJETOS COM ARDUINO E ANDROID

» USE SEU SMARTPHONE OU TABLET
PARA CONTROLAR O ARDUINO

SIMON
MONK

» série tekne



PROJETOS COM

ARDUINO

E ANDROID

O autor

Simon Monk é bacharel em cibernética e ciência da computação e doutor em engenharia de software. Desde seus tempos de escola, é aficionado em eletrônica. Ocasionalmente, publica artigos em revistas dedicadas à eletrônica. Também é o autor de *Programação com Arduino: começando com sketches* (publicado pela Bookman Editora).



M745p Monk, Simon.

Projetos com arduino e android [recurso eletrônico] :
use seu smartphone ou tablet para controlar o arduino / Simon
Monk ; tradução: Anatólio Laschuk. – Dados eletrônicos. –
Porto Alegre : Bookman, 2014.

Editedo também como livro impresso em 2014.

ISBN 978-85-8260-122-8

1. Programas de computador. 2. Arduino. 3. Android.
I. Título.

CDU 004.42

SIMON MONK

**PROJETOS COM
ARDUINO
E ANDROID**

**» USE SEU SMARTPHONE OU TABLET
PARA CONTROLAR O ARDUINO**

Tradução

Anatólio Laschuk

Mestre em Ciência da Computação pela UFRGS

Professor aposentado pelo Departamento de Engenharia Elétrica da UFRGS

Versão impressa
desta obra: 2014



2014

Obra originalmente publicada sob o título *Arduino + Android Projects for the Evil Genius: Control Arduino with Your Smartphone or Tablet*, 1st Edition
ISBN 007177596X / 9780071775960

Original edition copyright © 2012, The McGraw-Hill Global Education Holdings, LLC, New York 10020. All rights reserved.

Portuguese language translation copyright © 2014, Bookman Companhia Editora Ltda., a Grupo A Educação S.A. Company. All rights reserved.

Gerente editorial: *Arysinha Jacques Affonso*

Colaboraram nesta edição:

Editora: *Maria Eduarda Fett Tabajara*

Assistente editorial : *Danielle Oliveira da Silva Teixeira*

Capa e projeto gráfico: *Paola Manica*

Leitura final: *Susana de Azeredo Gonçalves*

Editoração eletrônica: *Techbooks*

Reservados todos os direitos de publicação, em língua portuguesa, à
BOOKMAN EDITORA LTDA., uma empresa do GRUPO A EDUCAÇÃO S.A.
Av. Jerônimo de Ornelas, 670 – Santana
90040-340 – Porto Alegre – RS
Fone: (51) 3027-7000 Fax: (51) 3027-7070

É proibida a duplicação ou reprodução deste volume, no todo ou em parte, sob quaisquer formas ou por quaisquer meios (eletrônico, mecânico, gravação, fotocópia, distribuição na Web e outros), sem permissão expressa da Editora.

Unidade São Paulo
Av. Embaixador Mamedo Soares, 10.735 – Pavilhão 5 – Cond. Espace Center
Vila Anastácio – 05095-035 – São Paulo – SP
Fone: (11) 3665-1100 Fax: (11) 3667-1333

SAC 0800 703-3444 – www.grupoa.com.br

IMPRESSO NO BRASIL
PRINTED IN BRAZIL

Para Linda, o amor de minha vida.



Agradecimentos

Sou grato à Linda por ter proporcionado tempo, espaço e apoio para que eu pudesse escrever este livro e por ter tolerado as confusões que meus projetos criaram pela casa.

Também sou grato aos meus filhos, Stephen e Matthew Monk, por terem mostrado interesse pelo que o pai deles estava fazendo e por terem dado todo tipo de ajuda enquanto eu trabalhava nos projetos.

Finalmente, eu gostaria de agradecer ao Roger Stewart, à Patricia Wallenburg, ao Mike McGee e a todos da McGraw-Hill, que novamente realizaram um ótimo trabalho. É um prazer trabalhar com essa equipe tão competente.



Sumário

INTRODUÇÃO 1

parte I

capítulo 1

PERIFÉRICOS ANDROID

ROBÔ BLUETOOTH 9

Construção 10	Resumo 22
Teoria 20	

capítulo 2

CONTADOR GEIGER COM ANDROID 23

O Open Accessory do Google 24	Teoria 36
Construção 25	Resumo 42

capítulo 3

SHOW DE LUZES COM ANDROID 43

Construção: a Base para Acessório	Usando o projeto 55
Droid 44	Teoria 56
Construção: o projeto show de	Resumo 60
luzes 49	

capítulo 4

CONTROLE REMOTO DE TV 61

Construção 62	Teoria 66
Usando o projeto 66	Resumo 68

capítulo 5

REGISTRADOR DE TEMPERATURA 69

Construção 70	Teoria 76
Usando o projeto 74	Resumo 77

capítulo 6

MEDIDOR DE DISTÂNCIA ULTRASSÔNICO 79

Construção 80	Teoria 84
Usando o projeto 84	Resumo 87

parte II**capítulo 7*****AUTOMAÇÃO RESIDENCIAL******CONTROLADOR DE AUTOMAÇÃO RESIDENCIAL 91***

Módulo de conexão de áudio	93	Teoria	109
Software Android	105	Resumo	116
Acesso à Internet	108		

capítulo 8***CONTROLADOR DE POTÊNCIA 117***

A eletrônica do controlador de potência	118	automação residencial	127
Construção do módulo controlador de potência	118	Configurando a sua casa	130
Integração com o controlador de		Teoria	131
		O sketch real	133
		Resumo	134

capítulo 9***TERMOSTATO INTELIGENTE 135***

Construção	136	Teoria	148
Usando o sistema	147	Resumo	152

capítulo 10***FECHADURA COM RFID 153***

Construção	154	Teoria	165
Usando o sistema	164	Resumo	168

capítulo 11***BANDEIROLAS DE SINALIZAÇÃO 169***

Construção	170	Resumo	176
Teoria	175		

capítulo 12***TEMPORIZADOR 177***

Construção	178	Resumo	187
Teoria	184		

apêndice***FUNDAMENTOS DE OPEN ACCESSORY 189***

Aprendendo a programação		A parte do Arduino	191
Android	190	A parte do Android	192
Programando o Arduino	190	Conclusão	198
O exemplo	190		

ÍNDICE 199



Introdução

Este é um livro de projetos em que as placas de microcontrolador (Arduino), fáceis de usar, são combinadas com telefones celulares Android e computadores do tipo tablet.

Este livro contém instruções detalhadas para a construção de diversos projetos que usam dispositivos Arduino e Android. Alguns dos projetos, como o Contador Geiger e o Medidor de Distância Ultrassônico, são basicamente acessórios eletrônicos para serem utilizados com seu celular Android.

Outros projetos do livro compõem um sistema completo de automação residencial, incluindo fechaduras elétricas e um controle remoto para comandar as luminárias, os aquecedores e outros aparelhos. Tais projetos permitem também que o controle da automação residencial seja feito através da Internet e de seu telefone Android.

» Arduino

O Arduino (Figura 1) é uma pequena placa de microcontrolador que contém uma conexão USB, tornando possível a ligação com um computador. Além disso, contém diversos terminais que permitem a conexão com dispositivos externos, como motores, relés, sensores luminosos, diodos a laser, alto-falantes, microfones e outros. Eles podem ser energizados pelo computador através do cabo USB, por uma bateria de 9 V ou por alguma outra fonte de alimentação. Um Arduino pode ser controlado diretamente pelo computador ou, então, pode trabalhar de forma autônoma. Neste caso, ele é primeiro programado pelo computador através da conexão USB e, em seguida, desconectado desse computador.

O projeto da placa é aberto. Isso significa que qualquer um pode construir placas compatíveis com o Arduino, o que gerou competição e resultou em placas de baixo custo.

As placas básicas são complementadas por placas acessórias (shields), que podem ser encaixadas em cima da placa do Arduino. Neste livro, usaremos três shields: um shield USB host que permitirá a conexão com dispositivos Android usando um cabo USB; um shield de motor para acionar as rodas de um pequeno robô; e um shield Ethernet que permitirá transformar o nosso Arduino em um pequeno servidor Web.

O software de programação do seu Arduino é fácil de usar e encontra-se disponível gratuitamente para computadores Windows, Mac e LINUX.

» Android

Android é o sistema operacional do Google que pode ser utilizado com celulares e tablets. O desenvolvimento de aplicativos Android é gratuito. As ferramentas de desenvolvimento de software são gratuitas e não há pagamento de taxas para fazer a distribuição dos aplicativos. Você também poderá distribuí-los sem que a comercialização tenha que ser feita através do Google.

Os aplicativos (ou simplesmente apps) de todos os projetos Android deste livro, como o mostrado na Figura 2, estão disponíveis em www.duinodroid.com. Entretanto, se desejar modificá-los, você poderá acessar gratuitamente os códigos-fontes no site.

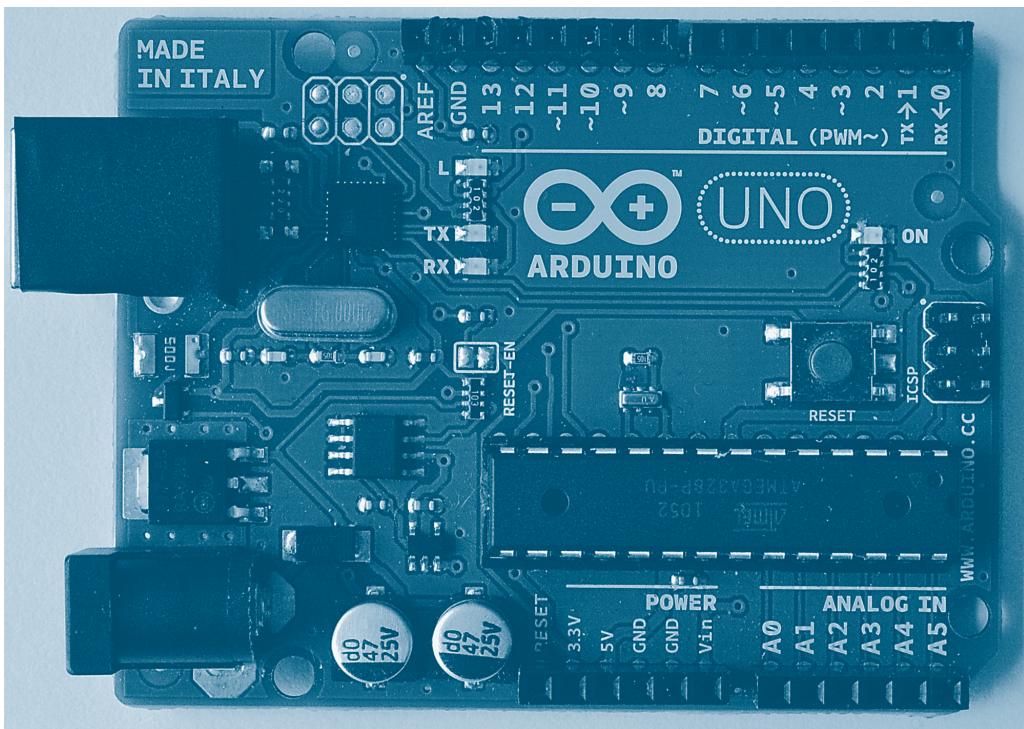


Figura 1 Uma placa de Arduino Uno.

» *Arduino e Android*

O Arduino é constituído de diversos circuitos eletrônicos que permitem a conexão de coisas entre si. Entretanto, pouco oferece para permitir a interação com um usuário ou a conexão sem fio. Por outro lado, o Android oferece muitos recursos de interação, mas nenhum recurso de conexão direta com circuitos eletrônicos.

A combinação dos dois permite que um projetista construa grandes coisas!

» *O Open Accessory do Android*

Na conferência Google de desenvolvimento, realizada em 2011 (Google IO 2011), o padrão Open Accessory foi anunciado. Esse padrão foi lançado para criar acessórios de hardware que podem ser

ligados ao dispositivo Android através da conexão USB. Está disponível para telefones celulares e tablets que funcionam com a versão Android 2.3.4 ou posterior.

Algo realmente bom a respeito desse padrão é que se baseia na tecnologia do Arduino. Isso é uma grande notícia para os entusiastas de Arduino. Cinco dos projetos deste livro (Contador Geiger, Show de Luzes, Controle Remoto de TV, Registrador de Temperatura e Medidor de Distância Ultrassônico) são baseados em Open Accessory.

Este livro introduz o conceito denominado "Droid Duino Base". Nesse caso, o microcontrolador já programado de uma placa de Arduino é retirado e transferido para um soquete na área de protótipo de um shield USB hospedeiro. Isso evita que seja necessário utilizar um Arduino diferente em cada projeto, reduzindo assim o tamanho e baixando o custo a menos de cem reais por projeto.

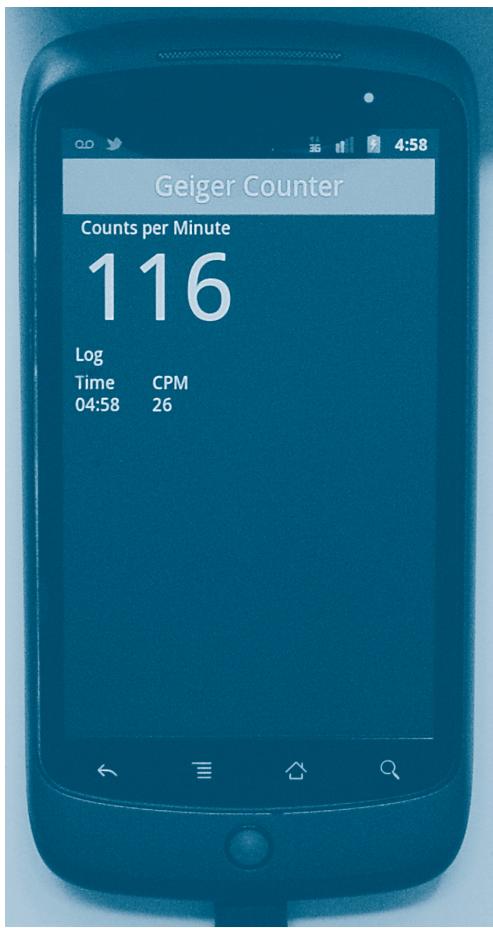


Figura 2 O aplicativo para o Contador Geiger.

Essa é a base utilizada em quatro projetos deste livro. Constitui um módulo prático para ser reaproveitado em seus próprios projetos com Open Accessory.

» **Amarino**

Em algumas situações, uma conexão com fio não é o que precisamos. Por exemplo, no primeiro projeto deste livro (Robô Bluetooth), a presença de fios soltos realmente atrapalharia a movimentação do robô. Entretanto, podemos usar uma parte da tecnologia denominada Amarino. Ela

permitirá controlar remotamente um pequeno robô utilizando um aplicativo Android instalado em um celular.

» **Interface de som**

A segunda seção deste livro trata da construção de um sistema de automação residencial. O controle desse sistema utiliza um tablet Android de baixo custo. Por sua vez, esse tablet comunica-se com um dispositivo Arduino que contém os circuitos eletrônicos de interface. Geralmente, esses tablets não têm capacidade para Bluetooth ou Open Accessory, de modo que construiremos uma interface eletrônica entre o tablet e um Arduino utilizando o conector de áudio.

Aqui, adotaremos a mesma abordagem empregada pelos computadores domésticos da década de 1980.

» **O livro**

Os detalhes de construção de todos os projetos deste livro são explicados passo a passo. Todos requerem a soldagem de alguns componentes, de modo que é necessário um conhecimento elemental de soldagem.

Os diagramas esquemáticos e os leiautes das placas de conexão são fornecidos.

Todos os sketches de Arduino e aplicativos de Android estão disponíveis gratuitamente. Assim, não há necessidade de saber programar. Entretanto, será dada a explicação de todo o software para quem quiser modificar os projetos ou compreender os princípios básicos antes de realizar os seus próprios projetos.

O livro também contém um apêndice com os fundamentos do Open Accessory do Android. Isso será útil para quem quiser saber mais a respeito do Open Accessory e aprender a programar nos ambientes do Arduino e do Android.

» Projetos

Cada um dos projetos deste livro está descrito no seu próprio capítulo. A maioria dos projetos pode ser construída isoladamente, mas todos os projetos de automação residencial dos Capítulos 8, 9, 10 e 11 necessitam do controlador de automação residencial do Capítulo 7.

Os projetos deste livro estão resumidos na tabela mostrada a seguir.

Na coluna de dificuldade, o número de estrelas de cada projeto dá uma ideia da facilidade de constru-

ção. Quanto mais estrelas, mais difícil é o projeto. Nenhum dos projetos requer soldagem de componentes de montagem superficial (SMD) nem mais do que uma simples placa perfurada com furos distanciados de 1/10 de polegada.

» Componentes

Todos os componentes utilizados são de fácil aquisição. Sempre que adequado, os fornecedores e também os números de código das partes são dados. No caso de componentes comuns, os núme-

Capítulo	Projeto	Notas	Dificuldade
1	Robô Bluetooth	Um projeto para controlar um pequeno veículo com o seu celular Bluetooth e um shield de motor para Arduino.	★★
2	Contador Geiger Android	Um projeto Android com Open Accessory usando Arduino Uno e um shield USB hospedeiro.	★★★★
3	Show de Luzes Android	Uma base para o Open Accessory do seu celular Android, capaz de acionar três painéis de LEDs e realizar um show de luzes sensível ao som.	★★★
4	Controle Remoto de TV	Um acessório programável de controle remoto infravermelho para o seu celular Android.	★★★
5	Registrador de Temperatura	Um registrador de temperatura que usa um celular Android para enviar leituras ao Pachube.	★★★
6	Medidor de Distância Ultrassônico	Um projeto Android com Open Accessory para medir distâncias.	★★★
7	Controlador de Automação Residencial	Uma unidade básica Android constituída de um tablet conectado a um Arduino por meio de uma interface de conexão de áudio.	★★★
8	Controlador de Potência para Automação Residencial	Um projeto para acrescentar controladores de lâmpadas e tomadas CA ao projeto de Controlador de Automação Residencial do Capítulo 7.	★★
9	Termostato Inteligente Residencial	Um projeto para acrescentar um controle remoto de aquecimento doméstico ao controlador de automação residencial usando módulos RF de dados de baixo custo.	★★★
10	Fechadura com RFID	Um projeto para controlar o acesso à sua casa usando etiquetas RFID. Inclui também uma conexão de RF com o controlador de automação residencial.	★★★
11	Bandeirolas de Sinalização	Um projeto para, pela rede, controlar duas bandeirolas que podem ser ativadas a partir de qualquer dispositivo conectado à Internet. Útil para chamar atendentes.	★★
12	Temporizador	Um temporizador baseado em Arduino de fácil construção.	★★★

ros de código são os usados pela empresa Farnell. Mesmo que você não os encomende dessa empresa, você poderá utilizar esses códigos para acessar o seu site e identificar exatamente quais são esses componentes antes de fazer pedidos a outros fornecedores.

A empresa SparkFun é uma fornecedora com entrega rápida e confiável de hardware para Arduino. No Reino Unido, a Proto-PIC tem uma boa variedade de placas e shields para Arduino a preços competitivos.

» Começando

Se você estiver mais interessado em projetos que usam o Open Accessory do Android, então o projeto "Show de Luzes Android" é um bom projeto para

começar. Nele você encontrará instruções para construir a "Base para Acessório Droid", o componente principal dos demais projetos que utilizam Open Accessory (com exceção do Contador Geiger).

Os leitores interessados em automatizar sua residência poderão começar pelo Capítulo 7, já que é a base dos projetos seguintes de automação residencial.

Se estiver interessado em saber mais sobre o uso do Arduino, você poderá consultar os seguintes livros do autor: *Programação com Arduino: começando com sketches* (Bookman Editora, 2013) e *30 Projetos com Arduino**.

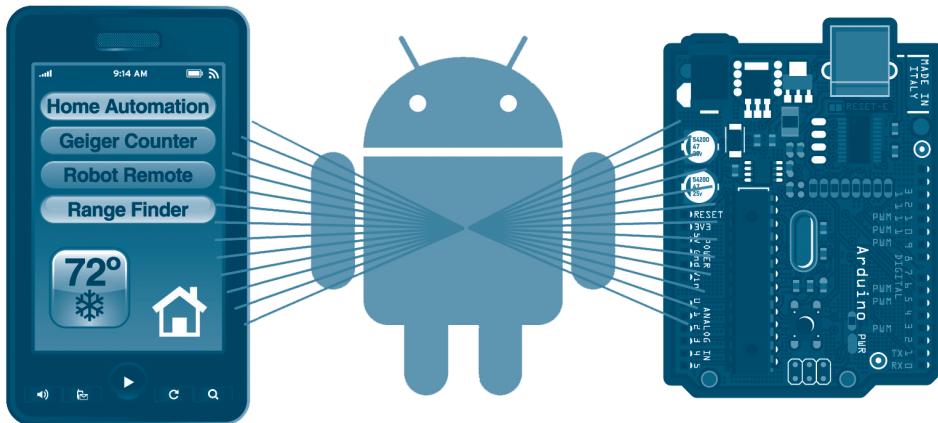
Códigos-fontes, aplicativos e muitos outros recursos podem ser obtidos acessando o site do livro em www.duinodroid.com (clique em "Arduino and Android Projects for the Evil Genius").

* Nota: Livro em produção pela Bookman Editora. Logo estará à disposição do leitor.



Parte I

Periféricos Android





» capítulo 1

Robô Bluetooth

O celular é um aparelho muito útil. Com ele podemos fazer compras, enviar mensagens e muitas outras coisas. Com o celular também é possível dirigir pequenos robôs usando um controle remoto baseado em Bluetooth. Neste capítulo, veremos como isso é possível.

Objetivos

- » Aprender a controlar um pequeno Robot baseado em Android utilizando um Shield de motor para Arduino e comunicação via Bluetooth
- » Conhecer o código do projeto
- » Entender as funções e variáveis constantes no código do projeto

Este projeto emprega um aplicativo Android simples (Figura 1-1) e um robô controlado por Arduino, que usa um módulo Bluetooth de baixo custo (Figura 1-2).

Os Arduinos são placas de microcontrolador muito populares que apresentam diversas vantagens, entre as quais se destacam:

- São fáceis de programar a partir de um computador Windows, Linux ou Mac.
- Muitos “shields” são facilmente encaixados em cima da placa do Arduino.
- Não são caros.



Figura 1-2 Um robô Bluetooth.

Todo o software do projeto pode ser acessado em www.duinodroid.com (clique em “Arduino and Android Projects for the Evil Genius”).

» Construção

A Figura 1-3 mostra o diagrama esquemático do projeto.

Os motores de acionamento do robô são controlados por um shield de motor, e o módulo Bluetooth é instalado na área de protótipos do shield, o que torna muito simples a realização do projeto, havendo poucas soldas a serem feitas.

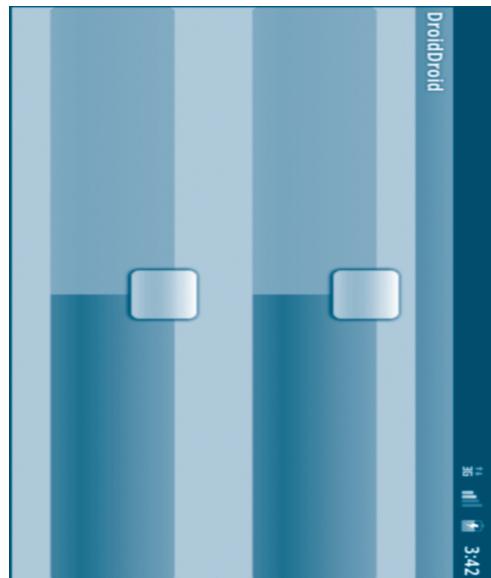


Figura 1-1 Um aplicativo de controle remoto.

» O que será necessário

Para realizar o projeto, além de um celular Android com Bluetooth (Android 2.1 ou posterior), você precisará dos componentes listados na tabela *Lista de Componentes*, a seguir.

Este projeto usa o Arduino Uno. O site oficial do Arduino (www.arduino.cc) oferece uma relação de fornecedores do Uno. Entretanto, se quiser economizar, você poderá usar um clone do Arduino Uno. O hardware do Arduino é “open-source”, isto é, todos os arquivos de projeto estão disponíveis sob uma licença Creative Commons, permitindo que outros fabricantes produzam os seus próprios Arduinos. Muitos o fazem, e uma busca na Internet fornecerá alternativas de menor custo para o “Uno” oficial.

No mercado, há muitos tipos diferentes de módulos Bluetooth. O módulo usado aqui é bem simples, com apenas quatro pinos para conexões de alimentação elétrica, recepção e transmissão. Esses módulos trabalham com 5V e são ideais para funcionar com um Arduino. Geralmente, são construídos em uma placa-base de quatro pinos sobre a qual é montada uma placa ainda menor, que contém o módulo Bluetooth propriamente dito. Podem ser comprados no Ebay por uns US\$ 15. É melhor comprar um no qual a placa menor já está soldada na placa principal porque as conexões são muito curtas e de soldagem difícil. Versões melho-

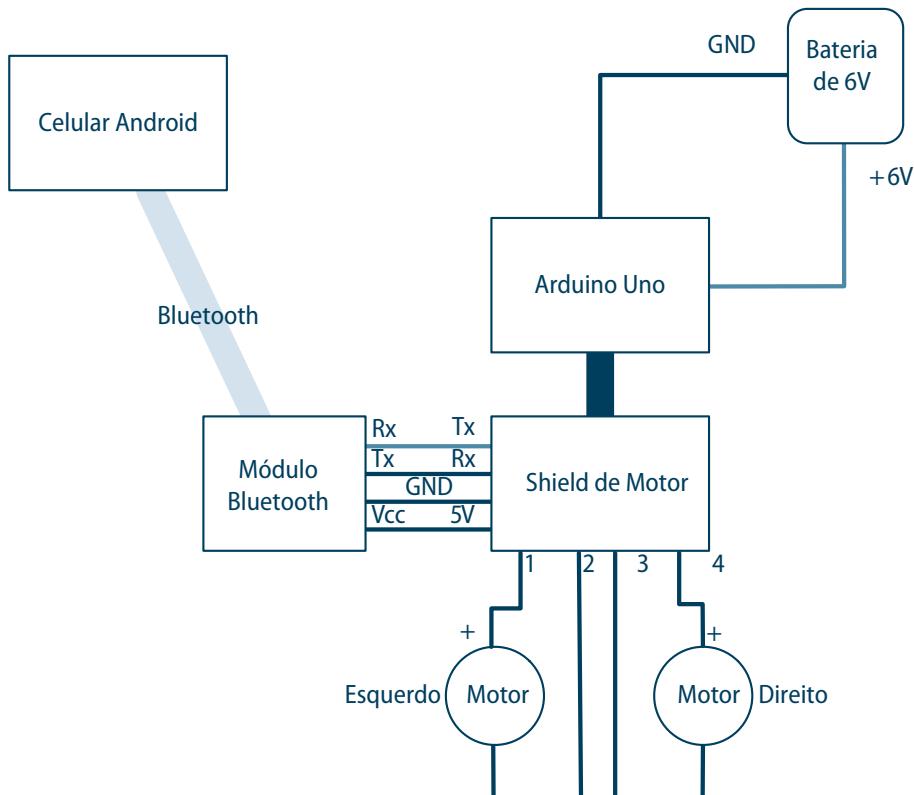


Figura 1-3 O diagrama esquemático.

res e mais caras estão disponíveis em fornecedores como Sparkfun, dentro da linha Bluesmirf. A diferença entre os módulos de baixo custo e os mais caros está na distância alcançada pela transmissão.

Os motores com engrenagens da empresa Pololu são ideais para esse tipo de aplicação. Apresentam uma caixa de redução com a relação correta e não são caros. Motores alternativos estão disponíveis, mas não compre motores que consomem mais do que 1 ampere. Se esse valor for excedido, é possível que o shield do motor não consiga fornecer a corrente necessária.

Com esse shield, todo o processo de acionamento dos motores fica bastante simplificado. Além disso, esse shield apresenta uma pequena área para protótipo em um dos lados, na qual você pode acrescentar os seus próprios componentes extras. No

nossa caso, é esse o local onde o módulo Bluetooth será instalado. Na lista de componentes, o kit de shield especificado é o básico, vindo sem as barras de pino macho nem os conectores KRE. Esse shield também está disponível como um kit que contém as barras de pino macho e os conectores KRE. Veja o site da Sparkfun para conhecer os detalhes.

Além desses componentes, você precisará também das seguintes ferramentas.

CAIXA DE FERRAMENTAS

- Uma furadeira elétrica com brocas
- Uma serra de arco ou uma ferramenta Dremel
- Uma pistola para cola a quente ou cola epóxi
- Um computador para programar o Arduino
- Um cabo de conexão USB do tipo A-B

LISTA DE COMPONENTES

Componente	Quantidade	Descrição	Fornecedor
Arduino Uno	1	Placa de Arduino Uno	www.arduino.cc
Ardumoto	1	Shield Ardumoto de motor	Sparkfun: DEV-09815
Módulo BT	1	Módulo Bluetooth TTL Bluesmif ou equivalente	eBay, Sparkfun
Barra de pino macho para PCB*	1	Barra dividida em duas seções de seis pinos e duas seções de oito pinos. Também conhecida simplesmente como alojamento ou como barra de pinos para placa de circuito impresso.	Farnell: 1097954
Conector KRE	3	Conector KRE de 2 vias com espaçamento de 3,5mm	Farnell: 1217302
Motores com engrenagens	2	Motor pequeno com engrenagens de plástico e redução 120:1	Pololu: 1125
Chave	1	Chave miniatura SPST (um polo, uma posição)	Farnell: 1661841
Suporte de pilha	1	Supporte para 4 pilhas AAA com terminais	Farnell: 1650687
Caixa	1	Caixa de plástico, 135 x 80 x 30mm	
Rodas	2	Rodas com aproximadamente 50 mm de diâmetro	Lojas de hobby
Rodízio	1	Rodízio pequeno	Lojas de ferragem

* N. de T.: Printed Circuit Board, ou seja, Placa de Circuito Impresso.

» Passo 1: soldie as barras de pino macho no Shield

O primeiro passo é soldar as barras de pino macho no shield de motor. A Figura 1-4 mostra a parte de baixo do shield com as barras de pino macho soldadas. Provavelmente, suas barras de pino macho virão em uma peça única comprida que deve ser cortada em seções de comprimento correto. Será necessário que você corte dois pedaços de seis pinos e dois de oito pinos.

A melhor forma de manter as barras de pino macho alinhadas corretamente é manter os pinos encaixados nas respectivas barras fêmeas na placa do seu Arduino enquanto você solda os pinos no shield. Entretanto, isso aquecerá o plástico das barras fêmeas, podendo amolecer-lo e tirar os pinos do alinhamento. Assim, você pode soldar rapidamente os pinos ou simplesmente soldar os pinos das extremidades de cada seção, de modo que a barra mantenha-se no local correto. A seguir, você

remove o shield da placa do Arduino e solda os demais pinos no shield.

Quando todos os pinos estiverem no lugar, a parte de cima do shield ficará como mostrado na Figura 1-5.

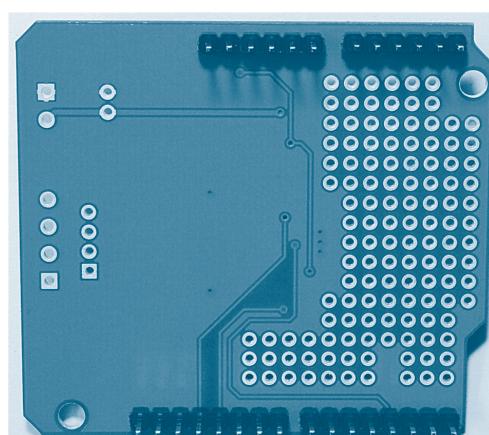


Figura 1-4 O shield de motor com as barras de pino macho soldadas.

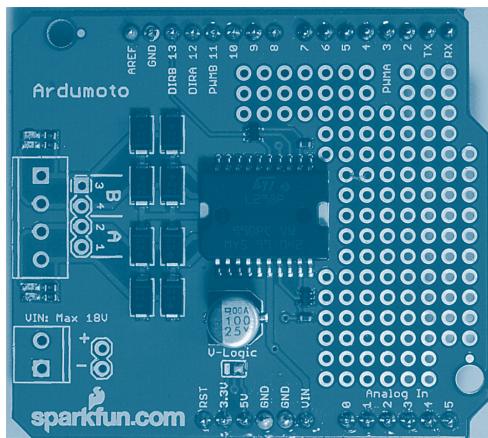


Figura 1-5 O lado de cima do shield de motor.

» Passo 2: instale os conectores KRE no Shield

Os conectores KRE são instalados perto dos canais A e B dos motores. Usaremos também um conector KRE para a alimentação elétrica porque será mais fácil fazer a conexão desse modo do que usando o jack principal de 2,1mm do Arduino.

Solde os quatro conectores KRE no lugar, com os orifícios de conexão voltados para fora do shield. A Figura 1-6 mostra o shield com os conectores KRE soldados e o shield encaixado em um Arduino.

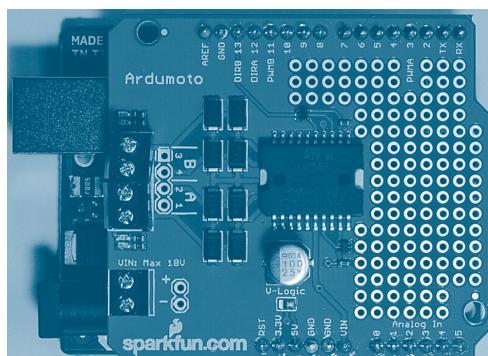


Figura 1-6 O shield com os conectores KRE soldados.

» Passo 3: instale o módulo Bluetooth

A Figura 1-7 mostra o módulo Bluetooth soldado juntamente com a fiação.

Antes de instalar o módulo Bluetooth, curve cuidadosamente os pinos com um alicate de modo que a placa fique paralela em relação ao shield. Comece soldando no lugar apenas o módulo. Em seguida, solda os quatro fios conforme a lista abaixo:

- +5V do módulo Bluetooth ao +5V do shield
- GND do módulo Bluetooth ao GND do shield
- TXD do módulo Bluetooth ao RX do shield
- RXD do módulo Bluetooth ao TX do shield

Observe o cruzamento dos fios de transmissão e recepção entre o Arduino e o módulo Bluetooth.

Essa foi a parte de eletrônica. Agora trataremos da construção do hardware do robô.

» Passo 4: instale os motores e o suporte de pilhas na caixa do robô

A Figura 1-8 mostra a posição dos motores. As caixas de engrenagens dos motores são coladas no interior da caixa do robô.

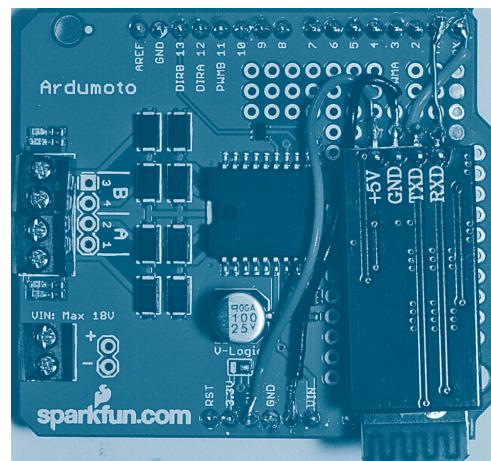


Figura 1-7 O shield completo.

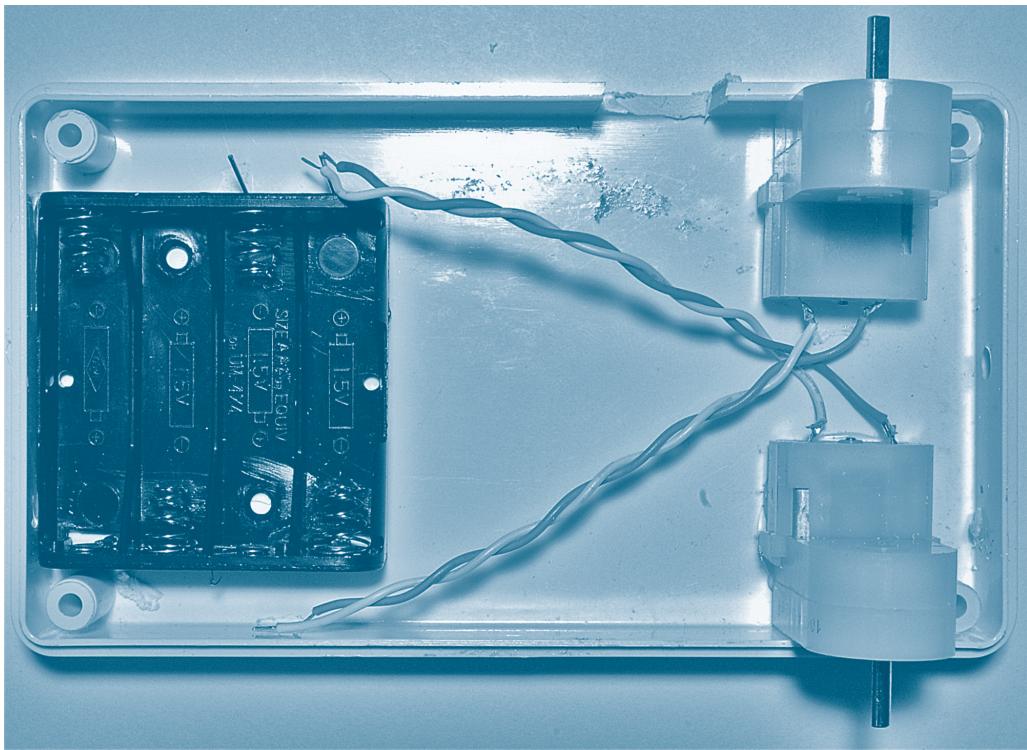


Figura 1-8 Os motores e o suporte de pilhas colados no interior da caixa.

Os motores e o suporte das pilhas são fixados na parte de baixo da caixa em ambas as extremidades, deixando espaço no meio para o Arduino e o shield.

» Passo 5: corte a parte de baixo da caixa e instale o rodízio

A Figura 1-9 mostra como a parte de baixo da caixa é recortada para permitir que os motores e suas caixas de engrenagens se sobressaiam da caixa.

Também é uma boa ideia fazer um furo próximo do módulo Bluetooth para ver se o LED do módulo está piscando ou não. A caixa que usamos foi aproveitada de um projeto anterior e tinha diversos furos. Isso não é ruim porque permite ventilação.

O rodízio é simplesmente o menor que conseguimos em uma loja de ferragem. Ele é colado diretamente na parte de baixo da caixa.

» Passo 6: fiação final

Usando conectores KRE, a fiação pode ser feita facilmente. A Figura 1-10 mostra o diagrama de fiação, e a Figura 1-11 mostra uma foto do interior do robô.

Os passos da fiação são:

1. Solde fios nos terminais dos motores. Esses fios devem ser suficientemente longos para serem conectados facilmente nos conectores KRE que foram instalados no shield de motor.
2. Solde um fio diretamente desde o terminal negativo (conector KRE) da alimentação elétrica do shield até o terminal negativo do suporte das pilhas.
3. Solde um fio mais longo desde o terminal positivo do suporte de pilhas até o terminal central da chave.

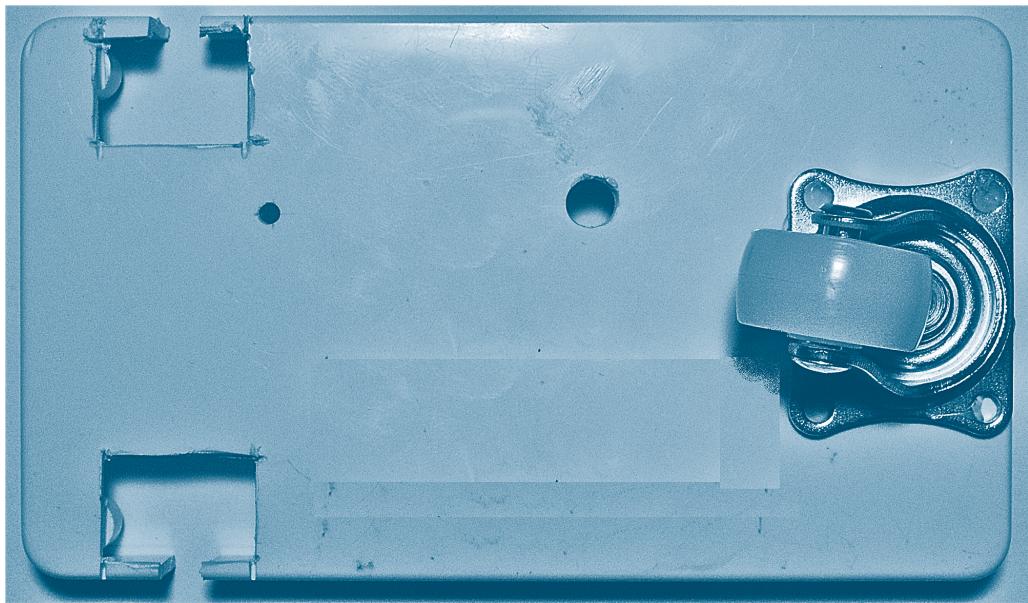


Figura 1-9 A parte de baixo da caixa.

4. Solde uma das extremidades de um fio mais curto em um dos lados da chave (não importa qual) e fixe a outra extremidade do fio no ter-

minal positivo (conector KRE) da alimentação elétrica do shield.

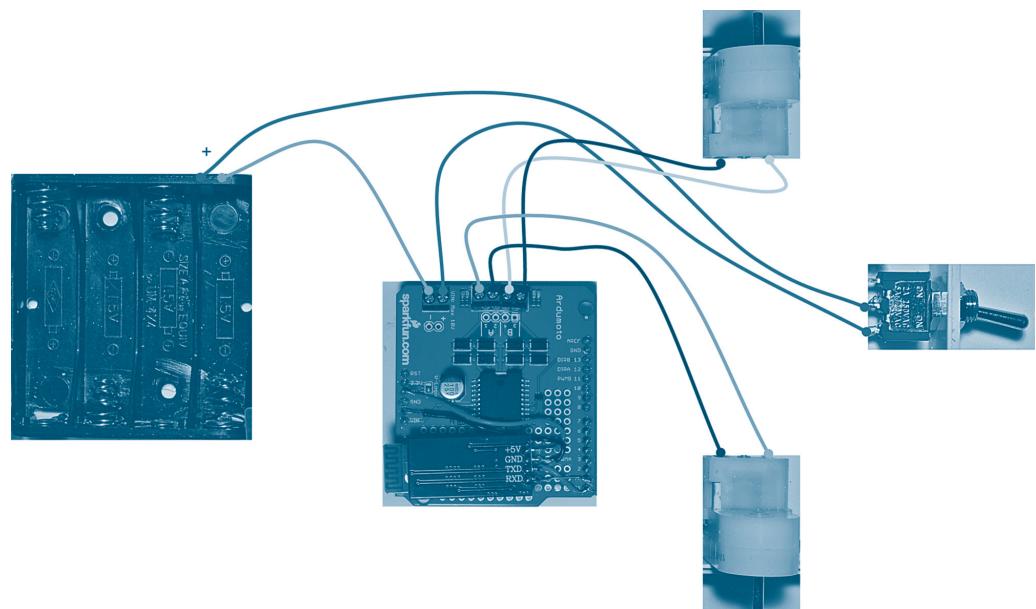


Figura 1-10 O diagrama de fiação.

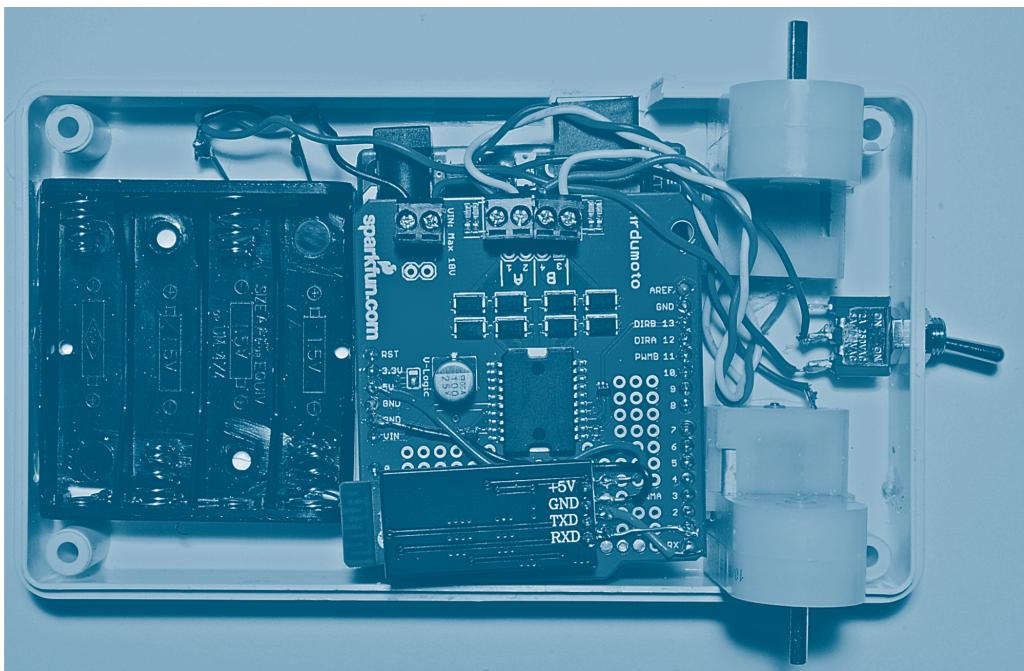


Figura 1-11 O interior do robô.

» Passo 7: teste dos motores

Antes de ir adiante e fazer a conexão com o módulo Bluetooth, precisamos primeiro preparar o nosso ambiente Arduino para instalar o programa de teste dos motores.

A placa de Arduino utilizada por nós (Arduino Uno) usa um ambiente de desenvolvimento especial que permite enviar os programas, ou “sketches”, como são conhecidos no mundo do Arduino, para a placa por meio de um cabo USB.

Precisamos fazer a instalação do ambiente Arduino. As instruções estão disponíveis no site oficial do Arduino (www.arduino.cc). Você deverá seguir-las para instalar o ambiente Arduino em seu computador. Nesse site, você encontrará instruções separadas para Windows, Linux e Mac. Neste livro, utilizamos uma placa de interface Arduino Uno e a versão 22 do software Arduino. Entretanto, se você quiser usar versões posteriores de Arduino, você não deverá encontrar problema.

Depois que o ambiente Arduino estiver preparado, você precisará instalar o sketch de teste do projeto. Esse e todos os demais sketches dos projetos deste livro estão disponíveis em um único arquivo zip que pode ser baixado de www.duinodroid.com (clique em “Arduino and Android Projects for the Evil Genius”).

Descompacte o arquivo zip e mova toda a pasta Arduino Android para a sua pasta de sketches. No Windows, a sua pasta de sketches estará em Meus Documentos/Arduino. No Mac, você a encontrará em Documents/Arduino/ e, no Linux, estará no diretório Sketchbook.

Depois de instalar a biblioteca, inicie novamente o software Arduino. A seguir, no menu File (arquivo), selecione sketches (ou SketchBook), seguido de Arduino Android, e então ch01_motor_test. Isso abrirá o sketch de teste dos motores, como mostrado na Figura 1-12.

Antes de realmente fazer funcionar os motores, talvez seja necessário alterar o valor de motorVolts

```

ch01_motor_test | Arduino 0022
ch01_motor_test

#define supplyVolts 6
#define motorVolts 5

int pwmLeftPin = 3;
int pwmRightPin = 11;
int directionLeftPin = 12;
int directionRightPin = 13;

void setup()
{
  pinMode(pwmLeftPin, OUTPUT);
  pinMode(pwmRightPin, OUTPUT);
  pinMode(directionLeftPin, OUTPUT);
  pinMode(directionRightPin, OUTPUT);
  setMotors(0, 0);
}

void loop()
{
  // forward
  setMotors(255, 255);
  delay(1000);
}

Done compiling.

Binary sketch size: 1634 bytes (of a 32256 byte maximum)

```

Figura 1-12 O sketch de teste dos motores.

localizado no início do sketch. Faça esse valor ser igual à tensão máxima de seus motores. Você precisará alterar esse valor se ele for diferente do valor da tensão dos motores da marca Pololu. Esses motores têm tensão nominal de 4,5V, mas trabalham bem também com 5V.

O módulo Bluetooth utiliza os pinos Rx e Tx do Arduino que também são usados pela interface. Assim, se o shield estiver conectado, nós não podemos programar o Arduino. Portanto, será necessário desinstalar o shield temporariamente.

Conekte a sua placa de Arduino ao computador usando o cabo USB. Precisamos dizer ao software Arduino qual é o tipo de placa que estamos usando. Para definir a placa usada, vá até o menu Tools (ferramentas) e selecione a opção Board (placa). Isso lhe dará uma lista semelhante à da Figura 1-13.

Selecione a opção correspondente ao tipo de placa que você está usando (Arduino Uno). A seguir, devemos fazer algo semelhante com a "Serial Port" (porta serial), que também faz parte do menu

Tools. Geralmente, a opção será a que está no topo da lista de portas (COM4, no Windows).

Agora, estamos prontos para transferir o sketch para a placa. Para isso, devemos clicar no ícone Upload (segundo a partir da direita na barra de ferramentas). Se aparecer uma mensagem de erro, confira o tipo de placa que você está usando e a conexão.

Depois de programar o Arduino com o sketch de teste dos motores, desconecte o cabo USB e reinstale o shield. Ligue a chave. Agora, os motores deverão passar pela sequência de teste.

- Ambos os motores para a frente
- Ambos os motores para trás
- Girar no sentido horário
- Girar no sentido anti-horário
- Pausa de cinco segundos

Se os motores não estiverem funcionando, verifique a fiação. Se um dos motores estiver indo para frente quando deveria estar indo para trás, inverta os fios nos conectores KRE desse motor.

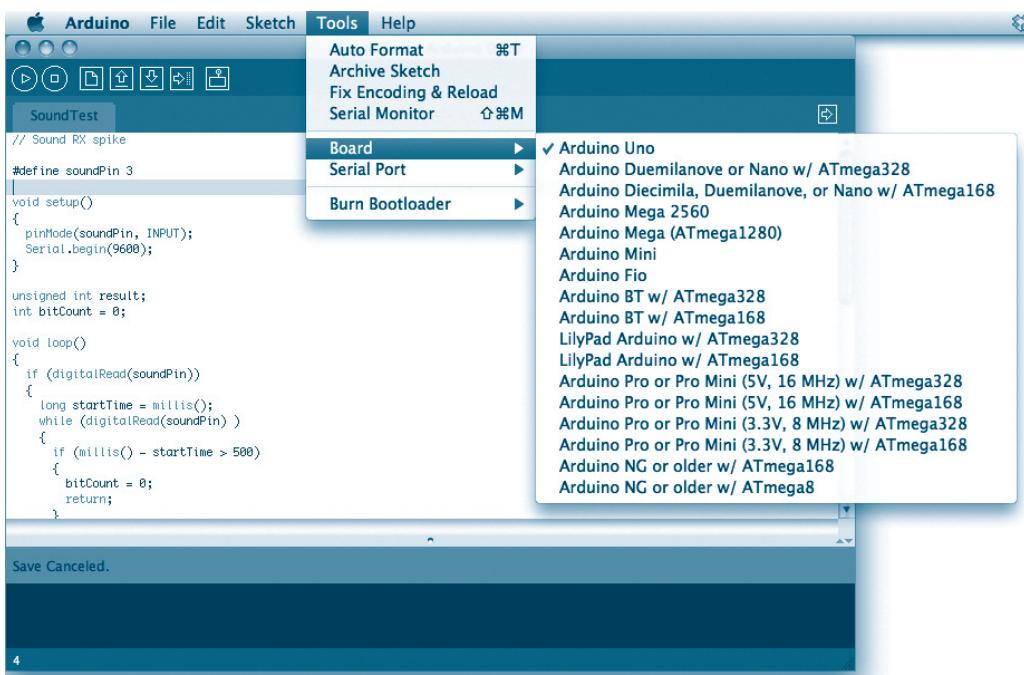


Figura 1-13 Selecionando o tipo de placa Arduino.

» Passo 8: instale o sketch de Arduino

Até agora, tudo está indo muito bem. A seguir, vamos para o próximo passo da instalação do sketch dos motores, os quais receberão comandos via Bluetooth.

O aplicativo Android usa uma tecnologia denominada **Amarino** (www.amarino-toolkit.net). Essa tecnologia aberta (open-source) simplifica grandemente o desenvolvimento de aplicações com Arduino e Bluetooth. Ela tem duas partes: uma biblioteca, que deve ser instalada no seu ambiente Arduino, e um aplicativo para o celular Android.

Para instalar a biblioteca, vá até a página de downloads no site Amarino (www.amarino-toolkit.net/index.php/download.html) e então clique em “MeetAndroid – Arduino Library”. Transfira o arquivo zip, descomponha-o e mova a pasta descompactada para a pasta libraries (bibliotecas) do Arduino. No Windows, a pasta libraries estará em Meus Documentos/Arduino. No Mac, você encontrará em Documents/Arduino/ e, no Linux, estará no diretório Sketchbook. Se a pasta libraries não estiver presente no seu Arduino, então você deverá criá-la*. Depois, reinicie o software Arduino.

Desligue a chave do robô e retire a placa do Arduino. A seguir, abra o sketch ch01_droid_droid no seu software Arduino.

Antes de transferi-lo para a placa, algumas alterações poderão ser necessárias. Primeiro, se você está usando motores diferentes, altere o valor de motorVolts.

Segundo, veja a documentação do seu módulo Bluetooth e verifique qual é a velocidade de comunicação com o Arduino. Frequentemente é 9600, mas pode ser mais elevada em alguns módulos.

* N. de T.: A pasta deve ser criada com o nome em inglês: libraries.

Finalmente, você pode transferir o sketch para a placa do mesmo modo que você fez com o sketch de teste. Se houver erros de compilação, provavelmente será porque a pasta com a biblioteca Amarino não está no lugar certo.

Desconecte o cabo USB da placa Arduino e reinstale o shield. Agora, chegamos ao ponto emocionante!

» Passo 9: instale o aplicativo Android

Diferentemente do iPhone, você pode baixar os seus aplicativos Android de qualquer lugar que desejar. Entretanto, isso significa que você deve ser cuidadoso e verificar antes se não irá baixar algo perigoso de um dado site. Após, você deverá programar o seu dispositivo Android para receber aplicativos de qualquer lugar.

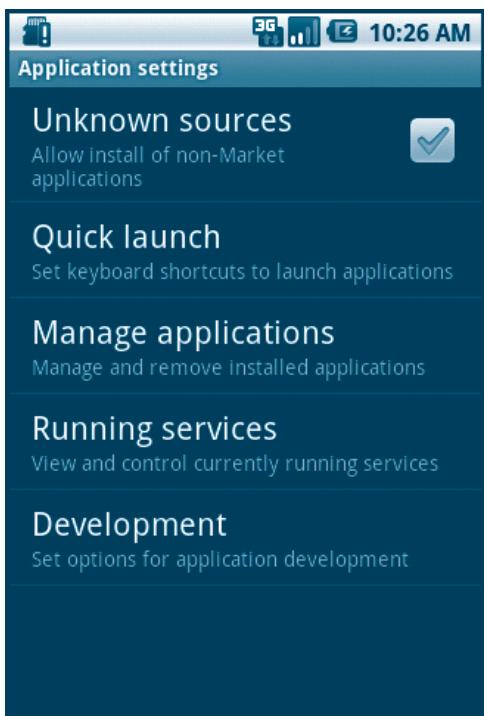


Figura 1-14 Alterando as configurações Android para permitir a instalação.

No celular utilizado aqui, a sequência de passos é: abrir “Configurações” (“Config.”), navegar até “Geriador de Aplicações” e ativar a opção “Fontes Desconhecidas”, como mostrado na Figura 1-14. No seu celular, a sequência para chegar até “Fontes Desconhecidas” poderá ser ligeiramente diferente. O manual do celular pode ser consultado.

Para usar o aplicativo do robô, devemos primeiro instalar o aplicativo de uso geral Amarino, que permitirá manipular os nossos dispositivos Bluetooth. Ele pode ser baixado do site Amarino. Para isso, use o navegador de Web do seu dispositivo Android e vá até [www.amarino-toolkit.net/index.php/download.html](http://amarino-toolkit.net/index.php/download.html). Então, clique em “Amarino – Android Application.”

Para instalar o aplicativo de controle do robô, abra o navegador de Web do seu dispositivo Android e vá até www.duinodroid.com. Clique em “Download” e então no aplicativo DroidDroid.

» Passo 10: ponha em funcionamento!

Antes de executar o aplicativo DroidDroid, precisamos usar o aplicativo Amarino (Figura 1-15).

Ligue o robô. Você deverá ver o LED do módulo Bluetooth piscando. Isso indica que o módulo ainda não está conectado a outro dispositivo. O aplicativo Amarino permitirá que nós o conectemos ao seu celular.

No menu principal mostrado na Figura 1-15, clique no botão “Acrecente dispositivo BT” (Add BT Device). Você verá uma lista de dispositivos Bluetooth (BT) que estão ao alcance.

Selecione o seu dispositivo nessa lista. Isso fará que você volte à página anterior na qual agora o seu dispositivo aparecerá incluído na lista de dispositivos. Ao clicar em “Conecte” (Connect), terá início o processo de conexão. Você deverá fornecer uma senha para o módulo Bluetooth. Provavelmente, será “1234”, mas também poderá ser “1111.” Consulte a documentação do módulo Bluetooth. Quando a conexão estiver estabelecida, a luz de pisca-pisca deverá se tornar constante. Anote o identificador ID

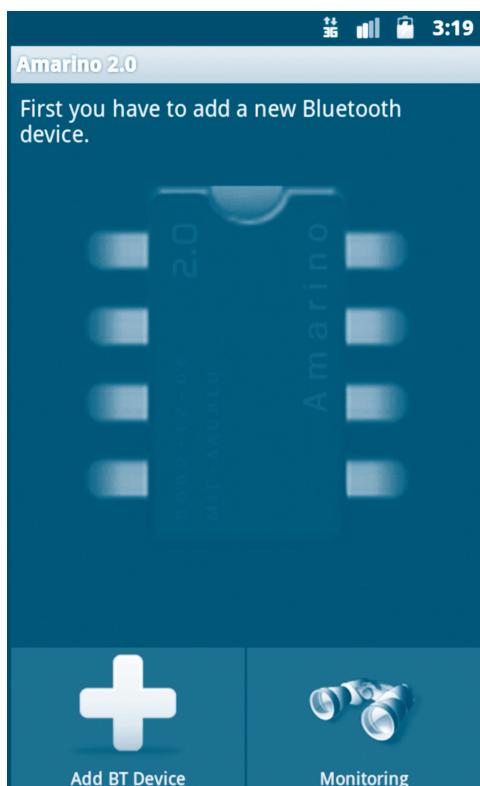


Figura 1-15 O Aplicativo Amarino.

de Bluetooth do dispositivo. É um número de seis partes com dois pontos separando cada par de dígitos. Esse ID será utilizado logo a seguir.

Inicie o aplicativo DroidDroid (Figura 1-16). A seguir, forneça o ID de Bluetooth do dispositivo que você anotou há pouco quando estava no aplicativo Amarino. Quando você clicar em “Set Device ID”, serão exibidos os controles principais (ver Figura 1-1).

Ao mover os controles para cima e para baixo, os motores esquerdo e direito do seu robô serão acionados.

» Teoria

O software deste e dos demais projetos do livro são abertos (open source), então você pode melhorá-los se desejar. Gostaríamos muito de ter notícias

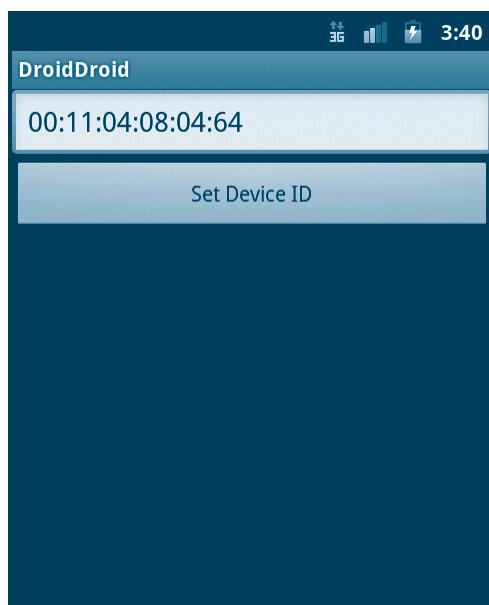


Figura 1-16 Ajustando o ID do dispositivo.

de qualquer aperfeiçoamento que você venha a fazer no software. Você pode nos contatar em www.duinodroid.com.

A seguinte descrição supõe que você já conhece a linguagem C utilizada na programação com o Arduino. Se você quiser aprender mais sobre a programação do Arduino, deve consultar o livro *Programação com Arduino: começando com sketches*, de Simon Monk, publicado pela Bookman Editora.

A listagem do sketch é a seguinte:

```
#include <MeetAndroid.h>
#define supplyVolts 6
#define motorVolts 5
#define baudRate 9600
MeetAndroid phone;
int left = 255; // ponto médio
int right = 255;
int pwmLeftPin = 3;
int pwmRightPin = 11;
int directionLeftPin = 12;
int directionRightPin = 13;
void setup()
{
    pinMode(pwmLeftPin, OUTPUT);
```

```

pinMode(pwmRightPin, OUTPUT);
pinMode(directionLeftPin, OUTPUT);
pinMode(directionRightPin, OUTPUT);
setMotors();

// ajuste a taxa de bauds usada para
// configurar o seu módulo Bluetooth
Serial.begin(baudRate);
phone.registerFunction(setLeft, 'l');
phone.registerFunction(setRight, 'r');
}
void loop()
{
    phone.receive();
}
void setLeft(byte ignore, byte count)
{
    int value = phone.getInt();
    left = value;
    setMotors();
}
void setRight(byte ignore, byte count)
{
    int value = phone.getInt();
    right = value;
    setMotors();
}
void setMotors()
{
    int vLeft = abs(left - 255) *
        motorVolts / supplyVolts;
    int vRight = abs(right - 255) *
        motorVolts / supplyVolts;
    int dLeft = (left > 255);
    int dRight = (right > 255);
    if (vLeft < 50)
    {
        vLeft = 0;
    }
    if (vRight < 50)
    {
        vRight = 0;
    }
    analogWrite(pwmLeftPin, vLeft);
    analogWrite(pwmRightPin, vRight);
    digitalWrite(directionLeftPin,
        dLeft);
    digitalWrite(directionRightPin,
        dRight);
}

```

O sketch começa com três constantes. Os valores de `supplyVolts` (tensão de alimentação) e `motorVolts` (tensão dos motores) são usados para ajustar a tensão fornecida aos motores. Assim, se você adaptar o projeto para funcionar com motores e pilhas diferentes, você precisará alterar esses valores.

A variável `baudRate` (taxa de bauds) deve ser a usada pelo módulo Bluetooth na comunicação com o Arduino.

A interface com o celular está inteiramente contida na biblioteca `MeetAndroid`. Para ter acesso, você deve criar uma instância dela – nesse caso, denominada “`phone`”.

As variáveis “`left`” (esquerdo) e “`right`” (direito) são usadas para ajustar a velocidade de cada motor. Elas têm um valor médio (central) de 255, significando que, em 255, o motor está parado, em 511, está à plena velocidade para frente e, em 0, está à plena velocidade para trás.

As quatro variáveis seguintes definem os pinos usados pelos motores. Os pinos são definidos pelo shield do motor e, portanto, não podem ser mudados. Cada motor é controlado por dois pinos. O pino “`pwm`” (power motor) controla a velocidade do motor: 0 é parado, 255 é velocidade máxima. O pino “`direction`” (direção)* altera o sentido de rotação do motor, 1 sendo para frente e 0 para trás.

A função “`setup`” (inicialização) define quais são os modos apropriados de cada pino (`pinMode`) e dá início (`begin`) ao funcionamento da porta serial (`Serial.begin`). Define também duas funções de comando dos motores – `setLeft` e `setRight` – que serão chamadas sempre que o sketch receber uma nova velocidade aos motores através das letras de comando “`l`” (left) ou “`r`” (right) dos motores esquerdo e direito, respectivamente.

* N. de T.: Os termos `pwm` e `direction` estão combinados com outros para formar os nomes das variáveis. Por exemplo, `pwmLeftPin` refere-se ao pino que aciona o motor esquerdo, e `directionRightPin` refere-se ao pino que controla o sentido de rotação do motor direito.

Tudo o que realmente precisamos na função “loop” (laço de repetição contínua) é chamar a função “receive” (receber) da biblioteca MeetAndroid. Ela verifica se há alguma mensagem chegando e chama as funções adequadas de comando dos motores.

Essas duas funções de comando são responsáveis pela atribuição de novos valores às variáveis “left” e “right”. Os parâmetros passados às funções de comando dos motores podem ser ignorados. Para obter os valores enviados pelo celular, as funções de comando dos motores usam a função “getInt”.

A função “setMotors” calcula os valores adequados das saídas analógicas ajustando-os para levar em consideração a diferença entre a tensão de alimentação e a tensão do motor. Também calcula o sentido de rotação de cada motor e define os valores adequados de saída.

```
private void updateLeft() {  
    Amarino.sendDataToArduino(this,  
        DroidDroid.DEVICE_ADDRESS, 'l',  
        (511 - left));  
}
```

A interface é muito simples. Você simplesmente chama o método sendDataToArduino (enviar dados para o Arduino). O primeiro argumento é a instância Android Activity (pense “tela”), e o segundo argumento é o Bluetooth ID (identificação do Bluetooth). O próximo parâmetro é um caractere que pode ser “l” (left) ou “r” (right). Esses caracteres são enviados ao Arduino para chamar a respectiva função que comanda cada um dos motores.

O argumento final é o valor inteiro enviado ao Arduino, que pode estar entre 0 e 511.

» O aplicativo Android

O aplicativo Android é a parte mais complexa do projeto. Só para aprender a programação Android seria necessário um livro inteiro. Muitos desses livros estão disponíveis e podem ser consultados. Contudo, podemos ao menos conhecer o trecho do código que envia o valor ao Arduino.

» Resumo

Essa é a primeira de uma série de coisas divertidas que você poderá fazer com o seu celular. No próximo capítulo, usaremos a nova tecnologia ADK do Google para criar um acessório contador Geiger para o nosso celular Android.



» capítulo 2

Contador Geiger com Android

Neste capítulo, mostraremos como construir um acessório contador Geiger para ser acoplado a um telefone celular Android.

Objetivos

- » Entender o que é o Open Accessory do Google
- » Usar a tecnologia Open Accessory para criar um contador Geiger para celular Android
- » Entender o funcionamento elétrico de um tubo Geiger-Müller
- » Entender as variáveis e funções constantes no sketch do Arduino para esse projeto



Figura 2-1 O contador Geiger Android.

O Google, desenvolvedor do extremamente bem-sucedido sistema operacional Android para dispositivos móveis, escolheu o Arduino como base para o seu kit de desenvolvimento denominado Open Accessory Development Kit, ou simplesmente ADK. Trata-se de uma especificação de protocolo e um conjunto de bibliotecas de software desenvolvidos pelo Google com o propósito de estimular usuários (isto é, nós) a desenvolver acessórios de hardware que podem ser conectados a um dispositivo Android através de uma conexão USB.

O Google provavelmente esperava que os acessórios desenvolvidos fossem caixas de som e outros acessórios domésticos sem graça para a sala de estar. No entanto, aproveitamos a oportunidade para desenvolver alguns acessórios bem mais emocionantes para o seu telefone celular, como este contador Geiger.

O contador Geiger usa um tubo Geiger-Müller (GM) de baixo custo, obtido no eBay por cerca de US\$ 20. O tubo não capta radiação alfa. Os tubos

capazes disso são mais caros e difíceis de obter, mas funcionariam bem com este projeto.

Todo o projeto deve custar menos de US\$ 100, incluindo a Arduino e o shield USB host.

ALERTA Este projeto gera cerca de 400V para o tubo GM. Esta tensão permanecerá armazenada nos capacitores mesmo depois de desligar o dispositivo. Isso poderá feri-lo, de modo que você deve tomar muito cuidado com a construção deste projeto. Além disso, se você não fizer a fiação de forma correta, uma alta tensão poderá passar ao telefone celular e danificá-lo. Não assumimos responsabilidade por qualquer dano que possa ocorrer ao seu celular quando você usar o acessório deste projeto. Não se exponha à radiação. Contente-se em medir a radiação de fundo.

» *O Open Accessory do Google*

O Open Accessory Development Kit (ADK) do Google baseia-se na tecnologia Arduino. Você pode comprar uma placa especial de desenvolvimento baseada no Arduino, com LEDs e diversos outros componentes de hardware já soldados, mas você pode conseguir mais flexibilidade usando uma placa Arduino padrão e um shield USB host.

O suporte proporcionado pelo Open Accessory está disponível apenas com celulares que utilizam o Android 2.3.4 ou posterior. Portanto, antes de encomendar componentes, verifique se o seu telefone suporta o modo de acessórios (Accessory Mode) e se tem Android 2.3.4 ou posterior.

A Figura 2-2 mostra o funcionamento do Android e do Arduino quando se usa o ADK para Android.

O celular Android funciona como um USB client (cliente). Isso significa que é o Arduino que está no comando da situação (mestre) atuando como USB host (hospedeiro). Ele deve iniciar a conexão com o celular Android. Quando faz isso, ele também pode colocar automaticamente o telefone no modo de acessório e iniciar um aplicativo específico. Neste

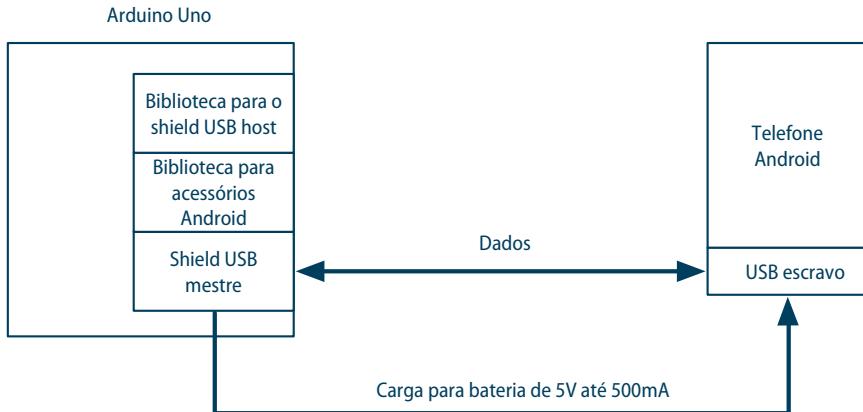


Figura 2-2 Arduino e Android, finalmente juntos.

caso, ele abrirá o aplicativo DroidGeiger que desenvolvemos para este projeto.

É necessário também que o Arduino forneça energia para carregar a bateria do celular. Assim, qualquer acessório que construirmos usando o nosso Arduino deve conseguir fornecer uma tensão de 5V e uma corrente de até 500mA para carregar o celular através do cabo USB. Isso significa que o acessório deve estar conectado a uma fonte de alimentação ou, como no caso deste projeto, a pilhas ou baterias de boa qualidade. Mais adiante, neste capítulo, veremos também como podemos contornar esse problema usando um cabo de conexão que não fornece energia para carregar a bateria.

» Construção

Como o robô do capítulo anterior, este projeto usa um shield que é instalado em cima do Arduino Uno. Nesse caso, usamos um shield USB host (hospedeiro). Você poderá pensar que isso não é necessário porque o Arduino já tem um conector USB. Infelizmente, a conexão USB do Arduino é do tipo cliente (client) e, para fazer uma conexão com um celular Android, precisamos de uma conexão USB do tipo hospedeiro (host). Por outro lado, o shield USB host tem uma área para protótipo na qual podemos soldar os demais componentes necessários ao projeto.

O diagrama esquemático deste projeto está na Figura 2-3.

O principal objetivo do circuito é gerar a tensão de 400V requerida pelo tubo GM. Você encontrará uma descrição mais detalhada do funcionamento desse circuito na seção *Teoria*, no final deste capítulo.

» O que será necessário

Além do telefone celular Android capaz de aceitar acessórios (Android 2.3.4 ou posterior), você precisará dos componentes listados na tabela *Lista de Componentes*, a seguir, para construir o módulo de conexão que envia as contagens e ativa o som dos cliques.

Para você construir um cabo USB que não faz a carga da bateria, você precisará de dois resistores de $1k\Omega$ em vez de apenas um. Você também precisará de um cabo de extensão USB fora de uso.

Como alternativa, se quiser uma duração maior de carga, use um suporte de pilhas que aceite seis pilhas do tipo AA para alimentar o projeto com 9V. Essa alternativa aparece em algumas das figuras.

O componente-chave deste projeto é o tubo GM. Esses tubos podem ser facilmente encontrados no eBay a partir de fornecedores internacionais, frequentemente de países da antiga União Soviética. O tubo que utilizamos aqui está descrito no eBay como "Russian Military GEIGER TUBE COUNTER CI-1".

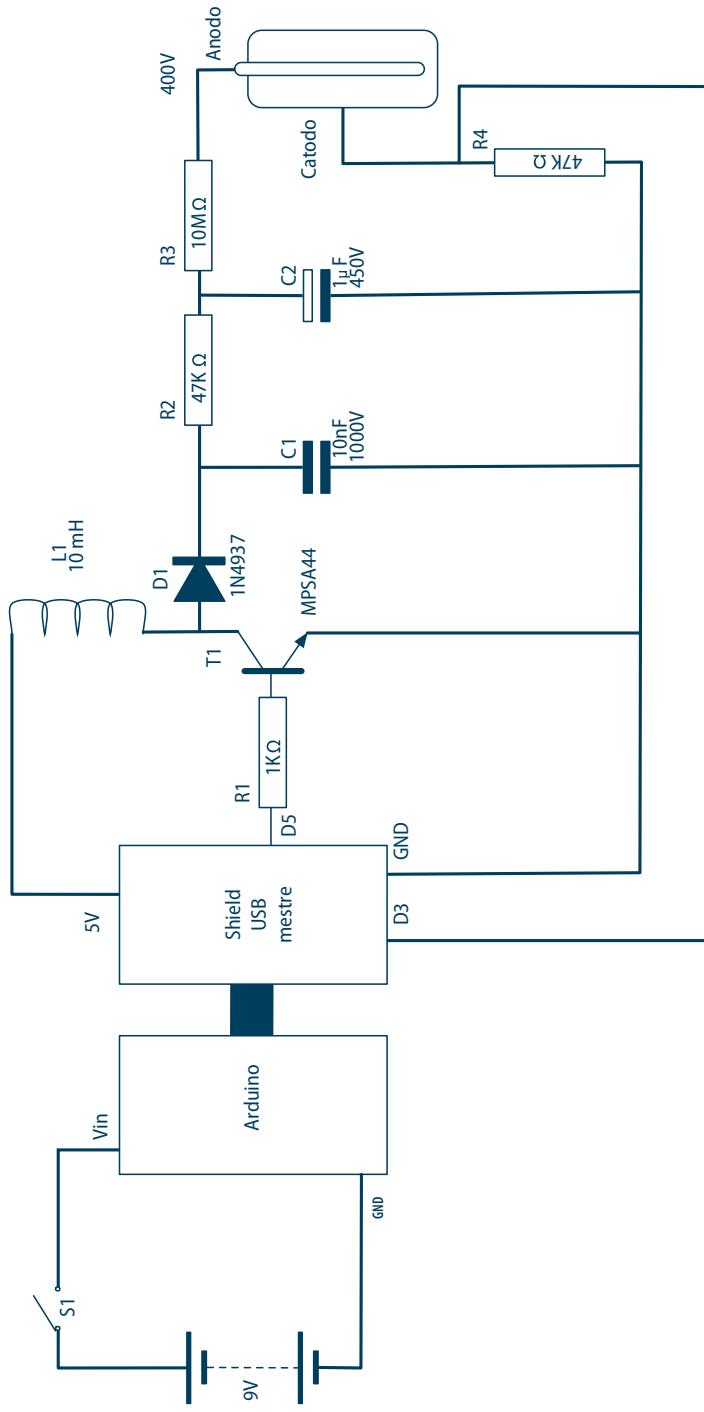


Figura 2-3 O diagrama esquemático.

O tubo tem as especificações que estão a seguir, mas outros tubos com características similares deverão funcionar igualmente bem. O nosso contador Geiger não se destina a situações críticas. Como ele não virá calibrado, você não deverá levar em conta a sua exatidão.

- Tipo: detector de radiação gama
- Tensão do anodo: 360–440V
- Comprimento do patamar: 80V
- Inclinação: 0,125%/V
- Resistência de carga: 10MΩ
- Comprimento: 90mm
- Diâmetro: 12mm

Usando um sketch de Arduino, veremos mais adiante como ajustar a geração da tensão de anodo para um valor entre 0 e 450V, que é a tensão máxima suportada por nossos capacitores.

Este projeto usa o Arduino Uno. O site oficial do Arduino (www.arduino.cc) lista fornecedores do Uno. Entretanto, se quiser economizar, você poderá usar

um clone do Arduino Uno. O hardware do Arduino é aberto (open-source). Isso significa que todos os arquivos de projeto estão disponíveis sob uma licença Creative Commons, permitindo que outros fabricantes produzam os seus próprios Arduinos. Muitos o fazem, e uma busca na Internet fornecerá alternativas de baixo custo para o Uno oficial.

Além desses componentes, você precisará também das seguintes ferramentas.

CAIXA DE FERRAMENTAS

- Uma furadeira elétrica com brocas
- Uma serra de arco ou uma ferramenta Dremel
- Uma pistola para cola a quente ou cola epóxi
- Parafusos autoatarraxantes de diversos tamanhos
- Um computador para programar o Arduino
- Um cabo de conexão USB do tipo A-B
- Um multímetro com uma escala de 1000V

LISTA DE COMPONENTES

Componente	Quantidade	Descrição	Fornecedor
Arduino Uno	1	Placa de Arduino Uno	www.arduino.cc
Shield USB	1	Shield USB host (hospedeiro) para Arduino	Sparkfun: DEV-09947
Barra de pino macho	1	Barra de pino macho dividida em duas seções de seis pinos e duas seções de oito pinos.	Farnell: 1097954
Chave	1	Chave miniatura SPST (um polo, uma posição)	Farnell: 1661841
Pilha recarregável	1	Pilha recarregável de NiMH e 9V do tipo PP3	
Clip para bateria	1	Clipe para bateria do tipo PP3	Farnell: 1183124
Caixa	1	Caixa de plástico	
R1 mais 1	2	Resistor de filme metálico de 1kΩ e 1/2 W	Farnell: 9339779
R2, R4	2	Resistor de filme metálico de 47kΩ e 1/2 W	Farnell: 9340637
R3	1	Resistor de filme metálico de 10MΩ e 1/2 W	Farnell: 1779379
C1	1	Capacitor cerâmico de 10nF e 1000V	Farnell: 1615007
C2	1	Capacitor eletrolítico de 1μF e 450V	Farnell: 1822752
D1	1	1N4937	Farnell: 9843663
T1	1	MPSA44	Farnell: 1574391
L1	1	Indutor de 10mH	Farnell: 1710435
Tubo	1	Tubo GM (veja detalhes no texto)	eBay
Clipes	2	Clipe (garra) de fusível para circuito impresso	Farnell: 1866097
Plugue	1	Plugue P4 de alimentação elétrica de 2,1mm	Farnell: 1200147

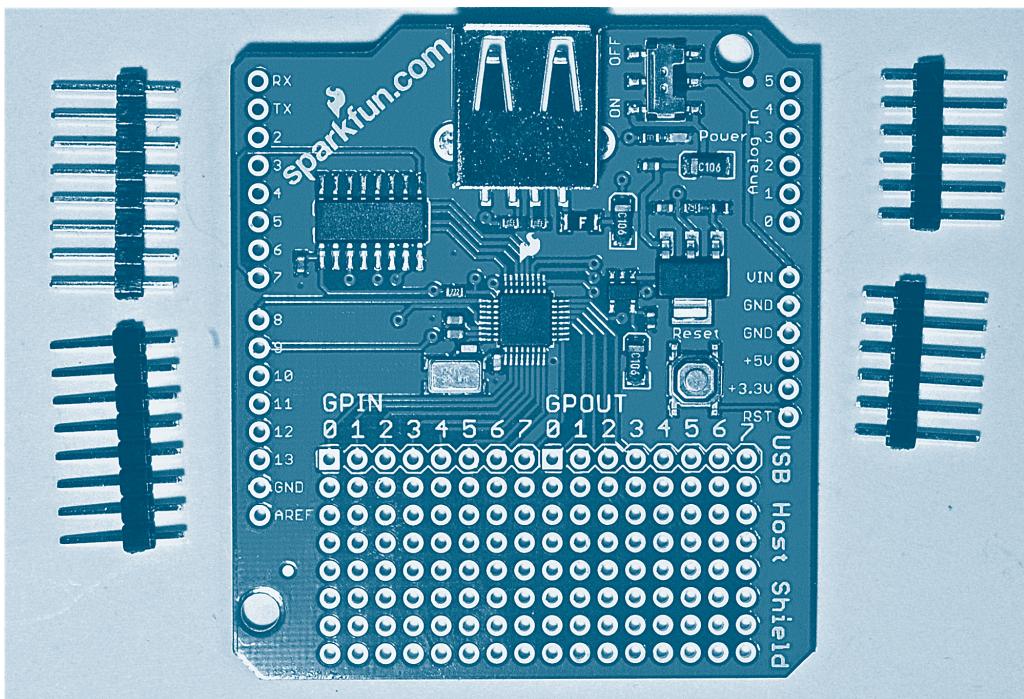


Figura 2-4 O shield USB com os pinos.

» Passo 1: solde as barras de pino macho no Shield

O primeiro passo é soldar as barras de pino macho no shield USB host. A Figura 2-4 mostra o shield com as barras de pino macho. Provavelmente, as barras de pino macho virão em uma peça única comprida que deve ser cortada em seções de comprimento correto. Será necessário que você corte dois pedaços de seis pinos e dois de oito pinos.

A melhor forma de manter os pinos do shield alinhados corretamente é mantê-los encaixados nos respectivos soquetes (barra fêmea) da placa do seu Arduino enquanto você solda os pinos no shield. Entretanto, isso poderá aquecer demais o plástico dos soquetes no Arduino, podendo amolecê-lo e desalinhá-lo. Para contornar isso, você poderá soldar rapidamente os pinos no shield ou então simplesmente soldar os pinos das extremidades de cada seção, de modo que a barra de pino

macho mantenha-se no local correto. A seguir, você remove o shield da placa de Arduino e solda os demais pinos.

Quando todos os pinos estiverem no lugar, a parte de cima do shield ficará como mostrado na Figura 2-5.

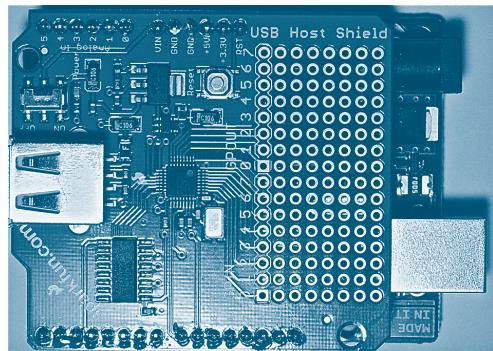


Figura 2-5 A parte de cima do shield USB montado em um Arduino.

» Passo 2: solde os componentes baixos

A Figura 2-6 mostra a disposição dos componentes na área de protótipo do shield.

Sempre é mais fácil soldar primeiro os componentes de perfil baixo. Assim, começaremos soldando os resistores e o diodo nos seus lugares. Não corte o excesso que sobra dos terminais debaixo da placa, porque iremos usá-los mais adiante para conectar os componentes.

O diodo deve ser encaixado com a orientação correta. No caso, a faixa pintada do diodo deve estar voltada para o lado do soquete USB do shield*.

A Figura 2-7 mostra os resistores e o diodo no lugar.

» Passo 3: solde os demais componentes

Agora, soldaremos os demais componentes (Figura 2-8).

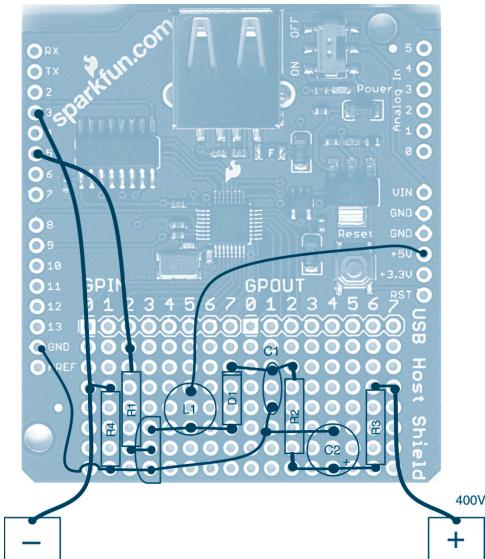


Figura 2-6 A disposição dos componentes na área de protótipo.

* N. de T.: Não confundir com o soquete USB da placa do Arduino.

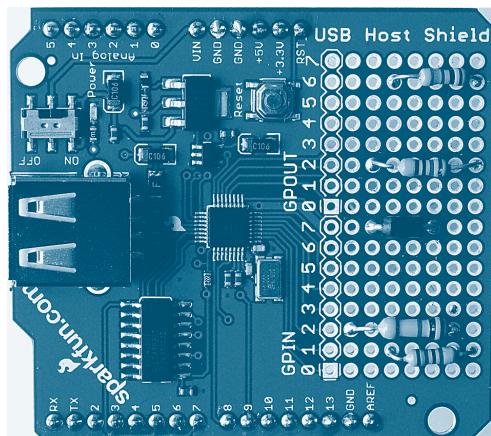


Figura 2-7 Resistores e diodo no shield.

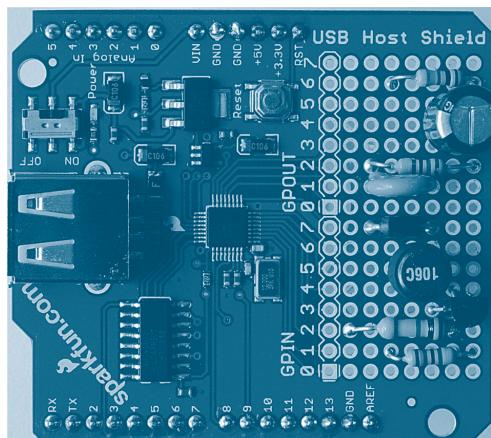


Figura 2-8 Os demais componentes no shield.

Tome cuidado para encaixar o transistor de forma correta. O capacitor eletrolítico C2 também deve ser inserido corretamente. O terminal positivo mais comprido deve ser encaixado no orifício mais próximo da borda da placa. O indutor pode ser colocado de qualquer modo.

Quando todos os componentes estiverem em seus lugares, a parte debaixo da placa ficará como mostrado na Figura 2-9. O próximo passo é curvar os terminais dos componentes para conectar os componentes entre si, como mostrado na Figura 2-6.

Agora, a parte debaixo da placa apresenta-se como mostrado na Figura 2-10.

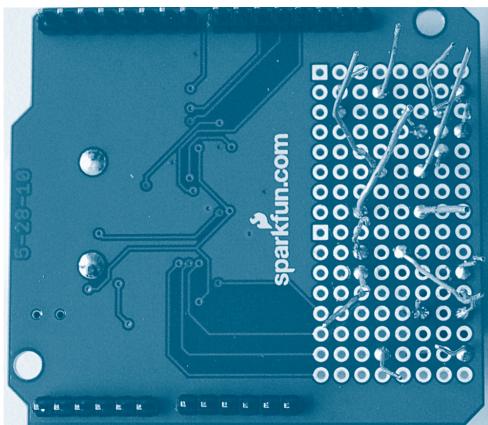


Figura 2-9 A parte debaixo da placa com todos os componentes nos seus lugares.

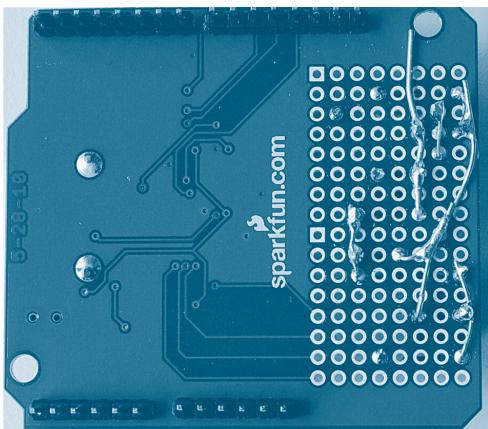


Figura 2-10 Os terminais de componente usados para fazer as conexões.

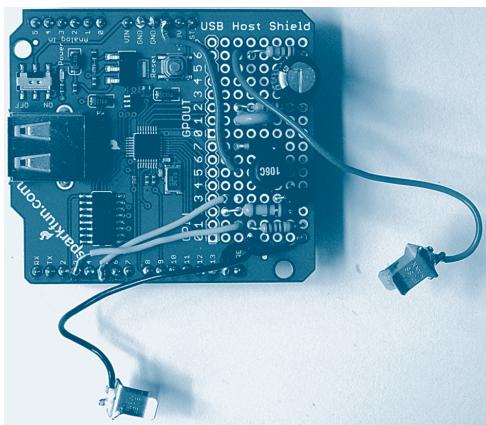


Figura 2-11 Clips para o tubo GM usando porta-fusível para CI.

quais devem ser soldados. Novamente, use a Figura 2-6 como guia para ver como são feitas as conexões dos fios.

Não conecte ainda o tubo GM. Primeiro, devemos fazer alguns testes.

» Passo 5: a fiação final

O projeto é energizado por uma bateria de 9V do tipo PP3 ou por seis pilhas do tipo AA alojadas em um porta-pilhas. Como este projeto consome energia do Arduino, faz sentido usar pilhas recarregáveis. De qualquer forma, se for usada uma bateria do tipo PP3, então deveremos usar um clip PP3 e colocar a chave em série com o fio positivo, como mostrado na Figura 2-12.

» Passo 4: solde os fios aos pinos do Arduino

As conexões finais que precisamos soldar envolvem alguns fios entre a área de protótipo e os pinos apropriados no Arduino. Após, deveremos soldar os fios que se conectam aos clips de fusível. O tubo GM será encaixado nesses clips. A Figura 2-11 mostra essas conexões. Na área de protótipo, os fios podem passar através dos furos que estão próximos dos terminais de cada componente, aos

» Passo 6: instale as bibliotecas Open Accessory

Se você ainda não o fez, instale o software do Arduino no seu computador. Você poderá encontrar instruções completas para isso no site oficial do Arduino em www.arduino.cc.

O Open Accessory do Google requer que duas bibliotecas (libraries) sejam instaladas no seu ambiente Arduino. A primeira é uma versão da biblio-

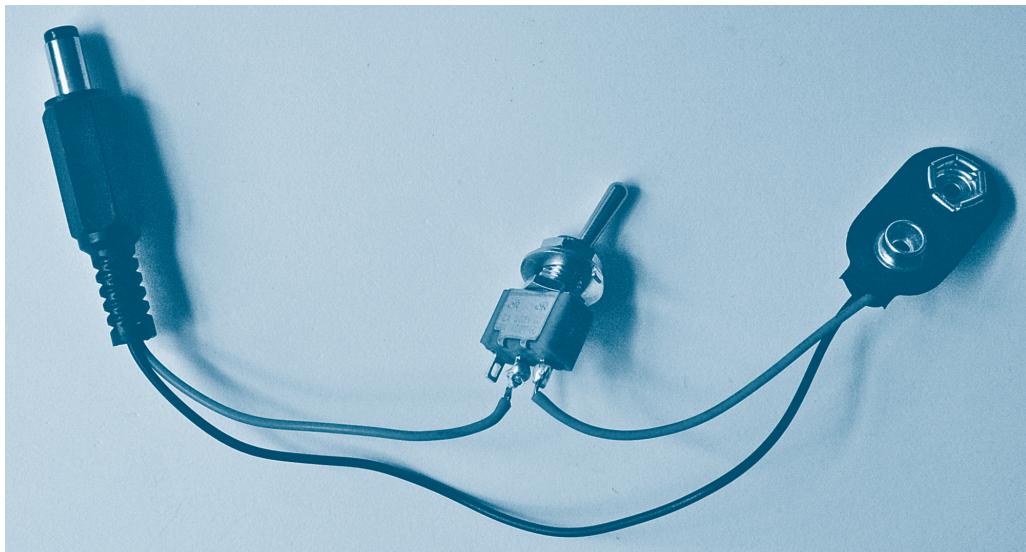


Figura 2-12 O pino de conexão elétrica e o clip para a bateria.

teca USB host (hospedeiro), que está preparada para trabalhar com o hardware padrão do Arduino. Ela pode ser obtida no site microbridge.googlecode.com/files/usb_host_patched.zip.

Se, por qualquer razão, você não conseguir baixar algum arquivo do software usado pelos projetos deste livro, acesse o site do livro (www.duinodroid.com), no qual você encontrará instruções para baixar o software de algum outro lugar.

Para instalar a biblioteca, baixe o arquivo zip, descompacte-o e mova a pasta descompactada para a pasta de bibliotecas (libraries) do Arduino. No Windows, a sua pasta libraries estará em Meus Documentos/Arduino. No Mac, você a encontrará em Documents/Arduino/ e, no Linux, estará no diretório Sketchbook. Se a pasta libraries não estiver presente no seu Arduino, então você deverá criá-la*. Depois, reinicie o software Arduino.

A segunda biblioteca – a AndroidAccessory propriamente dita – é baixada como parte da instalação do pacote ADK ("DOWNLOAD ADK package"),

que pode ser encontrado em <http://developer.android.com/tools/adk/adk.html>.

Faça download da biblioteca clicando em "Adk package". Um arquivo zip será baixado. Descompacte-o e você encontrará uma pasta denominada "ADK_release_0512". Ela contém dois arquivos e três pastas. A pasta que nos interessa é a denominada "firmware". Ela contém uma outra pasta de nome "arduino_libs" (bibliotecas do Arduino), dentro da qual há duas pastas, cada uma contendo uma biblioteca Arduino. Uma, "USB_Host_Shield", é desnecessária porque já instalamos uma versão sua. Entretanto, a outra, "AndroidAccessory", deve ser instalada.**

* N. de T.: A pasta deve ser criada com o nome em inglês: libraries.

** N. de T.: Quando o livro foi escrito, a versão vigente era "ADK_release_0512". Mas a Google lançou uma nova versão, a "ADK_release_20120606". Essa é a nova versão que você encontrará no site indicado no texto. Você verá que a pasta "firmware" deixou de existir, e a pasta "AndroidAccessory" que você deve baixar está dentro da "arduino_libs". O autor acredita que essa versão deve funcionar, mas, se você tiver problemas, poderá fazer uma pesquisa na Internet, onde ainda se pode encontrar a versão "ADK_release_0512", com a qual os exemplos foram originalmente desenvolvidos.

Para isso, simplesmente move toda a pasta para a pasta libraries (bibliotecas) do mesmo modo que você fez com a biblioteca USB host.

Você precisa reiniciar o software do Arduino para que ele inclua as novas bibliotecas.

» Passo 7: instale o sketch de Arduino

Se você construiu o robô Bluetooth do Capítulo 1, você já baixou o arquivo zip de www.duinodroid.com com todos os sketches usados neste livro. Se você ainda não o fez, faça isso agora. Descompacte o arquivo zip e move toda a pasta Arduino Android para a pasta de sketches. No Windows, a sua pasta de sketches estará em Meus Documentos/Arduino. No Mac, você a encontrará em Documents/Arduino/ e, no Linux, estará no diretório Sketchbook.

Você precisará reiniciar o software Arduino para que ele inclua os novos sketches.

A seguir, no menu File (arquivo) do software Arduino, selecione sketches (ou SketchBook), seguido de Arduino Android, e então o sketch ch02_Geiger_counter.

Conecte a sua placa de Arduino (sem o shield) ao computador via USB. Precisamos dizer ao software Arduino qual é o tipo de placa que estamos usando. Para definir a placa usada, vá até o menu Tools (ferramentas) e selecione a opção Board (placa).

Selecione a opção para o tipo de placa que você está usando (Arduino Uno). A seguir, devemos fazer algo semelhante com a opção de "Serial Port" (porta serial), que também faz parte do menu Tools. Geralmente, a opção será a que está no topo da lista de portas (COM4, no Windows).

Agora, estamos prontos para transferir o sketch para a placa. Para isso, devemos clicar no ícone Upload (segundo a partir da direita na barra de ferramentas). Se aparecer uma mensagem de erro, verifique o tipo de placa que você está usando e a conexão.

» Passo 8: teste a alimentação de alta tensão

Ainda não conectamos o tubo GM porque precisamos nos assegurar de estar gerando corretamente a alta tensão necessária para o tubo. Após, se for necessário, realizaremos ainda mais alguns ajustes.

Desconecte o Arduino do seu computador e encaixe o shield na placa do Arduino. De agora em diante, observe que partes do shield estarão com alta tensão sempre que o Arduino estiver sendo energizado pelo conector USB ou pela bateria. Portanto, cuidado para não encostar em fios desencapados. Não apenas isso, lembre-se de que o circuito armazena carga e que, mesmo após desligar a alimentação elétrica, ele permanecerá carregado com alta tensão durante um tempo considerável.

Coloque o multímetro na escala de 1000V CC e conecte as suas ponteiras aos pontos de teste mostrados na Figura 2-13. A ponteira negativa deve ser conectada no lado de R4 que está ligado ao terra

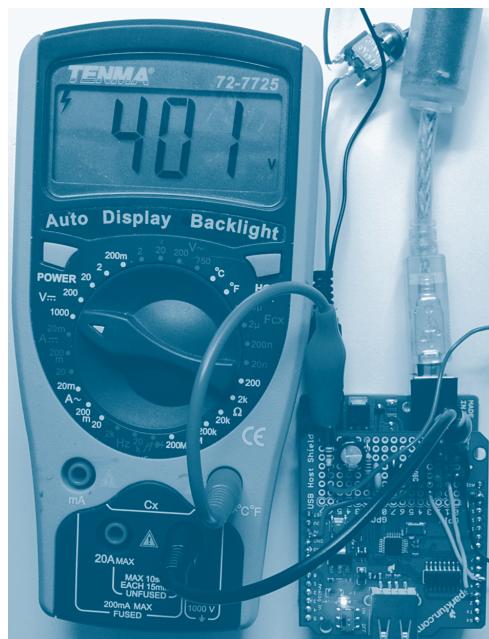


Figura 2-13 Testando a saída de alta tensão.

(GND). A ponteira positiva deve ser ligada no lado de R3 que está conectado ao terminal positivo do capacitor eletrolítico (não no lado que está ligado ao clip positivo do tubo GM).

Se as suas ponteiras não forem do tipo que tem garras (jacarés), então você deverá mantê-las apoiadas contra os pontos de teste depois de energizar a placa.

Insira o plugue da bateria no Arduino com o shield já instalado e ligue a chave. Se tudo estiver certo, você deverá ver uma tensão em torno de 400V. Se a tensão estiver dentro da tolerância do seu tubo GM, então esta parte está pronta.

Se a tensão estiver elevada ou baixa demais, você deverá modificar o sketch para alterar a tensão.

Examine a parte inicial do sketch. Você verá a linha:

```
int op = 45;
```

Podemos ajustar esse valor para mais ou para menos (não ultrapasse 200) para que a tensão suba ou desça, respectivamente.

A tabela seguinte mostra como a tensão variou com os componentes usados aqui. Provavelmente, os seus resultados serão ligeiramente diferentes.

Valor de "op"	Tensão (V)
45	400
50	425
70	500

Encontrar o valor correto é uma questão de tentativa e erro. Você poderá cuidadosamente conectar o cabo USB no seu computador e transferir novamente o sketch com um valor modificado de "op" sem precisar desligar tudo. Entretanto, se você fizer isso, poderá ocorrer um mau contato que resultará em 400V sendo aplicados à sua porta USB, podendo destruir o seu computador. Não diga depois que você não foi alertado. A alternativa mais segura é

desconectar tudo e, usando um alicate com isolação elétrica, colocar em contato os dois clips de conexão do tubo durante meio minuto. Essa operação descarregará os capacitores.

» Passo 9: instale o tubo GM

Desconecte e faça a descarga elétrica de tudo, porque agora estamos prontos para instalar o tubo GM e ver se funciona. Encaixe-o nos clips de fusível, cuidando para que o terminal positivo seja ligado ao clip positivo. O lado positivo do tubo costuma vir com uma marca vermelha ou com um sinal +. Em caso de dúvida, consulte a documentação que acompanha o seu tubo.

» Passo 10: instale o aplicativo Android

Agora, para instalar o aplicativo Android, você deve acessar o site www.duinodroid.com usando o navegador do seu telefone celular e seguindo o link de download do aplicativo DroidGeiger. Observe que talvez você tenha que habilitar a opção de "Fontes Desconhecidas", ou algo similar, nas configurações Android do celular. Com isso, o seu telefone terá permissão para baixar aplicativos vindos de fornecedores desconhecidos, como foi descrito no Passo 10 do Capítulo 1.

» Passo 11: teste

Depois de instalar o aplicativo, podemos colocar tudo sobre a bancada, como mostrado na Figura 2-14.

Ligue o seu telefone ao conector USB Host do shield. Após alguns segundos, o aplicativo DroidGeiger começará a ser executado automaticamente, e as leituras médias de contagem serão exibidas no display. Toda vez que ocorrer um evento de contagem, o telefone emitirá um som de clique e o símbolo de radiação aparecerá por um momento no lado direito.

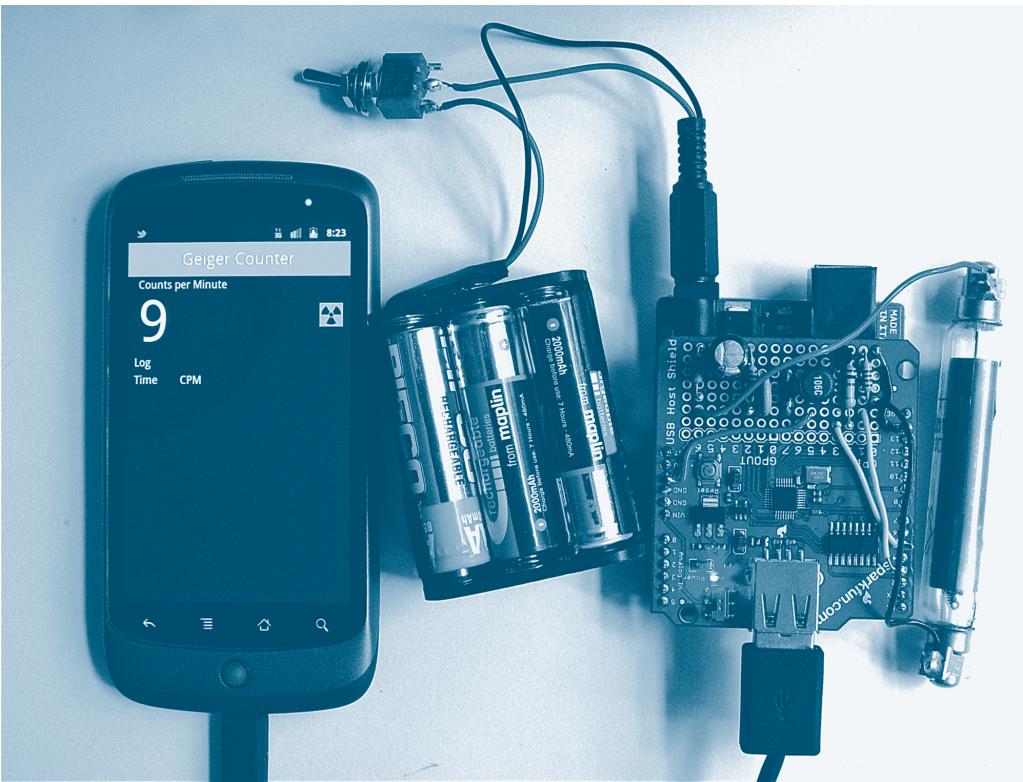


Figura 2-14 Testando o contador Geiger.

A leitura média baseia-se nos intervalos entre os eventos, passando por um processo de filtragem e suavização. Nesse algoritmo, ainda é possível fazer aperfeiçoamentos. Acompanhe o site do livro e conheça os melhoramentos feitos por nós e outros leitores.

A cada minuto, uma nova mensagem aparecerá na parte inferior da tela, na área onde está o registro das leituras (Log), mostrando o número de eventos (contagens) ocorridos no minuto anterior.

» Passo 12: acondicione o projeto

Para acondicionar este projeto, é necessário que a caixa escolhida acomode a chave, o tubo GM, a bateria e a placa de Arduino com o shield.

A caixa deverá ainda ter furos para a chave, o conector USB host do shield e quatro furos para fixar a placa do Arduino. Para isso, serão usados quatro parafusos autoatarraxantes pequenos (Figura 2-15 e Figura 2-16).

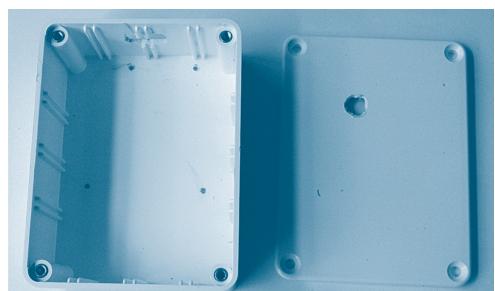


Figura 2-15 A caixa com os furos.

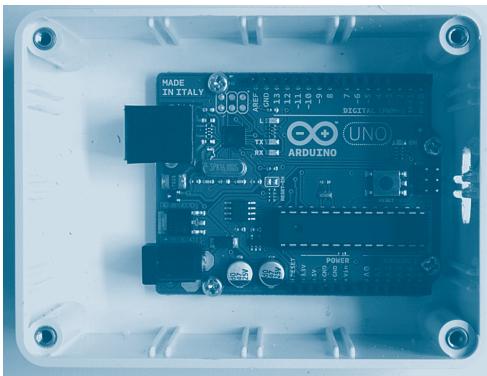


Figura 2-16 A placa de Arduino fixada na caixa.

O tubo GM pode ser instalado dentro da caixa porque a radiação gama detectada por ele não sofre atenuação significativa ao atravessar as paredes da caixa de plástico. Para que ele não fique solto dentro da caixa, clips ou grampos de fixação de cabo podem ser usados para prendê-lo no shield USB (Figura 2-17).

Se o tubo obtido por você for do modelo que capta radiação alfa, será necessário fazer um furo junto à parte do tubo indicada para isso. Essa "janela" é necessária porque a radiação alfa não consegue penetrar em plástico.

A bateria é mantida no lugar pelo conector USB da placa do Arduino. Como acabamento final, tudo que resta fazer é imprimir um rótulo de papel para ser colado na parte frontal da caixa (veja a Figura 2-1).

Este projeto utiliza uma bateria. Contudo, se você optou por usar uma bateria pequena de 9V, ela não conseguirá suprir a corrente de carga de 500mA que, conforme especifica o Open Accessory, o acessório deve fornecer. Para contornar essa situação, use o contador Geiger somente quando o seu telefone estiver completamente carregado ou então construa um cabo que não permite carga. Para isso, em um cabo USB de extensão, insira um resistor de $1k\Omega$ no fio que fornece +5V. Isso permitirá que o

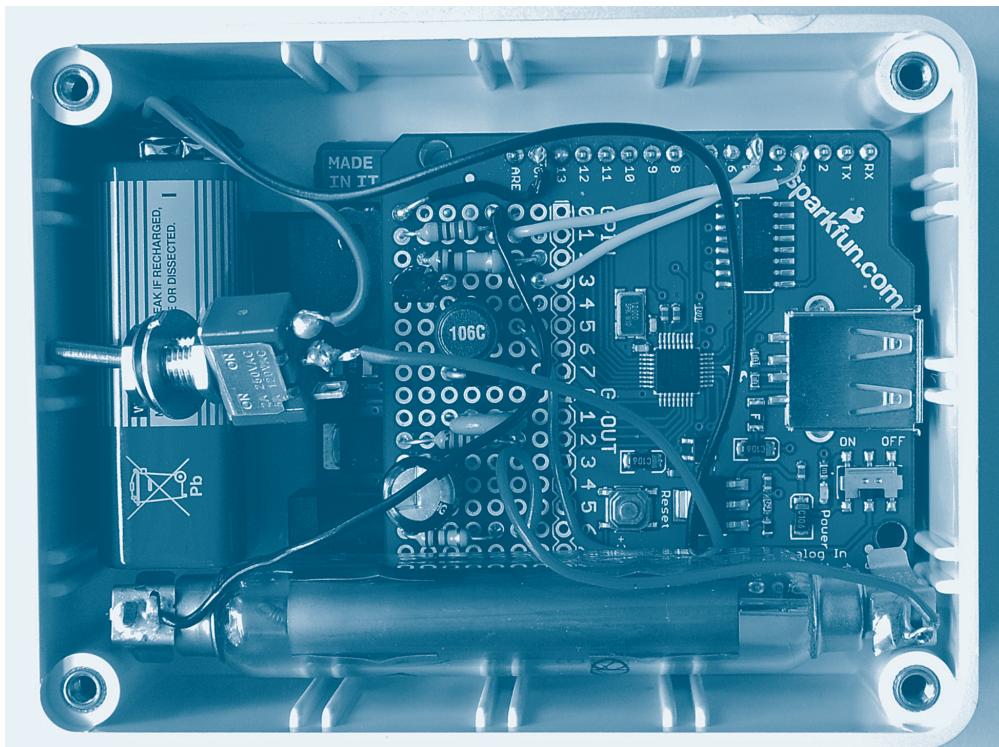


Figura 2-17 Os componentes no interior da caixa.

acessório funcione sem que forneça uma corrente significativa ao telefone.

A sequência de fotos a seguir mostra a construção do cabo.

Cuidadosamente desencapse, em um dos lados, uma parte do cabo com uma faca afiada e puxe para fora o feixe blindado de fios. A seguir, faça um corte no isolamento e localize o fio vermelho de +5V. Corte-o, desencapse e estanhe as terminações (Figura 2-18).

Solde o resistor de $1k\Omega$ no lugar. Então, enrole-o cuidadosamente com fita isolante para que não haja chance de ambos os terminais nus entrarem em contato com os fios ou o isolamento. (Veja a Figura 2-19.)

Agora, você poderá usar esse cabo para conectar o celular com o acessório neste e nos demais projetos deste livro com Open Accessory.

» Teoria

O software deste projeto e de todos os outros deste livro são abertos (open source). Portanto, você pode realizar os seus próprios melhoramentos. Gostaríamos muito de ter notícia de qualquer melhoria que você fizesse. Para entrar em contato conosco acesse www.duinodroid.com, onde você também encontrará todos os códigos-fontes para Android e Arduino.

Nesta seção, olharemos rapidamente o software usado no projeto, começando com o sketch para o Arduino. No entanto, antes disso, examinaremos a forma de gerar a tensão de 400V necessária ao tubo GM.



Figura 2-18 Construção de um cabo USB que não permite carga de bateria.



Figura 2-19 Fixando o resistor no local.

» Gerando 400V

O circuito utilizado para gerar 400V é comandado pelo Arduino. Para isso, os pulsos que acionam o indutor através de um transistor de alta tensão são produzidos pelo Arduino. Esses pulsos são coletados e suavizados por D1, C1, C2 e R2.

Os pulsos que vêm do Arduino são gerados em uma saída PWM. Esse tipo de saída digital envia pulsos de duração variável e costuma ser utilizado para realizar coisas como controlar a velocidade de um motor.

Usamos o mesmo mecanismo para produzir a tensão necessária ao tubo GM.

» O tubo Geiger-Müller

Para uma descrição completa dos tubos Geiger-Müller, acesse a Wikipedia. O princípio básico é que o tubo tem um catodo externo (terminal negativo) e um anodo interno (terminal positivo) encapsulados em um tubo de vidro que contém um gás. Os tubos mais caros, capazes de detectar radiação alfa*, têm uma janela de mica que permite a passagem de partículas alfa. No tubo usado pelo autor, essas partículas são barradas pelo vidro.

Quando uma partícula entra no tubo, ocorre a ionização do gás em seu interior. Isso torna o gás temporariamente condutor, causando um pulso de corrente que pode ser detectado.

Você poderá ver esses pulsos utilizando um osciloscópio para medir a tensão em R4. A Figura 2-20 mostra os pulsos.

A base de tempo do osciloscópio foi ajustada para 500ms, permitindo ver um intervalo em torno de dois segundos entre os pulsos. A escala Y está calibrada com 2 volts/divisão, resultando em uma amplitude de aproximadamente 5V. Os pulsos são extremamente estreitos.

* N. de T.: A radiação alfa consiste em núcleos de hélio dotados de velocidade muito elevada.

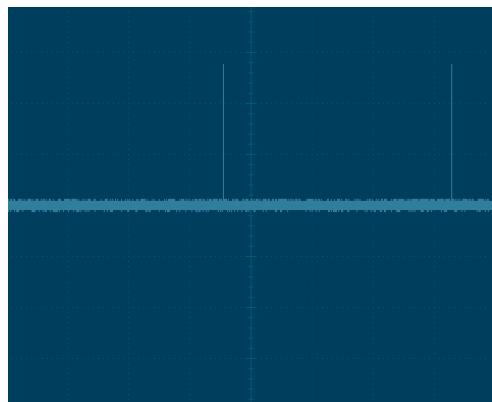


Figura 2-20 Um traçado de osciloscópio mostrando os pulsos.

» O sketch de Arduino

O sketch para o Arduino é muito simples. A biblioteca Android Accessory simplifica muito o processo de enviar dados para o dispositivo Android. No algoritmo que faz a média das contagens por minuto, ainda há muito espaço para melhoramentos, principalmente com contagens de baixa frequência. O algoritmo funciona melhor com as contagens de frequência elevada, como você verá caso tenha a sorte de encontrá-las.

Começamos incluindo as bibliotecas necessárias. A biblioteca Max3124e faz parte da biblioteca USB host que já instalamos.

A seguir, há uma definição de pino, "oscPin", para que o sinal PWM do Arduino acione a bobina que gera a alta tensão. Observe que não há definição de um pino para receber o sinal de pulso de retorno quando um evento é detectado. Para isso, usaremos uma interrupção no pino 3.

A variável "op" controla a largura (duração) do pulso, como foi descrito antes neste capítulo.

As variáveis do tipo "long", com nome "last" mais alguma coisa, são todas usadas para registrar tempos. A função "millis()" retorna um número "long" (32 bits) que representa o total de milissegundos decorridos desde que a placa do Arduino foi iniciada.

```

#include <Max3421e.h>
#include <Usb.h>
#include <AndroidAccessory.h>

int oscPin = 5;
int op = 45;
int minPulseSep = 50;
long lastEventTime = 0;
long lastTimerTime = 0;
long timerPeriod = 5001;
long lastLogTime = 0;
long logPeriod = 600001;
int count = 0;

float smoothingFactor = 0.6;
float instantaneousCPM = 0.0;
float smoothedCPM = 0.0;

AndroidAccessory acc("Simon Monk",
    "DroidGeiger",
    "Geiger Counter Accessory",
    "1.0",
    "http://www.duinodroid.com",
    "0000000012345678");

void setup()
{
    Serial.begin(9600);
    pinMode(oscPin, OUTPUT);
    analogWrite(oscPin, op);
    acc.powerOn();
    attachInterrupt(1, eventInterrupt, RISING); // Rising = para cima, subindo
}

void loop()
{
    if (acc.isConnected())
    {
        // a cada meio segundo, faça uma leitura instantânea e a integre (some)
        // à leitura média e, em seguida, envie-a
        long timeNow = millis();
        if (timeNow > (lastTimerTime + timerPeriod))
        {
            lastTimerTime = timeNow;
            integrateInstantReadingIntoSmooth();
            sendMessage('R', (int) smoothedCPM);
        }
        // a cada minuto, envie o total acumulado
        timeNow = millis();
        if (timeNow > (lastLogTime + logPeriod))
        {
            lastLogTime = timeNow;

```

```

        sendMessage('L', count);
        count = 0;
    }
}
delay(100);
}

void eventInterrupt()
{
    // ocorreu um evento, faça uma leitura
    calculateInstantCPM();
    count++;
    sendMessage('E', 0);
}

void calculateInstantCPM()
{
    // cpm instantânea = 60.000 / dt em milissegundos
    long timeNow = millis();
    long dt = timeNow - lastEventTime;
    if (dt > minPulseSep)
    {
        instantaneousCPM = ((float)logPeriod) / dt;

        lastEventTime = timeNow;
    }
}

void integrateInstantReadingIntoSmooth()
{
    smoothedCPM = smoothedCPM * smoothingFactor
        + instantaneousCPM * (1 - smoothingFactor);
}

void sendMessage(char flag, int cpm)
{
    if (acc.isConnected())
    {
        byte msg[4];
        msg[0] = 0x04;
        msg[1] = (byte) flag;
        msg[2] = cpm >> 8;
        msg[3] = cpm & 0xff;
        acc.write(msg, 4);
    }
}

```

lizada (reset). Isto é usado para determinar se certas ações deverão ser executadas.

As variáveis "timerPeriod" e "logPeriod" são constantes, representando respectivamente meio segundo e um minuto. São usadas no interior do

"loop" para enviar atualizações periódicas ao dispositivo Android (a cada meio segundo) e uma contagem a cada minuto.

As três variáveis seguintes do tipo "float" são todas usadas para obter um valor médio instantâneo que será enviado ao telefone.

Depois das definições de variável, a linha que começa com "Android Accessory" estabelece a conexão entre o celular e o Arduino. Os parâmetros são:

- Fabricante
- Modelo
- Descrição
- Versão
- URL
- Número de série

Desses, para que o aplicativo Android seja iniciado, somente fabricante, modelo e versão devem ser iguais aos parâmetros correspondentes que estão listados no arquivo *AndroidManifest** do projeto Android. Entretanto, é possível que os demais parâmetros apareçam na tela do celular.

A função "setup" dá início à comunicação serial. Isso não é necessário para que o sketch funcione. Entretanto, desse modo, a biblioteca USB mostra como a conexão está funcionando. Assim, se ocorrer algum problema de conexão com o telefone, você poderá abrir o Serial Monitor e obter algumas informações a respeito.

Mais algumas coisas interessantes ocorrem na função "setup". A "acc.powerOn" avisa o telefone que o acessório foi ligado e está pronto. A seguir, incluímos uma interrupção. O primeiro argumento é o número da interrupção. A interrupção 1 cor-

responde ao pino 3. Sempre que houver um sinal com sua tensão subindo (de 0 para 5V) no pino 3, o processador será interrompido e a função "eventInterrupt" (interrupção por evento) entrará em execução.

A função "loop" chama inicialmente a função "acc.isConnected" que testa se há uma conexão e, em caso negativo, tenta estabelecer uma. Se o acessório já estiver conectado, ela prossegue e, em caso contrário, o "loop" nada faz.

O loop tem dois mecanismos marcadores de tempo que executam ações a cada meio segundo e a cada minuto. O marcador de meio segundo chama uma função para atualizar o valor médio corrente e, em seguida, envia uma mensagem de atualização para o telefone. Essa mensagem contém um identificador "R" (de Reading, leitura de contagem), que é usado pelo telefone, e o novo valor calculado da leitura.

O marcador de tempo de um minuto envia uma mensagem contendo um identificador "L" (de Log), que será exibido na área de registro de leituras (log) da tela do telefone, além do valor total de contagens que foi acumulado no minuto anterior. Após o envio da mensagem, o contador é zerado.

Como mencionado antes, sempre que ocorrer um evento de captação de radiação no tubo GM, acionaremos um mecanismo de interrupção para chamar a função "eventInterrupt". Essa função atualiza a variável "instantaneousCPM" (contagem por minuto instantânea), incrementa o contador "count" e, em seguida, envia uma mensagem "E" de ocorrência de evento ("Event"). Uma mensagem de evento não envia o valor atualizado de leitura. Simplesmente avisa o telefone para fazer o ruído clássico de clique de contador Geiger e para fazer o símbolo de radiação piscar.

A função "sendMessage" (enviar mensagem) formata a mensagem enviando-a em seguida. Essa mensagem é composta de quatro bytes.

* N. de T.: Todo aplicativo Android deve conter, na pasta raiz de seu pacote de arquivos, um arquivo especial XML denominado "AndroidManifest", contendo informações fundamentais sobre o aplicativo e incluindo diversos parâmetros e indicando quais são os recursos necessários ao seu funcionamento.

O primeiro byte contém um código de identificação para todas as mensagens oriundas desse acessório (0x04). O segundo byte descreve o tipo de mensagem, podendo ser "R", "E" ou "L", dependendo de se a mensagem é "Reading", "Event" ou "Log". Os dois bytes finais são usados para representar o número inteiro de 16 bits com o valor da leitura.

» O aplicativo Android

O aplicativo Android é grande demais para mostrarmos a sua listagem inteira. Assim, se você estiver interessado, faça o download do arquivo-fonte de www.duinodroid.com. Agora, daremos uma olhada nas partes-chaves do código.

Uma característica útil do ADK Android é que faz o telefone iniciar a execução de um aplicativo sempre que um acessório Arduino é conectado ao telefone. Isso é obtido a partir dos dados descritos na configuração do arquivo `AndroidManifest.xml` do projeto.

A seção-chave desse arquivo é a seguinte:

```
<activity android:name="org_simonmonk
    .geiger.UsbAccessoryActivity"
        android:label="DroidGeiger"
        android:taskAffinity=""
        android:launchMode=
        "singleInstance">
    <intent-filter>
        <action android:name="android
            .hardware.usb.action.USB_
            ACCESSORY_ATTACHED"/>
    </intent-filter>
    <meta-data android:name="android
        .hardware.usb.action.USB_
        ACCESSORY_ATTACHED"
        android:resource="@xml/accessory_
        filter"/>
</activity>
```

O filtro de acessórios determina quais acessórios estão aptos a dar início à execução desse aplicativo. O filtro para esse aplicativo é como segue:

```
<resources>
    <usb-accessory manufacturer=
        "Simon Monk" model="DroidGeiger"
        version="1.0"/>
</resources>
```

Em outras palavras, o aplicativo será iniciado somente se o nome do fabricante (manufacturer), o modelo e a versão enviados pelo dispositivo Arduino forem iguais aos parâmetros anteriores listados nos recursos (resources).

Agora, vamos acompanhar o caminho seguido por uma mensagem vinda do Arduino.

Examinando o método "run" da classe `DroidGeigerActivity`, você encontra o seguinte trecho de código:

Esse método decodifica a mensagem original vinda da conexão USB e cria um objeto-mensagem para enviar a um Handler. Um Handler é o modo de um Android lidar com a thread de interface com o usuário.

```
case 0x4:
    if (len >= 3) {
        Message m = Message.obtain
            (mHandler, MESSAGE_TEMPERATURE);
        char flag = (char)buffer[i + 1];
        int cps = composeInt(buffer[i + 2],
            buffer[i + 3]);
        m.obj = new GeigerMsg(flag,
            countsPerQuarter);
        mHandler.sendMessage(m);
    }
    i += 4;
    break;
```

Como mostrado no snippet* de código a seguir, o Handler chamará o método handleGeigerMessage na classe InputController:

```
public void handleGeigerMessage
    (char flag, int reading) {
    if (flag == 'E') {
        mRadiationImage.setVisibility
            (ImageView.VISIBLE);
        mp.start();
    }
    else if (flag == 'R') {
        mRadiationImage.setVisibility
            (ImageView.INVISIBLE);
        mTemperature.setText("" + reading);
    }
    else if (flag == 'L') {
        String logText = mLogView
            .getText().toString();
        String timeFormatted = (String)
            DateFormat.format("hh:mm", new
            Date());
        mLogView.setText(logText + "\n" +
            timeFormatted + "\t\t\t" +
            reading);
    }
}
```

Dependendo do valor do identificador da mensagem, alterações específicas serão realizadas na interface do usuário.

» Resumo

Há muitos formas de estender este projeto. Por exemplo, o aplicativo Android poderia, com pouco trabalho, enviar os dados para outros celulares, permitindo o monitoramento remoto.

Poderia também enviar as leituras para um servidor de Web.

Esse aplicativo é básico e espera-se que os leitores o levem adiante fazendo todo tipo de coisas interessantes com ele.

No próximo capítulo, veremos um outro projeto usando Open Accessory. Desta vez, construiremos um projeto para produzir um show de luzes – porque os aficionados de Android e Arduino gostam de festas!

* N. de T.: Snippet é o nome dado a um pequeno trecho de código de programa que é usado em diversos locais do código.



» capítulo 3

Show de luzes com Android

Neste capítulo, mostraremos um projeto comandado por telefone Android que permite animar suas festas.

Objetivos

- » Construir um dispositivo para produzir um show de luzes usando o Open Accessory
- » Entender o que é uma Base para Acessório Droid e como montá-la
- » Entender como utilizar MOSFETs para acender e apagar LEDs
- » Entender o que é Modulação por Largura de Pulso
- » Entender as variáveis e funções constantes no sketch do Arduino para esse projeto

Depois de colocar o telefone em uma base para acessório Android, pode-se controlar painéis de LEDs com um toque de dedo, produzindo assim um show de luzes (Figura 3-1).

O acessório de controle do show de luzes está mostrado na Figura 3-2.

Para isso, o acessório utiliza três painéis de LEDs, cada um com 36 LEDs. Cada painel é de uma cor diferente: vermelho, verde e azul. A luminosidade de cada um pode ser controlada separadamente através do aplicativo Android, o qual monitora o nível sonoro através do microfone do telefone e faz as luzes piscarem de acordo.

» Construção: a Base para Acessório Droid

Como o contador Geiger, este projeto também é do tipo Open Accessory. Desta vez, entretanto, como só precisamos de poucos componentes além do Arduino, vamos utilizar um shield USB host e transformá-lo para que funcione como um Ardui-

no instalado fora de sua placa original. Em outras palavras, usaremos um Arduino para programar o seu microcontrolador ATMega328 e, em seguida, transferir esse microcontrolador para o shield. Isso significa que a placa do Arduino poderá ser usada em outro projeto.

Construiremos o nosso próprio Arduino que será montado em um shield USB host. Os poucos pinos de Arduino que necessitamos neste projeto serão ligados a um conector de seis pinos. Teremos três pinos PWM e um pino genérico de entrada e saída. A Base para Acessório Droid completa está mostrada na Figura 3-3.

Essa base será utilizada neste projeto e nos três seguintes.

O diagrama esquemático do projeto está mostrado na Figura 3-4.

Como você pode ver no diagrama, é necessário um mínimo de componentes para que o microcontrolador possa funcionar sem a placa de Arduino. Necessitamos de um oscilador a cristal, um resistor e dois capacitores. Isso é tudo.



Figura 3-1 O acessório para o show de luzes.

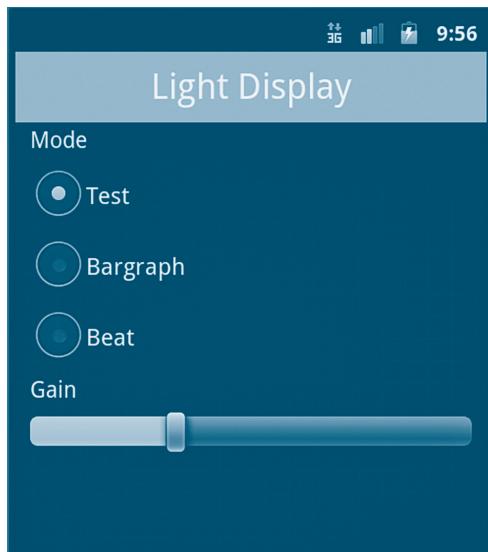


Figura 3-2 O aplicativo do show de luzes.

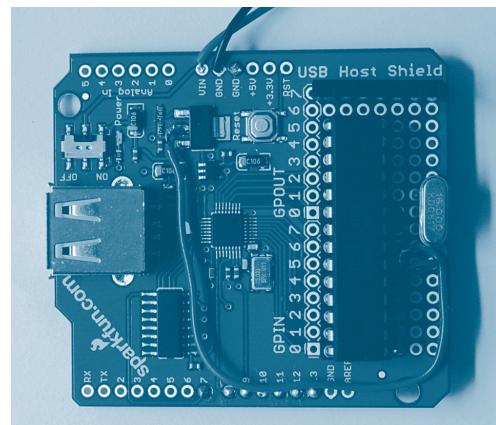


Figura 3-3 A Base para Acessório Droid.

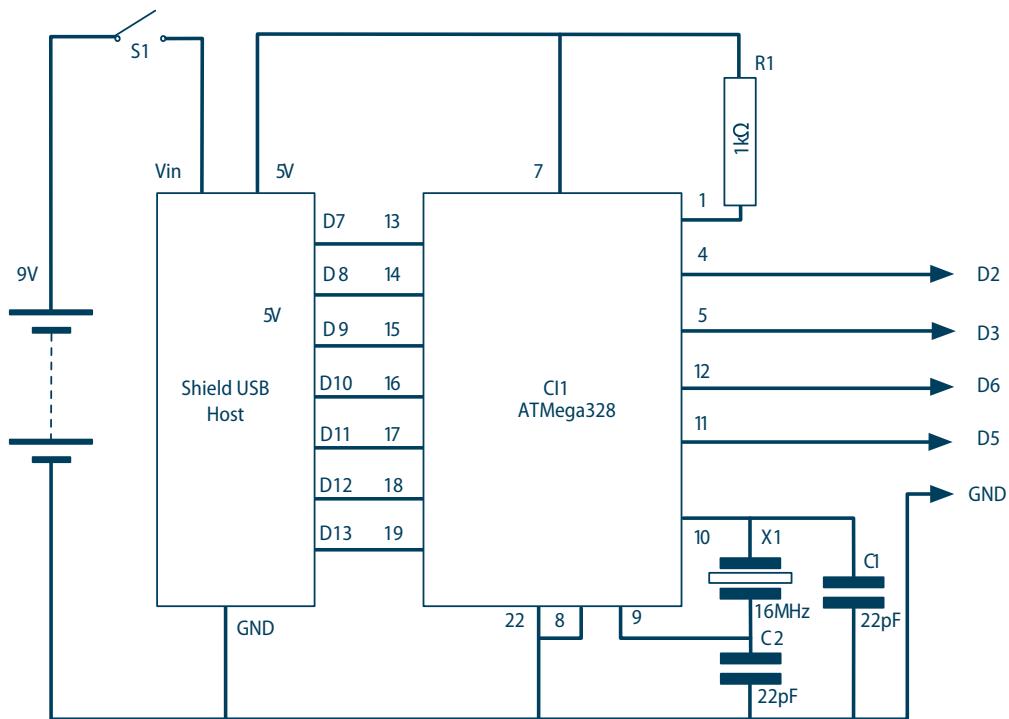


Figura 3-4 O diagrama esquemático.

LISTA DE COMPONENTES

Componente	Quantidade	Descrição	Fornecedor
Arduino Uno	1	Placa de Arduino Uno (para programar)	wwwarduino.cc
Shield USB	1	Shield USB host (hospedeiro) para Arduino	Sparkfun: DEV-09947
Microcontrolador	1	ATMega328 com bootloader	Sparkfun: DEV-10524
R1	2	Resistor de filme metálico de $1k\Omega$ e 1/2 W	Farnell: 9339779
C1, C2	2	Capacitor cerâmico de $22pF$	Farnell: 1600966
X1	1	Cristal de 16 MHz	Farnell: 1611761
Soquete para CI	1	Soquete para circuito integrado de 28 pinos	Farnell: 1824463
Barra de pino fêmea para PCB*	1	Barra de pino fêmea com seis soquetes. Também conhecida como alojamento ou barra de soquetes para placa de circuito integrado.	
Caixa	1	Caixa pequena para projetos	Lojas de eletrônica
Chave	1	Chave miniatura SPST (um polo, uma posição)	Farnell: 1661841

* N. de T.: Printed Circuit Board, ou Placa de Circuito Impresso.

» O que será necessário (Base para Acessório Droid)

A lista de componentes supracitada é da “Base para Acessório Droid”, como este módulo será denominado. Para construir o módulo do show de luzes, você precisará de outros componentes que estão listados mais adiante em seção que trata especificamente da construção do projeto.

Além desses componentes, você também necessitará das ferramentas seguintes.

CAIXA DE FERRAMENTAS

- Ferramentas para soldar
- Um computador para programar o Arduino
- Um cabo de conexão USB do tipo A-B

» Passo 1: solde o soquete do CI no lugar

Neste projeto, a maior parte da fiação será realizada na parte debaixo da placa. Por isso, em vez de começar com as conexões de fio, soldaremos primeiro o soquete de CI (circuit integrado). Isso facilitará a soldagem dos fios de conexão porque os

pontos de solda serão mais facilmente localizados tendo o soquete do CI como referência.

O diagrama de fiação para a placa de protótipo está mostrado na Figura 3-5.

A Figura 3-6 mostra a placa com o soquete no lugar.

Instale o soquete de tal forma que a reentrância do CI, que indica a posição do pino 1, fique como

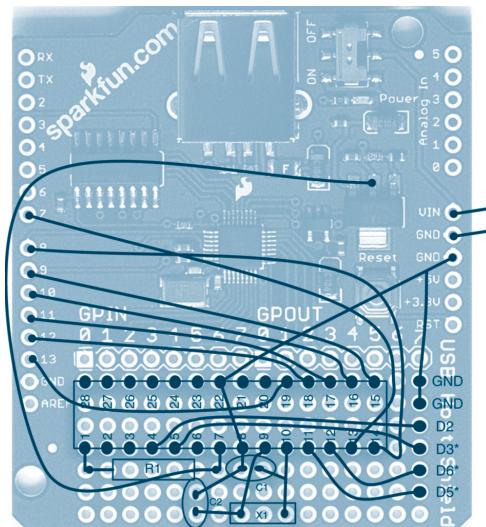


Figura 3-5 O diagrama de fiação.

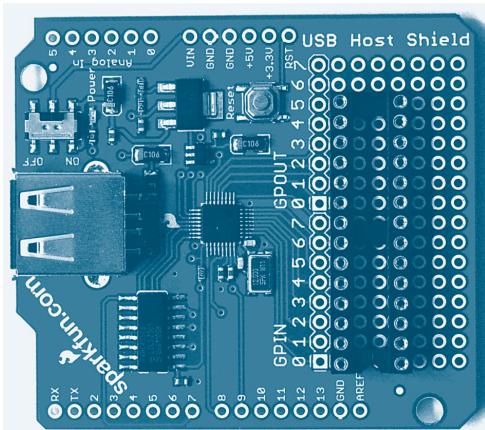


Figura 3-6 O shield com o soquete do Cl no lugar.

mostrado na Figura 3-5. Você só precisa soldar os pinos que realmente serão utilizados. É uma boa ideia soldar primeiro um pino de cada canto para manter o soquete no lugar.

» Passo 2: solde os fios de conexão no Shield USB

A seguir, devemos fazer as conexões com os pinos do soquete de Cl. Lembre-se de que, mais adiante, vamos inserir o microcontrolador nesse soquete. Portanto, ao fazer as soldas nos pinos do soquete, é como se você estivesse fazendo as soldas nos pinos do microcontrolador do Arduino. Use as Figuras 3-5 e 3-7 como referência.

Isso dará um pouco de trabalho. Corte os fios com o comprimento correto e descasque o mínimo necessário de suas extremidades. A seguir, solde cada fio no lugar, cuidando para não fazer pontes de solda entre os pinos do Cl.

» Passo 3: instale o cristal e os outros componentes

Agora, voltamos ao lado de cima da placa para soldar o cristal X1, os dois capacitores e o resistor. Veja a Figura 3-8.

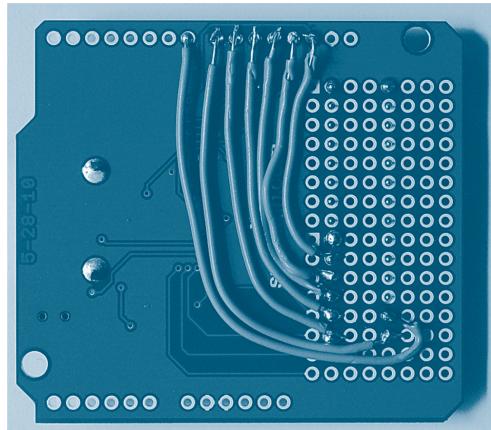


Figura 3-7 O shield com os fios de conexão soldados.

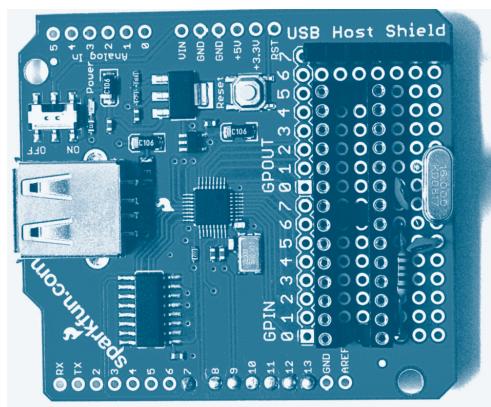


Figura 3-8 Os componentes instalados no lado de cima.

Solde os terminais dos componentes nos furos pelos quais devem passar, mas não corte ainda as partes em excesso. Nós vamos curvar essas partes para formar conexões entre os componentes. Use a Figura 3-5 como referência para fazer isso.

Aproveitamos também para soldar a barra de pino fêmea de seis soquetes, à qual os acessórios serão conectados.

O resultado final será como mostrado na Figura 3-9.

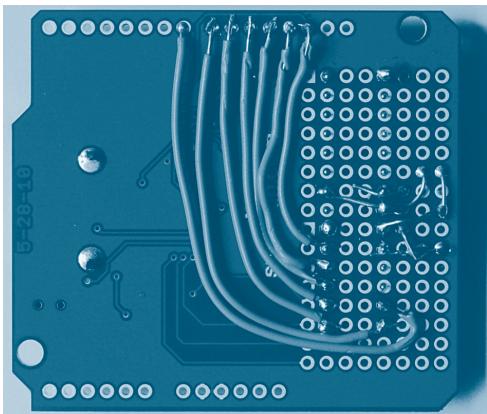


Figura 3-9 Conectando os componentes debaixo da placa.

» Passo 4: instale os fios das demais conexões

Agora, podemos voltar para a parte debaixo da placa e soldar os fios das demais conexões com a barra de pino fêmea de seis soquetes e com o terra. O resultado final está mostrado na Figura 3-10.

» Passo 5: conectando a fonte de alimentação de 5V

Como não temos uma placa de Arduino para fornecer os 5V, deveremos buscar esses 5V no shield. Para isso, devemos soldar cuidadosamente um fio entre

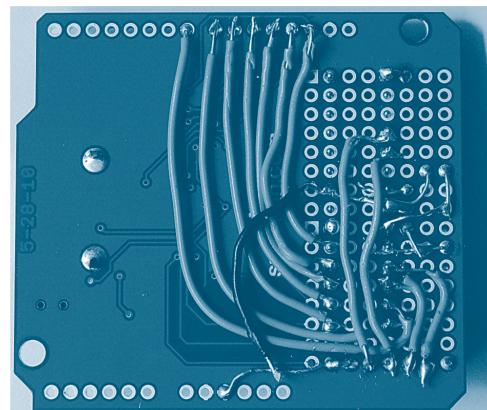


Figura 3-10 Soldando as conexões restantes.

o terminal central do regulador de tensão e o lado de R1 que se conecta ao pino 7 do CI (Figura 3-11).

» Passo 6: conecte a chave e os fios de alimentação elétrica

Tudo que resta fazer para completar a Base para Acessório Droid é instalar a chave e os fios de alimentação elétrica (Figura 3-12).

O acessório Droid pode ser usado em projetos tanto com bateria como com adaptador de voltagem CA/CC. Por enquanto, vamos conectar uma chave e dois fios, que podem ser ligados a um clip de bateria ou a um jack de alimentação elétrica, como no caso deste projeto de show de luzes.

» Passo 7: teste

Neste projeto, há muitas conexões, todas próximas entre si. Dedique algum tempo para inspecionar a placa e verificar se todas as conexões estão como descrito na Figura 3-5 e se não há pontes indesejáveis.

No final, utilizaremos este projeto de show de luzes para fazer um teste, mas poderemos fazer um teste inicial básico instalando primeiro o sketch Droid Geiger em um Arduino Uno. Em seguida, retiramos o microcontrolador da placa e voltamos a instalá-lo no soquete de CI do shield. Tome muito cuida-

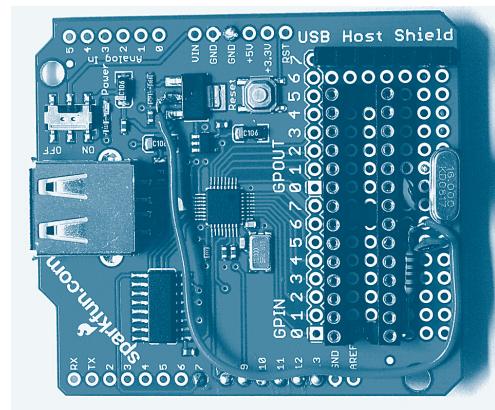


Figura 3-11 Conectando a alimentação de 5V.

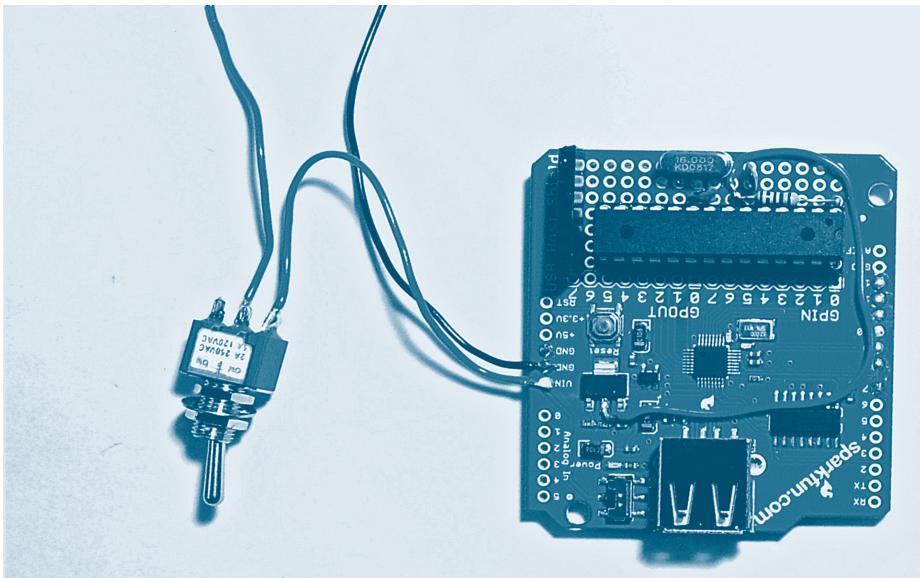


Figura 3-12 Instalando a chave e os fios de alimentação.

do para não dobrar os terminais no soquete do CI, porque isso acontece muito frequentemente.

Além disso, verifique se você encaixou o CI de forma correta, com as reentrâncias do CI e do soquete alinhadas.

Conecte uma bateria de 9V com os fios de alimentação da Base para Acessório Droid e encaixe o conector USB no seu celular Android que deverá estar com a bateria totalmente carregada. Se você instalou o aplicativo Droid Geiger do projeto do Capítulo 2, ele começará a ser executado e, obviamente, nenhuma leitura de contagem radioativa será vista na tela.

Se o aplicativo Droid Geiger não estiver instalado, você verá uma mensagem avisando que você pode baixá-lo do site deste livro.

Em ambos os casos, isso é um bom sinal.

» Construção: o projeto show de luzes

Agora que construímos a Base para Acessório Droid, o restante deste projeto (e dos próximos três projetos) é relativamente imediato.

O diagrama esquemático do projeto está mostrado na Figura 3-13.

O projeto usa os três pinos PWM da Base para Acessório Droid. Usa o pino D5 para controlar o módulo de LEDs vermelhos, o D6 para controlar os LEDs verdes e o D3 para os azuis. Cada um desses pinos aciona um transistor MOSFET que controla a potência elétrica dos módulos de LEDs. Esses módulos consomem potência demais para serem controlados diretamente pelo microcontrolador.

» O que será necessário (show de luzes)

A tabela *Lista de Componentes* apresentada adiante é para a parte do show de luzes do projeto. Supõe-se que você já construiu a Base para Acessório Droid descrita antes neste capítulo.

Cada um dos módulos de LEDs são constituídos por 36 LEDs de alto brilho e de montagem superficial (SMD). São projetados para funcionar com 12V. Na parte de trás, há superfícies autoadesivas que são utilizadas para colá-los à caixa. Entretanto,

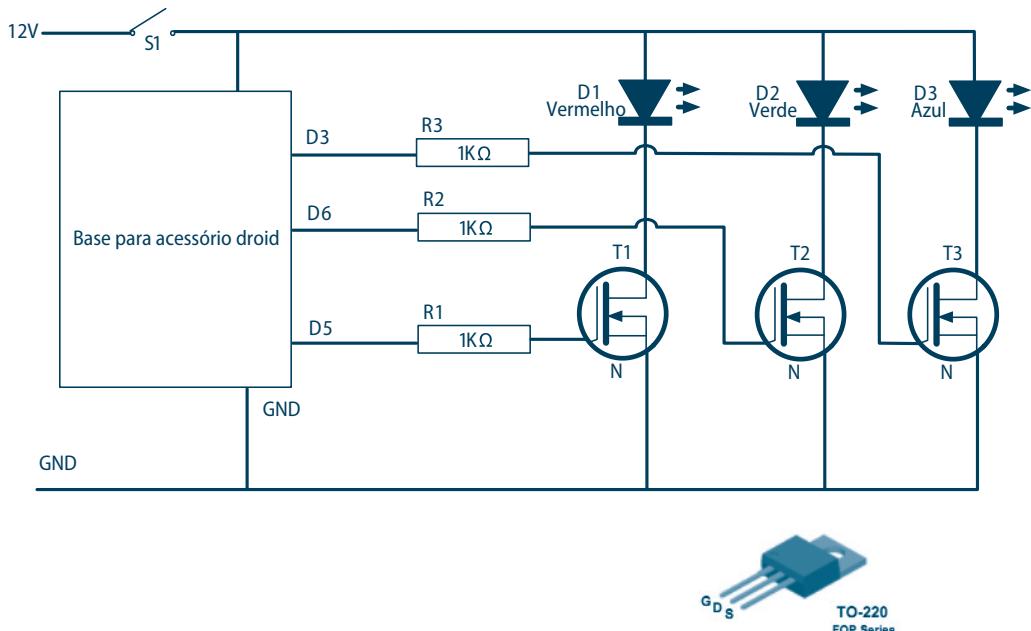


Figura 3-13 O diagrama esquemático do circuito responsável pelo show de luzes.

você poderá usar qualquer módulo de LEDs CC de 12V para as luzes, desde que os transistores sejam capazes de lidar com até 10W por canal. Nesse caso, você precisará de uma fonte de alimentação bem mais potente. Se os conectores dos seus módulos de LEDs não tiverem uma separação de 0,1 polegada entre os pinos, você precisará de um outro tipo de conector.

Além desses componentes, você precisará também das seguintes ferramentas.

CAIXA DE FERRAMENTAS

- Ferramentas para soldar
- Uma furadeira elétrica com brocas
- Parafusos autoatarraxantes pequenos
- Um computador para programar o Arduino
- Um cabo de conexão USB do tipo A-B

» Passo 1: construa uma Base para Acessório Droid

Veja a parte inicial deste capítulo.

» Passo 2: corte a placa perfurada no tamanho

Corte um pedaço da placa perfurada no tamanho de 20 por 10 furos. Para isso, a melhor maneira é fazer sulcos na placa com um estilete e então quebrá-la ao longo dos sulcos apoiando-a na beira de sua mesa de trabalho.

A Figura 3-14 mostra a placa perfurada já cortada.

A utilização de uma placa perfurada consiste em encaixar os componentes por cima e soldá-los por baixo. Depois, eles são conectados entre si usando os terminais que se sobressaíram e, se necessário, soldando fios extras.

LISTA DE COMPONENTES

Componente	Quantidade	Descrição	Fornecedor
Base para Acessório Droid	1	Veja o início deste capítulo	
R1-3	3	Resistor de filme metálico de $1k\Omega$ e $1/2 W$	Farnell: 9339779
T1-3	3	Transistores MOSFET FQP33N10 (ou equivalente)	Farnell: 9845534
D1	1	Módulo de LEDs vermelhos (veja <i>mais adiante</i>)	eBay
D2	1	Módulo de LEDs verdes (veja <i>mais adiante</i>)	eBay
D3	1	Módulo de LEDs azuis (veja <i>mais adiante</i>)	eBay
Placa perfurada	1	Placa no tamanho de 20 por 10 furos	Farnell: 1172145
Barra de pino fêmea para PCB	1	Barra de pino fêmea para PCB com seis soquetes	Farnell: 1218869
Conector KRE	1	Duas vias e espaçamento de 5mm	Farnell: 1641932
Caixa para projeto	1	Caixa grande para projeto	Lojas de eletrônica
Jack J4	1	Jack J4 de 2,1mm para fonte de alimentação	Farnell: 1217037
Adaptador de voltagem CA/CC	1	Fonte de alimentação de 12V e 1,5A	Lojas de eletrônica

A Figura 3-15 mostra a disposição dos componentes na placa.

» Passo 3: solde a barra de pino fêmea, o conector KRE e o transistor

A melhor forma de usar uma placa perfurada é colocando tudo no lugar de tal forma que possam se interconectar. Um bom ponto de partida é colocar a barra de pino fêmea, o conector KRE e um dos transistores na placa (Figura 3-16).

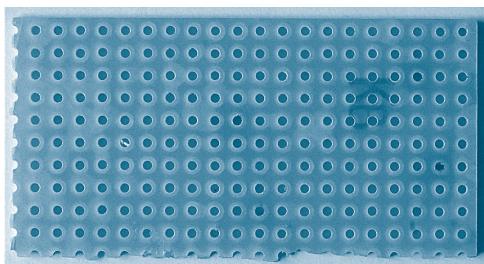


Figura 3-14 A placa perfurada já cortada.

A Figura 3-17 mostra a parte debaixo da placa. Pode-se ver que os terminais do transistor curvam-se o suficiente para alcançar uma das vias do conector KRE e um dos soquetes da barra de pino fêmea.

» Passo 4: solde os demais componentes

Agora podemos instalar os componentes que faltam. O mais fácil é simplesmente colocar os demais transistores e resistores em seus lugares na placa. Em seguida, curve os seus terminais nas direções corretas para se conectarem e, finalmente, soldê-los.

As Figuras 3-18 e 3-19 mostram as partes de cima e debaixo da placa depois da soldagem de todos os componentes.

A barra de pino macho usa apenas quatro das conexões, uma para o terra e três para R1, R2 e R3. Você pode ver na Figura 3-18 como os terminais dos resistores passam através dos furos da placa, primeiro para baixo e, em seguida, de volta para cima, terminando por se conectar à barra de pino macho na parte de cima da placa perfurada. So-

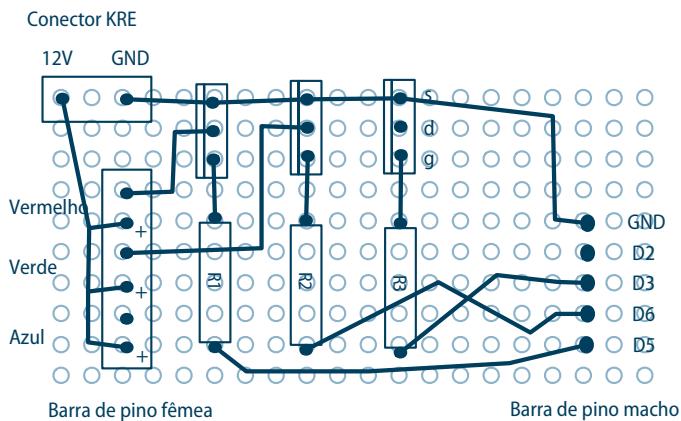


Figura 3-15 A disposição dos componentes na placa perfurada.

mente o resistor que está mais próximo da barra de pino macho conseguirá alcançá-la diretamente. Os terminais dos outros dois resistores provavelmente necessitarão ser aumentados soldando fios extras.

Observe o fio em zigue-zague, no lado direito da Figura 3-19, conectando o terminal positivo da alimentação elétrica às três conexões positivas dos LEDs. O melhor é curvar os fios adequadamente e, em seguida, soldá-los no lugar.

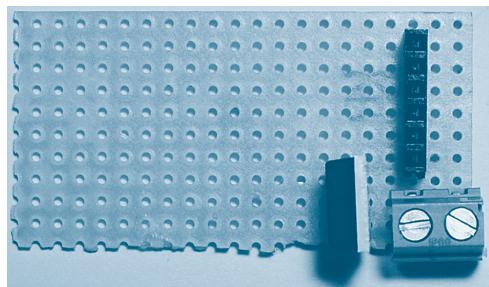


Figura 3-16 Instalando os primeiros componentes.

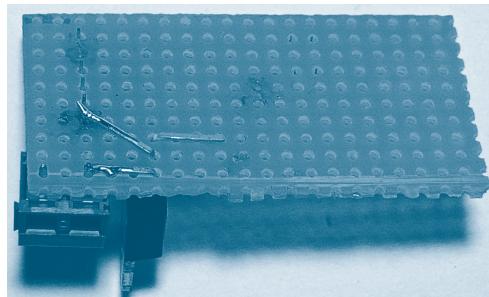


Figura 3-17 A parte debaixo da placa com os primeiros componentes instalados.

» Passo 5: ligue tudo

A Figura 3-20 mostra como tudo é ligado junto.

Tome cuidado para que os conectores dos LEDs sejam ligados corretamente.

Como você pode ver na Figura 3-20, um outro fio foi conectado entre a chave e o terminal positivo do conector KRE da placa perfurada.

Precisamos também conectar os fios de alimentação elétrica do shield do acessório Droid e conectá-los a um jack de alimentação elétrica de 2,1mm para ligarmos a nossa fonte de alimentação de 12V à placa (veja a Figura 3-20).

» Passo 6: instale o sketch do Arduino

Para instalar o sketch no microcontrolador, primeiro programaremos o chip. A seguir, iremos retirá-lo com cuidado da placa do Arduino e, depois, encaixá-lo no soquete da Base para Acessório Droid.

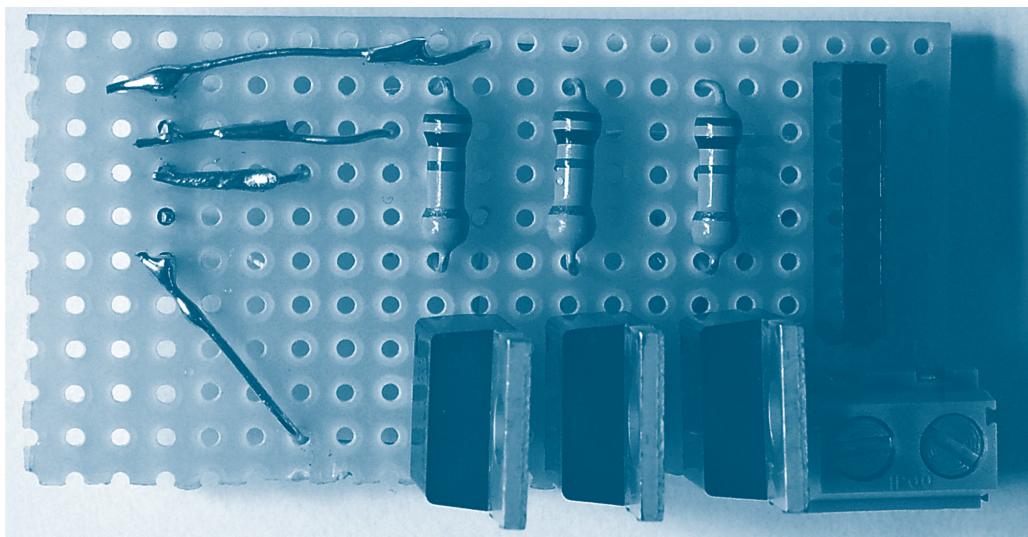


Figura 3-18 O lado de cima da placa depois de completada.

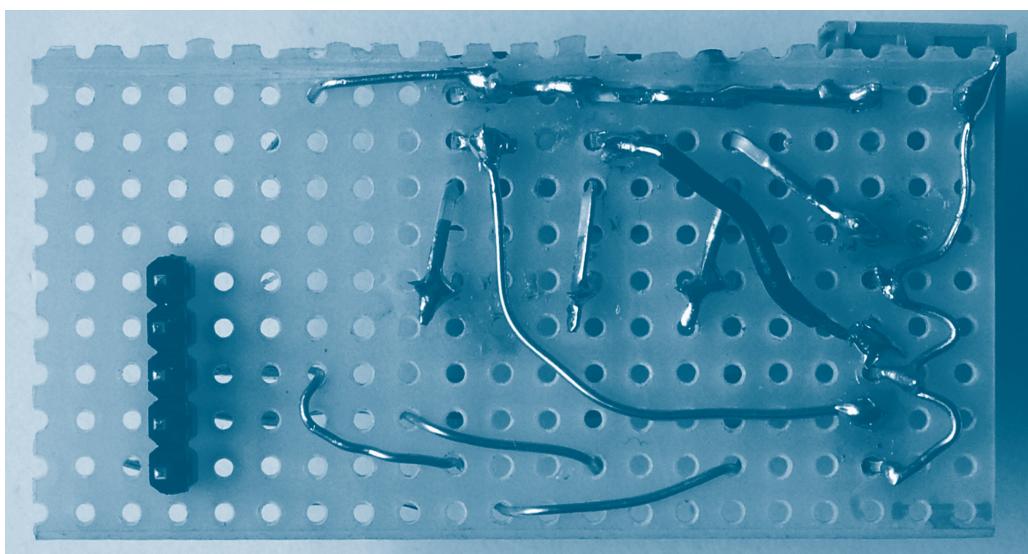


Figura 3-19 A parte debaixo da placa depois de completada.

O sketch usa as mesmas bibliotecas dos projetos dos Capítulos 2 a 5. Se, até agora, você não construiu qualquer desses projetos, consulte as instruções do Passo 6 (“Instale as Bibliotecas Open Accessory”), na seção *Construção* do Capítulo 2.

O sketch denomina-se “ch03_light_show” e faz parte do arquivo zip que contém todos os sketches do livro, podendo ser baixado de www.duinodroid.com. Veja detalhes no Capítulo 1.

Quando o sketch estiver instalado no microcontrolador, remova cuidadosamente o chip do mi-

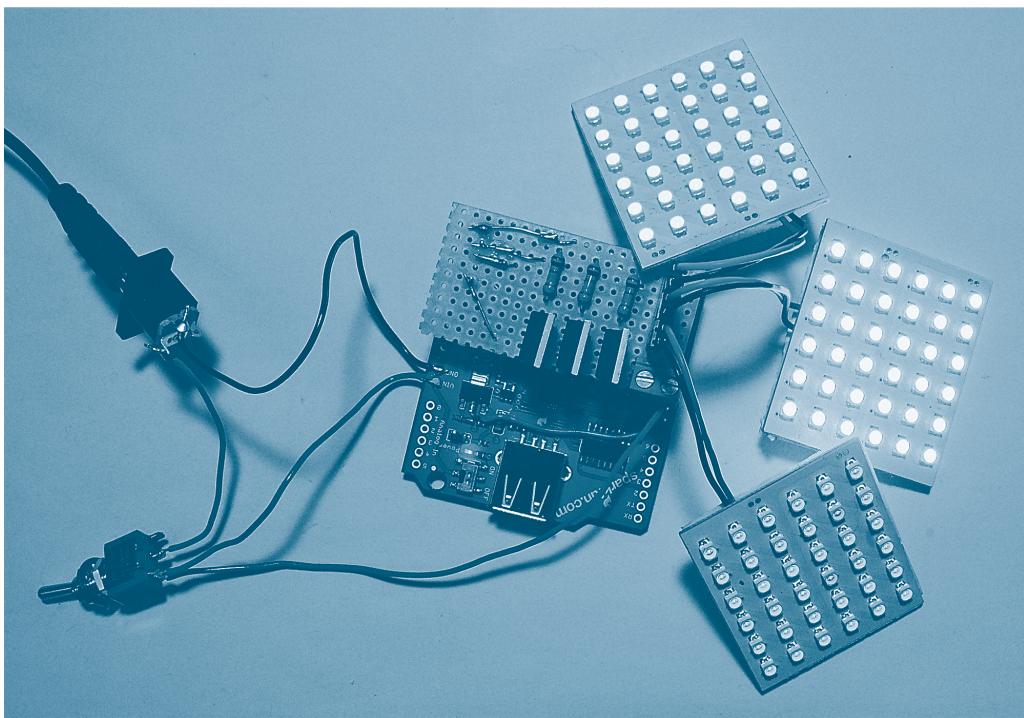


Figura 3-20 Conectando tudo.

crocontrolador, levantando cada extremidade um pouco de cada vez para que os pinos não se dobram. A seguir, encaixe-o no seu soquete na Base para Acessório Droid, verificando se está com a orientação correta.

» Passo 7: instale o aplicativo Android

Se o seu telefone celular tiver conexão com a Internet, você poderá passar por cima da instalação do aplicativo Android e esperar até que o software Open Accessory do celular solicite-o na primeira vez em que você conectar o acessório do show de luzes. Caso contrário, visite www.duinodroid.com e baixe o instalador APK.

Se ocorrer algum problema, consulte o Passo 10 na seção *Construção* do Capítulo 2.

» Passo 8: teste

Chegou o momento de testar o nosso projeto. O sketch inicia o acessório no modo de “teste”, de modo que, mesmo sem conectar o celular, podemos ver os LEDs em ação se ligarmos a fonte de 12V.

A seguir, podemos conectar o acessório ao celular. Esse projeto funciona também como uma estação para carregar a bateria do telefone, de modo que, enquanto as luzes estiverem piscando, o telefone também estará sendo carregado.

Quando o telefone for conectado, ele iniciará a execução do aplicativo “Show de Luzes Duino”.

» Passo 9: acondicione o projeto

O projeto pode ser instalado em uma caixa suficientemente grande para servir de base para os

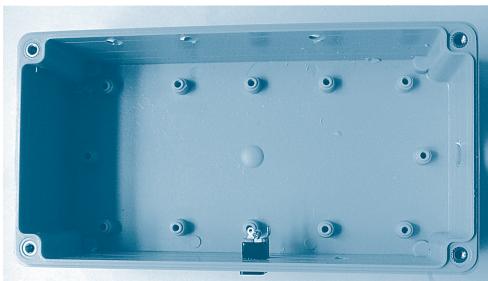


Figura 3-21 A caixa do projeto com os furos.

três painéis de LEDs autoadesivos. O celular é conectado por meio de um cabo USB dentro da caixa.

A Figura 3-21 mostra a caixa com os furos para a chave, o cabo USB e o jack de alimentação elétrica, além dos três furos para os painéis de LEDs.

Agora, fixe o shield em uma das pequenas colunas com orifício no fundo da caixa de plástico, usando um parafuso autoataraxante pequeno, e instale também a chave e o jack de alimentação elétrica (Figura 3-22).

Finalmente, podemos fixar os painéis de LEDs autoadesivos no lado da caixa e passar os fios através

dos furos e conectá-los aos soquetes da barra de pino fêmea.

Verifique se os LEDs estão conectados com a orientação correta e se os cabos são suficientemente longos para alcançar a barra de pino fêmea antes que você cole os painéis. Para isso, remova a película de proteção da superfície autoadesiva (Figura 3-23).

» Usando o projeto

O projeto tem três modos. O primeiro é o modo de teste que simplesmente faz os LEDs piscarem mesmo quando o acessório não está conectado ao telefone celular.

Quando você coloca nos modos de "Bargraph" ou "Beat", o dispositivo torna-se sensível ao som. Ele utiliza o microfone do celular e, portanto, responderá aos sons do Music Player e a sons externos. Se você quiser, poderá ligar a saída de áudio a um amplificador de som.

A sensibilidade da unidade é ajustada usando o controle deslizante de "gain" (ganho).

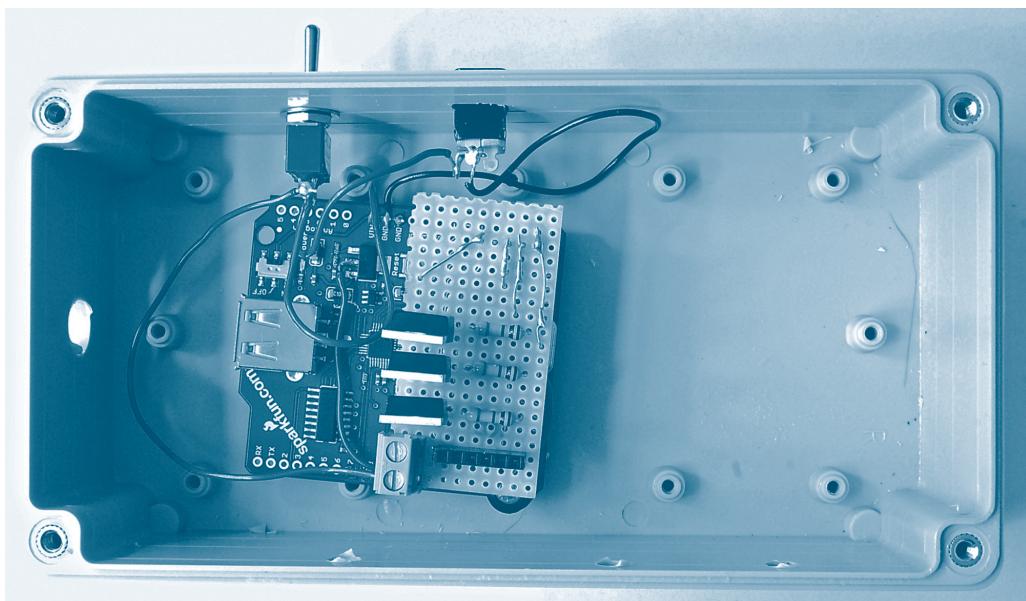


Figura 3-22 Instalando a chave e o jack de alimentação elétrica.

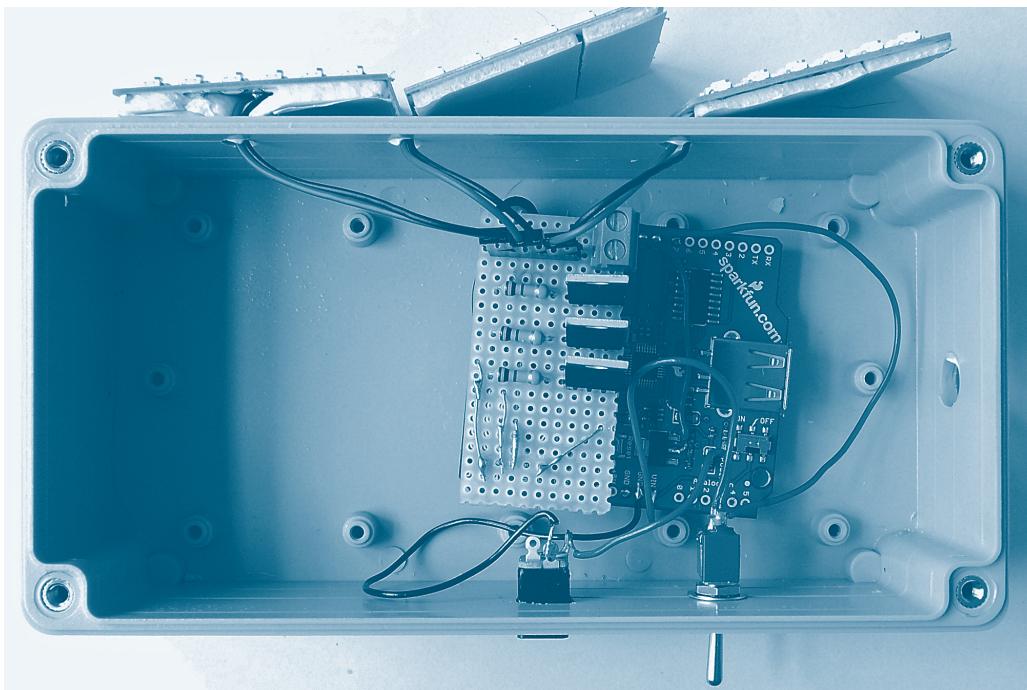


Figura 3-23 Fixando os painéis de LEDs.

» Teoria

O software deste e de todos os outros projetos deste livro são abertos (open source). Assim, você pode desenvolver os seus próprios acréscimos e aperfeiçoamentos. Gostaríamos muito de ter notícia de qualquer melhoria que você fizesse. Para entrar em contato conosco acesse www.duino-droid.com, onde você também encontrará todos os códigos-fontes para Android e Arduino.

Nesta seção, olharemos rapidamente o sketch de Arduino, mas, antes disso, veremos como utilizar MOSFETs para acender e apagar LEDs e discutiremos rapidamente a modulação por largura de pulso.

» MOSFETs

Um MOSFET (Metal Oxide Semiconductor Field Effect Transistor) é um tipo de transistor excelente para o chaveamento de correntes elevadas. Quan-

do está completamente cortado (desligado), ele tem uma resistência muito elevada e, quando está saturado (ligado), ele tem uma resistência muito baixa. Isso significa que, quando usado como uma chave que está completamente fechada ou aberta, ele gera muito pouco calor.

Como estamos usando modulação por largura de pulso (veja a próxima seção) no controle de brilho dos LEDs, essa técnica funciona muito bem – com uma dissipação de potência muito baixa na forma de calor.

Os transistores MOSFETs diferenciam-se dos transistores bipolares normais porque são controlados por tensão em vez de por corrente. Em outras palavras, o que determina se ele está ligado ou não é a tensão presente no terminal de “porta” de um MOSFET. É necessário que flua apenas uma corrente diminuta pela porta. Isso o torna ideal para fazer chaveamento a partir de dispositivos de baixa corrente como o Arduino.

Os MOSFETs que utilizaremos são denominados MOSFETs de nível lógico porque a tensão de porta necessária para ligar o MOSFET é inferior a 5V. Desse modo, o MOSFET poderá ser ligado e desligado diretamente a partir dos pinos do Arduino.

» Modulação por largura de pulso

Modulação por Largura de Pulso (PWM*) é uma técnica de controlar potência. No projeto do Capítulo 1, já utilizamos essa técnica para controlar a velocidade dos motores por meio da função “analogWrite”. Aqui, também a utilizaremos para controlar o brilho dos módulos de LEDs.

A Figura 3-24 mostra o sinal produzido por um pino PWM do Arduino.

O pino PWM oscila com uma frequência em torno de 500 Hz. O valor relativo de tempo em que o pino está em nível alto (HIGH) varia de acordo com o valor dado pela função “analogWrite”. Assim, examinando a Figura 3-24, se a saída estiver alta durante 5% do tempo, então o que estivermos acionando, seja lá o que for, receberá apenas 5% da potência total. Se, entretanto, a saída estiver com 5V durante

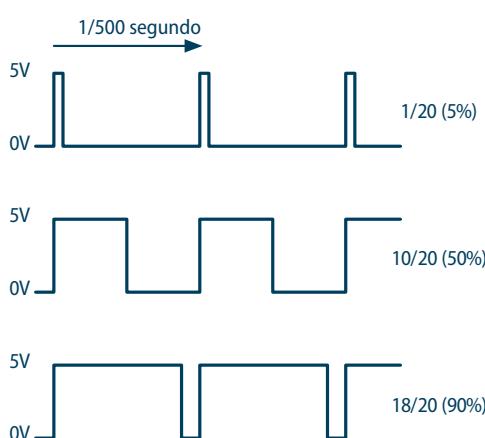


Figura 3-24 Modulação por largura de pulso.

* N. de T.: Pulse Width Modulation (PWM).

te 90% do tempo, então a carga receberá 90% da potência total.

Quando acionamos um motor com PWM, o motor não arranca e para 500 vezes por segundo. Devido à inércia de rotação, o motor recebe um impulso de intensidade variável a cada 1/500 de segundo. O efeito resultante é um controle suave da velocidade do motor.

Os LEDs podem responder muito mais rapidamente do que um motor, mas o efeito visível é o mesmo. Não podemos ver os LEDs acendendo e apagando com essa velocidade. Como resultado, o que percebemos é o brilho mudando de intensidade.

» O sketch de Arduino

O sketch deste projeto é muito simples. A diferença deste projeto em relação ao anterior (contador Geiger) é que neste a atividade maior do sketch está na recepção de comandos do acessório. No projeto do contador Geiger, a comunicação dava-se no sentido oposto.

Começamos pela inclusão das bibliotecas que serão necessárias. A biblioteca Max3124e já foi incluída. Ela faz parte da biblioteca USB host que foi instalada anteriormente.

A seguir, temos a constante “cycleTime”, que é utilizada quando o acessório está no modo de teste. Ela define o intervalo de tempo entre as mudanças de cor.

As três variáveis seguintes definem as intensidades do brilho para os LEDs vermelhos, verdes e azuis, podendo assumir valores entre 0 (apagado) e 255 (aceso ao máximo)

Depois de definir as variáveis, a linha que começa com “AndroidAccessory” estabelece a conexão entre o telefone e o Arduino da mesma forma que foi feito no sketch do contador Geiger.

A função “setup” inicia a comunicação serial e inicializa os pinos de saída.

A função “loop” chama “acc.isConnected” para verificar se chegaram mensagens utilizando a função

```

#include <Max3421e.h>
#include <Usb.h>
#include <AndroidAccessory.h>

#define redPin 5
#define greenPin 6
#define bluePin 3

#define cycleTime 10

int red = 0;
int green = 85;
int blue = 170;

boolean randomMode = true;

AndroidAccessory acc("Simon Monk",
    "DroidLightShow",
    "Light Show Accessory",
    "1.0",
    "http://www.duinodroid.com,
    "0000000012345678");

void setup()
{
    Serial.begin(9600);
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
    acc.powerOn();
}

void loop()
{
    byte msg[3];
    if (acc.isConnected())
    {
        int len = acc.read(msg, sizeof(msg), 1);
        if (len > 2 && msg[0] == 1) // acende o vermelho
        {
            red = msg[2];
        }
        if (len > 2 && msg[0] == 2) // acende o verde
        {
            green = msg[2];
        }
        if (len > 2 && msg[0] == 3) // acende o azul
        {
            blue = msg[2];
        }
        if (len > 2 && msg[0] == 4) // modo de teste ativado
    }
}

```

```

    {
        randomMode = true;
    }
    if (len > 2 && msg[0] == 5) // modo de teste desativado
    {
        randomMode = false;
    }
}
if (randomMode)
{
    changeColors();
}
showColors();
delay(cycleTime);
}

void changeColors()
{
    red++;
    if (red > 255) red = 0;
    green++;
    if (green > 255) green = 0;
    blue++;
    if (blue > 255) blue = 0;
}

void showColors()
{
    analogWrite(redPin, red);
    analogWrite(greenPin, green);
    analogWrite(bluePin, blue);
}

```

"acc.read". O primeiro byte da mensagem é o código do tipo de mensagem. O valor "1" é para a intensidade dos LEDs vermelhos, "2" para os LEDs verdes e "3" para os LEDs azuis. Os códigos "4" e "5" ativam e desativam o modo de teste, respectivamente.

A maior parte do restante do sketch trata do modo de teste. Nesse modo, o brilho de cada LED varia repetidamente de 0 a 255. Cada painel de LEDs tem brilhos iniciais independentes.

» O aplicativo Android

O aplicativo Android é grande demais para ser listado aqui por inteiro. Assim, se estiver inte-

ressado, você poderá baixar o código-fonte em www.duinodroid.com. A seguir, examinaremos as partes mais importantes do código.

A maior parte do aplicativo Android é similar ao que encontramos no aplicativo do contador Geiger. O leitor interessado poderá rever a descrição dessa parte na seção *Teoria* do capítulo anterior.

No que se refere ao sketch de Arduino, a diferença principal está no sentido da comunicação. No projeto deste capítulo, o Android é o dispositivo que envia mensagens ao Arduino. A seção do código que trata disso é a seguinte:

```
public void sendCommand(byte command,
    byte target, int value) {
    byte[] buffer = new byte[3];
    if (value > 255) value = 255;
    buffer[0] = command;
    buffer[1] = target;
    buffer[2] = (byte) value;
    if (mOutputStream!= null &&
        buffer[1]!= -1) {
        try {
            mOutputStream.write(buffer);
        } catch (IOException e) {
            Log.e(TAG, "write failed", e);
        }
    }
}
```

Esse código pode ser encontrado na classe DroidSoundDisplayActivity. A mensagem é cons-

truída na forma de um array de três bytes que é enviada à fila de saída USB.

Para amostrar o som, empregamos uma biblioteca GPL* bem popular que é utilizada no projeto Blinkendroid. As duas classes utilizadas podem ser encontradas no pacote “org.cbase.blinkendroid.audio”. Elas proporcionam um mecanismo eficiente que retorna uma seção dos dados amostrados. A seguir, essa seção é analisada pela classe “Visualizer” que determina quais mensagens devem ser enviadas ao Arduino.

» Resumo

Com isso, completamos o segundo de nossos projetos Open Accessory e o primeiro que utiliza o módulo “Base para Acessório Droid” que criamos. Esse módulo será a base dos projetos dos três capítulos seguintes.



» capítulo 4

Controle remoto de TV

Neste projeto, utilizando um telefone celular, mostraremos como construir um controle remoto para a sua TV por meio de um aplicativo Android.

Objetivos

- » Construir um acessório programável de controle remoto para TV baseado em Android e utilizando luz infravermelha
- » Aprender a utilizar circuitos integrados para transmitir e receber luz infravermelha
- » Entender como funcionam os controles remotos de infravermelho

O software de controle Android do projeto está mostrado na Figura 4-2.

O software permite a programação de oito botões que enviam comandos por IR (infrared, ou infravermelho).

Esses comandos também podem ser usados com outros aparelhos. O controle remoto pode ser “treinado” com outros controles remotos diferentes, funcionando como um “controle remoto universal”.

» Construção

Se você construiu a Base para Acessório Droid do Capítulo 3, então, na realidade, haverá pouca coisa a fazer, porque os poucos componentes podem ser colocados em uma placa perfurada pequena.

O diagrama esquemático do projeto está mostrado na Figura 4-3.

O projeto utiliza todos os pinos de entrada e saída da nossa Base para Acessório Droid. O pino D3 é utilizado para acionar o transmissor de IR (infravermelho). O chip de recepção de IR usa os pinos D6 e D2 para fornecer o GND e a tensão positiva do CI. O pino D5 recebe o sinal IR do CI receptor de IR.

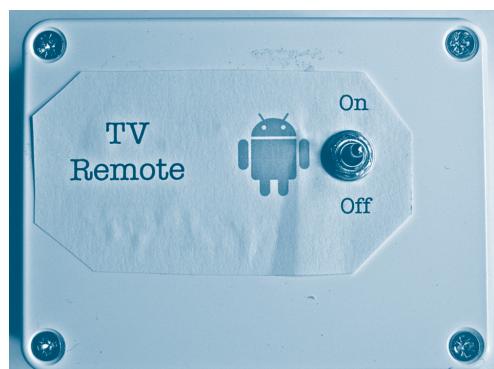


Figura 4-1 O acessório Android para controle remoto de TV.

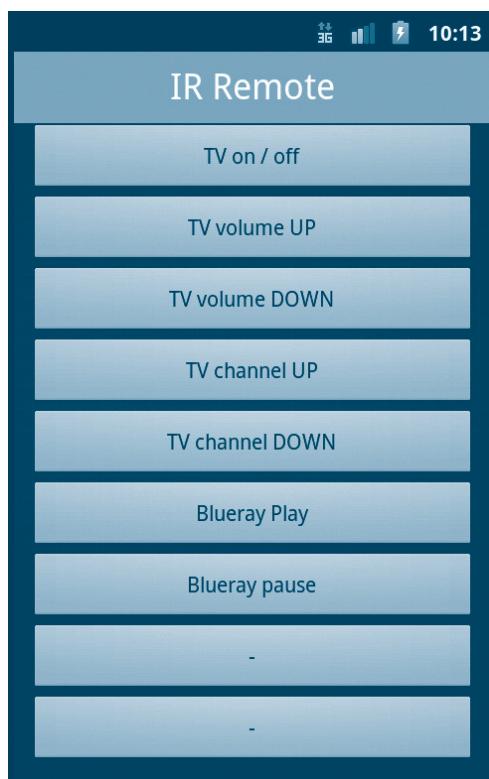


Figura 4-2 O aplicativo Android para controle remoto de TV.

» O que será necessário

Além do telefone celular Android, capaz de funcionar com acessórios (Android 2.3.4 ou posterior), e de todos os componentes necessários para construir a Base para Acessório Droid, você precisará dos componentes listados na tabela *Lista de Componentes*, a seguir.

Se você planeja construir o cabo USB que não permite carga de bateria, você precisará também de um resistor de $1\text{k}\Omega$ e um cabo USB de extensão. Veja o Capítulo 2 para os detalhes de construção desse cabo, que impedirá seu telefone celular de drenar as baterias dos seus acessórios.

Como alternativa para duração mais prolongada de carga, você pode usar um suporte para seis pilhas AA capaz de fornecer 9V.

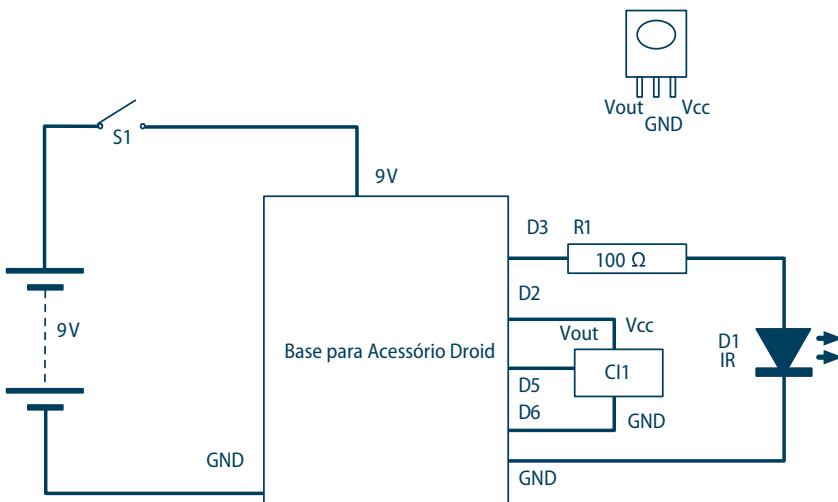


Figura 4-3 O diagrama esquemático.

Além desses componentes, você precisará também das seguintes ferramentas.

CAIXA DE FERRAMENTAS

- Ferramentas para soldar
- Uma furadeira elétrica com brocas
- Parafusos autoatarraxantes de diversos tamanhos
- Um computador para programar o Arduino
- Um cabo de conexão USB do tipo A-B

» Passo 1: construa uma base para Acessório Droid

Essa é a unidade descrita no Capítulo 3 que contém um shield USB host, um microcontrolador ATmega328, programado no seu Arduino, e alguns outros componentes.

» Passo 2: corte a placa perfurada no tamanho correto

A placa perfurada com trilhas deve ser cortada no tamanho correto. Para isso, a melhor maneira é fazer

sulcos na placa com um estilete e então quebrá-la ao longo desses sulcos apoiando-a na beira de uma mesa. Você também pode usar uma tesoura grande, mas o resultado não será tão bom (veja a Figura 4-4).

Não há necessidade de fazer cortes nas trilhas ou soldar fios de conexão. A disposição dos componentes está mostrada na Figura 4-5. Observe que a figura mostra o lado dos componentes e das trilhas (cobre). Observe também que a barra de pino macho foi colocada no lado sem cobre da placa, com os pinos orientados de tal forma que possam ser encaixados na Base para Acessório Droid. Desse modo, o lado de cima da placa vai para baixo em direção ao acessório Droid, no qual se encaixa. O LED de IR e o chip de recepção IR estão voltados para o lado oposto à barra de pino macho (para à direita na figura).

» Passo 3: solde os componentes

Apenas os três componentes e a barra de pino macho precisam ser soldados na placa.

Verifique se o LED de IR está com a orientação correta. O terminal longo positivo é o que deve ser conectado a um dos terminais do resistor (Figura 4-6).

LISTA DE COMPONENTES

Componente	Quantidade	Descrição	Fornecedor
Base para Acessório Droid	1	Veja o Capítulo 3 para os componentes e as instruções de construção	
R1	1	Resistor de filme metálico de 100Ω e 1/2 W	Farnel: 9339760
R	Opcional	Resistor de filme metálico de $1k\Omega$ e 1/2 W	Farnel: 9339779
D1	1	LED transmissor de IR de 5mm, para 940nm	Farnell: 1020634
Cl1	1	Cl receptor de IR, para 940nm	Farnell: 4142822
Clip para bateria	1	Clip para bateria de 9V	Farnell: 1183124
Placa perfurada com trilhas	1	Oito trilhas de seis furos	Farnell: 1201473
Barra de pino macho	1	Barra de pino macho de seis vias	Farnell: 1097954
Caixa	1	Caixa de plástico	Lojas locais

» Passo 4: faça a fiação

A Figura 4-7 mostra a placa perfurada conectada à base.

Você também deverá soldar uma chave e um clip de bateria à Base para Acessório Droid, como mostrado na Figura 4-7.

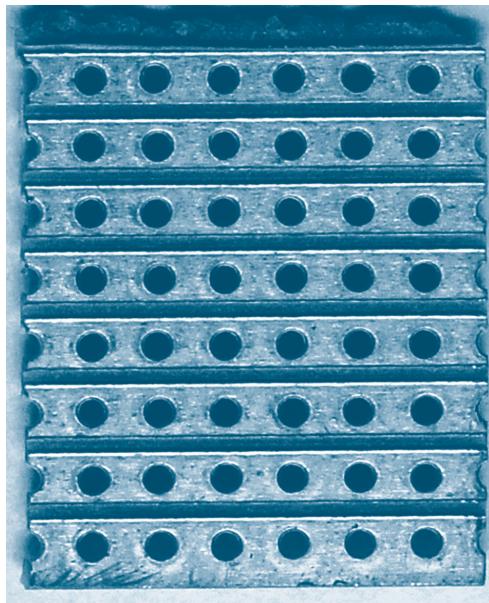


Figura 4-4 A placa perfurada cortada no tamanho correto.

» Passo 5: instale o sketch do Arduino

Para instalar o sketch no microcontrolador, primeiramente vamos programar o chip. Após, ele será retirado com cuidado da placa do Arduino e encaixado no soquete da Base para Acessório Droid.

Este sketch usa somente bibliotecas Android. Consulte as instruções do Passo 6 do Capítulo 2 sobre a instalação das bibliotecas Android no software do seu Arduino.

Portanto, instale o sketch (`ch04_tv_remote`) na placa do Arduino. A seguir, com a placa desligada, remova cuidadosamente o Cl do microcontrolador e encaixe-o no soquete da Base para Acessório Droid, verificando se está com a orientação correta.

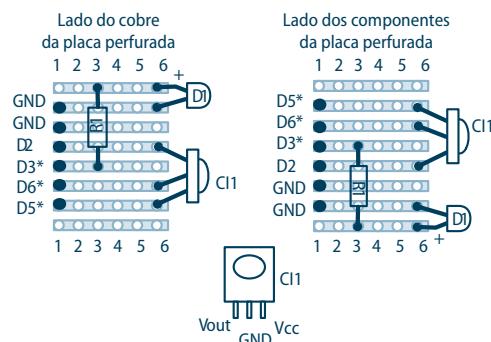


Figura 4-5 A disposição dos componentes na placa.

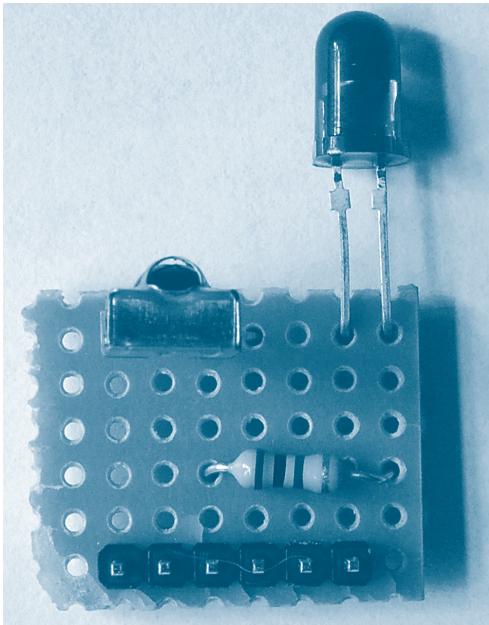


Figura 4-6 A placa perfurada completa.

» Passo 6: instale o aplicativo Android

Se seu telefone celular tiver conexão com a Internet, então, na primeira vez que conectar o acessório para controle remoto de TV, você poderá pular a instalação do aplicativo Android e esperar até que o software Open Accessory do celular solicite-a. Caso contrário, acesse www.duinodroid.com e baixe o instalador APK deste projeto.

Se ocorrer algum problema, consulte o Passo 10 do Capítulo 2.

» Passo 7: teste

Sempre é uma boa ideia testar o projeto antes de acondicioná-lo em uma caixa. Para isso, conecte tudo, como mostrado na Figura 4-7.

Ligue a chave. O celular iniciará a execução do aplicativo “Droid TV Remote” ou, se ainda não estiver

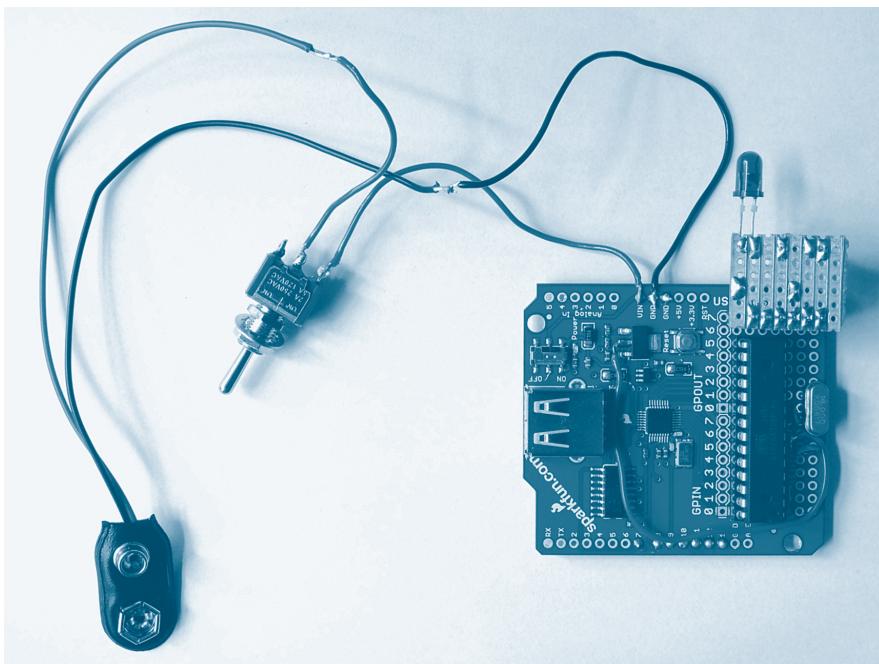


Figura 4-7 A placa perfurada conectada à base.

instalado, aparecerá no display o URL do local de onde pode ser baixado.

Observe também que, neste projeto, a Base para Acessório Droid tem um clip de bateria de 9V ligado aos cabos de alimentação.

Neste projeto, o celular Android faz pouco mais do que servir de teclado, passando comandos ao Arduino. O treinamento do controle remoto e o armazenamento dos códigos são feitos no Arduino. Começaremos pelo treinamento do controle remoto. Para isso, utilizaremos os comandos enviados por um de nossos controles remotos comuns. Primeiro, selecionamos o botão que queremos programar (apertando-o). A seguir, tecemos o botão “Menu” e selecionamos a opção “Program” (programar). Depois, levamos o controle remoto para próximo do receptor de IR e pressionamos um botão. Se o código for recebido, o aplicativo avisará que o código foi salvo.

Sempre que você programar o controle remoto desse modo, o último botão apertado será programado.

Agora, aponte o seu controle remoto para o dispositivo que era controlado pelo controle remoto original e aperte o primeiro botão do controle remoto. Deveremos obter o mesmo efeito que tínhamos com o controle remoto original.

Como primeira escolha, o comando para colocar o dispositivo em modo “standby” é sempre uma boa ideia para começar o treinamento. Observe que o transmissor de IR não tem o mesmo brilho que um controle remoto comum. Assim, o nosso controle remoto Android de TV exigirá que você esteja mais próximo do dispositivo a ser controlado do que quando você usa o controle remoto original.

Se você olhar o LED de IR através de uma camcorder ou uma câmera digital, você poderá ver o LED funcionando. Assim, se você não tiver certeza se o controle remoto está enviando códigos, você poderá utilizar esse método para fazer uma verificação.

» Passo 8: acondicione o projeto

O processo de acondicionamento deste projeto é muito semelhante ao dos dois projetos anteriores. Primeiro, disponha os componentes na caixa (Figura 4-8) e, em seguida, verifique quais furos serão necessários levando em conta as posições e os diâmetros.

Serão necessários orifícios para:

- O cabo USB
- Os parafusos da base da caixa para fixar a placa no lugar
- A chave
- O receptor e transmissor de IR

» Usando o projeto

Você poderá alterar o nome de um botão qualquer se mantê-lo apertado por um tempo (um ou dois segundos). Isso abrirá a tela de configurações (settings) onde o nome ou rótulo (label) do botão poderá ser editado (Figura 4-9).

» Teoria

Nesta seção, veremos como funciona o controle remoto de IR (infravermelho).

» Controles remotos de IR

Os controles remotos de infravermelho enviam comandos na forma de uma sequência de pulsos. Esses pulsos são modulados por uma frequência portadora em torno de 40kHz. A Figura 4-10 mostra a forma de onda de uma mensagem típica.

O comprimento dos códigos variam entre 12 e 32 bits. Geralmente, um “1” é indicado pelo que se denomina uma “marca” (LED transmitindo) e um “0”,

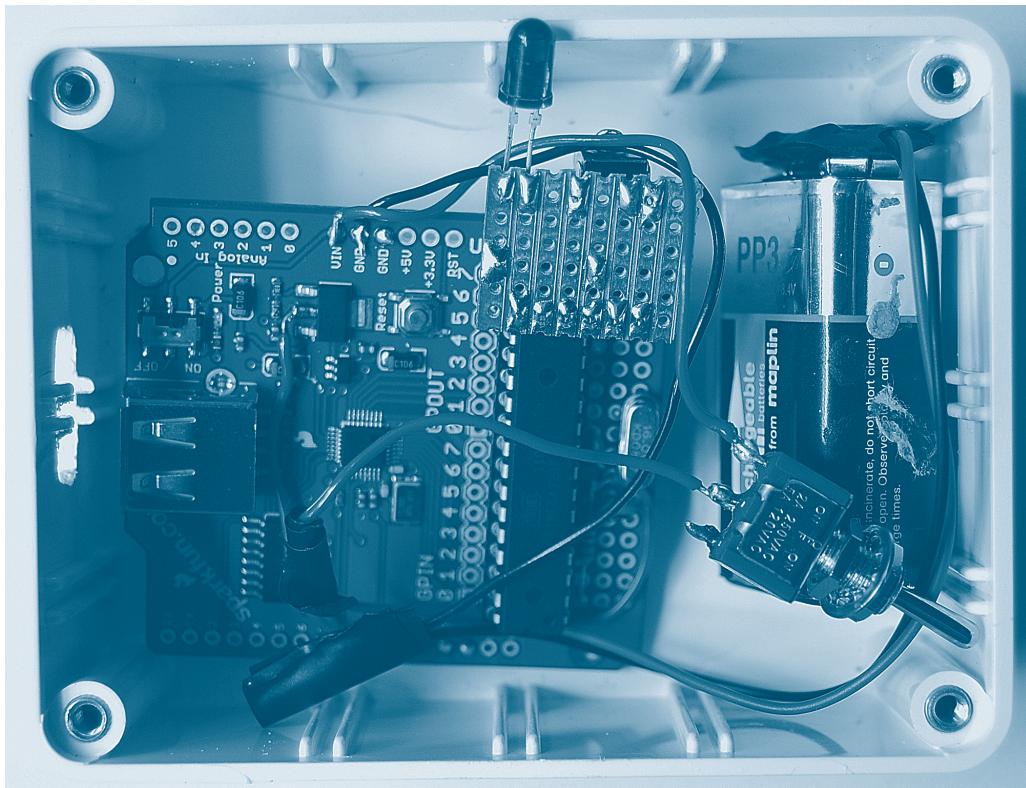


Figura 4-8 Disposição dos componentes na caixa.

por um espaço (LED apagado). Frequentemente, o controle remoto repete a mensagem diversas vezes quando um botão é apertado.

Um padrão, ou norma, é definitivamente um conceito em que menos é mais. Infelizmente, no mundo dos controles remotos de IR, há muitos padrões. Por isso, é mais prático gravar os sinais

que você deseja enviar do que tentar sintetizá-los sabendo a marca e o modelo do aparelho que é comandado por seu controle remoto. Simplesmente há muitas representações diferentes para os mesmos códigos.

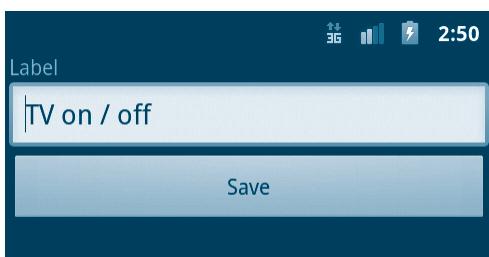


Figura 4-9 Editando o nome de um botão.

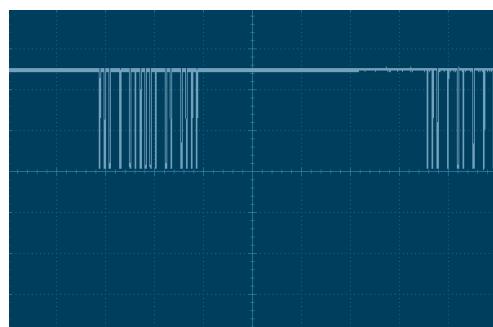


Figura 4-10 O traçado em osciloscópio de uma mensagem infravermelha.

» Resumo

Agora que podemos controlar a nossa TV usando o telefone celular, o nosso próximo projeto será criar outro dispositivo Open Accessory. Desta vez, será um registrador de temperatura que grava temperaturas e, em seguida, as transmite sem fio para o site Pachube de telemetria.



» capítulo 5

Registrador de temperatura

Neste capítulo, mostraremos o projeto de construção de um registrador de temperatura. Além de exibir no telefone a temperatura atual e as leituras anteriores, os valores são enviados ao site Pachube, por onde podem ser acessados.

Objetivos

- » Usar o Open Accessory para construir um registrador de temperaturas que usa um celular Android para enviar leituras a um servidor Web
- » Aprender a utilizar circuitos integrados sensores de temperatura
- » Conhecer o site Pachube
- » Conhecer o trecho do aplicativo Android responsável por transmitir registro de temperaturas e um servidor Web – Pachube

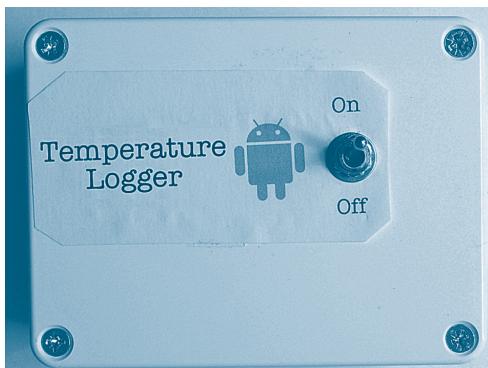


Figura 5-1 O acessório Android registrador de temperatura.

Uma característica interessante deste registrador de temperatura é que utiliza o celular Android para também transmitir as leituras feitas por minuto ao site



Figura 5-2 O aplicativo Android registrador de temperatura.

Pachube*. Nesse site, você poderá fazer todo o tipo de análise e de confecção de gráficos (Figura 5-3).

» Construção

Se você já construiu a Base para Acessório Droid do Capítulo 3, então, na realidade, haverá pouca coisa para fazer. De fato, há um mínimo de soldas. A tarefa maior será provavelmente acondicionar o projeto dentro de uma caixa.

O diagrama esquemático do projeto está mostrado na Figura 5-4.

O projeto utiliza apenas três dos pinos de entrada e saída da Base para Acessório Droid. O pino D2 é usado como entrada digital de um sensor de temperatura e o D6, para fornecer tensão ao sensor.

O sensor de temperatura usado neste projeto é um dispositivo muito interessante, combinando um sensor de temperatura e um processador que envia a temperatura em forma digital. Isso permite que ele seja instalado a uma distância do Arduino, sem que o comprimento do fio altere a exatidão das medidas de temperatura.

» O que será necessário

Você precisará de um telefone celular Android capaz de funcionar com acessórios (Android 2.3.4 ou posterior) e de todos os componentes necessários para construir a Base para Acessório Droid. Além disso, você precisará apenas de um sensor de temperatura, de um resistor e de alguns conectores. A tabela *Lista de Componentes*, a seguir, mostra uma lista completa do que você precisará.

* N. de T.: Em 2011, a empresa LogMeIn adquiriu o Pachube (www.pachube.com), e alterou o nome para Cosm (www.cosm.com), mantendo os serviços que vinham sendo prestados, ainda que sob outra forma. Em maio de 2013, a LogMeIn reestruturou o Cosm, denominando-o Xively (xively.com) e dando um caráter comercial a seus serviços. O Xively oferece recursos para projetar e desenvolver dispositivos visando transformá-los em produtos comerciais.

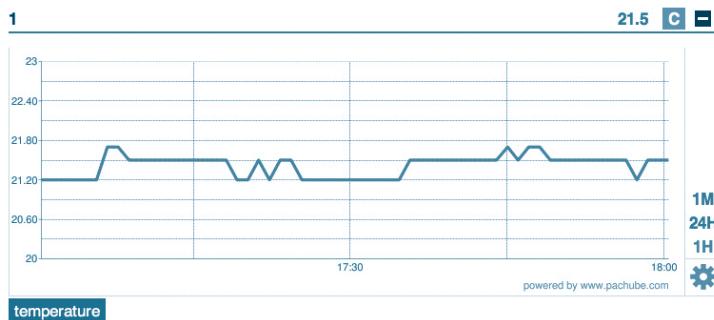


Figura 5-3 O site Pachube.

Os únicos componentes eletrônicos deste projeto são o circuito integrado sensor de temperatura e um resistor de pull-up.

Se você planeja construir o cabo USB que não permite carga de bateria, você precisará também de um resistor de $1\text{k}\Omega$ e um cabo USB de extensão. Veja o Capítulo 2 para os detalhes de construção desse cabo, que impedirá o seu telefone celular de drenar as baterias dos seus acessórios.

Como alternativa para uma duração de carga mais prolongada, você pode usar um suporte para seis pilhas AA capaz de fornecer 9V.

Além desses componentes, você precisará também das seguintes ferramentas.

CAIXA DE FERRAMENTAS

- Ferramentas para soldar
- Uma furadeira elétrica com brocas
- Parafusos autoatarraxantes
- Um computador para programar o Arduino
- Um cabo de conexão USB do tipo A-B

» Passo 1: construa uma Base para Acessório Droid

Esse é o módulo descrito no Capítulo 3. É constituído de um shield USB host, um microcontrolador ATMega328, programado no Arduino, e alguns ou-

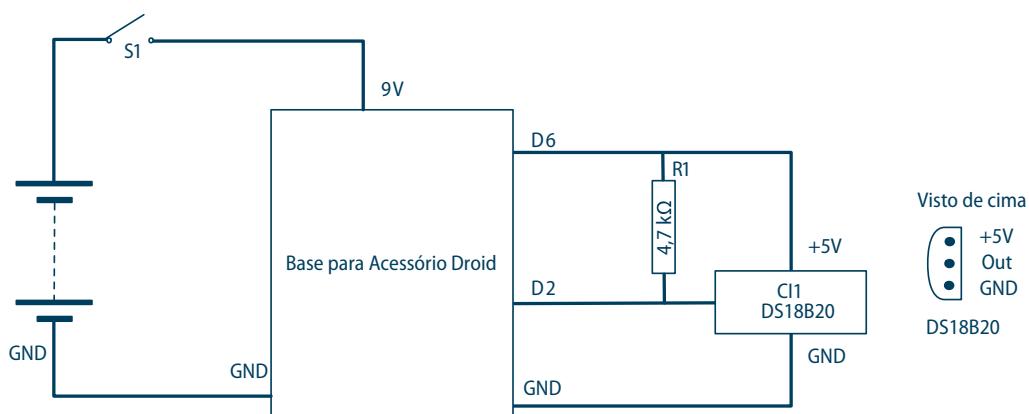


Figura 5-4 O diagrama esquemático.

LISTA DE COMPONENTES

Componente	Quantidade	Descrição	Fornecedor
Base para Acessório Droid	1	<i>Veja o Capítulo 3 para os componentes e as instruções de construção</i>	
R1	1	Resistor de filme metálico de $4,7\text{k}\Omega$ e 1/2 W	Farnel: 9340629
R	Opcional	Resistor de filme metálico de $1\text{k}\Omega$ e 1/2 W	Farnel: 9339779
Cl1	1	Circuito integrado sensor de temperatura – DS18B20	SparkFun: SEN-00245
Conector KRE	1	Conector KRE de três vias	Farnell: 1641933
Clip para bateria	1	Clip para bateria de 9V	Farnell: 1183124
Caixa	1	Caixa de plástico	Lojas locais

etros componentes. Siga as instruções do Capítulo 3 para construí-lo.

para Acessório Droid (Figura 5-6). Verifique se o Cl de três pinos pequeno está conectado corretamente. O lado chato do chip deve ficar para cima.

» Passo 2: instale os componentes no conector KRE

Como há poucos componentes neste projeto, poderemos fixar o Cl sensor de temperatura e o resistor em um conector KRE (Figura 5-5). Este conector, por sua vez, pode ser encaixado diretamente na barra de pino fêmea (com seis soquetes) da Base

» Passo 3: instale o sketch do Arduino

Para instalar o sketch no microcontrolador, primeiro vamos programar o chip e, a seguir, retirá-lo com cuidado da placa do Arduino e encaixá-lo no soquete da Base para Acessório Droid.

Este sketch utiliza duas bibliotecas para o módulo de temperatura (OneWire e DallasTemperature).

O procedimento de instalação de uma biblioteca é o mesmo de todas as bibliotecas (veja o Capítulo 1). Você pode baixar as bibliotecas OneWire e Dallas em um único arquivo zip de www.hacktronics.com/code/OneWire-v2.zip.

A seguir, instale o sketch (ch05_temp_logger) na placa do Arduino. Após, com a placa desligada, remova cuidadosamente o Cl do microcontrolador e encaixe-o no seu soquete na Base para Acessório Droid, verificando se está com a orientação correta. A melhor maneira de remover o Cl é, usando uma chave de fenda pequena, erguê-lo com cuidado, levantando cada extremidade um pouco de cada vez.

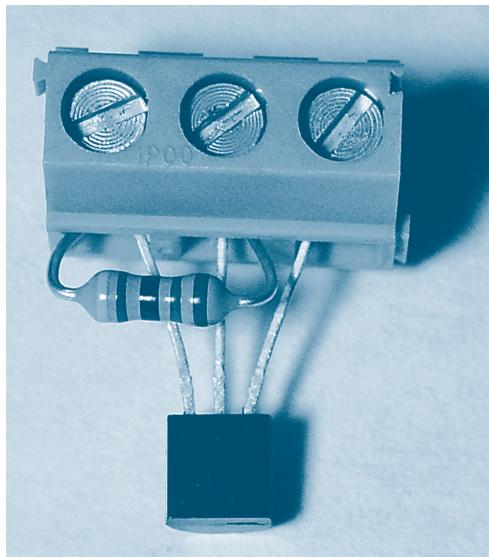


Figura 5-5 O conector KRE.

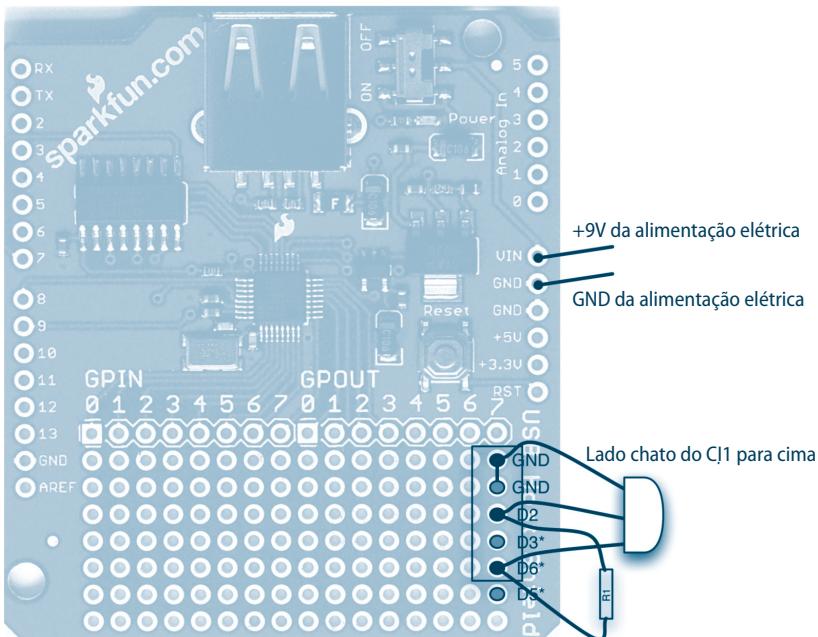


Figura 5-6 O diagrama de fiação.

» Passo 4: instale o aplicativo Android

Se seu telefone celular tiver conexão com a Internet, você poderá pular a instalação do aplicativo Android. Para isso, na primeira vez em que você conectar o acessório registrador de temperatura, espere até que o software Open Accessory do celular solicite a instalação. Caso contrário, visite www.duinodroid.com e baixe o instalador APK.

Se ocorrer algum problema, consulte o Passo 10 do Capítulo 2.

» Passo 5: teste

Nesse projeto, a Base para Acessório Droid tem um clip de bateria de 9V ligado aos cabos de alimentação.

Sempre é uma boa ideia testar o projeto antes de colocar tudo junto em uma caixa. Desse modo, conecte a bateria e o celular (Figura 5-7).

Ligue a chave. O celular iniciará a execução do aplicativo “Droid Temp Logger” (Registrador de Temperatura Droid) ou, se não ainda estiver instalado, aparecerá no display o URL de onde pode ser baixado.

» Passo 6: acondicione o projeto

O processo de acondicionamento deste projeto é muito semelhante ao dos três projetos anteriores. Primeiro, disponha os componentes na caixa (Figura 5-8) e, em seguida, verifique quais furos serão necessários levando em conta as posições e os diâmetros.

Serão necessários orifícios para:

- O cabo USB
- Os parafusos da base da caixa para fixar a placa no lugar
- A chave
- Um orifício próximo do sensor de temperatura

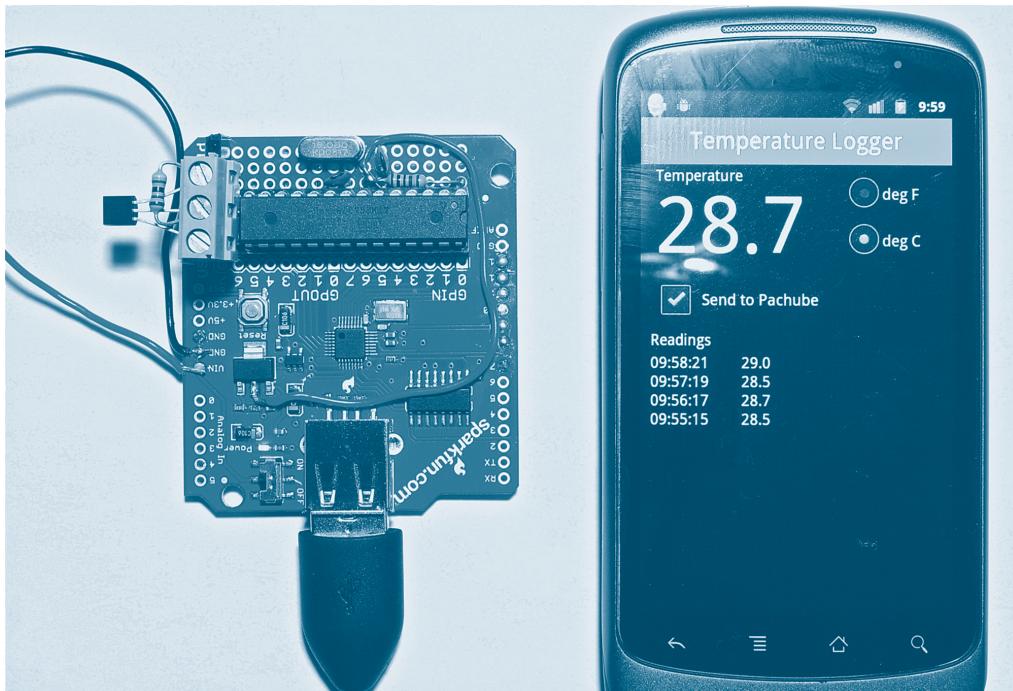


Figura 5-7 Teste.

» Usando o projeto

Para tirarmos o máximo desse projeto, precisamos criar uma conta no site Pachube*. Isso permitirá que as leituras de temperatura que chegam ao celular Android sejam enviadas automaticamente para a conta no Pachube.

Comece criando uma nova conta Pachube no site www.pachube.com. Essa conta é gratuita para

membros da classe “Basic” (básica), o que será ótimo para os nossos propósitos.

Após criar a conta, você deverá clicar em “Create a Feed” (Criar um Feed) para dar um destino aos nossos dados. A Figura 5-9 mostra isso. O único campo obrigatório é um nome para o “feed”. Você também deverá criar um “stream”. Para isso, clique em “add stream” (adicone stream) e forneça os detalhes mostrados na Figura 5-10. Após, clique em “Save” para salvar. Agora, você verá um resumo do seu novo “feed”.

Anote o número – nesse caso, 29933. Você deverá fornecer esse número no seu aplicativo Android. A outra informação que você precisará é a chave pessoal API da sua conta. Você poderá encontrar essa chave clicando em “My API Keys” (minhas chaves API) no lado direito do site. Essa chave será longa, algo como B74W43hXupkN6J_rEsM1exa30X-6TjP8cNFFDYS2i6r.

Você também necessitará fornecer a chave ao aplicativo Android, mas será mais fácil se você entrar

* N. de T.: Nesta seção, o autor descreve como o registrador de temperatura, depois de exibir as temperaturas no celular, pode enviar os dados ao site Pachube. Como foi comentado na nota de rodapé anterior, o site Pachube não existe mais na forma descrita neste texto. Por essa razão, as informações desta seção devem ser vistas como uma referência inicial para guiar o leitor no novo site (xively.com), substituto do Cosm e do Pachube. Com um objetivo comercial, o Xively propõe-se a oferecer recursos para projetar e desenvolver dispositivos que em seguida serão transformados em produtos comerciais. Depois de acessar a página inicial do site (xively.com), você poderá acessar a página https://xively.com/dev/tutorials/xively_develop/ para conhecer melhor a nova orientação e a forma como você poderá conectar os seus dispositivos.

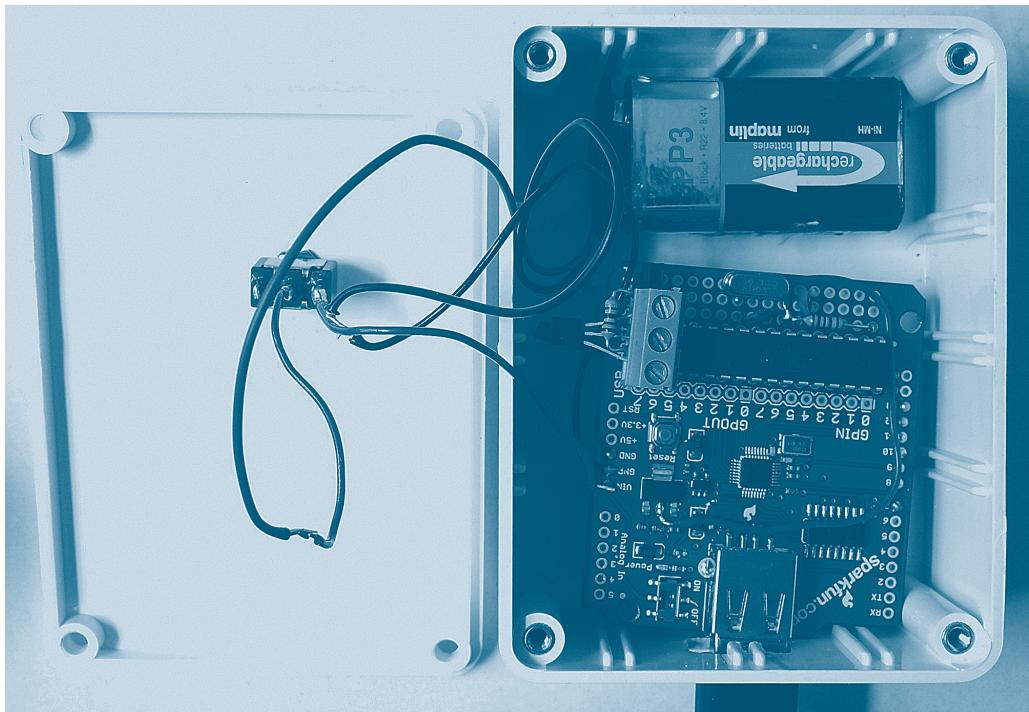


Figura 5-8 Disposição dos componentes na caixa.

```
private void sendPachubeReading() {  
    SharedPreferences settings =  
        mHostActivity.getSharedPreferences(SettingsActivity.PREFS_NAME, 0);  
    String pachubeFeed = settings.getString("PACHUBEFEEDID", null);  
    String pachubeKey = settings.getString("PACHUBEKEY", null);  
    if (pachubeFeed == null || pachubeKey == null) {  
        alert("Go to Settings and add entries for Parachube Feed ID and Key.");  
    }  
    else {  
        try {  
            Pachube p = new Pachube(pachubeKey);  
            Feed f = p.getFeed(Integer.parseInt(pachubeFeed));  
            Double reading = Double.parseDouble(mTemperature.getText().toString());  
            f.updateDatastream(1, reading);  
        }  
        catch (PachubeException e) {  
            alert(e.errorMessage);  
        }  
    }  
}
```

The screenshot shows the 'Create New Feed' page on the Pachube website. The feed name is 'Evil Genius Cat Bed Temperatures'. Below it is a text area for a description. A 'TAGS' field contains 'Comma-separated descriptive keywords for this Feed'. A 'LOCATION NAME' field has 'Name of location'. A 'LOCATION MAP' section includes a world map with a marker set on Brazil. Below the map are fields for 'LATITUDE', 'LONGITUDE', and 'ELEVATION' (set to 'Metres'). Under 'EXPOSURE' and 'DISPOSITION', 'INDOOR' is selected. The 'DOMAIN' is set to 'Physical'. A 'WEBSITE' field says 'This will be publicly visible'. Under 'FEED STATUS', 'PUBLIC' is selected. In the 'DATASTREAMS' section, there is one entry for 'temperature' with unit 'C'. At the bottom are 'Cancel' and 'Save' buttons.

Figura 5-9 Criando um stream no Pachube.

na sua conta Pachube por meio do navegador do seu celular e, em seguida, copiar a chave e colá-la.

Agora que estamos com tudo pronto, conecte o acessório para dar início ao aplicativo "TemplLogger" (registrar de temperatura), clique no botão "Menu" do celular e selecione "Configurações".

Forneça o identificador ID do "Feed", como está mostrado na Figura 5-11.

Após entrar com os dados, clique em "Save" para salvá-los e, em seguida, use o botão "Back" (voltar) do telefone para retornar à tela principal.

Clique na "checkbox" para entrar no Pachube. Deixe o registrador de temperatura funcionando por

The screenshot shows the 'Edit Feed' page for 'Evil Genius Cat Bed Temperatures'. It includes fields for 'TAGS', 'LOCATION NAME', 'LATITUDE', 'LONGITUDE', 'ELEVATION', 'EXPOSURE', 'DISPOSITION', and 'DOMAIN' (set to 'Indoor' and 'Physical'). Under 'FEED FORMATS', 'JSON' is selected. A 'WEBSITE' field contains '/feeds/29933'. A 'CONTACT EMAIL' field is empty. In the 'DATASTREAMS' section, there is one entry for 'temperature'. At the bottom are 'Pachube home', 'Plans & Pricing', 'About us', 'Find Feeds', 'Blog', 'Support', and 'Developer Docs' links.

Figura 5-10 O resumo do "feed".

The screenshot shows a mobile device screen with a dark blue background. It displays the 'Pachube Feed ID' as '29933' and the 'Pachube API Key' as 'B74W43hXupkN6J_rEsM1exa30X6TjP8cNFFDYs2i6r'. There is a 'Save' button at the bottom.

Figura 5-11 Configurando o identificador ID e a chave para o Pachube.

um tempo. Após, você poderá ir até o site do Pachube e ver os seus dados (Figura 5-12).

>> Teoria

A parte interessante deste projeto é a capacidade de registrar temperaturas e transmiti-las remotamente sem fio para um servidor de Web. Nesta se-

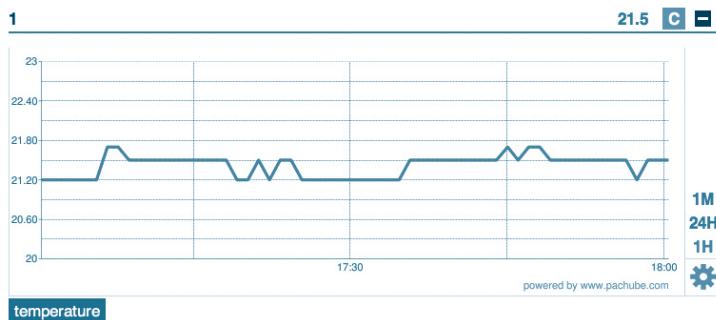


Figura 5-12 O gráfico de temperatura.

ção, examinaremos o trecho do aplicativo Android responsável por isso.

» O aplicativo Android

A estrutura do aplicativo Android é a mesma utilizada em todos os outros projetos Open Accessory deste livro. Para compreender os detalhes de como isso funciona, consulte algum dos outros capítulos envolvendo Open Accessory.

O envio de dados para o Pachube é executado por uma biblioteca denominada JPachube (code.google.com/p/jpachube). Ela se encarrega de todas as solicitações HTTP necessárias para enviar dados ao Pachube. O código que está mostrado na página 69 foi retirado da classe InputController. O código completo do aplicativo está disponível em www.duinodroid.com.

O código é bem claro. Primeiro, obtemos o ID do Feed e a chave Pachube no sistema de preferências. Se não houver valores, então será solicitado que o usuário vá à página de configurações e faça as alterações necessárias.

Após, as classes Pachube e Feed são usadas para enviar os dados. No caso de ocorrer um erro, uma mensagem será apresentada ao usuário na forma de um alerta.

» Resumo

Agora, temos outro projeto Open Accessory em nossas mãos. No próximo capítulo, passaremos da medição de temperatura para a medição de distância usando um medidor ultrassônico.



» capítulo 6

Medidor de distância ultrassônico

Neste capítulo, apresentaremos o projeto de construção de um medidor de distância ultrassônico. Buscamos inspiração em pessoas envolvidas com negócios imobiliários. É extremamente difícil para essas pessoas fazer medições de salas e terrenos usando uma fita métrica tradicional e, ao mesmo tempo, manipular um celular. Por isso, é muito comum essas pessoas utilizarem um medidor ultrassônico.

Objetivos

- » Criar um dispositivo Open Accessory para medir distâncias
- » Analisar o funcionamento de medidores ultrassônicos de distância
- » Aprender a fazer a interface do medidor de distância com um sketch de Arduino

Nós nos inspiramos nos agentes imobiliários para construir o projeto final de acessório Android desta primeira parte do livro. Trata-se de um medidor de distância ultrassônico (Figura 6-1).

O software de controle Android do projeto está mostrado na Figura 6-2.

Na prática, qualquer projeto pode ser melhorado com a inclusão de um laser. Por essa razão, decidimos acrescentar um laser ao nosso acessório para apontar a direção de medição da distância.

ALERTA Você deve tomar certas precauções em relação ao uso de lasers: (1) Nunca aponte um laser para os olhos de alguma pessoa ou animal. (2) Para verificar se ele está aceso, resista à tentação de observá-lo diretamente. Em vez disso, aponte-o para um papel ou alguma outra superfície colorida.

» Construção

Se você já construiu a Base para Acessório Droid do Capítulo 3, então haverá realmente muito pouca coisa a fazer. De fato, quase não há soldas. A tarefa maior será provavelmente acondicionar o projeto dentro de uma caixa.

O diagrama esquemático do projeto está mostrado na Figura 6-3.



Figura 6-1 O acessório Android medidor de distância ultrassônico.



Figura 6-2 O aplicativo Android medidor de distância ultrassônica.

O projeto utiliza todos os quatro pinos de entrada e saída da nossa Base para Acessório Droid. Ele usa um pino para fornecer a tensão de +5V ao módulo ultrassônico, dois pinos para os sinais de "trigger" (gatilho) e "echo" (eco) e o quarto pino para controlar a ativação do laser.

» O que será necessário

Você precisará de um telefone celular Android capaz de funcionar com acessórios (Android 2.3.4 ou posterior) e de todos os componentes necessários para construir a Base para Acessório Droid. Além

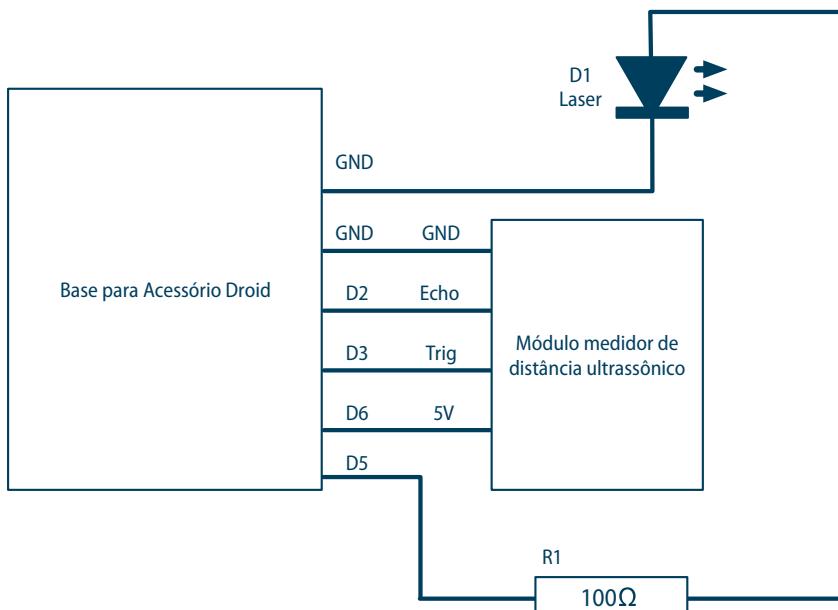


Figura 6-3 O diagrama esquemático.

disso, você precisará apenas de um módulo medidor de distância ultrassônico, um resistor e alguns conectores. A tabela *Lista de Componentes*, a seguir, mostra um lista completa do que você precisará.

O componente-chave deste projeto é o módulo medidor de distância ultrassônico. A Figura 6-4 mostra o módulo que foi utilizado pelo autor. Esse módulo requer apenas quatro pinos para funcionar. Além de GND (terra ou 0V) e 5V, o módulo tem uma entrada de “trigger” (gatilho) e uma saída de “echo”

(eco). A folha de especificações desse componente diz que você deve aplicar um pulso de 10 microssegundos à entrada de “trigger” e, em seguida, verificar quanto tempo decorre antes que o pino de “echo” indique que o som retornou. Na seção *Teoria*, mais adiante, você verá detalhes sobre o funcionamento do medidor de distância ultrassônico.

A construção será mais fácil se você encontrar um módulo similar ao usado aqui. As interfaces desses dispositivos são todas muito semelhantes e, se você

LISTA DE COMPONENTES

Componente	Quantidade	Descrição	Fornecedor
Base para Acessório Droid	1	Veja o Capítulo 3 para os componentes e as instruções de construção	
Módulo ultrassônico medidor de distância	1	Veja a descrição no texto	
R1	1	Resistor de filme metálico de 100Ω e 1/2 W	Farnel: 9339760
Bateria recarregável	1	Bateria recarregável PP3 de NiMH e 9V	
Clip para bateria	1	Clip para bateria do tipo PP3	Farnell: 1183124
D1	1	Módulo de diodo laser de 5mW	eBay
Caixa	1	Caixa de plástico	Lojas locais

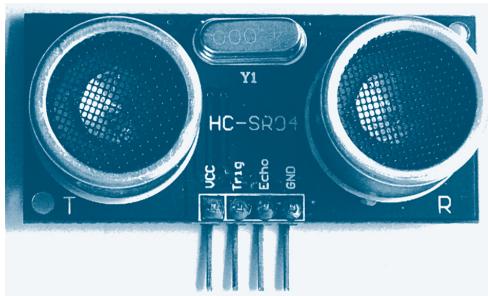


Figura 6-4 O módulo medidor de distância ultrasônico.

encontrar um módulo com a mesma disposição de pinos, será de grande ajuda. O autor obteve o seu módulo no eBay. Lá havia muitos dispositivos similares em oferta.

Se você planeja construir o cabo USB que não permite carga de bateria, você precisará também de dois resistores de $1\text{k}\Omega$ em vez de apenas um. Também será necessário um cabo USB de extensão. Veja o Capítulo 2 para os detalhes de construção desse cabo. Ele impedirá que seu celular descarregue as baterias dos seus acessórios.

Como alternativa para dispor de carga por mais tempo, você pode usar um suporte para seis pilhas AA capaz de fornecer 9V.

Além desses componentes, você precisará também dos seguintes equipamentos.

CAIXA DE FERRAMENTAS

- Uma furadeira elétrica com brocas
- Parafusos autoatarraxantes
- Um computador para programar o Arduino
- Ferro de soldar e solda
- Um cabo USB do tipo A-B

ATMega328 programado no Arduino e alguns outros componentes.

» Passo 2: solde o laser e o resistor

O módulo laser terá dois terminais: uma conexão positiva vermelha e uma negativa preta. Provavelmente, você precisará encurtar os terminais deixando-os com uns 5 cm. Descasque e estanhe as extremidades dos terminais. Cubra o terminal negativo com uma camada espessa de solda. Esse terminal será encaixado diretamente na barra de pino fêmea (de seis soquetes) da nossa Base para Acessório Droid. Encurte os terminais do resistor R1 de 100Ω e solde uma de suas extremidades no terminal positivo do laser. A outra extremidade do resistor também será encaixada na barra de pino fêmea no soquete de D5. Por isso, cubra-a também com um pouco de solda deixando-a mais espessa.

O módulo laser completo está mostrado na Figura 6-5.

Após, encaixe o módulo ultrassônico diretamente nos soquetes adequados da barra de pino fêmea. Use a Figura 6-3 como orientação para os pinos.

» Passo 3: instale o sketch do Arduino

Para instalar o sketch no microcontrolador, primeiro vamos programar o chip. Em seguida, ele será retirado com cuidado da placa do Arduino e encaixado no soquete da Base para Acessório Droid.

O sketch usa as mesmas bibliotecas dos projetos dos Capítulos 2 a 5. Se, até agora, você não construiu esses projetos, consulte o Passo 6 na seção Construção do Capítulo 2.

Após a instalação do sketch no microcontrolador, remova-o cuidadosamente, levantando cada extremidade um pouco de cada vez para não dobrar os seus pinos. A seguir, encaixe-o na Base para Acessório Droid, verificando se está com a orientação correta.

» Passo 1: construa uma Base para Acessório Droid

Essa é a unidade descrita no Capítulo 3 que contém um shield USB host, um microcontrolador

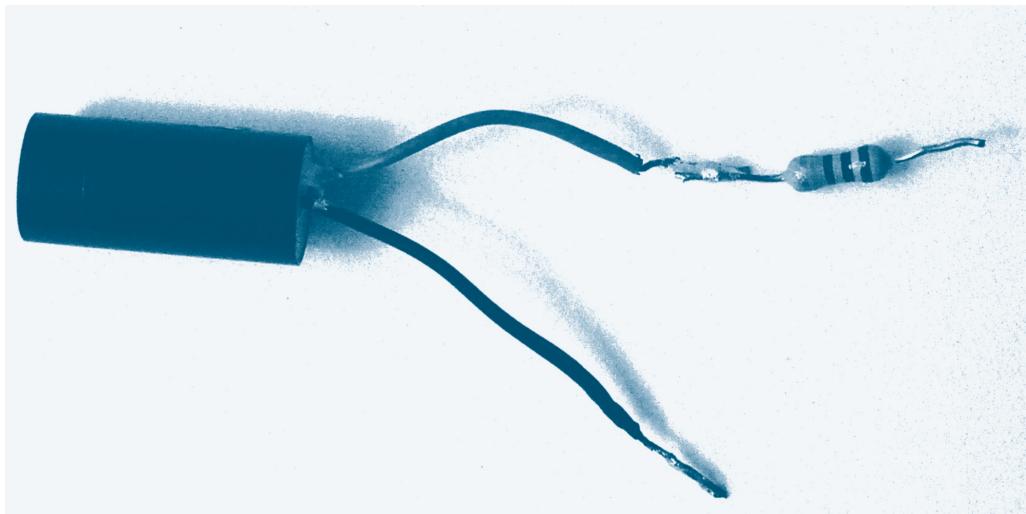


Figura 6-5 O módulo laser.

» Passo 4: instale o aplicativo Android

Se o seu telefone celular tiver conexão com a Internet, você poderá pular a instalação do aplicativo Android. Para isso, na primeira vez que você conectar o acessório medidor de distância ultrassônico, espere até que o software Open Accessory do celular solicite a instalação. Caso contrário, visite www.duinodroid.com e baixe o instalador APK.

Se ocorrer algum problema, volte ao Capítulo 2, na seção *Construção*, e leia o Passo 10.

» Passo 5: teste o projeto

Sempre é uma boa ideia testar o projeto antes de acondicioná-lo em uma caixa. Para isso, conecte tudo, como mostrado na Figura 6-6.

Agora, ligue a chave. Isso fará o celular iniciar a execução do aplicativo Droid medidor de distância. Se o aplicativo ainda não estiver instalado, o URL do local onde ele se encontra deverá aparecer no display.

Toque no botão “LASER ON” e verifique se o laser liga e desliga. Em seguida, experimente passar a mão na frente do transdutor ultrassônico e observe as leituras de distância variando na tela do Android. Se você clicar na leitura atual, ela será salva.

» Passo 6: acondicione o projeto

Disponha os componentes na caixa e verifique onde será necessário fazer furos. A seguir, marque a posição dos furos com um lápis (Figura 6-7).

Faça furos para o transdutor ultrassônico, o módulo laser, o soquete USB e faça dois furos pequenos para fixar a placa do shield no fundo da caixa. Você também precisará de um furo no lado da caixa para a chave.

Use parafusos autoatarraxantes pequenos para fixar o shield na caixa (Figura 6-8). Eles devem ser suficientemente pequenos para manter a placa no lugar sem atravessar o fundo da caixa, a fim de não arranharem alguma coisa.



Figura 6-6 Teste.

» Usando o projeto

Agora, o aplicativo Android deverá estar funcionando e mostrando continuamente a distância até o alvo da leitura (Figura 6-9).

Para unidade de distância, você pode escolher entre polegada e centímetro usando os respectivos botões. Se, a qualquer momento, você tocar no valor exibido de leitura, ele será incluído na lista de leituras que está na parte debaixo da tela.

O botão “LASER ON” permite ligar e desligar o laser.

O código fonte do aplicativo está disponível gratuitamente como software open-source. Para isso, acesse o site do livro (www.duinodroid.com). Sinta-se à vontade para fazer experimentos como

código e melhorá-lo. Você poderá adaptar esse projeto para outras aplicações interessantes, como:

- Detectar automaticamente se algo se moveu, enviando um SMS ou uma solicitação de Web.
- Calcular a área de uma sala tomando as suas duas dimensões.

» Teoria

Nos capítulos anteriores, nós já examinamos acessórios Android em número suficiente, sendo desnecessário nos repetirmos aqui.

Entretanto, nós analisaremos o funcionamento dos medidores ultrassônicos de distância e veremos como fazer a interface com um sketch de Arduino.

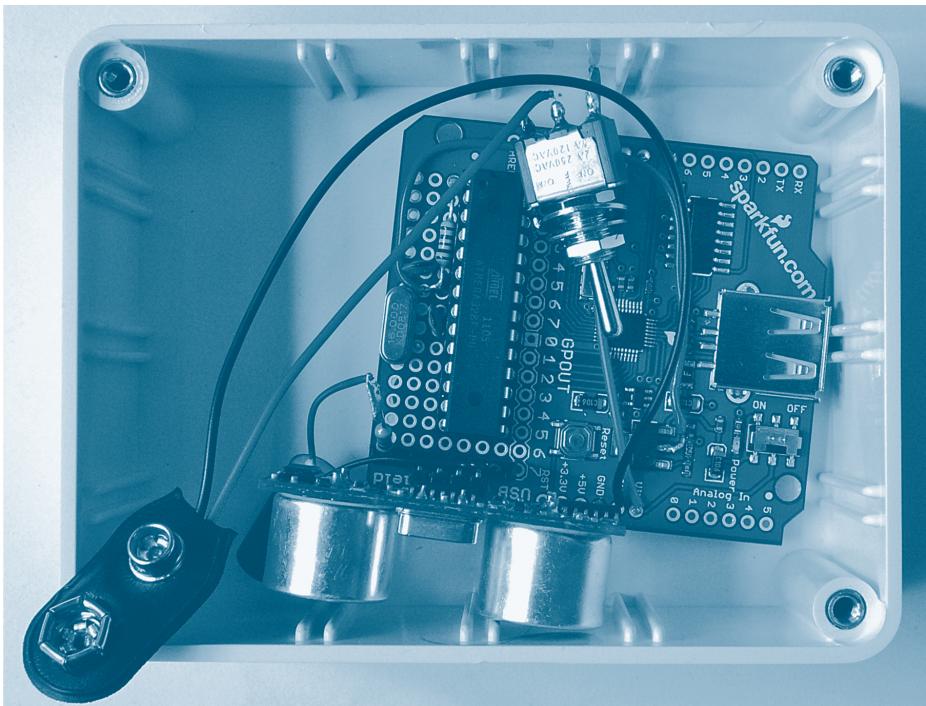


Figura 6-7 Disposição dos componentes na caixa.

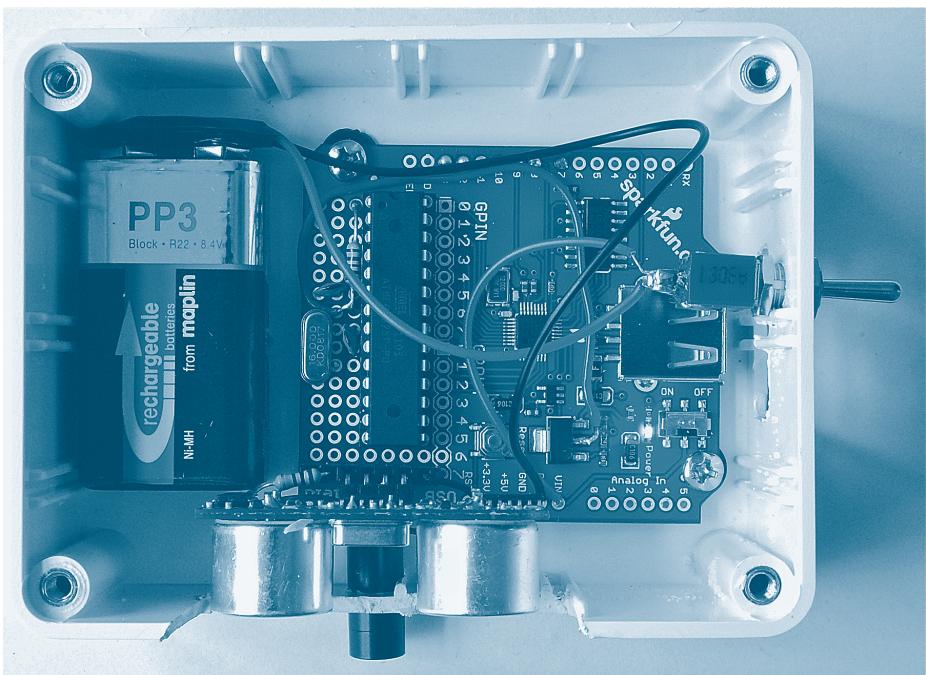


Figura 6-8 Os componentes fixados no interior da caixa.



Figura 6-9 O aplicativo Android.

» Medidores ultrassônicos de distância

Os medidores ultrassônicos de distância trabalham do mesmo modo que os sonares usados em navios e submarinos. Uma onda sonora é enviada por um transmissor, bate em um objeto e retorna a um receptor. Como você sabe, utilizando a velocidade do som poderemos calcular a distância até o objeto refletor do som. Basta medir o tempo que leva para o pulso sonoro ir até o objeto e retornar ao receptor (Figura 6-10).

O som usado é de alta frequência e, por isso, é denominado *ultrassônico*. A maioria dos módulos opera com uma frequência em torno de 40 kHz. Poucas pessoas conseguem ouvir sons acima de 20 kHz.

O código de Arduino para medir distância está inteiramente contido na função “takeSounding”. Ela envia um pulso único de 10 microssegundos ao pino “trigger” (gatilho) do módulo ultrassônico e, em seguida, utiliza a função interna de Arduino “pulseIn” para medir quanto tempo leva até que o pino “echo” (eco) vá para nível alto.

Esse intervalo é a quantidade de tempo necessária para a onda sonora viajar desde o transmissor, bater no alvo, ser refletida e retornar ao sensor.

```
long takeSounding()
{
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    long duration = pulseIn(echoPin,
        HIGH);
    long distance =
        microsecondsToCentimeters(duration);
    if (distance > 500)
    {
        return lastDistance;
    }
    else
    {
        lastDistance = distance;
        return distance;
    }
}
```

A seguir, devemos converter esse tempo (milissegundos) em distância (centímetros). Se não houver reflexão, porque não há objeto próximo o suficiente, ou se o objeto estiver refletindo a onda sonora para outra direção (sem retornar diretamente ao receptor), então o tempo do pulso será

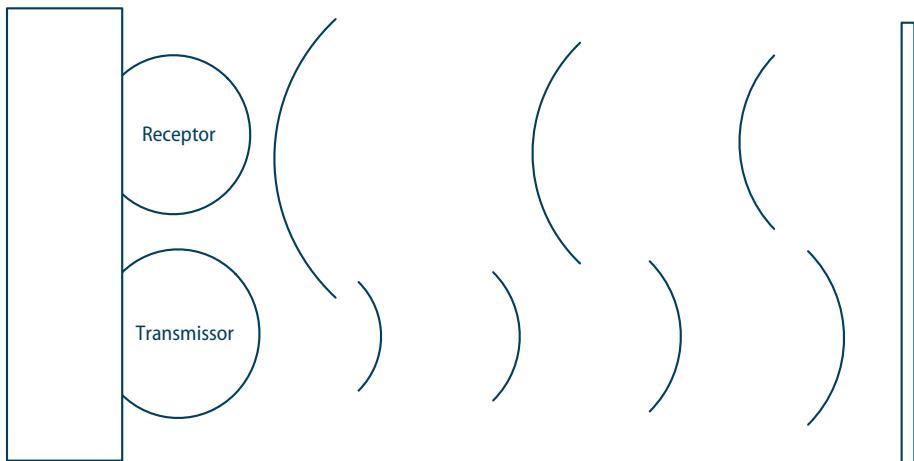


Figura 6-10 Medição ultrassônica de distância.

muito longo e a distância será aparentemente muito grande.

Para filtrar essas leituras muito demoradas, desconsideramos qualquer medição superior a 5m e adotamos a última leitura realizada.

A função “microsecondsToCentimeters” (microssegundos para centímetros) é, na realidade, muito simples.

```
long microsecondsToCentimeters (long
    microseconds)
{
    return microseconds / 29 / 2;
}
```

A velocidade do som é aproximadamente 343 m/s no ar seco a 20 °C, ou 34.300 cm/s.

Expressando de outro modo, temos $(34.300/1.000.000)$ cm/microsssegundo.

Isso corresponde a 0,0343 cm/microsssegundo.

Invertendo, é $(1/0,0343)$ microsssegundo/cm.

Portanto, 29,15 microsssegundos/cm.

Concluímos que 291,5 microsssegundos indicam uma distância de 10 cm.

A função “microsecondsToCentimeters” arredonda 29,15 para 29 e, em seguida, divide a resposta por 2, porque não queremos o tempo completo de ida e volta do som. Queremos apenas a distância até o objeto.

Na realidade, há muitos fatores que afetam a velocidade do som, de modo que a abordagem adotada aqui dá apenas uma resposta aproximada. Tanto a temperatura quanto a umidade do ar afetam a medição.

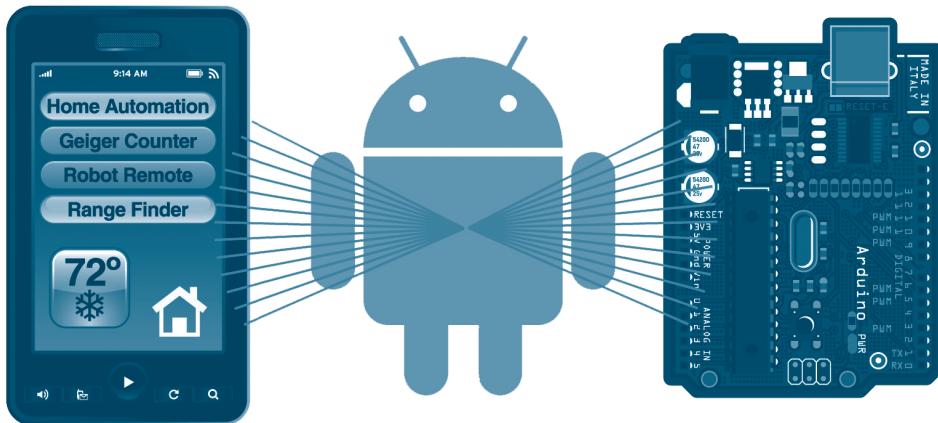
» Resumo

Esse foi o último de nossos acessórios Android. O restante do livro será dedicado a combinar Android e Arduino de um modo diferente, propiciando um sistema de automação residencial com base em um tablet Android de baixo custo. Como esses dispositivos ainda não estão preparados para utilizar o protocolo Android de Open Accessory, nós mesmos teremos que encontrar um modo de conectá-los a um Arduino.



Parte II

Automação residencial





» capítulo 7

Controlador de automação residencial

Neste capítulo, você aprenderá a construir um controlador de automação residencial. Este projeto constitui a unidade central do sistema de automação residencial desenvolvido aqui. Será o sistema de controle de muitos projetos deste livro.

Objetivos

- » Entender o que é um sistema de automação residencial
- » Aprender a usar uma interface de conexão de áudio entre o Android e o Arduino
- » Entender o funcionamento do módulo de conexão de áudio
- » Entender como funcionam a codificação de dados com som, a eletrônica da interface de áudio e a decodificação dos sons no Arduino

Com o sistema de automação residencial é possível controlar remotamente as luminárias, as tomadas, o sistema de aquecimento e até mesmo o acesso à porta da frente. Além disso, o controlador de automação residencial apresenta uma tecnologia que auxilia ligar e desligar os dispositivos automaticamente de forma programada, mantendo assim as suas contas de luz sob controle.

A Figura 7-1 mostra a unidade de controle do sistema de automação residencial, e a Figura 7-2 mostra como o controlador de automação residencial vincula-se aos outros projetos deste livro.

Examinando a Figura 7-2, pode-se ver que o controlador baseia-se em um tablet Android e em uma placa de Arduino bastante vinculados entre si.

As placas de Arduino são ótimas para adicionar circuitos eletrônicos a um computador, mas não são boas para funcionar como uma interface de visual agradável e uso fácil. O máximo que se consegue é um pequeno display e alguns botões. É aqui que entra o tablet Android de baixo custo (em torno de 100 dólares). Ele apresenta uma tela grande de toque, WiFi, saída de áudio e, algumas vezes, até uma webcam. Além disso, um tablet Android pode funcionar como um servidor de Web, de modo que você pode acessar o nosso sistema de automação residencial a partir de qualquer dispositivo ligado à Web, seja em casa seja em qualquer outro lugar através da Internet.

Quando concluirmos a construção, o controlador de automação residencial estará conectado a:

- Tomadas e luminárias controladas por RF
- Um termostato “inteligente” que usa outro Arduino
- Uma fechadura com RFID

Por enquanto, construiremos apenas os circuitos básicos para o Arduino e o tablet Android. À medida que o livro avançar, a cada capítulo acrescentaremos outros componentes ao projeto.

Há diversos maneiras para enviar comandos do tablet Android ao Arduino. Poderíamos:

- Equipar nosso Arduino com um shield de Ethernet ou WiFi e realizar a comunicação por meio de uma rede.
- Conectar um módulo Bluetooth ao Arduino e, se nosso tablet Android tivesse Bluetooth, enviar dados desse modo.
- Usar a saída de áudio do tablet Android para enviar comandos ao Arduino.
- Usar uma conexão USB e o ADK do Google (isso requer Android 2.3.4 ou posterior).

Em outros capítulos, utilizamos Ethernet e Bluetooth. Usamos também um enlace de radiofrequência (RF) e uma conexão USB usando o ADK do Android. No entanto, neste projeto, usaremos uma interface de “áudio” entre o Android e o Arduino. Isso traz a vantagem de ser um laço de conexão física – em outras palavras, um fio vai diretamente do tablet Android ao Arduino. Além disso, muitos tablets Android de baixo custo não têm Bluetooth ou Android 2.3.4 (necessário para o ADK). Se não fosse por isso, o uso de ADK seria uma boa opção.

O tablet Android produzirá sons e empregará a mesma técnica que os primeiros computadores domésticos usavam para gravar programas em fitas-cassetes.

A utilidade desse tipo de conexão não se limita à automação residencial. Poderá ser usado em qualquer situação em que haja necessidade de enviar comandos de um tablet Android a um Arduino.

Power		
	On	Off
Outlet a	On	Off
Outlet 2	On	Off
Lights 1	On	Off
Lights 2	On	Off
All	On	Off

Figura 7-1 O controlador do sistema de automação residencial.

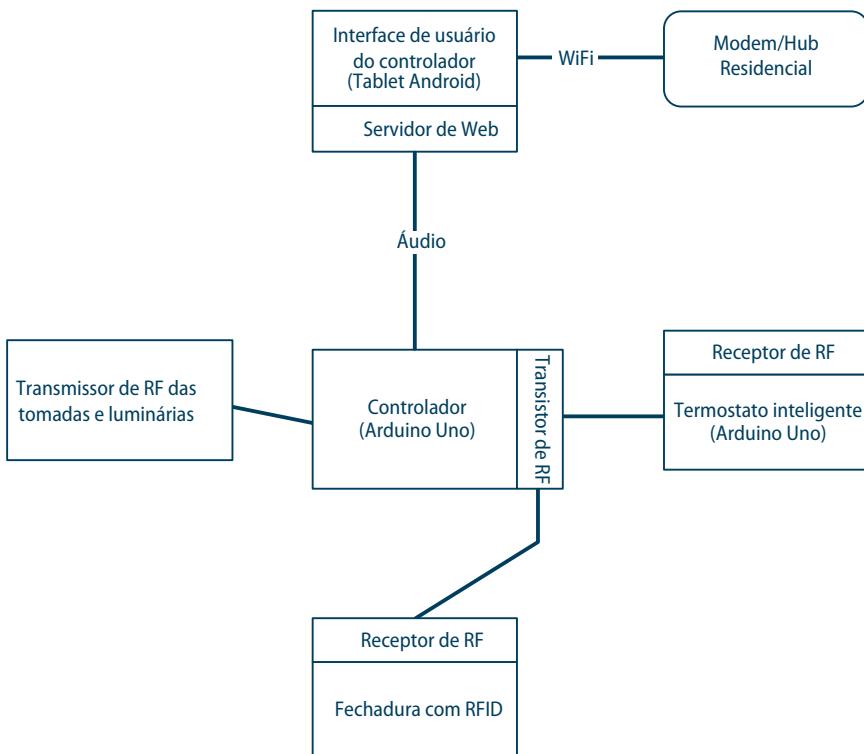


Figura 7-2 O sistema de automação residencial.

Este projeto está dividido em três seções: construção do hardware da conexão de áudio, programação do tablet Android e programação do Arduino.

» Módulo de conexão de áudio

O diagrama esquemático do módulo de conexão de áudio está mostrado na Figura 7-3.

A seção *Teoria*, no final deste capítulo, contém mais informações sobre o funcionamento dessa conexão, mas por enquanto será suficiente saber que o dispositivo Android produz uma sequência de 16 pulsos sonoros de 1 kHz. Um pulso longo indica o valor lógico 1 e um pulso curto, o valor lógico 0. Esses pulsos são detectados e enviados a uma entrada digital do Arduino, que os converte em um número.

» O que será necessário

Você precisará dos componentes listados na tabela *Lista de Componentes*, a seguir, para construir o módulo da conexão de áudio.

Obviamente, os componentes mais caros deste projeto são o tablet Android e a placa do Arduino. O tablet Android utilizado aqui custou em torno de 100 dólares no eBay. Ele tem uma tela de sete polegadas e usa o Android 2.1. Ele foi projetado para servir de substituto do iPad. A sua qualidade é bem baixa, mas o que poderíamos esperar a esse preço?

No tablet, deve estar instalado o Android versão 2.1 ou posterior. Deve ter também WiFi e um jack de saída de áudio. Não há necessidade de Bluetooth.

A placa de Arduino mais atual é a Uno. O site oficial do Arduino (www.arduino.cc) apresenta uma lista de fornecedores do Uno. Entretanto, se quiser eco-

LISTA DE COMPONENTES

Componente	Quantidade	Descrição	Fornecedor
Arduino Uno	1	Placa de Arduino Uno	www.arduino.cc
Tablet Android	1	Tablet Android de baixo custo com Android versão 2.1 ou posterior, com jack de saída de áudio e WiFi	eBay
Adaptador de Voltagem CA/CC	1	Adaptador de Voltagem CA/CC de 15V e 1A	Farnell: 1176620
R1, R3	2	Resistor de filme metálico de $1k\Omega$ e 1/2 W	Farnell: 9339779
R2	1	Resistor de filme metálico de $1M\Omega$ e 1/2 W	Farnell: 9339809
R4	1	Resistor de filme metálico de $100k\Omega$ e 1/2 W	Farnell: 9339795
R5	1	Resistor de filme metálico de $33k\Omega$ e 1/2 W	Farnell: 9340424
C1	1	Capacitor eletrolítico de $1\mu F$ e 16V	Farnell: 1236655
C2	1	Capacitor cerâmico de $220nF$	Farnell: 1216441
C3,C4	2	Capacitor eletrolítico de $100\mu F$ e 16V	Farnell: 1136275
C5	1	Capacitor cerâmico de $100nF$	Farnell: 1200414
D1	1	1N4001	Farnell: 1458986
Cl1	1	Amplificador operacional CMOS 7611	Farnell: 1018166
Cl2	1	Regulador de tensão linear 7909	Farnell: 7202164
Soquete para Cl	Opcional	Soquete circuito integrado de oito pinos	Farnell: 1101345
	1	Placa perfurada com 17 trilhas de 12 furos	Farnell: 1201473
	1	Plugue P4 de alimentação elétrica de 2,1 mm	Farnell: 1200147
	1	Jack J4 de alimentação elétrica de 2,1 mm	Farnell: 1217038
Plugue de 3,5mm	1	Plugue P2 conector de áudio estéreo de 3,5mm	Farnell: 1267389
Fios		Fios rígidos e flexíveis diversos (veja o texto)	Lojas locais de eletrônica
Caixa		Caixa ABS de projeto para acondicionar o tablet Android	Farnell, Radio Shack e outros
Fita perfurada de metal			Lojas de hobby ou ferragem

nomizar, você poderá utilizar um clone do Arduino Uno ou do Arduino Duemilanove, que também funcionará muito bem.

Você pode comprar um plugue de áudio novo, mas também poderá usar algum par de fones antigo, do qual poderá cortar o plugue deixando-o com cerca de 25cm de cabo. Nesse caso, ele já virá soldado ao cabo. O mesmo argumento aplica-se ao plugue de alimentação elétrica de 2,1mm. Pode ser também que você já tenha um adaptador de voltagem CA/CC fora de uso, do qual poderá cortar um pedaço de cabo junto com o conector.

Se, como no nosso caso, a caixa de projeto que você pretende utilizar para acondicionar tudo não tiver espaço sobrando suficiente, então use plu-

ges em 90° (joelho), que se sobressairão muito menos do tablet Android. Você também pode curvar os cabos de um plugue normal de 3,5mm para impedir que ele fique muito para fora.

Como você pode ver na Figura 7-19, o tablet Android está fixado no interior de uma caixa plástica de projeto. Ela precisa ser suficientemente grande para conter o tablet, mas também suficientemente funda para conter os demais componentes eletrônicos que serão necessários na caixa. Entre esses componentes, estarão o Arduino, a interface de áudio e, mais adiante, incluiremos o controle remoto de RF, um módulo Bluetooth e um transmissor de RF. Isso significa que a caixa deverá ter uma altura extra de uma polegada (2,5 cm) a mais que a es-

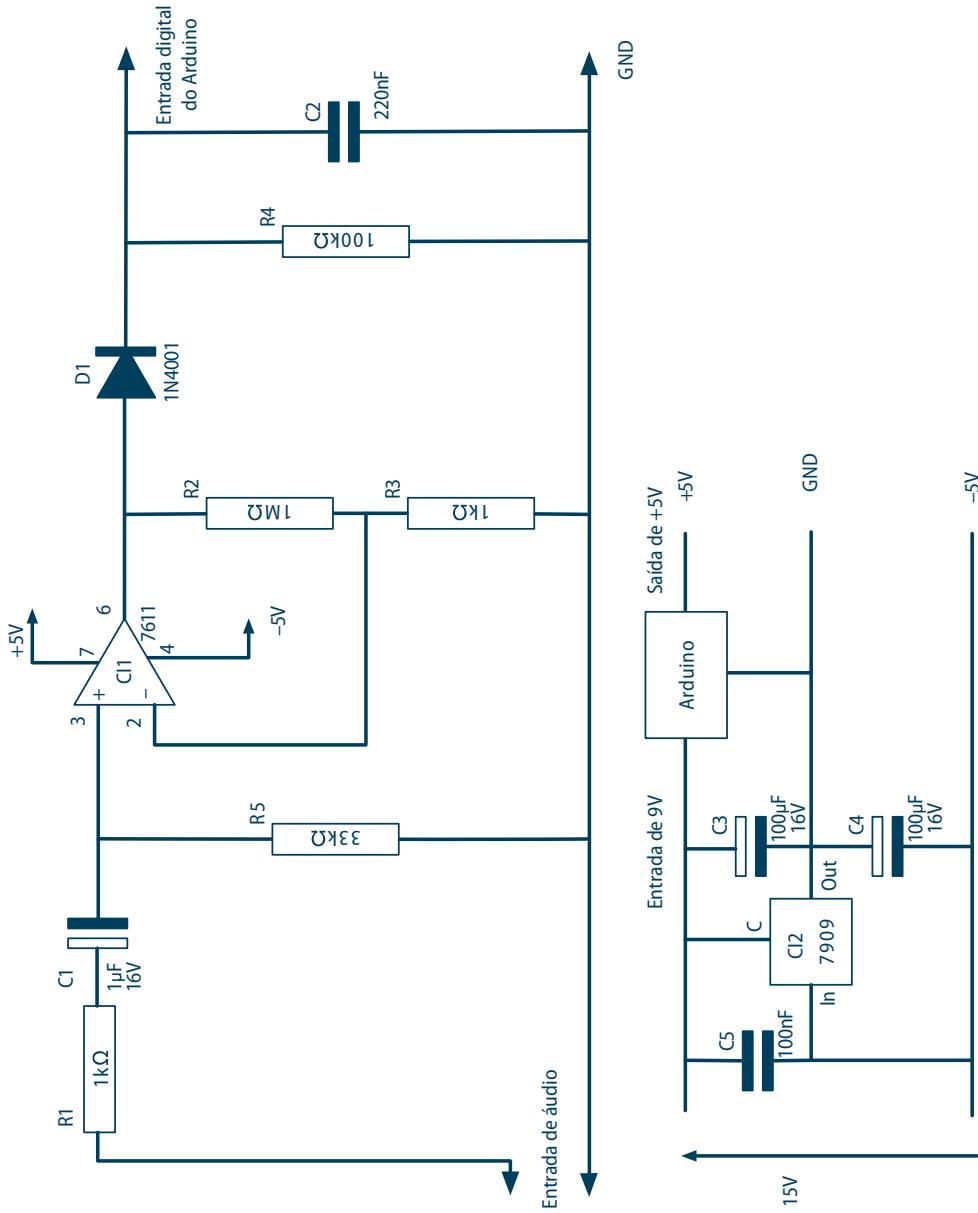


Figura 7-3 O diagrama esquemático do módulo de conexão de áudio.

pessura do tablet. A caixa que usamos aqui tem uma altura total de cerca de 6,5 cm.

Para todos os projetos deste livro, você precisará de um estoque básico de fios. Para fazer ligações na placa perfurada, o melhor é usar fio rígido fino de cobre. Quando se tratar de fazer conexões entre componentes do projeto, como entre a placa perfurada da interface de áudio e a placa do Arduino, o melhor é usar fio flexível fino. É bom ter, como no nosso caso, uma caixa de sucata com fios de antigos equipamentos domésticos. Sempre é útil dispor de diversas cores utilizando o vermelho para as conexões positivas e o preto ou o azul para as conexões negativas e do terra. Além dessas, podemos usar alguma outra cor, como amarelo ou laranja, para as conexões de sinal.

Além desses componentes, você precisará das ferramentas da lista seguinte.

CAIXA DE FERRAMENTAS

- Ferramentas para soldar
- Fios rígidos e flexíveis de diversas cores
- Um multímetro
- Uma furadeira elétrica com brocas
- Um estilete ou uma tesoura forte
- Um cortador ou estilete rotativo opcional tipo Olfa ou uma microrretífica tipo Dremel

» Passo 1: prepare a placa perfurada

O módulo de conexão de áudio é construído em um pequeno pedaço de placa perfurada (Figura 7-4).

Se esta é a primeira vez que você trabalha com placa perfurada, então você deverá primeiro conhecer algumas coisas a respeito.

Uma placa perfurada é uma placa com furos separados de 1/10 de polegada (2,5 mm). Atrás dos furos, há trilhas de cobre. Os terminais dos componentes entram pelo lado sem cobre e saem pelo outro lado, sendo soldados nas trilhas. A Figura 7-5 mostra a placa perfurada do módulo de conexão de áudio visto do lado das trilhas.

Observe que algumas das trilhas de cobre estão cortadas no lugar onde o CI (circuito integrado) deve ser instalado. Esses locais estão marcados com um "X" na placa perfurada da Figura 7-4. Os cortes nas trilhas podem ser feitos usando uma microrretífica com ponta de desbaste. A ponta é encostada no furo em que se deve fazer o corte e, em seguida, fazendo movimentos circulares com a microrretífica, a trilha de cobre é desbastada sem aumentar muito o diâmetro do furo.

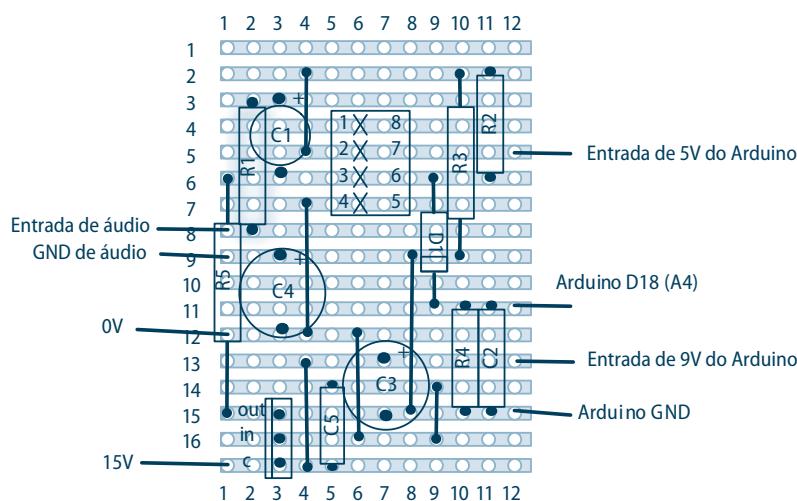


Figura 7-4 A disposição dos componentes do módulo de conexão de áudio.

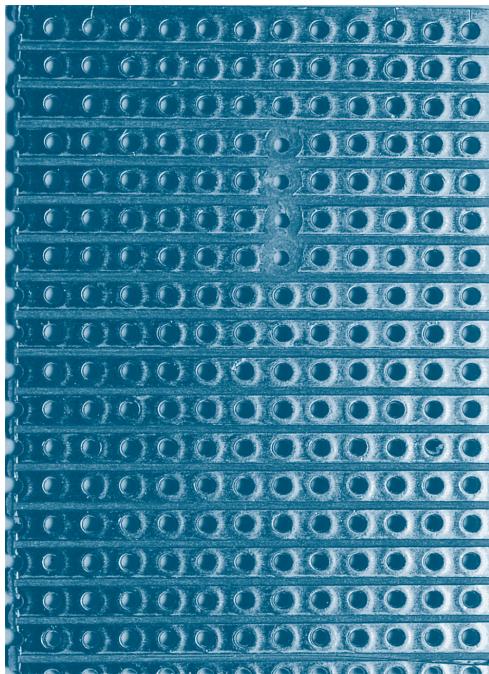


Figura 7-5 A placa perfurada pronta.

Comece cortando um pedaço de placa perfurada com 17 trilhas de 12 furos. Você pode fazer isso com uma tesoura grande, mas nesse caso as bordas poderão ficar irregulares. Um acabamento melhor pode ser conseguido fazendo sulcos na placa com um estilete e quebrando-a ao longo dos sulcos na beira de uma mesa. Tome cuidado ao quebrar a placa, pois podem surgir arestas afiadas.

Usando as Figuras 7-4 e 7-5 como guia, faça quatro cortes nas trilhas.

» Passo 2: solde as conexões

A regra para soldar componentes em placa perfuradas – ou em placas de circuito impresso – é soldar primeiro os componentes baixos. Assim, depois de colocar um componente no lugar e virar a placa de cabeça para baixo, o peso da placa manterá o componente na sua posição enquanto você o solda.

Neste projeto, temos que soldar seis conexões na placa e começaremos por elas.

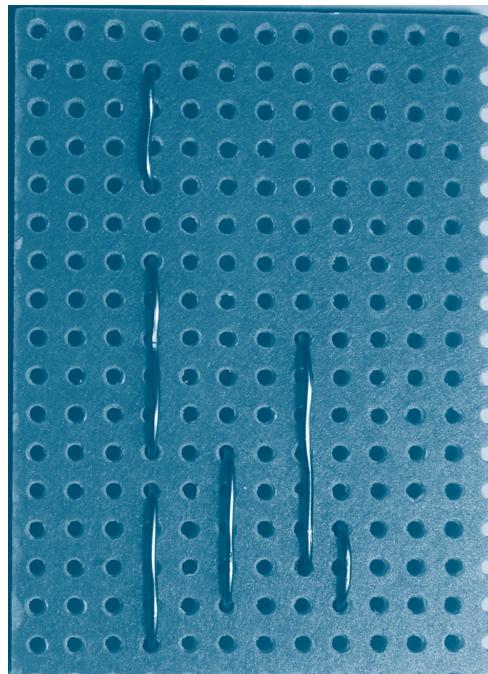


Figura 7-6 A placa perfurada com as ligações.

Usando as Figuras 7-4 e 7-6 como referência, corte fios rígidos no tamanho aproximadamente correto, soldê-os no lugar e corte o que sobrar dos fios.

» Passo 3: solde os resistores e o diodo

Os próximos componentes baixos são os resistores e o diodo (Figura 7-7). Não solde o resistor R5 ainda. Isso será feito após a instalação dos cabos de conexão de áudio.

Os resistores podem ser inseridos com qualquer orientação, mas, quando você soldar o diodo no lugar, ele deverá estar como mostrado na Figura 7-7, de forma que a faixa que envolve o diodo fique para baixo, como na figura.

» Passo 4: solde o CI

Como opção, você pode usar um soquete para o circuito integrado. Isso evita danificar o CI por aquecimento excessivo. Se você decidir soldar o

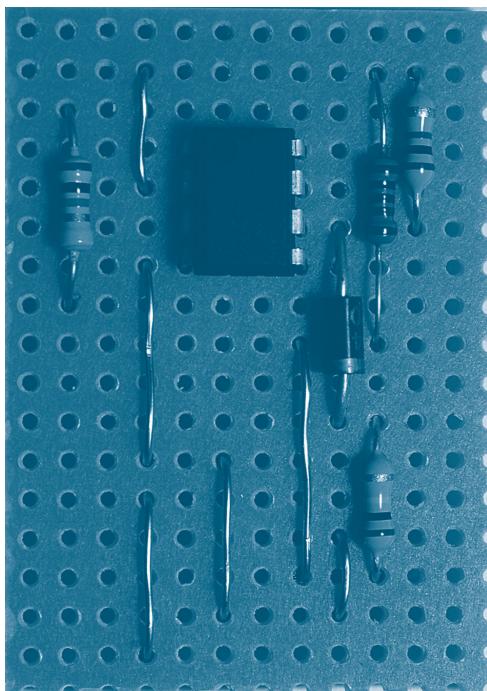


Figura 7-7 A placa perfurada com os resistores e o diodo.

CI diretamente, tente minimizar o tempo em que você mantém os pinos aquecidos. Se surgir uma conexão difícil, levando tempo para a solda derreter, espere até que o CI esfrie antes de soldar o pino seguinte.

Verifique se o CI está com a orientação correta. A marca pequena em cima do CI indica o pino 1 e deve estar em cima à esquerda. Veja a Figura 7-8.

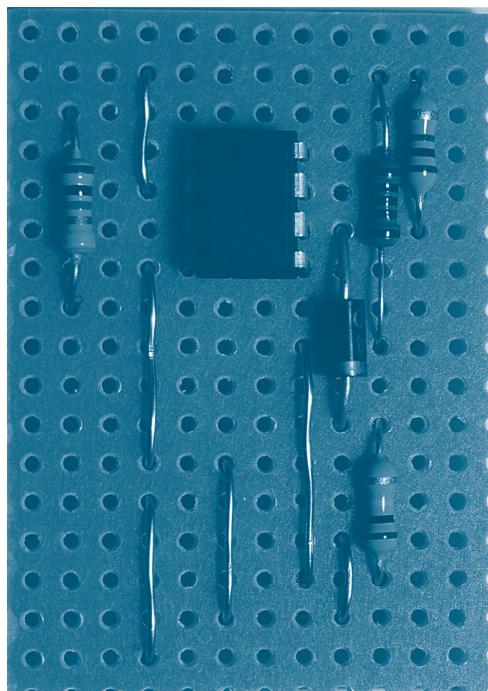


Figura 7-8 A placa perfurada com o CI no lugar.

O regulador de tensão tem a placa de metal voltada para o lado esquerdo da placa.

A Figura 7-9 mostra a placa perfurada pronta.

Observe que ainda não colocamos o resistor R5. Iremos soldá-lo após colocar os cabos de conexão da placa.

Neste ponto, uma inspeção completa da placa é uma boa ideia. Verifique a parte de cima da placa vendo se todos os componentes e as conexões estão nos lugares indicados pela Figura 7-4. Examine também a parte debaixo da placa verificando se todos os cortes de trilha estão nos lugares corretos e se não há pontes acidentais de solda entre as trilhas.

» Passo 5: solde os demais componentes

Depois que o CI estiver no lugar, ainda haverá quatro capacitores e o regulador de tensão para soldar. Os capacitores eletrolíticos são polarizados. Isso significa que eles devem ser colocados seguindo uma orientação correta. O terminal mais comprido é o positivo. O terminal negativo costuma ter um símbolo de diamante próximo dele.

» Passo 6: instalando os cabos

Precisamos conectar a nossa interface de áudio ao tablet Android e à nossa placa de Arduino. Assim, o próximo passo será soldar alguns cabos na placa (Figura 7-10).

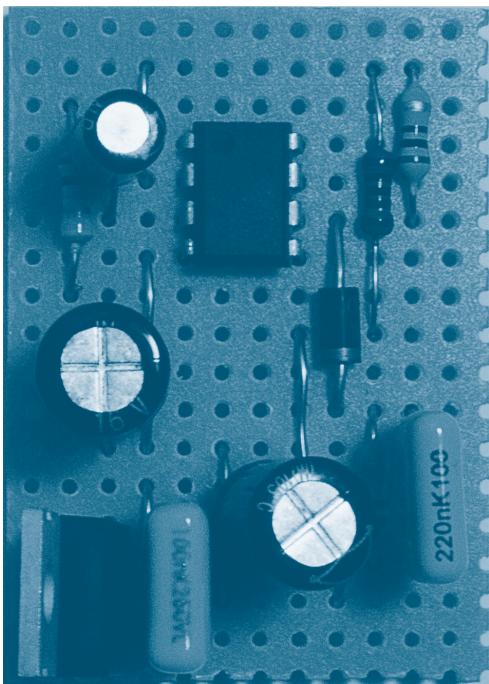


Figura 7-9 A placa perfurada pronta.

Vamos começar com o cabo de áudio. Aqui você tem duas opções. Você pode comprar um jack estéreo de 3,5 mm e soldar os cabos ou, então, você pode comprar ou aproveitar um par de fones, sem uso e de baixo custo, e cortar uns 25 cm de cabo junto com o conector. Desse modo, você terá um cabo pronto. O cabo provavelmente terá uma malha de fio na camada externa que atua como blindagem. Internamente, terá dois fios isolados em cores vermelha e branca para os canais esquerdo e direito da saída de som do dispositivo Android. Só precisamos de um canal. Portanto, podemos cortar um dos fios internos (digamos o branco). Ao fazer isso, verifique se não ficou algum resíduo de fio nu fazendo curto-circuito com a blindagem. Isso poderia danificar o dispositivo Android.

Junte e torça os fios da malha de blindagem. Em seguida, coloque um pouco de solda para mantê-los juntos. Com cuidado, desencapse a ponta do fio vermelho e coloque também um pouco de solda.

Enfie o fio de blindagem e o vermelho nos furos correspondentes da placa perfurada (entrada de áudio e GND de áudio) e solde-os por baixo nas trilhas, como mostrado nas Figuras 7-10 e 7-4.

Além do cabo de áudio, precisaremos de um cabo ligado a um jack de alimentação elétrica para receber os 15V do adaptador de voltagem CA/CC. Precisaremos também de outro cabo ligado a um plugue de baixa tensão para fornecer alimentação elétrica ao Arduino.

O cabo com jack é feito soldando fios flexíveis curtos entre a placa de interface de áudio e um jack de 2,1 mm de alimentação elétrica. Entretanto, para o plugue de alimentação elétrica do Arduino, podemos fazer um cabo usando fio flexível e um plugue, ou cortando a extremidade do cabo de um adaptador de voltagem CA/CC fora de uso ou estragado.

Agora, podemos soldar o resistor R5 por cima dos cabos, no lado esquerdo da placa. Verifique também se ele não está encostando nas partes nuas do cabo. Veja a Figura 7-11.

Finalmente precisamos de fios para conectar a alimentação de 5V do Arduino à placa de interface de áudio e dessa placa até a conexão A4 da placa do Arduino.

» Passo 7: teste

Com isso, completamos a construção da placa. Agora, precisamos testá-la.

Em vez de testar o software completo do sistema de automação residencial, vamos baixar um aplicativo de teste para o dispositivo Android e um sketch de teste para o Arduino em www.duinodroid.com.

Vamos iniciar instalando o aplicativo Android no nosso dispositivo Android.

Diferentemente do iPhone, você pode baixar os seus aplicativos Android de qualquer lugar que desejar. Isso significa que você deve verificar se não está baixando nada perigoso. Além disso, você

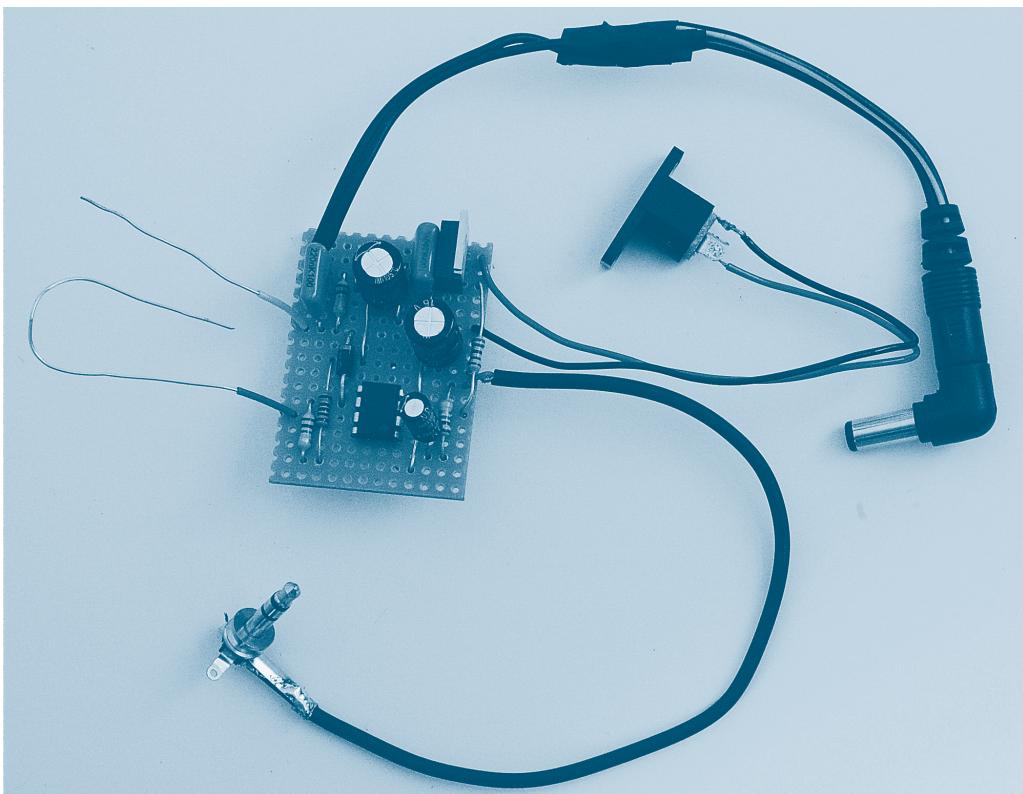


Figura 7-10 A placa perfurada com os cabos instalados.

deve programar o seu dispositivo Android adequadamente.

Abra o aplicativo Android de “Configurações”, navegue até “Aplicações” e ative a opção “Fontes desconhecidas”, como mostrado na Figura 7-12*.

Para fazer a instalação do aplicativo de teste, abra o aplicativo de navegação no seu dispositivo Android e vá para www.duinodroid.com. Clique em “Downloads” e, em seguida, clique em “Sound Interface Test App” (aplicativo de teste da interface de áudio).

O aplicativo começará a ser baixado e, logo em seguida, você poderá executá-lo (Figura 7-13).

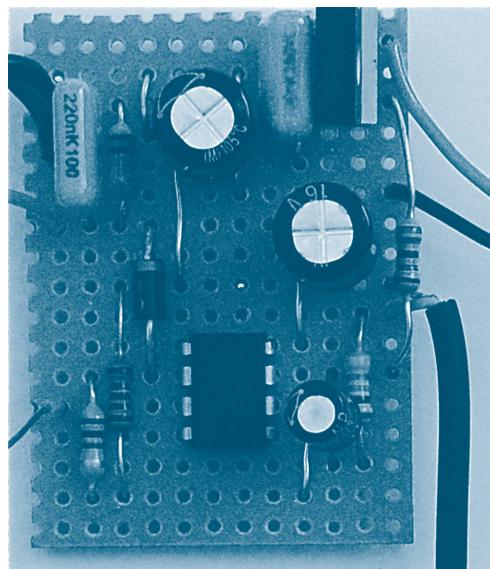


Figura 7-11 A placa com o resistor R5 no lugar.

* N. de T.: Essa sequência aplica-se ao celular que utilizamos aqui. No seu caso, a sequência poderá ser diferente.

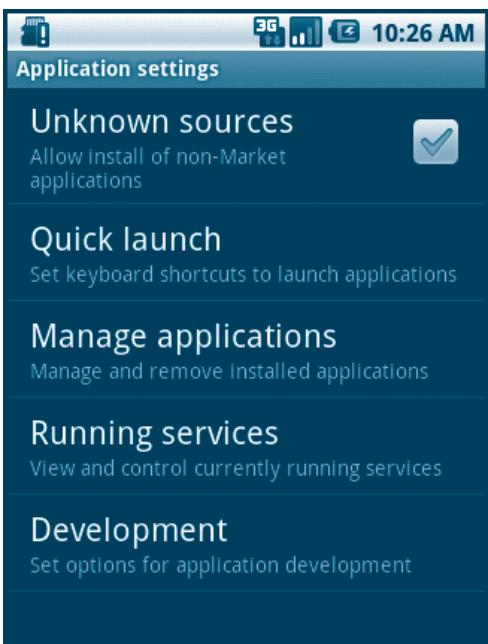


Figura 7-12 Mudando as configurações para permitir baixar aplicativos de fontes desconhecidas.

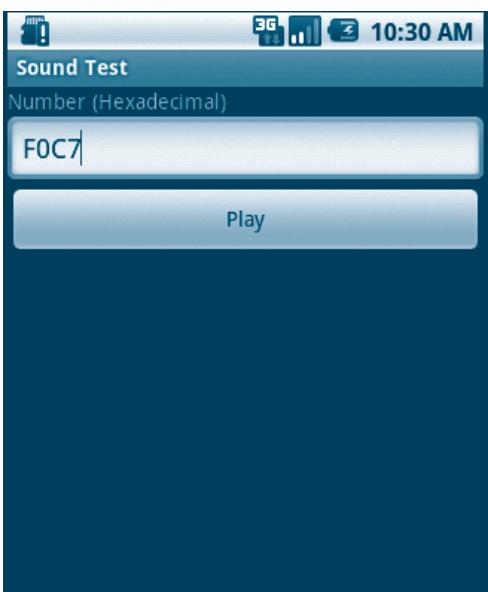


Figura 7-13 O aplicativo Android de teste para a interface de áudio.

O aplicativo de teste permite que você entre com um número hexadecimal (hex) de quatro dígitos. Quando você pressionar o botão Play (tocar), o número será convertido em pulsos sonoros. Como veremos mais adiante, esses são os pulsos que o software completo de automação residencial produz para serem enviados ao Arduino, que por sua vez reconstruirá o número. Podemos fazer um teste da geração dos pulsos sonoros sem conectar o celular à interface de áudio, simplesmente entrando com um número (cada dígito deve ser de 0 a 9 ou de A a F).

Experimente entrar com "0000" e clicar em "Play". A seguir, entre com "FFFF" e clique em "Play". Você deverá notar a diferença entre os dois conjuntos de pulsos sonoros saindo dos alto-falantes do dispositivo Android.

Agora, precisamos preparar o ambiente do nosso Arduino para completar a conexão e fazer o Arduino informar quais são os números que estamos enviando.

A placa de Arduino utilizada (Arduino Uno) usa um ambiente de desenvolvimento especial que permite enviar os programas, ou "sketches" como são conhecidos no mundo do Arduino, para a placa por meio de um cabo USB.

Precisamos instalar o ambiente Arduino. Em vez de repetir aqui as instruções que estão disponíveis no site oficial do Arduino (www.arduino.cc), acesse esse site e siga as instruções que lá estão para instalar o ambiente Arduino em seu computador. Nesse site, você encontrará instruções separadas para Windows, Linux e Mac. Neste livro, estamos usando uma placa de interface Arduino Uno e a versão 22 do software Arduino. Entretanto, você não deverá encontrar problema se usar versões posteriores de Arduino.

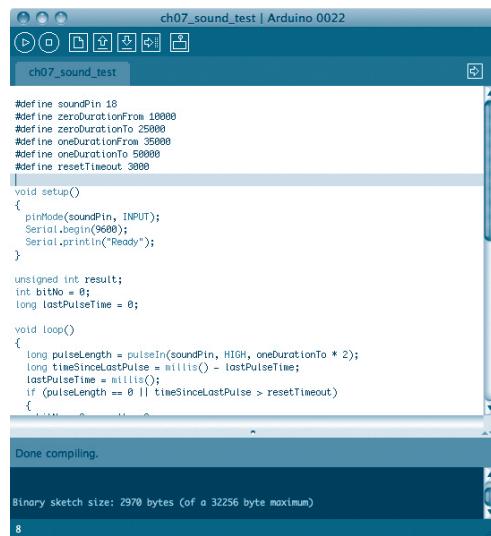
Logo que o ambiente Arduino estiver preparado, você precisará instalar o sketch de teste do projeto. Na realidade, todos os sketches dos projetos deste livro estão disponíveis em um único arquivo zip que pode ser baixado de www.duinodroid.com.

Descompacte o arquivo zip e mova toda a pasta Arduino Android para a sua pasta de sketches. No Windows, a sua pasta de sketches estará em Meus Documentos/Arduino. No Mac, você a encontrará em Documents/Arduino/ e, no Linux, estará no diretório Sketchbook.

Reinic peace software Arduino. Em seguida, no menu File (arquivo), selecione sketches (ou SketchBook), seguido de Arduino Android, e então ch07_sound_test. Isso abrirá o sketch Sound Test (teste da interface de áudio), como mostrado na Figura 7-14.

Na seção *Teoria*, no final deste capítulo, veremos como esse sketch decodifica o som que vem do dispositivo Android e gera o número. Por enquanto, estaremos satisfeitos apenas em utilizá-lo. Esse sketch simplesmente espera por uma sequência de 16 pulsos sonoros. Em seguida, repassa o número recebido e decodificado ao Serial Monitor do software do Arduino. Em outras palavras, usaremos o nosso computador como um monitor para nos informar o que o Arduino está recebendo do dispositivo Android.

Conecte a sua placa de Arduino ao computador via USB e transfira o sketch para a placa clicando no ícone Upload (segundo a partir da direita na barra



```
#define soundPin 18
#define zeroDurationFrom 10000
#define zeroDurationTo 25000
#define oneDurationFrom 35000
#define oneDurationTo 50000
#define resetTimeout 3000

void setup()
{
    pinMode(soundPin, INPUT);
    Serial.begin(9600);
    Serial.println("Ready");
}

void loop()
{
    long pulseLength = pulseIn(soundPin, HIGH, oneDurationTo * 2);
    long timeSinceLastPulse = millis() - lastPulseTime;
    lastPulseTime = millis();
    if (pulseLength == 0 || timeSinceLastPulse > resetTimeout)
    {
        // Process the received number
    }
}
```

Done compiling.

Binary sketch size: 2970 bytes (of a 32256 byte maximum)

8

Figura 7-14 O sketch “Sound Test” (teste da interface de áudio).

de ferramentas). Se aparecer uma mensagem de erro, você deverá informar quais são os tipos de placa e conexão que você está utilizando. Para definir a placa usada, vá até o menu Tools (ferramentas) e selecione a opção Board (placa). Isso lhe dará uma lista semelhante à da Figura 7-15.

Selecione a opção correspondente ao tipo de placa que você está usando. A seguir, faça a mesma coisa com a Serial Port (porta serial), que também faz parte do menu Tools. Geralmente, a opção será a que está no topo da lista de portas. Normalmente costuma ser a COM4, no Windows.

Quando você clica no ícone de “Upload”, você deve ver os LEDs da placa do Arduino piscando bastante. Isso indica que a placa do Arduino está recebendo o sketch.

Agora, podemos fazer as conexões da nossa placa de interface de áudio e iniciar o teste. A Figura 7-16 mostra a interface de áudio conectada à placa de Arduino e, em particular, as conexões de A4 e +5V.

Encaixe o plugue de áudio na saída de som do tablet e o plugue de 15V, do adaptador de voltagem CA/CC, no jack do cabo de alimentação da placa de interface de áudio. A seguir, encaixe no Arduino o plugue de 9V, vindo da placa de interface de áudio, e conecte os fios de sinal digital (A4) e 5V do Arduino, como mostrado na Figura 7-16.

Agora que tudo está conectado, forneceremos um número ao aplicativo de teste Android para ser enviado ao Arduino através da interface de áudio. Por sua vez, o Arduino repassará o número ao computador através do cabo USB.

Para ver as mensagens vindas do Arduino, abra o Serial Monitor do Arduino no seu computador (Figura 7-17). Isso permite ver qualquer comunicação serial vinda do Arduino.

Entre com um número – por exemplo, F0C7 – no aplicativo de teste Android e pressione “Play”. Se você ouvir o som, o plugue de áudio não está ligado no tablet Android. Conecte-o e tente novamente.

Quase imediatamente, você verá a seguinte mensagem aparecer no Serial Monitor: “Arduino recei-

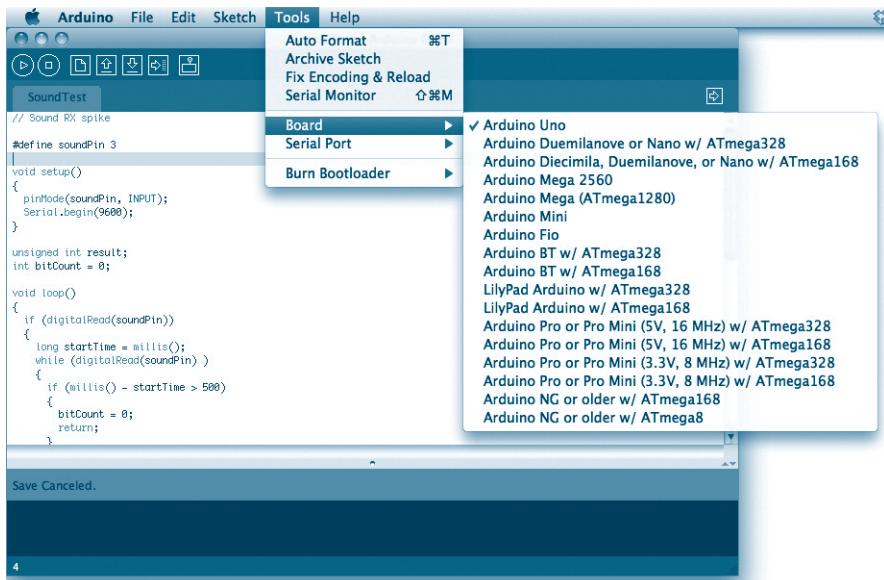


Figura 7-15 Seleção da placa de Arduino utilizada.

ved: F0C7" (Arduino recebeu: F0C7). Se você ver um número diferente ou nenhum número, tente ajustar o volume do tablet Android. Provavelmente, você deverá ajustar o volume para três quartos do máximo.

» Passo 8: acondicione o controlador

Um dos objetivos ao montar o tablet Android em uma caixa é evitar que seja danificado, de modo que, se for necessário retirá-lo do sistema de automação residencial por alguma razão, isso será possível.

Quando se faz a caixa para o controlador, a parte mais difícil é cortar, na tampa da caixa, uma janela suficientemente grande para permitir acesso à tela de toque do tablet Android. Isso ficará bem mais fácil se você usar uma ferramenta de corte circular do tipo Dremel ou outra que tenha um disco de corte.

Planeje cuidadosamente o local onde o corte deverá ser feito na tampa e marque-o a lápis. Quando

você estiver absolutamente seguro de que todas as coisas estão nos seus lugares, corte a janela (Figura 7-18).

Observe que o autor incluiu também um corte em V para permitir que o botão "Voltar" (Back) possa ser pressionado no tablet.

Precisamos de mais alguns furos na caixa, incluindo um para o cabo de força do tablet Android. Pode-se conseguir isso fazendo, no lado da caixa, um furo com diâmetro suficiente para deixar passar completamente o cabo, sem necessidade de modificá-lo.

Precisamos também de um pequeno orifício próximo da chave do tablet Android. Ele deve ter diâmetro suficiente para deixar passar uma chave de fenda, que será usada para ligar o dispositivo.

O último furo que precisamos fazer na caixa é para o jack do cabo de alimentação elétrica que soldamos para a placa de interface de áudio.

A posição exata desses furos dependerá do tamanho da sua caixa e do seu tablet Android. No próximo capítulo, faremos mais alguns furos na placa para fixar a placa perfurada que contém a eletrôni-

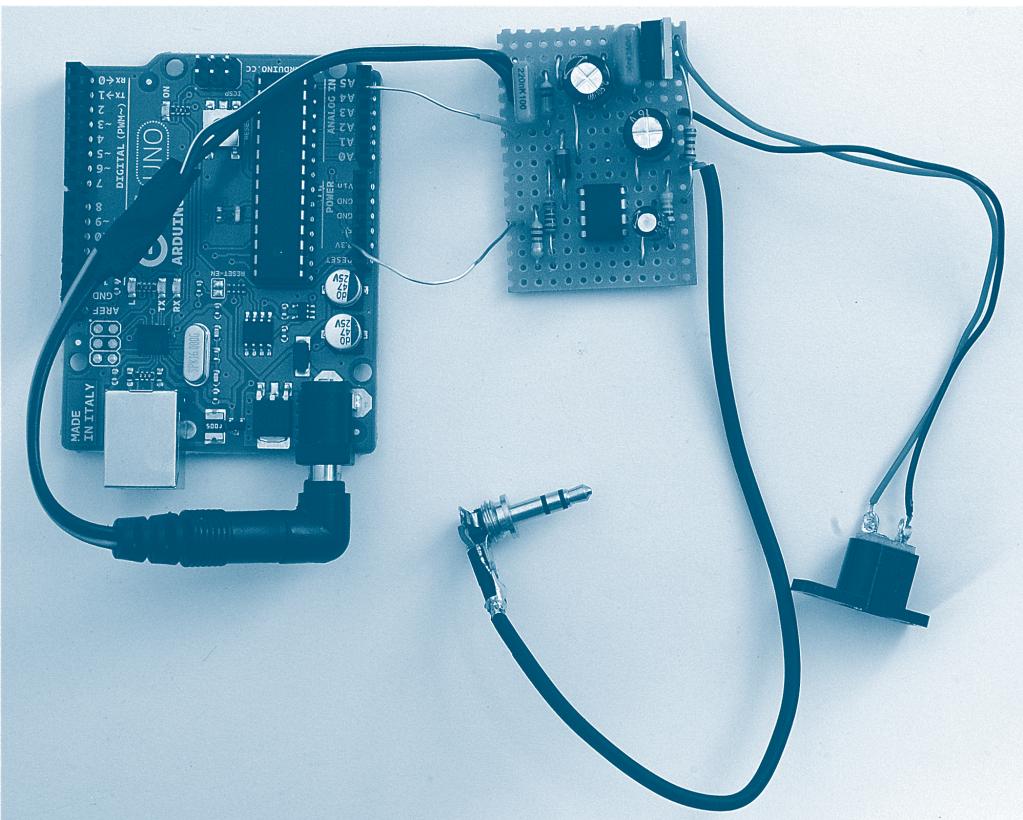


Figura 7-16 Fazendo as ligações para testar a interface de áudio.

ca do controle de potência. Opcionalmente, faremos um furo para permitir que um conector USB seja anexado à placa de Arduino. Por essa razão, talvez você queira esperar até o próximo capítulo antes de finalizar a construção da caixa. No entanto, a maioria dos entusiastas é conhecida por sua

falta de paciência. Portanto, é possível que você “sinta necessidade” de fazer logo os furos para ver como ficarão na caixa.

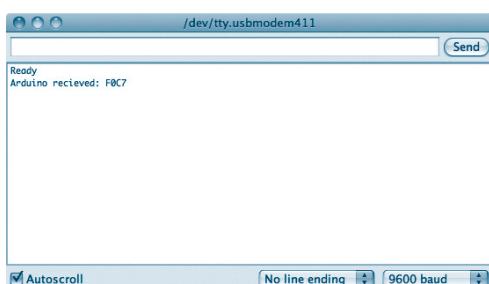


Figura 7-17 O Serial Monitor.

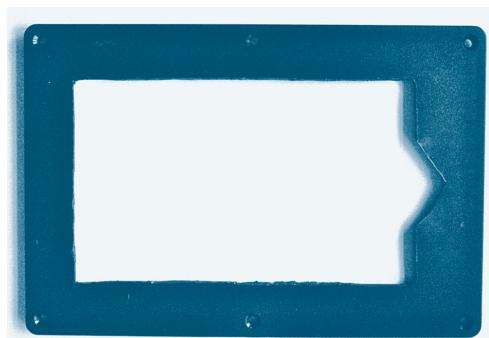


Figura 7-18 A tampa da caixa.

Para fixar o tablet na tampa, pode-se utilizar um pedaço de fita perfurada de metal. Esse material está disponível em lojas de ferragens e de hobby e é útil porque pode ser curvado e cortado facilmente para assumir a forma desejada. A fita de metal foi dobrada para encaixar sobre a parte de trás do tablet. Os seus furos são grandes o suficiente para permitir a passagem dos parafusos de fixação na base da caixa. Veja a Figura 7-19.

À medida que avançarmos nesta parte do livro dedicada à automação residencial, mais coisas serão acrescentadas a essa caixa de projeto.

» Software Android

Agora que você testou a conexão de áudio e acondicionou o tablet em uma caixa, chegou a vez de examinar o aplicativo Home Automation (automação residencial). Assim como o aplicativo de teste da interface de áudio (Sound test), este também pode ser baixado de www.duinodroid.com. Os códigos-fontes de ambos os aplicativos estão disponíveis gratuitamente para o caso de você querer modificá-los para seus próprios objetivos. Depois de baixado e instalado, deixe conectados a interface de áudio e o Arduino para que você possa ver as mensagens que o tablet Android está enviando.

A estrutura do software Android (Figura 7-20) não é usual porque, na realidade, contém um pequeno servidor de Web. Esse servidor atende a solicita-



Figura 7-19 A fixação do tablet.

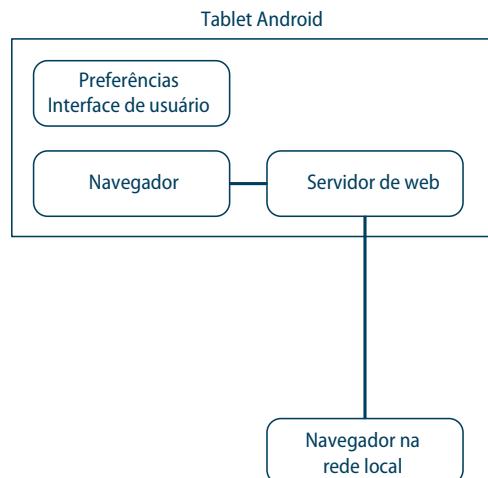


Figura 7-20 A estrutura do aplicativo Android.

ções HTML enviadas pelo navegador (browser) do dispositivo, de modo que a principal interface de usuário do aplicativo é o navegador de Web que é executado no mesmo tablet Android em que está instalado o pequeno servidor de Web.

Isso significa que outros dispositivos da rede local também poderão se conectar ao tablet Android e, se você configurar corretamente o seu modem sem fio residencial, você também poderá se comunicar através da Internet com o aplicativo Home Automation (automação residencial).

Se você permitir o acesso pela Internet, será necessário fazer um alerta aqui. Se alguém estiver navegando na Internet e encontrar o seu servidor de automação residencial, praticamente nada haverá em termos de segurança que você possa fazer para impedir que esse alguém assuma o controle de sua casa. Portanto, talvez seja melhor não se arriscar.

A Figura 7-21 mostra a primeira tela que você vê quando o aplicativo é iniciado.

A primeira linha diz que o servidor está funcionando com o URL exibido. Nesse caso, “<http://10.0.2.15:8080/home?>”.

Primeiro, contudo, vamos clicar em “Preferences” (preferências), que abrirá uma segunda tela (Figura 7-22).

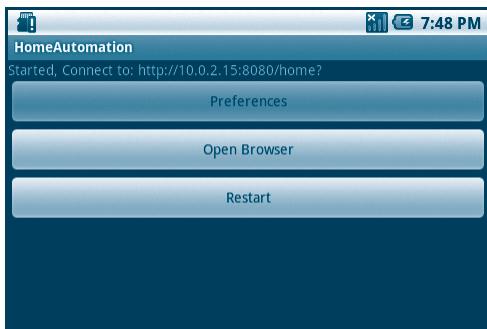


Figura 7-21 A tela inicial do aplicativo de automação residencial*.



Figura 7-23 O menu principal.



Figura 7-22 A tela de preferências**.

A tela de Preferences permite que você mude os nomes das tomadas e luminárias que serão controladas. Se você descer até o final da tela, verá uma opção para criar uma senha (Password). Essa senha será necessária quando você abrir a porta. Não se preocupe em configurar as preferências agora. Você poderá retornar a qualquer momento e alterá-las. Por enquanto, simplesmente clique em "Voltar" (Back) para retornar ao menu principal e, em seguida, clique em "Abra o navegador" (Open Browser), o qual abrirá o navegador do dispositivo Android na página principal do sistema de automação (Figura 7-23).

Se o seu computador estiver na mesma rede que o seu dispositivo Android, você poderá copiar o

URL que está no topo da Figura 7-23 e digitá-lo no navegador do seu computador para ver a mesma tela. Isso significa que você pode controlar a sua casa de qualquer dispositivo conectado a sua rede doméstica usando apenas um navegador. Isso inclui laptops, netbooks, iPhones, iPads e iPods, assim como a maioria de outros smartphones.

Como funciona o aplicativo?

No menu principal, teremos diversas opções de controle da casa. Infelizmente, por enquanto, nada elas farão, porque ainda não construímos o hardware para controlar as tomadas, os aquecedores e a porta. No entanto, podemos ao menos ter uma ideia das opções que aos poucos estarão disponíveis à medida que o nosso trabalho avançar nesta parte do livro.

Clicando em "Power" (potência, energia elétrica), serão abertas as opções de controle de tomadas e luminárias (Figura 7-24).

Nessa tela, você poderá ligar (On) e desligar (Off) tomadas e luminárias individualmente. Lembre-se de que você pode, a qualquer momento, voltar ao início do aplicativo Home Automation para alterar as preferências dando nomes diferentes aos canais. Simplesmente clique no botão "Android Home" e traga o aplicativo Home Automation para a frente.

Tente clicar em "On" (ligar), próximo da primeira tomada. Você verá um número surgir no Serial Monitor. Desconecte o plugue de áudio do tablet Android e você deverá ouvir os diferentes sons que você produz quando cada botão é pressionado.

* N. de T.: Preferences = Preferências, Open Browser = Abra o navegador e Restart = Reinicie.

** N. de T.: Outlet = Tomada, Lights = Luminárias e Password = Senha.

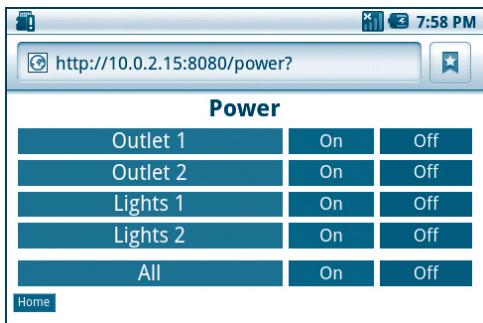


Figura 7-24 As tomadas de energia elétrica (outlet) e as luminárias (light).

Na parte debaixo da janela do navegador, você verá um botão verde de nome “Home” (página inicial). Clique nele para retornar ao menu principal e, em seguida, clique em “Bedtime” (hora de dormir). Quando você está pronto para ir dormir, esse recurso é interessante (Figura 7-25) porque pode ser utilizado para desligar tudo. Após pressionar o botão, passarão 10 minutos antes que todas as tomadas e luminárias sejam desligadas. Isso dá tempo suficiente para chegar até seu quarto antes que tudo fique escuro.

Se mudar de ideia, você sempre poderá clicar em “Back” (voltar), antes que o tempo marcado pelo temporizador expire, e programar novamente as luminárias e as tomadas para que não sejam desligadas.

A próxima opção (Figura 7-26) do menu principal é “Timers” (temporizadores, marcadores de tempo).

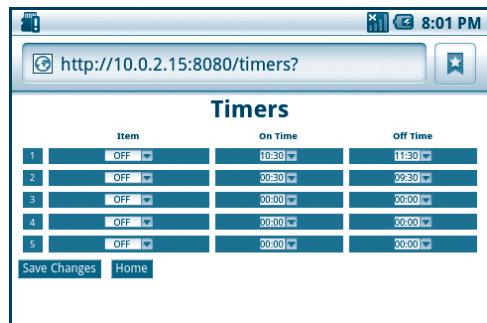


Figura 7-26 Temporizadores (Timers).

Essa opção permite que até cinco eventos ocorram em momentos previamente programados.

Para cada evento, você pode escolher uma luminária ou uma tomada para ser ligada em um horário e desligada em outro horário. Se você ajustar o “off time” (horário de desligar) com um horário anterior ao de “on time” (horário de ligar), então o sistema entenderá que o horário de desligar será no dia seguinte.

As alterações nos temporizadores não serão salvas até que você clique em “Save Changes” (salve as alterações).

De volta ao menu principal, a próxima opção (Figura 7-27) será “Door Lock” (fechadura da porta).

Preocupado com a questão de segurança, você deve fornecer uma senha que coincida com a senha definida em Preferences antes que você possa trancar ou destrancar a porta.

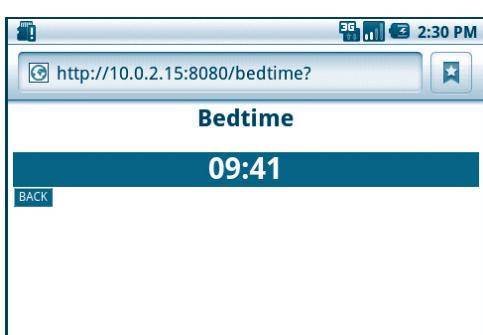


Figura 7-25 Hora de dormir (Bedtime).

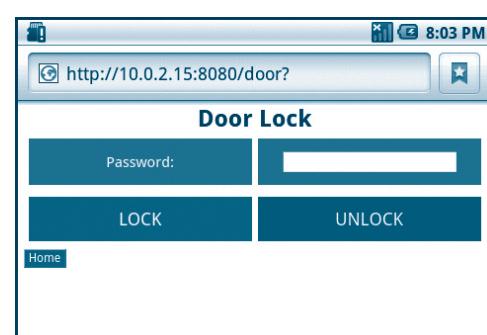


Figura 7-27 A tela de fechadura da porta (Door Lock).

A opção final do menu principal permitirá que você programe um perfil de temperatura para o sistema de aquecimento (Figura 7-28). As temperaturas que você define para os diversos horários do dia serão transferidas ao Arduino e, sem seguida, enviadas para o dispositivo “Termostato Inteligente” do Capítulo 9. Essa transmissão utiliza um enlace de RF Arduino-Arduino. Veja mais detalhes desse recurso no Capítulo 9.

» Acesso à Internet

Provavelmente, com o navegador de um computador conectado à rede WiFi de sua casa, você já tentou usar o URL do seu controlador de automação residencial para acessá-lo. Lembre-se de que seu tablet Android está funcionando como um servidor de Web. Desse modo, você poderia ir mais além e acessar o controlador de automação residencial de qualquer lugar pela Internet. Entretanto, para isso, você deverá alterar algumas configurações na sua rede WiFi doméstica. Isso é assim porque, por configuração inicial, o roteador WiFi não permite que um computador da sua rede doméstica possa ser acessado diretamente pela Internet. Para tornar isso possível, você tem que permitir o acesso ao tablet Android através da porta 8080.

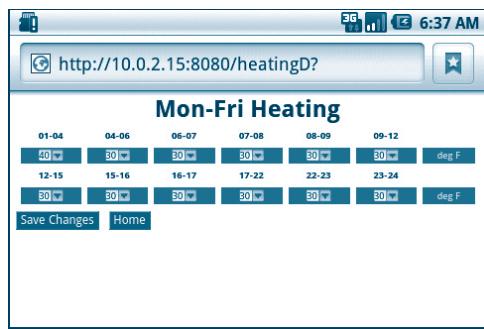


Figura 7-28 A programação do sistema de aquecimento de segunda a sexta-feira (Mon-Fri Heating).

Tecnicamente, você está tentando abrir a porta 8080 para permitir a entrada de solicitações (requests) HTTP destinadas ao tablet Android.

Para cada modelo de roteador WiFi, o procedimento para fazer isso é ligeiramente diferente. As instruções seguintes são do roteador utilizado aqui. No seu caso, você verá que as coisas serão ligeiramente diferentes.

Usando o seu navegador, o primeiro passo é fazer “log in” na página de gerenciamento do seu roteador.

Na página principal de status (Figura 7-29), após o “log in”, você verá alguns números. Procure algo como “external IP address” (endereço IP externo) e anote o “IP address”. Você precisará dele quando estiver se conectando de fora da sua rede.

Agora, você deverá procurar por uma opção semelhante a “NAT Virtual Server” (Figura 7-30). Isso permitirá que qualquer solicitação Web destinada ao seu roteador seja redirecionada para o tablet Android. Observe a alteração marcada com um círculo. O ponto-chave é obter corretamente o endereço local de IP do tablet Android. Observe que isso poderá mudar, a menos que você tenha ajustado o “lease time DHCP” para ser o maior número de dias possível. Você encontrará essa opção na página de configuração DHCP do seu hub doméstico.

Quando você achar que fez as alterações necessárias no seu roteador, você poderá fazer um teste pedindo que alguém de fora da sua rede doméstica faça um acesso. Para causar uma impressão profunda nas pessoas, utilize o navegador de um smartphone que tenha 3G e que esteja com o WiFi desligado.

Desse modo, se seu endereço IP externo for 98.76.543.21, entre com o URL <http://98.76.543.21:8080/home?> na barra de endereço do navegador do seu celular. Você deverá ver a página inicial aparecer na sua tela. Se você achou isso emocionante, espere até poder realmente ligar e desligar coisas!

Service Information					
LAN Interface					
IP Address	Subnet	MAC Address	Status		
192.168.1.1	255.255.255.0	00:21:63:3b:c5:00	<input checked="" type="checkbox"/>		
Port	Speed	Duplex	Status		
1	-	-	<input checked="" type="checkbox"/>		
2	-	-	<input checked="" type="checkbox"/>		
3	-	-	<input checked="" type="checkbox"/>		
4	-	-	<input checked="" type="checkbox"/>		
WAN Interface					
PVC	VPI/VCI	IP Address	Subnet	Gateway	Encapsulation
PVC-0	0 /38	92.16.0.22	255.255.255.255	92.16.0.1	Route -PPPoA <input checked="" type="checkbox"/>

Figura 7-29 A página de status do roteador.

NAT - Virtual Server	
NAT - Virtual Server	
Virtual Server for	Single IP Account
Rule Index	1
Application	Home Automation
Protocol	TCP
Start Port Number	8080
End Port Number	8080
Local IP Address	192.168.1.4
Start Port(Local)	8080
End Port(Local)	8080

Figura 7-30 A configuração do servidor virtual NAT.

» Teoria

Este é um projeto complexo que contém muitos conceitos teóricos interessantes para serem explorados.

» Codificação de dados com som

Há algumas décadas, o computador não tinha um disco rígido (HD) nem um disco flexível. Tinha apenas alguns fios que levavam a um velho gravador cassete de áudio.

Para carregar um jogo no computador, era preciso colocar primeiro uma fita-cassete no gra-

vador, rebobiná-la e apertar Play (com o botão Pause pressionado). Após, era necessário digitar LOAD (carregar) no computador, liberar o botão Pause e, então, apertar ENTER logo que a parte inicial da fita, que precedia os dados gravados, terminava de passar pela cabeça de leitura do gravador. Em seguida, uma série de sons estranhos era ouvida por diversos minutos, antes que o programa massivo de 3 quilobytes terminasse de ser carregado.

Esses sons estranhos representavam os dados do programa que tinham sido codificados na forma de uma série de sons. Usamos exatamente essa mesma abordagem para fazer a comunicação do nosso tablet Android com o Arduino.

O traçado no osciloscópio da Figura 7-31 mostra 16 bits de dados codificados como sons.

Esse traçado é a saída bruta da saída de áudio. Você pode ver 16 pulsos: quatro longos, quatro curtos, dois longos, três curtos e, finalmente, três longos. Um pulso de duração curta representa um 0 e um pulso de duração longa representa um 1. Desse modo, se escrevermos o número em binário, teremos 1111 0000 1100 0111, que em hexadecimal é F0C7, o número que enviamos.

Se ampliarmos o sinal (Figura 7-32), veremos que, na realidade, cada pulso é constituído de uma onda senoidal de frequência mais elevada, estando modulada na forma de pulsos. Examinando bem de perto o sinal, podemos ver que o período de cada ciclo é exatamente uma divisão ou 1 ms. Portanto, a frequência desse sinal é 1 kHz. Examinando novamente a Figura 7-31, pode-se ver que é possível inserir cerca de quatro pulsos em cada divisão de 250 ms. Portanto, cada pulso pode ser encaixado em um intervalo de aproximadamente 62 ms. Quando olhamos o código Android para gerar esse sinal, vemos que esse número é, na realidade, 64 ms e que um pulso longo dura 32 ms e um curto, 8 ms.

Além de baixar o aplicativo Android que está pronto para ser usado, os códigos-fontes completos para o aplicativo de teste da interface de áudio (Sound Test) e para o controlador de automação residencial são softwares abertos, estando disponíveis sob uma licença GPL. Você pode baixar os arquivos dos códigos-fontes de www.duinodroid.com.



Figura 7-31 Traçado de osciloscópio da saída do tablet Android.

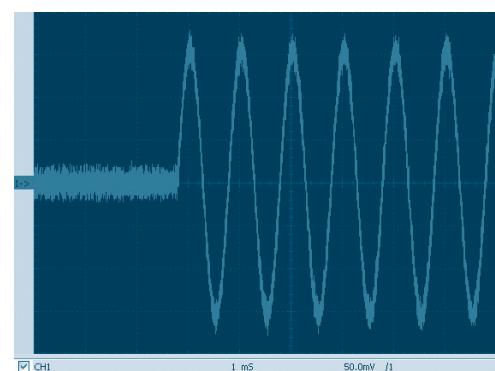


Figura 7-32 Ampliação do sinal.

A classe principal que gera os pulsos sonoros corretos é denominada Beeper, e a sua listagem é dada a seguir:

As constantes no início do arquivo definem a frequência e a duração dos pulsos que o sistema

```
import android.media.AudioFormat;
import android.media.AudioManager;
import android.media.AudioTrack;

public class Beeper {

    private final static int SAMPLE_RATE = 16000;

    private final static int ONE_DURATION = 32;
    private final static int ZERO_DURATION = 8;
    private final static int BIT_DURATION = 64;
    private final static int DURATION = BIT_DURATION * 32;
```

```
private final static float f = 1000.0f;

private short[] buffer = null;

public void beep(int word) {
    AudioTrack at;
    int bufsizbytes = DURATION * SAMPLE_RATE / 1000;
    int bufsizsamps = bufsizbytes / 2;
    buffer = new short[bufsizsamps];
    fillbuf(word, bufsizsamps);
    try {
        at = new AudioTrack(AudioManager.STREAM_MUSIC, SAMPLE_RATE,
            AudioFormat.CHANNEL_CONFIGURATION_MONO,
            AudioFormat.ENCODING_PCM_16BIT, bufsizbytes,
            AudioTrack.MODE_STATIC);
        at.setStereoVolume(1.0f, 1.0f);
        at.write(buffer, 0, bufsizsamps);
        at.play();
    } catch (IllegalArgumentException e) {
        e.printStackTrace();
    }
}

void fillbuf(int word, int bufsizsamps) {
    double omega, t;
    double dt = 1.0 / SAMPLE_RATE;
    t = 0.0;
    omega = (float) (2.0 * Math.PI * f);
    for (int i = 0; i < bufsizsamps; i++) {
        if (toneRequired(t, word)) {
            buffer[i] = (short) (32000.0 * Math.sin(omega * t));
        } else {
            buffer[i] = 0;
        }
        t += dt;
    }
}

boolean toneRequired(double t, long word) {
    int ms = (int) (t * 1000);
    int bitIndex = ms / BIT_DURATION;
    int bit = (int) ((word >> (15 - bitIndex)) & 1);
    int msWithinBit = ms - (bitIndex * BIT_DURATION);

    if (bit == 1 && msWithinBit < ONE_DURATION) {
        return true;
    }
    if (bit == 0 && msWithinBit < ZERO_DURATION) {
        return true;
    }
    return false;
}
}
```

utilizará. Essa classe do sistema de interface de áudio poderia ser estendida facilmente para permitir taxas mais velozes e quantidades maiores de dados. Para isso, deveríamos aumentar a frequência da portadora (f) e a taxa de amostragem (`SAMPLE_RATE`), além de diminuir a duração dos pulsos. Entretanto, como não há muita coisa para enviar, preferimos dar prioridade à confiabilidade e não ao desempenho. Observe que, se você realmente pretender mudar esses parâmetros, então provavelmente terá que alterar também a eletrônica.

O único método público da classe é denominado “beep”. Ele recebe o número de 16 bits a ser enviado e converte-o em uma série de pulsos.

Na realidade, para gerar os pulsos, deveremos reunir em um buffer cada uma das partes que constituem a forma de onda e, em seguida, executar esse conteúdo como se fosse um arquivo de áudio. O método “fillBuffer” (preencher o buffer) encarrega-se da maior parte desse trabalho, gerando a onda senoidal com o uso da função trigonométrica seno (`sin`). Isso só ocorre quando o método “toneRequired” (som requerido) diz que um som deve ser gerado. Para isso, ele verificará qual bit deve ser codificado como som, retornando “true” (verdadeiro) se o tempo dentro do bit for inferior à duração do bit, dependendo se é um 1 ou um 0.

» A eletrônica da interface de áudio

O diagrama esquemático da interface de áudio está repetido na Figura 7-33, em que o ponto de teste “A” aparece destacado.

A tarefa da eletrônica é receber o pequeno sinal do dispositivo Android – provavelmente em torno de 300mV pico a pico (Figura 7-31) – e produzir uma série de pulsos de nível lógico (0–5V) para o Arduino. Isso é obtido através de uma série de estágios, como está resumido na Figura 7-34.

O primeiro passo é amplificar o sinal do tablet Android. Faremos isso utilizando um CI de amplificador operacional configurado para um ganho de 1000. Isso significa que o sinal de saída será 1000 vezes maior que o sinal de entrada. Bem, se esse fosse realmente o caso, então a saída teria 300V. No entanto, a saída é limitada pelas tensões de alimentação do amplificador operacional, que, no caso, são +5V e -5V. Desse modo, o sinal será amplificado até um máximo possível limitado. Entretanto, mesmo que o sinal do tablet seja muito fraco, ainda será suficientemente amplificado para atingir um nível apropriado.

Antes de avançar para o próximo estágio, vale a pena examinar como o circuito recebe a alimentação elétrica. O sinal vindo do tablet Android oscila para ambos os lados em torno da tensão do terra (GND). Para amplificá-lo, o amplificador operacional necessita de uma fonte de alimentação simétrica, ou seja, precisará de GND, +5V e -5V. Usando de forma engenhosa a fonte interna de 5V do Arduino, poderemos obter as tensões de +5V e -5V, além de 9V para o Arduino.

Examinando a metade debaixo da Figura 7-33, vemos que a nossa alimentação elétrica básica é de 15V vinda do adaptador de voltagem CA/CC. O seu lado mais negativo fornece os -5V do amplificador operacional. A seguir, utilizamos um regulador de tensão negativa de 9V para fornecer uma tensão regulada que está 9V abaixo dos 15V. Essa tensão resultante será o GND do amplificador operacional e do Arduino. Essa queda de 9V também alimenta o Arduino. Os +5V do regulador de tensão do Arduino fornecem os +5V do amplificador operacional.

Após a amplificação, o próximo estágio do sinal será a sua retificação. Se aplicarmos tensões negativas a um pino de entrada, o nosso Arduino poderá ser danificado. Por isso, precisamos nos assegurar de que a tensão estará sempre entre GND e +5V. O diodo realiza essa tarefa. Entretanto, ainda continuamos com o problema de que cada pulso é,

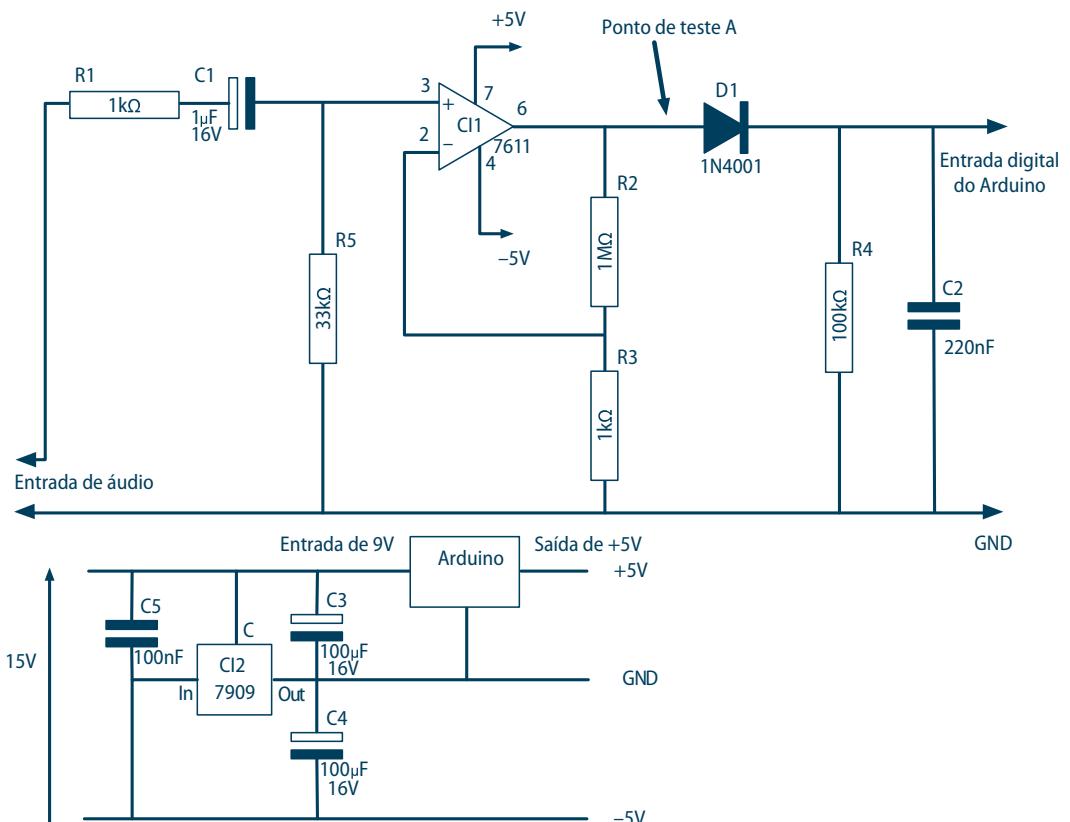


Figura 7-33 O diagrama esquemático da interface de áudio.

na realidade, uma onda de 1kHz, sendo necessário remover esse sinal senoidal e ficar apenas com o “envelope” dos pulsos. Para conseguir isso, usaremos o filtro passa-baixa constituído pelo resistor R2 e o capacitor C2. Esse filtro produz também um

pouco de distorção no sinal, mas isso não é importante no nosso caso.

A Figura 7-35 mostra os traçados de osciloscópio do sinal. O traçado de cima mostra o sinal amplifi-

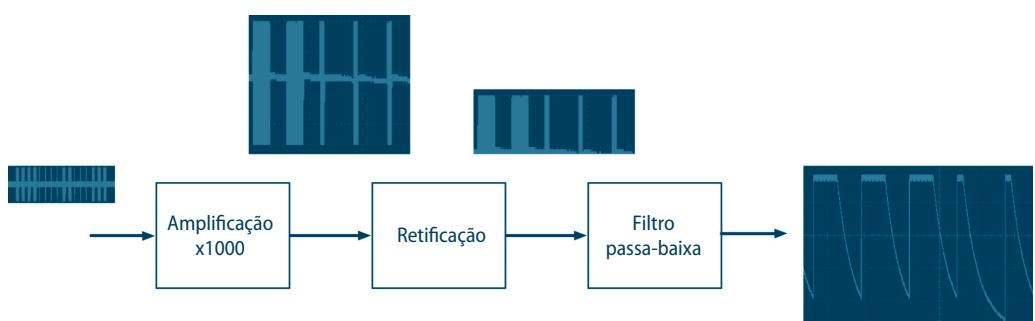


Figura 7-34 Diagrama de blocos da eletrônica.

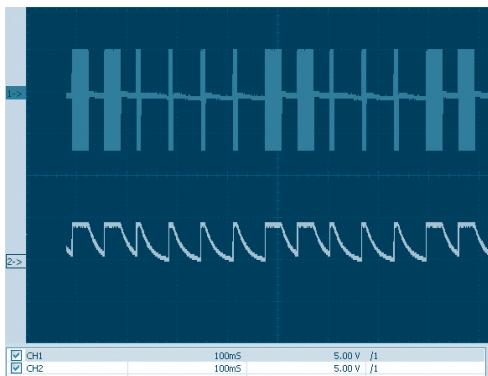


Figura 7-35 Retificação e filtragem.

cado no ponto de teste A e o traçado debaixo mostra o sinal depois da retificação e do filtro passa-baixa.

A Figura 7-36 é uma vista detalhada do sinal final. A entrada digital do Arduino tratará qualquer coisa acima da linha de 3V como sendo um 1 e qualquer coisa abaixo como sendo um 0. Portanto, nesse nível, um pulso longo terá uma duração de 40 ms e um pulso curto, cerca de 20 ms.

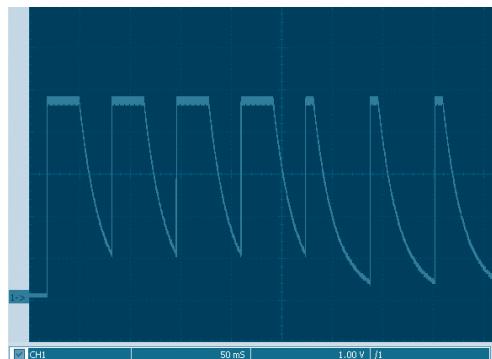


Figura 7-36 Vista ampliada do sinal final.

» Decodificação dos sons no Arduino

Finalmente, o nosso sinal está no Arduino. Agora, precisamos apenas de um modo de decodificar os pulsos de duração variável convertendo-os em um número.

O sketch de teste capaz de fazer essa decodificação está mostrado na listagem seguinte.

```
#define soundPin 18
#define zeroDurationFrom 10000
#define zeroDurationTo 25000
#define oneDurationFrom 35000
#define oneDurationTo 50000
#define resetTimeout 3000

void setup()
{
    pinMode(soundPin, INPUT);
    Serial.begin(9600);
    Serial.println("Ready");
}

unsigned int result;
int bitNo = 0;
long lastPulseTime = 0;

void loop()
{
    long pulseLength = pulseIn(soundPin, HIGH, oneDurationTo * 2);
    long timeSinceLastPulse = millis() - lastPulseTime;
```

```

lastPulseTime = millis();
if (pulseLength == 0 || timeSinceLastPulse > resetTimeout)
{
    bitNo = 0; result = 0;
}
else
{
    if (pulseLength >= zeroDurationFrom && pulseLength <= zeroDurationTo)
    {
        result = result << 1;
        bitNo++;
    }
    else if (pulseLength >= oneDurationFrom && pulseLength <= 50000)
    {
        result = (result << 1) + 1;
        bitNo++;
    }
    else
    {
        Serial.print("Error pulseLength="); Serial.println(pulseLength);
    }
}
if (bitNo == 16)
{
    Serial.print("Arduino recieved: ");
    Serial.println(result, 16);
    bitNo = 0; result = 0;
}
}

```

A primeira coisa que fazemos no sketch é definir algumas constantes para as durações máxima e mínima dos pulsos. Há também um tempo de espera (timeout) após o qual o software desiste de ler o número e começa tudo de novo desde o primeiro bit.

Qualquer “sketch” de Arduino, como é denominado um programa no mundo do Arduino, contém as funções “setup” (inicialização) e “loop” (laço de repetição). Quando o Arduino é submetido a um “reset” (inicialização), a função “setup” é executada uma vez, ao passo que a função “loop” é repetida um número indefinido de vezes.

A função “setup” define que o pino A4 será o pino pelo qual receberemos os pulsos. Em seguida, ela inicializa a conexão serial para que o Arduino possa enviar mensagens ao Serial Monitor pela con-

xão USB e, finalmente, envia a mensagem “Ready” (pronto) ao computador.

A seguir, vamos definir algumas variáveis. A variável “result” (resultado) conterá, no final, o número enviado ao Arduino. O valor de “result” é construído literalmente bit a bit, e a variável “bitNo” é utilizada para contar quantos bits já foram lidos. Finalmente, a variável “lastPulseTime” (tempo do último pulso) contém o instante em que o último pulso foi recebido. Isso permite ativar o mecanismo do tempo de espera, para o caso de não chegar mais algum pulso.

A maior parte das atividades desse sketch ocorre na função “loop”. Ela é chamada repetidas vezes indefinidamente. A cada vez, a primeira coisa que acontece é chamar a função “pulseIn” (entrada de pulso) da biblioteca Arduino. Essa função espera

por um pulso, mas, se nenhum chegar dentro do período de tempo especificado pelo terceiro argumento (nesse caso, o dobro da duração de um “um”), a função retornará um 0.

As linhas seguintes verificam se o tempo de espera expirou. Isso pode ocorrer porque “pulseIn” expirou ou porque o tempo decorrido desde a chegada do último pulso foi excedido. Se for esse o caso, começamos novamente desde o início, e as variáveis bitNo e result voltam a ser inicializadas.

Caso contrário, testamos a duração do pulso para verificar se é um 1 binário ou um 0 binário. Em seguida, atualizamos a variável “result” e incrementamos o contador de bits.

A última coisa que fazemos na função “loop” é verificar se já lemos todos os 16 bits. Se for este o caso, ela enviará o número resultante à porta serial, de onde será repassado ao Serial Monitor.

» Resumo

Com isso concluímos o projeto descrito neste primeiro capítulo da segunda parte do livro. Agora, temos um controlador básico para automação residencial. No próximo capítulo, faremos algo útil com ele: ligaremos e desligaremos os nossos aparelhos elétricos.



» capítulo 8

Controlador de potência

No capítulo anterior, você aprendeu como construir um controlador de automação residencial. Neste capítulo, como primeira aplicação, utilizaremos o sistema de automação residencial para controlar luminárias e tomadas.

Objetivos

- » Acrescentar controladores de lâmpadas e tomadas ao controlador de automação residencial
- » Entender como funciona e como construir uma placa controladora de potência
- » Aprender a integrar a placa controladora de potência ao controlador de automação residencial
- » Conhecer os relés utilizados neste projeto
- » Examinar o sketch do Arduino utilizado neste projeto

É perfeitamente possível comprar unidades de automação residencial com controle remoto que se conectam a luminárias e tomadas utilizando o protocolo X10. Entretanto, elas são muito caras e, para baixar custo, decidimos utilizar tomadas com controle remoto de RF (radiofrequência) à venda em lojas de material elétrico (Figura 8-1). Por uns 25 dólares, pode-se dispor de três ou quatro tomadas CA comandadas por controle remoto, além do próprio controle remoto.

Observe que o modelo mostrado destina-se ao mercado inglês, cujas tomadas têm uma configuração própria. Unidades similares com formatos diferentes estão disponíveis para outros mercados, como o americano, o asiático, o europeu continental e outros.

Modificaremos o controle remoto para que possa ser acionado pelo Arduino do nosso sistema de automação residencial. De fato, o Arduino é quem “apertará” os botões do controle.

Utilizaremos essas tomadas com controle remoto de baixo custo para controlar as tomadas das luminárias e de outros aparelhos. Para as luminárias, precisaremos modificar as ligações para torná-las adequadas ao controle das luminárias residenciais.



Figura 8-1 Uma tomada de parede com controle remoto.

» A eletrônica do controlador de potência

Para desenvolver este projeto, foi necessária uma certa dose de engenhosidade. Utilizaremos um controle remoto já existente e alguns relés pequenos do tipo reed para fazer o equivalente a apertar os botões do controle remoto.

Tudo isso será construído em uma placa, permitindo que o Arduino comande o controle remoto das tomadas. Se quiséssemos, poderíamos utilizar esse módulo de forma inteiramente independente do tablet Android para controlar diretamente as tomadas a partir do nosso computador usando a conexão USB. Na realidade, antes de começar a ligar coisas à interface de áudio e ao tablet Android, isso é exatamente o que faremos para realizar os testes.

A Figura 8-2 mostra a placa com a montagem. Observe que os componentes foram dispostos em uma placa perfurada grande porque estamos reservando espaço para acrescentar mais componentes à medida que avançarmos com os outros projetos do sistema de automação residencial. A placa perfurada utilizada não tem trilhas de cobre, como as utilizadas em alguns projetos anteriores. Ela tem apenas orifícios separados de 1/10 de polegada, sem nenhuma trilha na face oposta.

» Construção do módulo controlador de potência

A Figura 8-3 mostra o diagrama esquemático do módulo controlador de potência.

O Arduino tem oito relés-reed de baixa potência conectados às saídas digitais. Quando uma das entradas está em nível alto, ela energiza a bobina do relé, fechando a chave do relé. Os terminais dessa chave são conectados e soldados diretamente aos terminais do controle remoto, que são os que se fecham quando um botão é apertado.

O uso de relés pode parecer antiquado, mas uma abordagem mais elaborada usando transistores

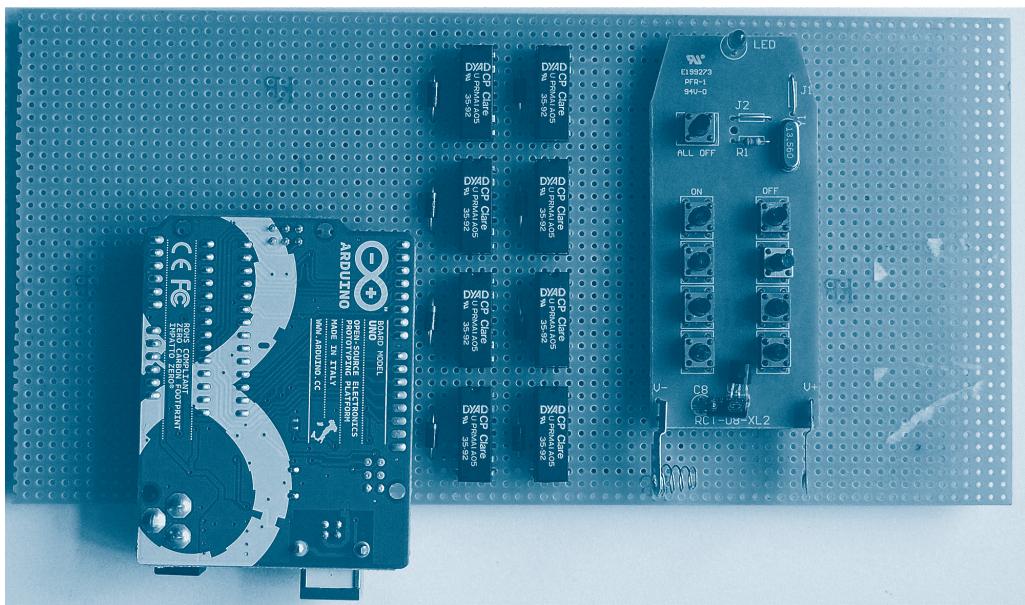


Figura 8-2 Controle de tomadas efetuado pelo Arduino.

dependeria muito do tipo de controle remoto usado.

» O que será necessário

Você precisará dos componentes listados na tabela *Lista de Componentes*, a seguir, para construir o módulo controlador de potência.

No mercado, há diversos kits de tomadas com controle remoto, e você deve escolher um que seja o mais semelhante possível ao usado aqui. Utilizamos aqui um kit com tomadas destinadas ao mercado inglês, que custou menos de 10 dólares e que contém três tomadas e um controle remoto. Foi comprado em um supermercado local. Nos Estados Unidos, há outros kits similares, como os das empresas La Crosse (RS-204U), Stanley (31164) e Woods (13568). Desses, o da La Crosse é o que mais se aproxima do que utilizamos aqui. De qualquer forma, vale a pena dar uma boa olhada por aí porque novos kits estão sempre surgindo no mercado. O sistema que utilizamos está mostrado na Figura 8-1.

As características que são importantes em um kit de tomadas com controle remoto são:

- Essencial: deve ser com RF (radiofrequência) e NÃO com infravermelho.
- Essencial: deve usar botões separados para ligar e desligar cada tomada. Alguns controles remotos têm botões que alternam a função de comando entre ligar e desligar. Não use um controle remoto deste tipo.
- Desejável: quatro canais. Se você conseguir somente um kit com três pares de botões de ligar e desligar, então deixe de lado um dos pares de relés.
- Desejável: configuração dinâmica de tomadas. Alguns controle remotos têm configuração fixa, isto é, o botão 1 comandará somente a tomada que tem o rótulo com 1 impresso nela. Outros, como o usado aqui, permitem “programar” qualquer tomada para que fique associada a qualquer canal do controle remoto. Isso permite que você configure grupos de tomadas que podem todas ser comandadas a

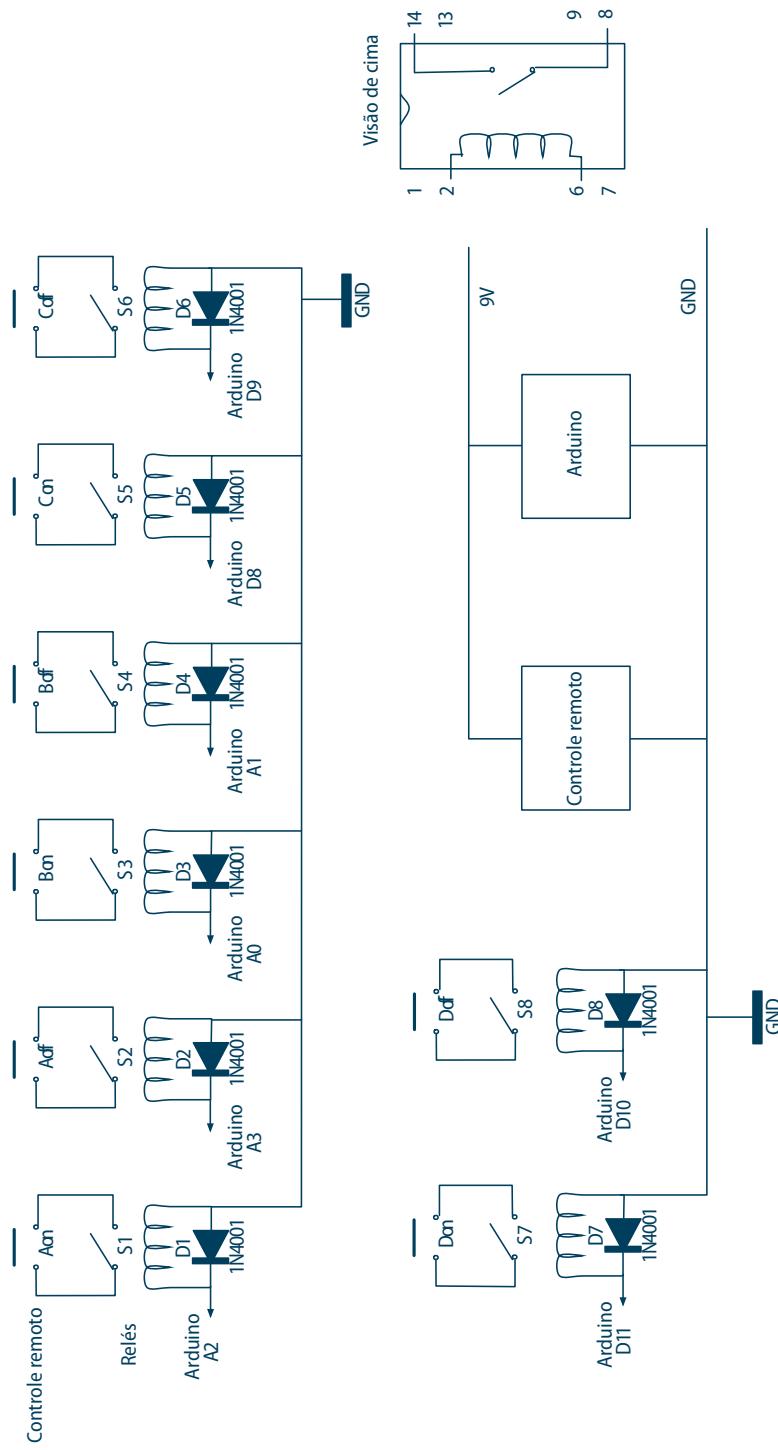


Figura 8-3 Diagrama esquemático do módulo controlador de potência.

LISTA DE COMPONENTES

Componente	Quantidade	Descrição	Fornecedor
Kit de tomadas com controle remoto	1	Veja a descrição que está no texto	eBay
Relés 1-8	8	Relés-reed de 5V, um contato NA	eBay
D1-8	8	Diodo 1N4001	Farnell: 1458986
Placa perfurada sem trilhas	1		Farnell: 1172145
Barra de pino macho	1	Barra de 40 pinos	Farnell: 1097954
Porcas e parafusos	4	Porcas e parafusos pequenos para fixar a placa perfurada	

partir do mesmo canal, proporcionando assim muita flexibilidade ao sistema. Na realidade, você poderá comprar tomadas extras para usar com o mesmo controle remoto.

Você precisará de oito relés-reed. Se você comprá-los de um fornecedor comum de componentes, eles poderão sair caros. No entanto, no eBay, você poderá encontrar lotes de 10 relés por um preço mais em conta. Procure os que têm a mesma disposição de pinos que os utilizados aqui (veja a Figura 8-3). Não deverá ser difícil encontrá-los porque a configuração dos pinos de relés está se tornando padrão.

Além desses componentes, você também precisará das seguintes ferramentas.

CAIXA DE FERRAMENTAS

- Ferramentas para soldar
- Fios flexíveis e rígidos de diversas cores
- Um multímetro

Para fazer uma comparação, um controle remoto de outro fabricante está mostrado nas Figuras 8-6 e 8-7. As semelhanças entre os dois estão bem claras.

» Passo 2: soldando fios à PCB* do controle remoto

Com base na Figura 8-4, vamos soldar fios a cada uma das oitos chaves na parte principal da placa e vamos soldar dois fios aos conectores da pilha na parte inferior da placa. Caso o seu controle remoto tenha botões “All on” (tudo ligado) e “All off” (tudo desligado), desconsidere-os neste projeto.

Se olharmos a placa por baixo (Figura 8-5), veremos os locais onde as chaves estão soldadas.

Na realidade, cada botão tem quatro terminais de conexão, mas só precisa encontrar dois que entram em curto quando se aperta o botão. No caso do controle remoto utilizado aqui, que tem um pequeno LED, isso poderá ser feito usando um pedaço curto de fio e observando o LED brilhar quando um botão é pressionado. O procedimento é o seguinte:

1. Coloque a pilha no controle remoto.
2. Escolha um botão e identifique as suas quatro conexões.
3. Use o fio para colocar em curto duas das conexões do botão enquanto observa se o LED brilha.

* N. de T.: Printed Circuit Board, ou Placa de Circuito Impresso.

» Passo 1: desmonte o controle remoto

A primeira coisa a destacar é que você estará quebrando a garantia do produto porque o controle remoto será desmontado e fios serão soldados à placa com os circuitos.

O controle remoto que utilizamos é o mostrado nas Figuras 8-4 e 8-5.

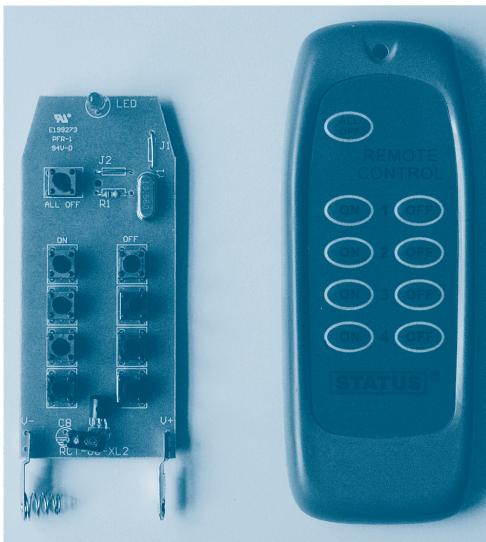


Figura 8-4 O controle remoto visto por cima.

4. Anote qual é a combinação de conexões que faz o LED acender.
5. Repita o procedimento para os demais botões. Provavelmente, você descobrirá que o mesmo padrão se repete em cada botão. De qualquer forma, verifique.

Agora, corte pedaços de fio rígido (10mm) e soldê-os nos pontos de contato da placa de circuito impresso (PCB) do controle remoto, como mostrado na Figura 8-8. Observe também os fios ligados aos terminais da pilha.

» Passo 3: coloque todos os componentes na placa perfurada

Agora, montaremos o controle remoto, os relés-reed e o Arduino em um pedaço de placa perfurada. Essa placa é semelhante à placa perfurada trilhada que utilizamos antes, mas sem as trilhas. Portanto, é apenas uma placa com furos distanciados de 1/10 de polegada entre si. A ideia é encaixar os componentes pela parte de cima e soldá-los na parte debaixo juntamente com os fios. A Figura 8-2

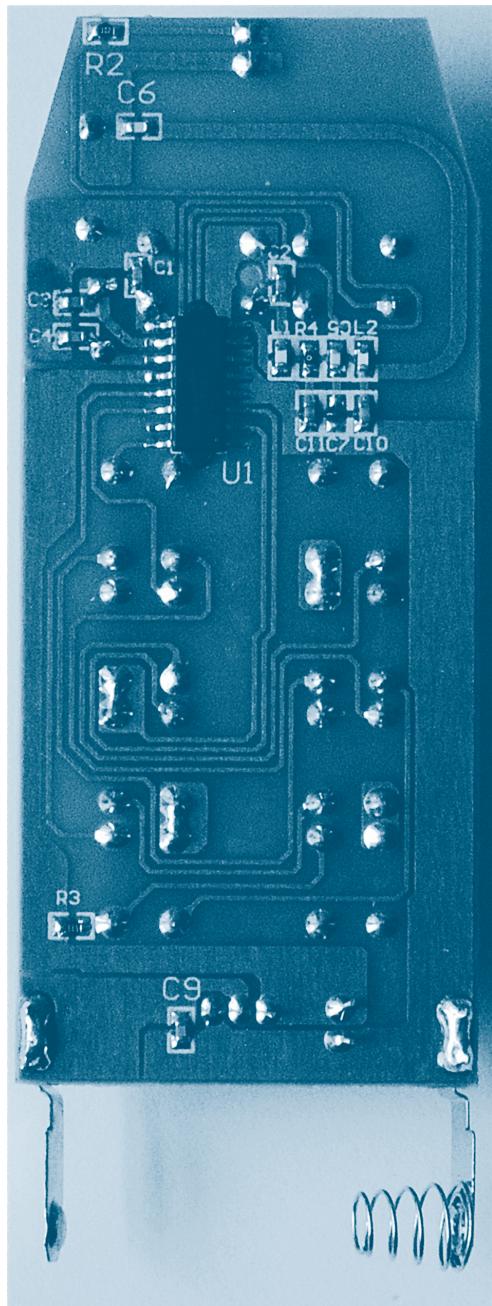


Figura 8-5 O controle remoto visto por baixo.

é o que queremos, e a Figura 8-9 mostra a disposição exata.

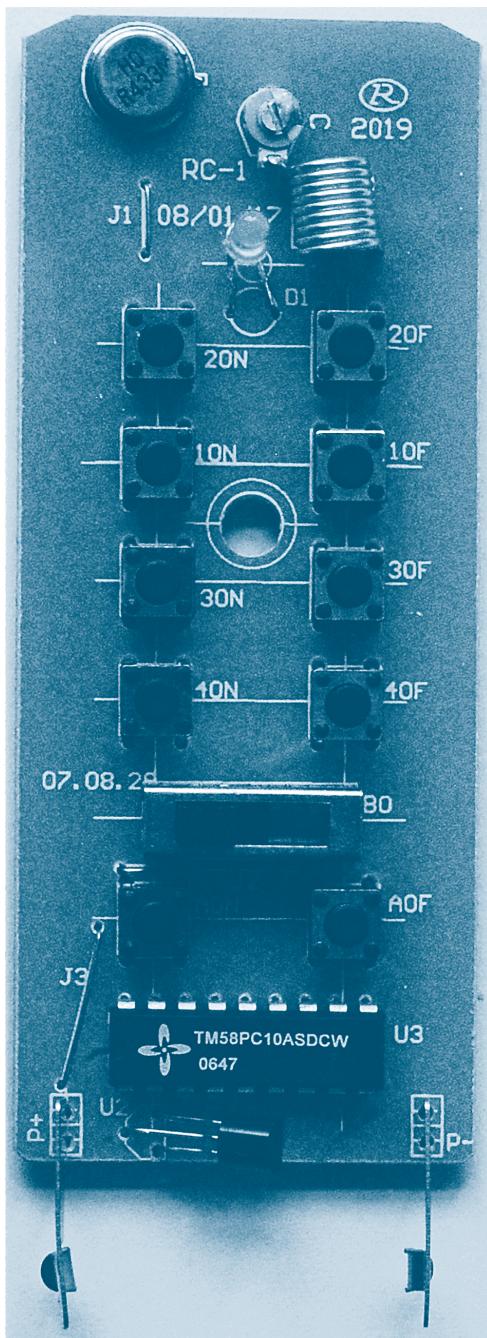


Figura 8-6 O controle remoto de outro fabricante visto por cima.

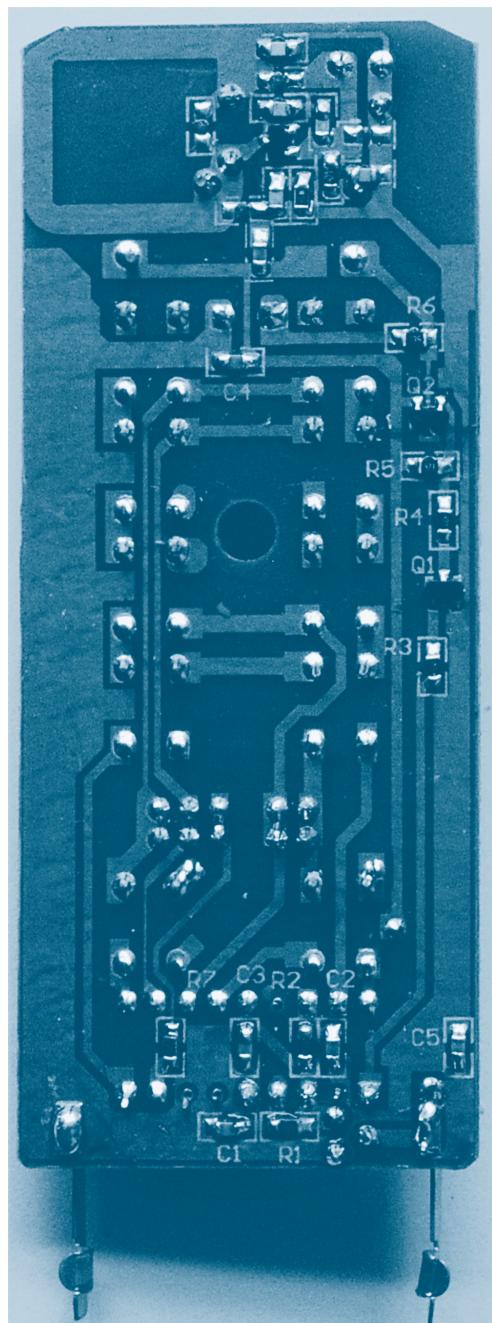


Figura 8-7 O controle remoto de outro fabricante visto por baixo.

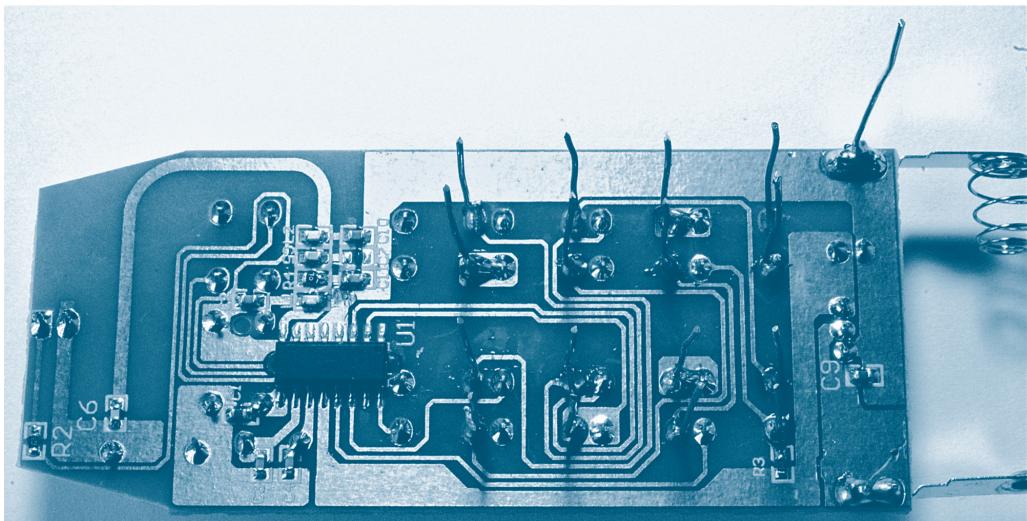


Figura 8-8 Soldagem dos fios às chaves dos botões.

No caso do seu controle remoto, é possível que seja necessário fazer ajustes na disposição dos componentes.

Pode ser trabalhoso fazer todos os fios do controle remoto ficarem alinhados perfeitamente com os furos da placa perfurada. Pode ser útil deixá-los tão retos quanto possível antes de encaixá-los nos furos da placa. Você pode usar uma chave de fenda para guiar os fios desalinhados até os seus furos. Quando todos os fios estiverem nos devidos lugares, dobre as pontas de alguns deles para assegurar que permaneçam no lugar.

Agora, podemos colocar os relés-reed, verificando se estão com a orientação correta (veja a disposição dos pinos na Figura 8-3). Em seguida, coloque os diodos, novamente conferindo a sua orientação.

A instalação da placa do Arduino é feita por barras de pinos e soquetes. Como você pode ver na Figura 8-10, nem todas as quatro barras de soquetes do Arduino são utilizadas. Há duas razões para isso. Primeiro, não precisamos de todas as conexões. Segundo, por alguma razão, de conhecimento dos projetistas do Arduino, o espaçamento entre os grupos de soquetes em um dos lados da placa do Arduino (o lado esquerdo na Figura 8-10 ou o

direito na Figura 8-2) não se encaixa corretamente no espaçamento de 1/10 de polegada da placa perfurada.

O Arduino será instalado usando barras de pino macho (ou simplesmente barras de pinos) na placa perfurada. Prepare três barras de pino macho, uma com oito pinos e duas com seis pinos cada.

A melhor maneira de colocar os pinos na posição correta é encaixar as barras de pinos nos soquetes da placa do Arduino (Figura 8-10) e, em seguida, encaixar os pinos nos respectivos furos da placa perfurada.

» Passo 4: solde os fios de conexão

Como são muitas as conexões que devem ser feitas, com os fios cruzando-se entre si, usaremos fio rígido isolado. Quando todas as ligações estiverem prontas, a parte de trás da sua placa deverá ser como a da Figura 8-10.

Não é uma má ideia fazer uma cópia da Figura 8-9 e marcar as conexões com uma caneta à medida que forem feitas.

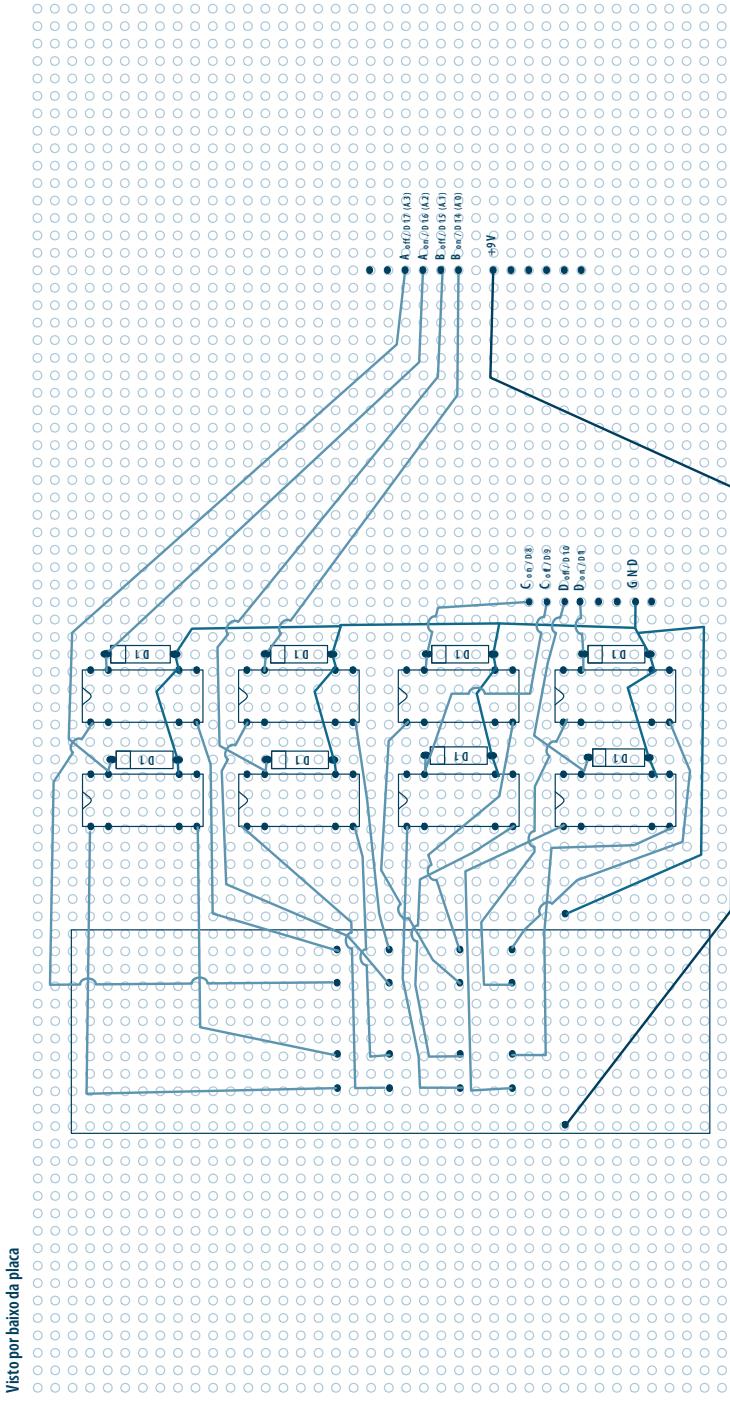


Figura 8-9 A disposição dos componentes na placa perfurada.

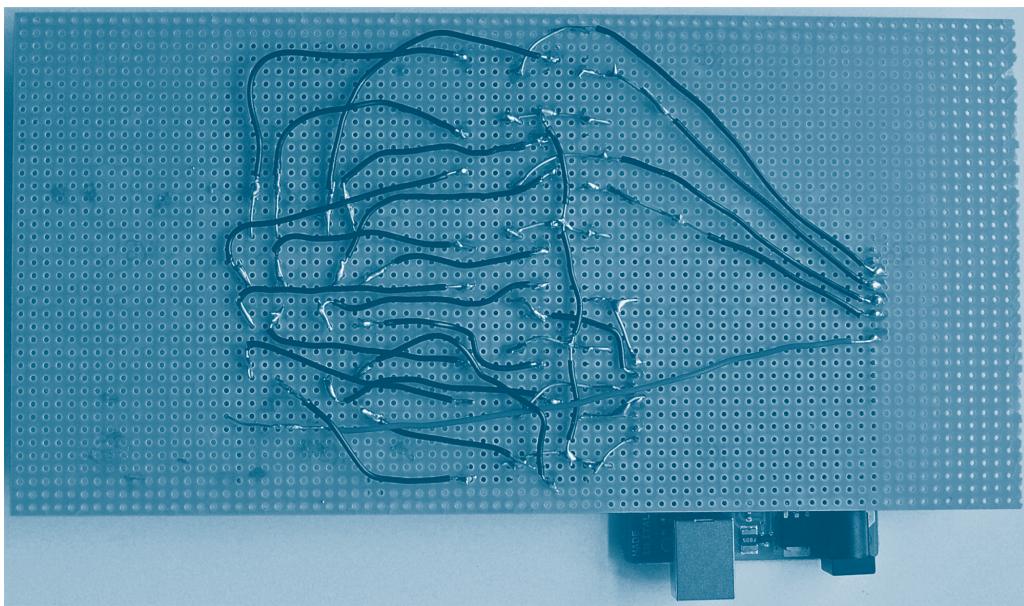


Figura 8-10 A parte de trás da placa perfurada.

Quando você terminar, volte ao início e confira tudo novamente.

Agora, estamos prontos para testar a placa.

» Passo 5: teste

Para testar a nossa criação, precisamos de uma fonte de alimentação de 9V. Se você não dispõe de uma, use o adaptador de voltagem CA/CC de 15V e a saída de 9V da interface de áudio que você construiu no Capítulo 7.

O site www.duinodroid.com tem um sketch de teste de potência (ch08_test_power) que podemos usar. Como já baixamos todos os sketches, tudo que falta fazer é transferir o sketch para a placa do Arduino. Se você não sabe como fazer isso, consulte o Passo 7 do Capítulo 7.

Conecte a alimentação de 9V ao Arduino e, no software do Arduino, abra o Serial Monitor (Figura 8-11).

As opções mostradas pelo sketch são as letras A a D, maiúsculas e minúsculas. No Serial Monitor,

quando você apertar em um “A” maiúsculo e, em seguida, clicar em “Return” (ou “Send”), o comando “A” será enviado ao Arduino. Esse, por meio de um relé, efetivamente “apertará” o botão On (ligar) do canal A do controle remoto. Se você apertar em um “a” minúsculo, o botão Off (desligar) será “pressionado”.

Para iniciar, simplesmente envie cada um dos comandos por vez e observe se o LED do controle remoto acende. É como se você estivesse apertando manualmente os botões. Se isso não acontecer para um dado comando, confira o diagrama de fiação da Figura 8-9 e tente localizar o problema na sua placa.

Agora, vamos à parte mais legal!

Encaixe uma das tomadas de controle remoto em uma tomada comum na parede. Em seguida, conecte uma luminária de mesa na tomada com controle remoto e verifique se a chave que faz parte da própria luminária está ligada. Se a sua tomada com controle remoto for do tipo que precisa de programação, configure-a para o primeiro canal do seu controle remoto. No caso do controle remoto



Figura 8-11 O Serial Monitor durante a execução do sketch para testar o controle de potência.

utilizado aqui, isso é feito apertando, e mantendo apertado, o botão que está no painel da frente da tomada até que o LED pisque e, em seguida, apertando um botão no controle remoto.

Verifique se o controle remoto ainda funciona normalmente apertando os seus botões para ligar e desligar a luminária. Agora, faça a mesma coisa enviando os comandos "A" e, em seguida, "a" ao Arduino através do Serial Monitor.

Repita esse procedimento com cada um dos canais do controle remoto para assegurar que tudo está funcionando corretamente.

Esse projeto é bem útil por si mesmo porque permite comandar dispositivos remotamente desde o seu computador. Entretanto, o nosso propósito principal é integrá-lo ao controlador de automação residencial. Isso é exatamente o que faremos na próxima seção.

» Integração com o controlador de automação residencial

Agora sabemos que a nossa placa controladora de potência trabalha corretamente com o Arduino. Chegou o momento de dar mais um passo e

integrá-la ao resto do controlador de automação. Para isso, vamos encaixar a placa de interface de áudio do Capítulo 7 na placa perfurada e fazer as conexões dessa placa aos pinos A4 e +5V do Arduino. A Figura 8-12 mostra o diagrama de fiação do projeto com as novas ligações.

A Figura 8-13 mostra a parte de cima da placa com a interface de áudio incluída. Observe que agora o Arduino recebe a alimentação elétrica através do cabo com plugue de 9V que vem da interface de áudio. Vê-se também o jack para o adaptador de voltagem CA/CC e o pino de áudio de 3,5mm pronto para ser conectado à saída de som do tablet Android.

Antes de instalar tudo na caixa, vamos fazer um teste de bancada. Para isso, vamos utilizar o aplicativo completo Android Home Automation (automação residencial) e o sketch "ch07_sound_test" que foi utilizado no capítulo anterior com a placa de Arduino. Isso permitirá verificar se as mensagens que são enviadas do tablet Android estão sendo recebidas corretamente. Após, instalaremos o sketch apropriado de Arduino. Portanto, transfira o sketch "ch07_sound_test" para a placa do Arduino.

Confira se tudo está conectado corretamente, isto é:

- Conecte o adaptador de voltagem CA/CC de 15V à placa da interface de áudio.

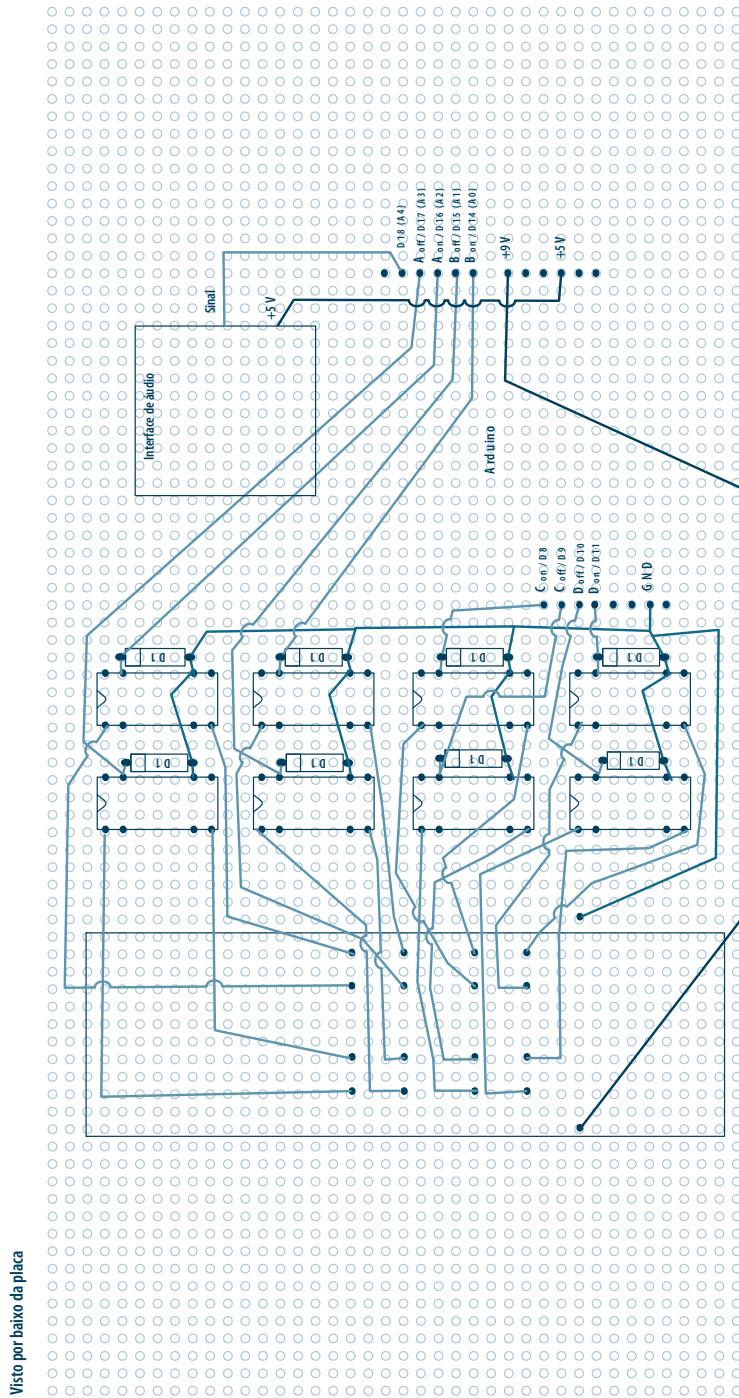


Figura 8-12 O diagrama de fiação com a interface de áudio.

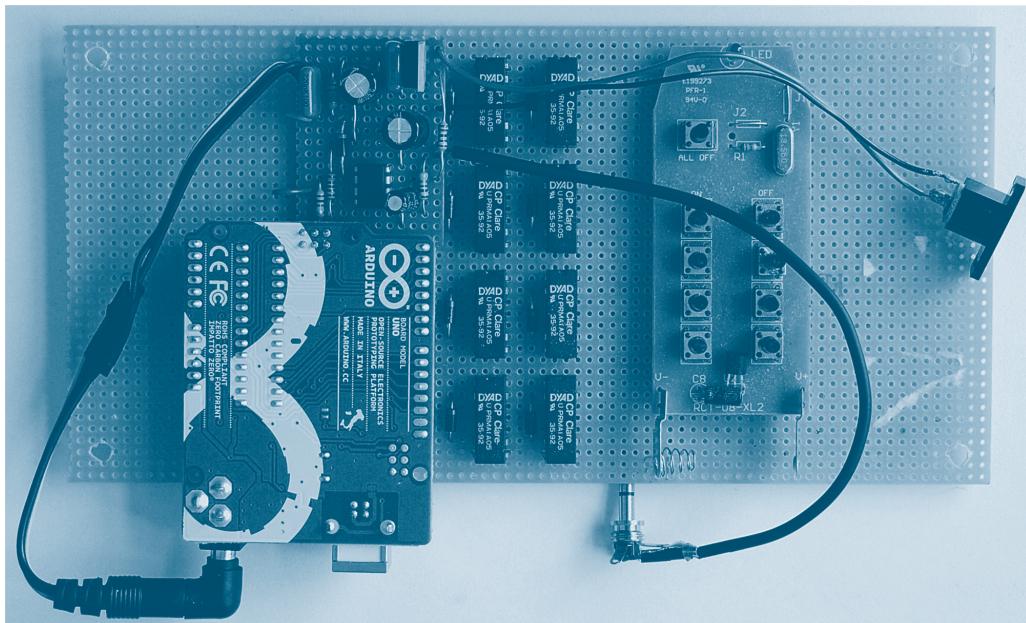


Figura 8-13 A placa do controlador de potência com a interface de áudio instalada.

- Conecte o plugue de 9V da placa de interface de áudio ao Arduino.
- Conecte o plugue de áudio de 3,5mm ao tablet Android.
- Conecte a placa de Arduino ao seu computador e abra o Serial Monitor.
- Abra o aplicativo Home Automation (automação residencial) no tablet Android e verifique se o som está ativado.

Abra o navegador (Open Browser) do aplicativo Home Automation e toque no botão “Light and Power” (luminárias e tomadas). Em seguida, toque no botão “On” (ligar) de “Outlet 1” (tomada 1) para ligar a tomada 1. O Serial Monitor deve mostrar o número 0101. Toque no botão “Off” (desligar) da mesma tomada. Deve aparecer 0100.

Agora podemos substituir o sketch de teste pelo real. Como sempre, ele pode ser baixado de www.duinodroid.com juntamente com os demais sketches do livro. O sketch é denominado “ch08_home_automation”.

A maneira mais fácil de testar o sistema é: ligar uma luminária de mesa a uma das tomadas de controle remoto, executar o aplicativo Home Automation no tablet Android, iniciar o navegador e ir para a página Lights And Power (luminárias e potência). Se a tomada for do tipo que deve ser programada para responder a um dado comando, então deveremos fazer isto primeiro: aperte e mantenha apertado o botão do controle remoto na frente da tomada até que o LED pisque e, no navegador, toque no botão “On”, que deve ser associado a essa tomada. Sugiro usar a primeira tomada. O LED na tomada deve parar de piscar para indicar que a programação foi efetuada.

Se você estiver utilizando o tipo de tomada com controle remoto que já vem com os números previamente programados, então não será necessário executar a etapa de programação das tomadas.

Agora, você poderá acender e apagar a luminária usando o tablet Android. Fique à vontade para testar as várias características do controlador de automação residencial. Tente, inclusive, conectá-lo a

partir de outro computador da sua rede ou mesmo pela Internet (veja o Capítulo 7).

Após se recuperar da emoção de testar o sistema, você deverá acondicionar tudo na caixa do projeto.

No Capítulo 7, construímos os circuitos básicos do projeto e os instalamos em uma caixa. Agora, precisamos acomodar a placa perfurada de controle das tomadas. Para isso, devemos fazer mais alguns furos na caixa do projeto. Precisaremos de orifícios para os quatro parafusos que fixarão a placa perfurada, de um orifício para o jack da fonte de 15V que está ligado à placa de áudio e, à sua escolha, de um orifício suficientemente grande para que um cabo USB seja conectado à placa do Arduino. Este último orifício só será necessário se você pretender modificar o software do Arduino.

A placa perfurada deve ser fixada à base da caixa do projeto com porcas e parafusos (Figura 8-14). Você deve também acondicionar e fazer nova soldagem do jack de alimentação elétrica.

» Configurando a sua casa

É muito fácil conectar as tomadas e, se elas forem programáveis, é muito provável que você deseje associar um grupo delas a um dos canais do seu controlador de automação residencial. Por exemplo, com a utilização de diversas tomadas com controle remoto, você poderá reunir todos os dispositivos que devem ser desligados à noite.

A Figura 8-15 mostra a configuração de tomadas que utilizamos. Todas as luminárias do andar térreo (Downstairs Lights) estão agrupadas em um canal – isso é muito prático na hora de ir dormir.

Os aparelhos de áudio, imagem e TV da sala de estar estão reunidos em outro canal, e os dormitórios dos atendentes (Minion Dorm) estão nos dois canais restantes.

Como no caso de tomadas, seria realmente muito útil poder controlar lâmpadas de teto e de parede. Entretanto, isso é mais difícil do que simplesmente

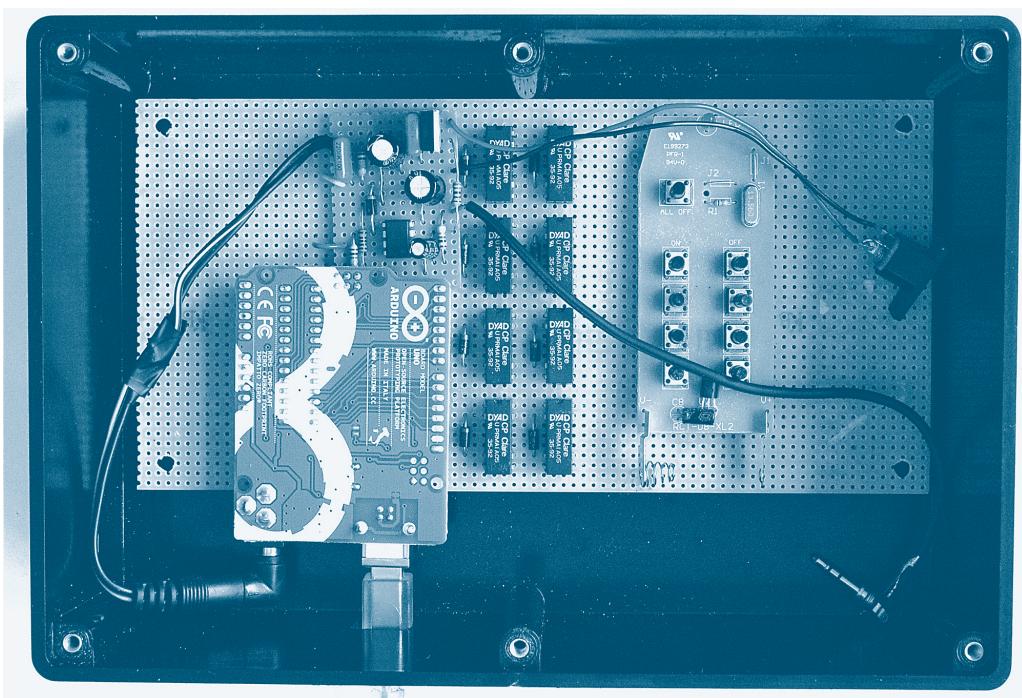


Figura 8-14 O interior da caixa do projeto com a placa perfurada instalada.

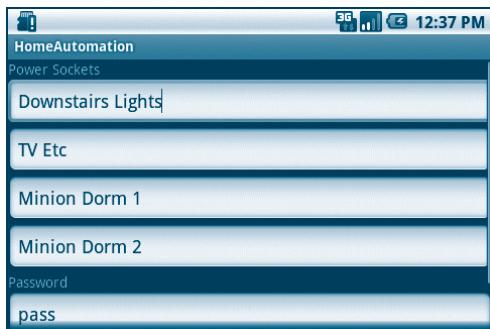


Figura 8-15 Um exemplo de configuração de automação residencial.

controlar uma tomada. A forma mais direta requer, infelizmente, um certo conhecimento especializado para fazer esse tipo de fiação. Portanto, isso só deve ser feito com o auxílio de um eletricista devidamente qualificado.

Utilizando uma barra de bornes de conexão, você poderá converter uma tomada com controle remoto em um comando de luminária (veja a Figura 8-16). Não faça isso a menos que você esteja devidamente qualificado para alterar a fiação de sua casa.

» Teoria

Nesta seção, examinaremos o software de nosso projeto. Em particular, veremos os sketches

de Arduino que utilizamos: "ch08_test_power" e "ch08_home_automation". No entanto, primeiro analisaremos uma parte da eletrônica utilizada neste projeto. Para isso, vamos examinar os relés utilizados para simular os toques nos botões do controle remoto.

» Relés

Em termos eletrônicos, os relés pertencem a uma tecnologia antiga. Eles já estavam disponíveis antes das válvulas termiônicas e eram usados como chaves para controlar o fornecimento de energia elétrica e para realizar comutação telefônica. Em muitas aplicações, eles foram substituídos por transistores e circuitos integrados. Entretanto, eles ainda têm seu lugar no mundo de alta tecnologia.

O tipo de relé utilizado neste projeto é denominado relé-reed (Figura 8-17). Eles não são adequados para controlar correntes elevadas, mas são ótimos para o caso do nosso projeto. Apresentam também a grande vantagem, diferentemente da maioria dos relés, de poderem ser controlados diretamente por uma placa de Arduino sem necessidade de transistores.

O que exatamente é um relé? Bem, ele tem duas partes eletricamente isoladas entre si. Há uma bobina que atua como um eletroímã e uma chave que é ativada pelo eletroímã. No caso do relé-reed, a chave é constituída de duas lâminas metálicas

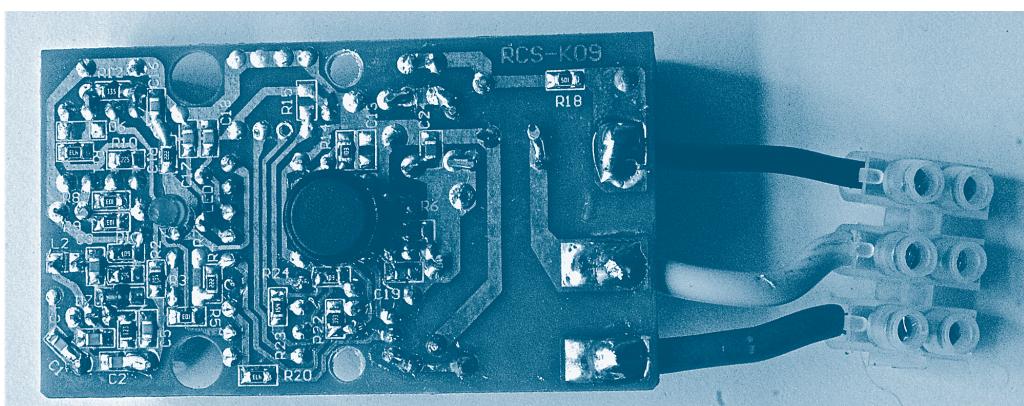


Figura 8-16 Alteração de uma tomada com controle remoto para que seja usada com uma luminária.

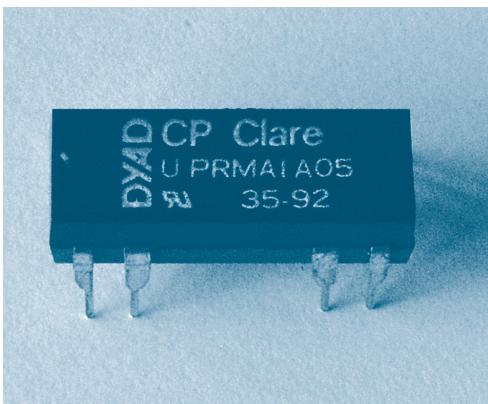


Figura 8-17 Um relé-reed.

delgadas que fazem contato elétrico dentro de uma cápsula de vidro lacrada. Isso os torna muito confiáveis e duráveis.

```
// Sketch de teste do controlador de
potência

#define AonPin 16
#define AoffPin 17
#define BonPin 14
#define BoffPin 15
#define ConPin 8
#define CoffPin 9
#define DonPin 11
#define DoffPin 10

#define ButtonPressPeriod 500

int onPins[] = {AonPin, BonPin, ConPin,
    DonPin};
int offPins[] = {AoffPin, BoffPin,
    CoffPin, DoffPin};

void setup()
{
    for (int i = 0; i < 4; i++)
    {
        pinMode(onPins[i], OUTPUT);
        pinMode(offPins[i], OUTPUT);
    }
    Serial.begin(9600);
    Serial.println("Ready enter one of:
        A a B b C c D d");
}
```

```
void loop()
{
    int channel = 0;
    if (Serial.available())
    {
        char ch = Serial.read();
        if (ch >= 'a' && ch <= 'd')
        {
            channel = ch - 'a';
            pressButton(channel, offPins);
        }
        else if (ch >= 'A' && ch <= 'D')
        {
            channel = ch - 'A';
            pressButton(channel, onPins);
        }
    }
}

void pressButton(int channel, int
    column[])
{
    digitalWrite(column[channel], HIGH);
    delay(ButtonPressPeriod);
    digitalWrite(column[channel], LOW);
}
```

» Sketch de teste

Aqui está o código do sketch que utilizamos antes para testar as tomadas.

Por meio de comandos `#define`, a primeira coisa é especificar todos os pinos do Arduino que utilizaremos. Observe que estamos usando os pinos analógicos do Arduino na forma digital. O software do Arduino possibilita que isso seja feito, permitindo que você utilize os pinos analógicos A0 a A5 como se fossem pinos digitais. Para isso, simplesmente some 14 ao número do pino analógico. Assim, o pino 16 é o pino analógico A2 ($16 = 2 + 14$) operando na forma digital, o 17 é o A3 e assim por diante.

Em seguida, definimos outra constante – `ButtonPressPeriod` (período de pressionamento de botão) – que especifica o tempo em milissegundos durante o qual cada relé deve permanecer ativado para simular um pressionamento de botão.

Para facilitar o acesso aos pinos associados aos relés, nós os agrupamos em dois arrays, um para os

pinos ligados ("onPins") e outro para os pinos desligados ("offPins").

A função "setup" define que todos esses pinos serão de saída, além de inicializar a conexão serial com o Serial Monitor.

Como de costume, a maior parte das atividades ocorre na função principal "loop," a qual verifica se um comando está sendo enviado pelo Serial Monitor (Serial.available). Em caso afirmativo e se estiver entre "a" e "d" ou "A" e "D," ela verifica qual é o número do canal (entre 0 e 4) e, em seguida, chama a função "pressButton" (pressiona botão) usando o canal como argumento da função. Se for o caso, usa também como argumento o array de pinos ligados ("onPins") ou o de pinos desligados ("offPins").

Em seguida, a função "pressButton" ativa o pino de saída adequado que, por sua vez, aciona o respectivo relé para "pressionar" o botão e ativar o canal.

» O sketch real

O sketch real do controlador de automação residencial é uma combinação do sketch de teste da interface de áudio do Capítulo 7 e do sketch de teste de potência que acabamos de analisar.*

```
#include <VirtualWire.h>

#define soundPin 18
#define zeroDurationFrom 10000
#define zeroDurationTo 25000
#define oneDurationFrom 35000
#define oneDurationTo 50000
#define resetTimeout 3000

#define AonPin 16
#define AoffPin 17
#define BonPin 14
#define BoffPin 15
#define ConPin 8
#define CoffPin 9
```

* N. de T.: Mais informações sobre a biblioteca VirtualWire podem ser obtidas em <http://www.airspayce.com/mikem/arduino/> e <http://www.airspayce.com/mikem/arduino/VirtualWire.pdf>.

```
#define DonPin 11
#define DoffPin 10

#define ButtonPressPeriod 1000

int onPins[] = {AonPin, BonPin, ConPin,
    DonPin};
int offPins[] = {AoffPin, BoffPin,
    CoffPin, DoffPin};
int remote = 0;

void setup()
{
    pinMode(soundPin, INPUT);
    for (int i = 0; i < 4; i++)
    {
        pinMode(onPins[i], OUTPUT);
        pinMode(offPins[i], OUTPUT);
    }
    vw_set_ptt_pin(5);
    vw_set_rx_pin(4);
    vw_set_ptt_inverted(true);
    vw_setup(2000);
    Serial.begin(9600);
}

unsigned int result;
int bitNo = 0;
long lastPulseTime = 0;

void loop()
{
    long pulseLength = pulseIn(soundPin,
        HIGH, oneDurationTo * 2);
    long timeSinceLastPulse = millis() -
        lastPulseTime;
    lastPulseTime = millis();
    if (pulseLength == 0 || timeSinceLastPulse > resetTimeout)
    {
        bitNo = 0; result = 0;
    }
    else
    {
        if (pulseLength >=
            zeroDurationFrom
            && pulseLength <= zeroDurationTo)
        {
            result = result << 1;
            bitNo++;
        }
        else if (pulseLength >=
            oneDurationFrom && pulseLength
```

```

        <= 50000)
    {
        result = (result << 1) + 1;
        bitNo++;
    }
}
if (bitNo == 16)
{
    processWord(result);
    bitNo = 0; result = 0;
}
}

void processWord(int message)
{
    int device = message >> 8;
    int action = message & 0x00FF;
    Serial.print("Device: ");
    Serial.print(device);
    Serial.print(" action: ");
    Serial.println(action);
    if (device > 0 && device <= 4)
    {
        processPower(device, action);
    }
    else
    {
        processRadio(device, action);
    }
}
void processPower(int device, int
action)
{
    if (action)
    {
        pressButton(device, onPins);
    }
    else
    {
        pressButton(device, offPins);
    }
}

void processRadio(int device, int
action)
{
    uint8_t msg[2];
    msg[0] = (uint8_t)device;
    msg[1] = (uint8_t)action;
    vw_send((uint8_t *)msg, 2);
    delay(400);
}

void pressButton(int channel, int
column[])
{
    int pin = column[channel - 1];
    digitalWrite(column[channel-1],
HIGH);
    delay(ButtonPressPeriod);
    digitalWrite(column[channel-1], LOW);
    delay(ButtonPressPeriod);
}

```

Usando `#define`, como no sketch anterior, começamos definindo alguns valores constantes, como os pinos que serão usados para os relés e as durações dos pulsos que vêm da placa de interface de áudio (veja a seção *Teoria* do Capítulo 7).

Neste caso, após todos os 16 bits serem reunidos, eles serão passados na forma do argumento “`result`” para a função “`processWord`”, a qual fará a decomposição em dispositivo (“`device`”) e ação (“`action`”). Se o dispositivo estiver no intervalo 1 a 4, então saberemos que se trata de um dos canais que controlam as tomadas (power, potência) e chamaremos a função “`pressButton`”, como fizemos no sketch anterior.

Esse sketch inclui também os códigos para abrir a fechadura da porta e controlar o termostato – ambos tratados pela função “`processRadio`”. Por enquanto, podem ser desconsiderados.

» Resumo

Agora, o nosso controlador de automação residencial tornou-se um dispositivo útil. Além de ajustar tempo, podemos ligar e desligar coisas estando em qualquer lugar. No próximo capítulo, ampliaremos esse projeto mais um pouco acrescentando um meio de comandar um termostato para controlar o sistema de aquecimento.



» capítulo 9

Termostato inteligente

Nos capítulos anteriores, aprendemos diversos recursos que podem ser controlados pelo Arduino. Neste capítulo, vamos analisar uma aplicação muito especial que poderá proporcionar conforto térmico à sua casa: mostraremos como elaborar um sistema de aquecimento com controle remoto.

Objetivos

- » Criar um meio para comandar um termostato a fim de controlar o sistema de aquecimento de uma casa
- » Aprender a acrescentar o projeto do termostato ao controlador de automação residencial usando comunicação RF de dados de baixo custo
- » Conhecer o circuito integrado sensor de temperatura utilizado neste projeto
- » Conhecer os módulos RF para a comunicação de Arduino para Arduino
- » Examinar o sketch do Arduino utilizado neste projeto

Este projeto apresenta a palavra final em regulação de temperatura*. Não só atua como termostato independente como também tem um canal sem fio de RF, possibilitando a comunicação com o controlador do sistema de automação residencial. Dessa forma, temperaturas diferentes poderão ser programadas para períodos diferentes do dia.

O sistema de automação residencial informará o termostato do seu novo ajuste de temperatura a cada 10 segundos por meio do enlace de RF. Para dar um retorno e informar que tudo está funcionando, o LED piscará duas vezes sempre que receber uma mensagem.

Para diminuir o tamanho do projeto e reduzir o custo, o circuito utiliza o microcontrolador de um Arduino. Após ser programado, esse microcontrolador é retirado da placa do Arduino e depois é instalado em uma placa perfurada. O Arduino é utilizado para programar o microcontrolador, que, em seguida, é substituído por outro chip de microcontrolador. Um ATMega328, do tipo utilizado em placas de Arduino, pode ser comprado facilmente por cerca de cinco dólares.

A Figura 9-1 mostra o termostato com uma chave de seleção, um botão de controle e um LED. A chave permite selecionar entre os modos de ajuste manual e programado.

A maior parte do software de comando do termostato inteligente faz parte do controlador de automação residencial. Esse software exibe telas gráficas de toque, com um visual agradável, proporcionando um mecanismo para configurar o perfil de temperatura (Figura 9-2). Há duas páginas para o ajuste do perfil de temperatura. Uma para os dias úteis e outra para os fins de semana. O dia é dividido em 12 períodos de tempo. Por exemplo, 01-04 indica

* N. de T.: Estamos partindo de um sistema de aquecimento comandado eletricamente. O sistema é controlado por um termostato, isto é, um sensor de temperatura ajustável que compara a temperatura atual do ambiente com a temperatura desejada que foi previamente ajustada. Dependendo do resultado dessa comparação, o termostato ativa um relé de potência que liga o módulo de aquecimento propriamente dito. Neste capítulo, subsumiremos o termostato já existente pelo termostato inteligente.

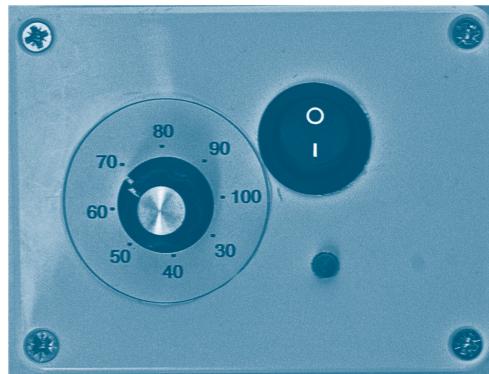


Figura 9-1 Um termostato inteligente usado para aquecimento.

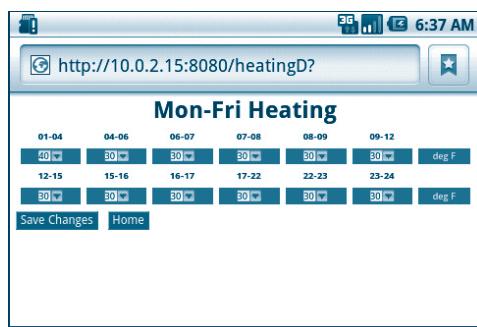


Figura 9-2 O controlador de automação residencial – controle da temperatura de aquecimento**.

o período que vai desde 1 hora até 4 horas da madrugada. A temperatura de cada período é ajustada selecionando-a de uma lista.

Mudando a chave de seleção, o termostato deixa de utilizar a temperatura enviada pelo controlador de automação residencial e passa a usar a temperatura indicada no seu botão de ajuste manual de temperatura.

» Construção

Precisamos construir o termostato, mas também precisamos modificar o controlador de automação

** N. de T.: Mon-Fri heating = aquecimento de segunda à sexta-feira.

residencial, incluindo um transmissor de RF e atualizando o software do Arduino.

O melhor é construir primeiro o termostato. Desse modo, poderemos utilizá-lo para testar se as alterações feitas no controlador de automação residencial foram bem-sucedidas.

Além do CI (círcuito integrado) do microcontrolador e alguns outros componentes de apoio, há apenas um transistor, um relé e um módulo-receptor de RF na placa perfurada. Utilizaremos uma fonte de alimentação de um carregador de telefone celular fora de uso. Se você não dispõe de um carregador como esse, então você poderá usar um adaptador de voltagem CA/CC comum.

A Figura 9-3 mostra as relações existentes entre o termostato, o sistema central de aquecimento e o controlador de automação residencial.

A Figura 9-4 mostra o diagrama esquemático do termostato.

ALERTA Neste projeto, tensões elevadas residenciais são utilizadas, e a fiação do seu sistema de aquecimento é modificada. A eletricidade de uso residencial mata muitas pessoas a cada ano e dá início a centenas de incêndios só nos Estados Unidos.

Você só poderá instalar este sistema se você estiver qualificado adequadamente e se tiver um bom entendimento das implicações de segurança acarretadas quando se altera a fiação elétrica de uma casa.

» O que será necessário

Você necessitará dos componentes listados na tabela *Lista de Componentes*, a seguir, para construir o termostato.

Você também precisará de um Arduino Uno ou Duemilanove para programar o circuito integrado (CI2) ATMega328 ou ATMega168. Assegure-se de comprar um circuito integrado que já venha com uma versão instalada do “bootloader” do Arduino.

No mercado, há muitos módulos disponíveis de RF, tanto receptores como transmissores. A maioria apresenta a mesma configuração de pinos que foi utilizada neste projeto. De qualquer forma, consulte as especificações dos módulos que você pretende adquirir e compare as configurações de pinos com as da Figura 9-4, para se assegurar de que são as mesmas.

A escolha do relé dependerá do país em que você vive. A sua bobina deverá ser de 5 ou 6V. Os contatos

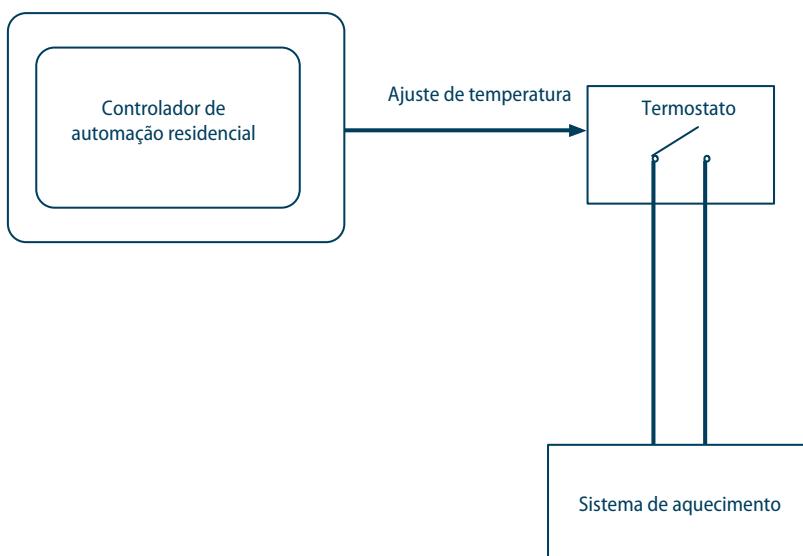


Figura 9-3 O controlador de automação residencial e o termostato.

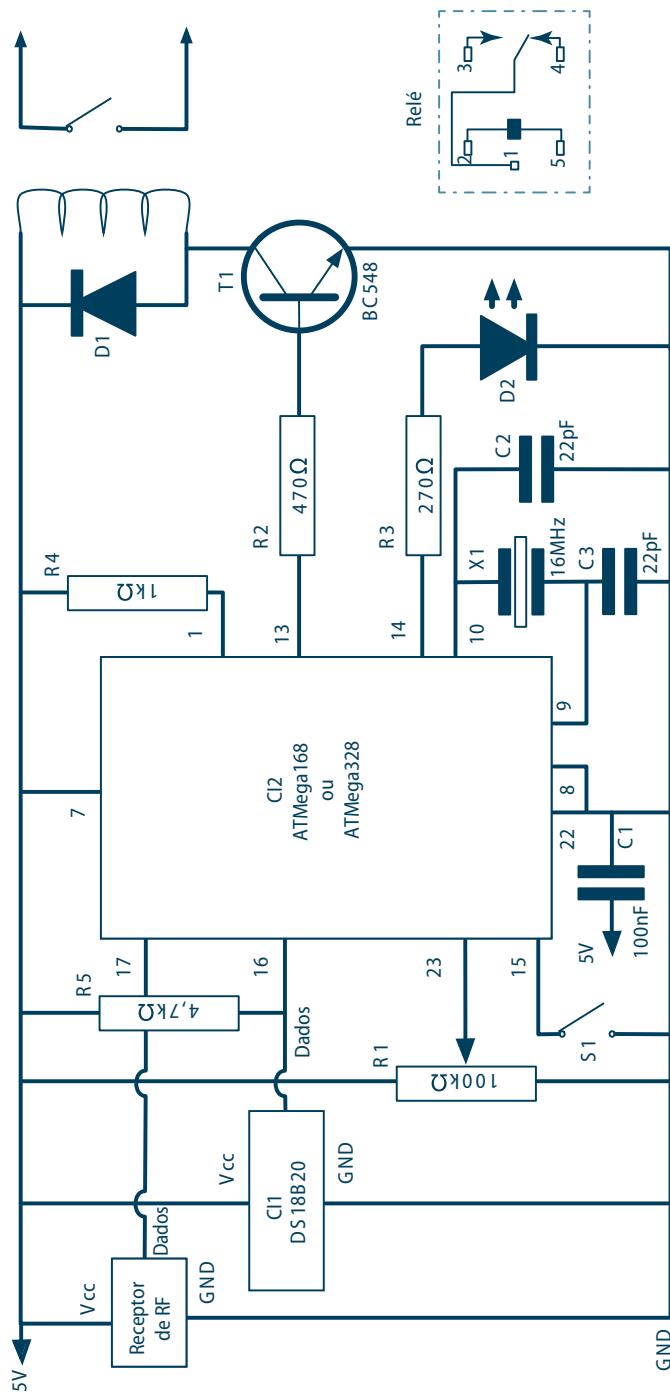


Figura 9-4 Diagrama esquemático do termostato.

LISTA DE COMPONENTES

Componente	Quantidade	Descrição	Fornecedor
Microcontrolador	1	ATMega328 com bootloader	Sparkfun: DEV-10524
Receptor de RF	1	Módulo receptor de 433 MHz	Farnell: 1304026
Transmissor de RF	1	Módulo transmissor de 433 MHz	Farnell: 1304024
Relé	1	Relé com bobina de 5 ou 6V	Farnell: 1455502
R1	1	Potenciômetro de 100kΩ	
Botão	1	Botão redondo com seta para ajustar o potenciômetro R1 (ver Figura 9-1)	Farnell: 2056847
R2	1	Resistor de filme metálico de 470Ω e 1/2 W	Farnell: 1099883
R3	1	Resistor de filme metálico de 270Ω e 1/2 W	Farnell: 9340300
R4	1	Resistor de filme metálico de 1kΩ e 1/2 W	Farnell: 9339779
R5	1	Resistor de filme metálico de 4,7kΩ e 1/2 W	Farnell: 9340629
C1	1	Capacitor cerâmico de 100nF	Farnell: 1200414
C2, C3	2	Capacitor eletrolítico de 22pF	Farnell: 1600966
X1	1	Crystal de 16MHz	Farnell: 1611761
D1	1	1N4001	Farnell: 1458986
D2	1	LED vermelho de 5mm	Farnell: 1712786
Cl1	1	Circuito integrado DS18B20	Sparkfun: SEN-00245
T1	1	BC548	Farnell: 1467872
S1	1	Chave de balancim (ver Figura 9-1)*	Farnell: 1634645
Soquete para Cl	1	Soquete para circuito integrado de 28 pinos	Farnell: 1824463
Fonte de alimentação	1	Fonte de alimentação de 5V	Veja descrição no texto
Barra de conexão	1	Barra com dois bornes de conexão de 2A	Loja de ferragem
Placa perfurada	1	Placa trilhada com 29 trilhas de 24 orifícios	Farnell: 1201473
Caixa	1	Caixa de plástico	Lojas locais

* N. de T.: Quando essa chave estiver fechada, o termostato estará no modo manual de seleção de temperatura, utilizando como referência de temperatura um valor ajustado manualmente. Quando estiver aberta, o termostato estará no modo automático, utilizando o valor enviado pelo controlador de automação residencial.

deverão ter uma especificação similar à do termostato que está sendo substituído. Se você não tiver certeza das especificações de tensão e corrente necessárias, você deverá consultar um eletricista qualificado ou um engenheiro. O relé especificado neste projeto foi o que se mostrou adequado ao sistema de aquecimento central utilizado aqui. Não temos como saber qual é o relé adequado ao sistema de aquecimento que você está usando. Portanto, faça a sua própria verificação.

Como fonte de alimentação, aproveitaremos o carregador fora de uso de algum celular antigo para for-

necer 5V ao termostato. Se você não dispuser de um, compre um adaptador de voltagem CA/CC de 5V.

Além desses componentes, você usará as ferramentas da lista seguinte.

CAIXA DE FERRAMENTAS

- Ferramentas para soldar
- Estilete e régua
- Uma pistola para cola a quente ou cola epóxi
- Fios flexíveis de diversas cores
- Um multímetro

» Passo 1: prepare a placa perfurada

A Figura 9-5 mostra a disposição dos componentes na placa perfurada para o termostato.

Corte a placa no tamanho correto. A maneira mais simples de fazer isso é riscar fortemente a placa com um estilete e, em seguida, quebrá-la na beira de uma mesa. Você também pode usar uma tesoura, mas o resultado não será tão bom.

Como você pode ver, há diversos cortes para fazer na placa. Como na placa de interface de áudio do Capítulo 7, use uma microrretífica para fazer cortes nos lugares indicados das trilhas. Na Figura 9-5, a placa é vista de cima.

Quando os 19 cortes forem feitos, a sua placa será semelhante à mostrada na Figura 9-6.

» Passo 2: solde as conexões e o soquete do Cl

O próximo passo é soldar as conexões de fio na placa (Figura 9-7). De novo, há diversas conexões para fazer (11). Portanto, tome cuidado e assegure-se de que todas sejam feitas e estejam nos lugares corretos.

» Passo 3: solde os resistores e o diodo

Agora, começaremos a soldar os componentes. Como sempre, começaremos pelos componentes mais baixos, os resistores e o diodo, e depois soldaremos os capacitores. Quando soldar o diodo, verifique se ele está na orientação correta. O lado com a faixa deve estar voltado para a parte de cima da placa.

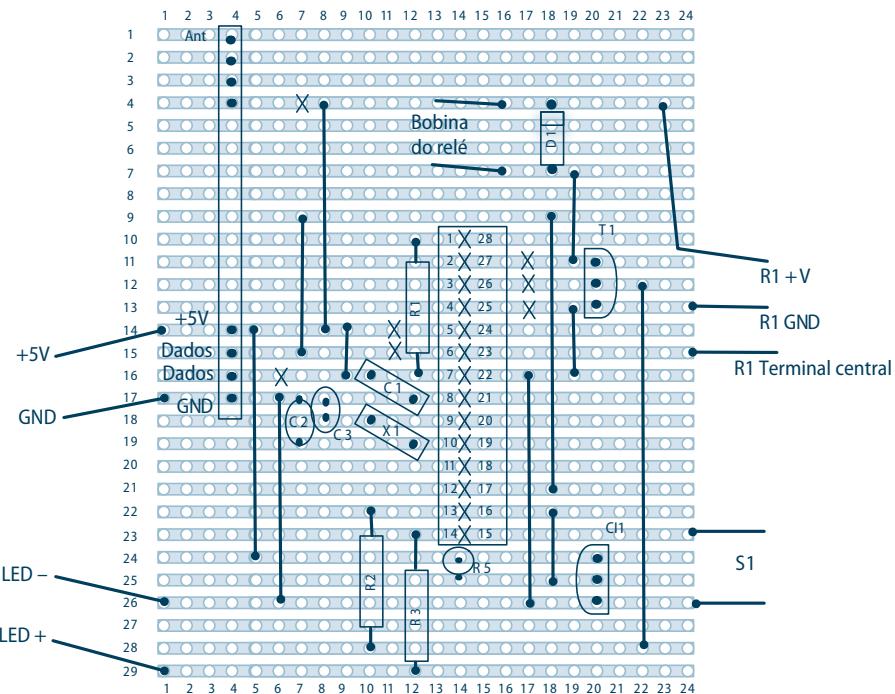


Figura 9-5 A disposição dos componentes na placa perfurada.

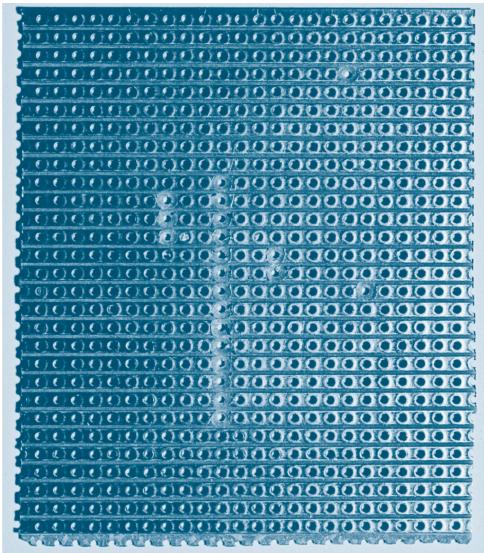


Figura 9-6 A placa perfurada pronta (lado das trilhas).

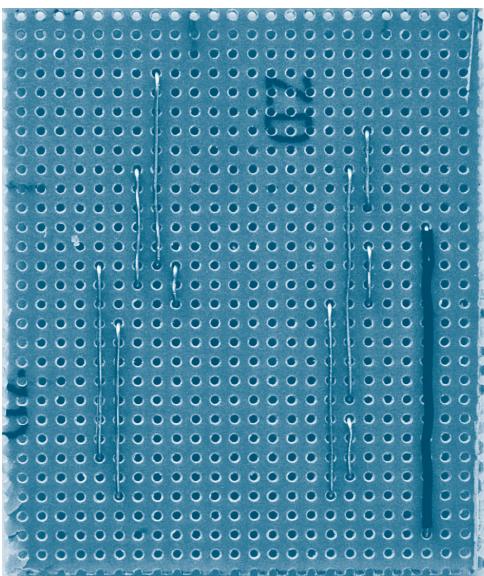


Figura 9-7 A placa perfurada com as conexões de fio.

O resistor R5 é montado verticalmente. Portanto, espere até o próximo passo quando soldaremos os capacitores e os componentes de perfil mais alto.

Quando todos eles estiverem soldados em seus lugares, você poderá soldar o soquete do circuito

integrado. Ele deverá ser soldado de forma que a marca em corte usada para indicar o lado do pino 1 esteja voltada para o lado de cima da placa.

A Figura 9-8 mostra a placa com todos esses componentes no lugar.

» Passo 4: solde os demais componentes

Agora, os demais componentes podem ser instalados. Tome cuidado para que CI1 e T1 estejam na orientação correta e para que você não esquente demais os seus terminais.

Utilizaremos apenas os quatro pinos debaixo do módulo de recepção* de RF. Entretanto, se o seu termostato estiver muito afastado do controlador de automação residencial, você precisará soldar um fio no pino ANT para atuar como antena. O fio deve ter em torno de 10cm.

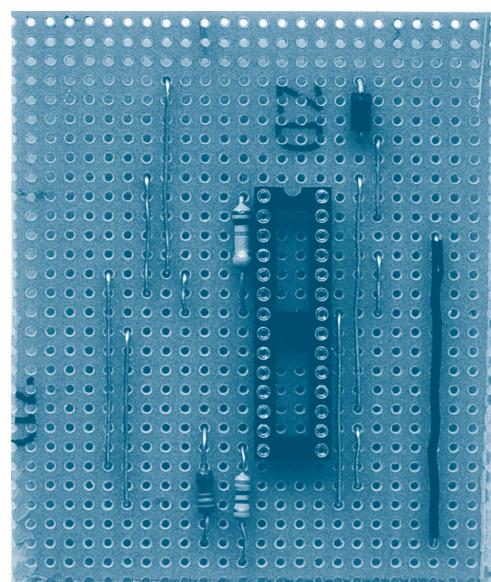


Figura 9-8 A placa perfurada com os resistores e os capacitores.

* N. de T.: O módulo de recepção de RF aparece à esquerda na Figura 9-5 e embaixo na Figura 9-9. Combinando as duas figuras, você poderá identificar os pinos do módulo.

Quando tudo estiver conectado, a placa ficará como na Figura 9-9.

Neste ponto, uma inspeção completa da placa é uma boa ideia. Verifique a parte de cima da placa verificando se todos os componentes e as conexões estão nos lugares indicados pela Figura 9-5. Examine também a parte debaixo da placa conferindo se todos os cortes de trilha estão nos lugares corretos e se não há pontes acidentais de solda entre as trilhas.

» Passo 5: programe e instale o microcontrolador

Precisamos instalar no microcontrolador o sketch de Arduino que controla o termostato. A maneira mais simples de fazer isso é usando uma placa de Arduino.

O mais fácil a fazer é programar o microcontrolador que já está na placa do Arduino e, depois de transferi-lo para a placa perfurada, instalar na placa do Arduino o microcontrolador que você comprou.

O sketch utiliza três bibliotecas, duas para o sensor de temperatura (OneWire e DallasTemperature) e a biblioteca VirtualWire para o receptor de RF. Provavelmente, você já instalou a biblioteca VirtualWire no Capítulo 8. Portanto, falta instalar ainda as duas outras bibliotecas no ambiente Arduino do seu computador.

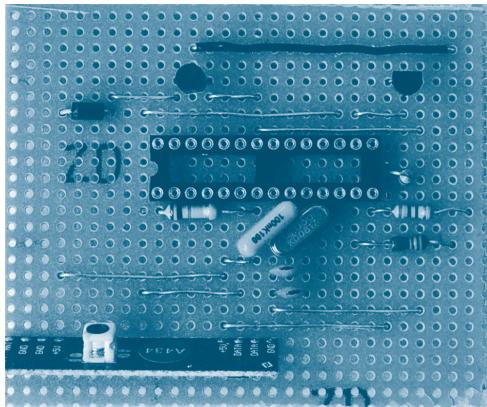


Figura 9-9 A placa perfurada completa.

O procedimento é o mesmo para todas as bibliotecas (veja Capítulo 1). As bibliotecas OneWire e Dallas podem ser baixadas de www.hacktronics.com/code/OneWire-v2.zip.

Agora, instale o sketch (ch09_thermostat) na placa do Arduino. Então, com a alimentação elétrica desligada, remova cuidadosamente o circuito integrado da placa e encaixe-o no soquete da placa perfurada, verificando se está na orientação correta.

Ao remover o circuito integrado, tome cuidado porque os pinos podem facilmente se curvar ou quebrar. Levante o microcontrolador pelas extremidades com uma faca, um pouco de cada vez até que se solte. Você também tem que tomar cuidados com a eletricidade estática, encostando primeiro em alguma coisa aterrada e descarregando a eletricidade do seu corpo antes de tocar nos pinos. Uma pulseira antiestática de aterramento é outra opção.

» Passo 6: ligando tudo

Agora, a placa está pronta. Portanto, após uma inspeção completa para verificar se não há pontes indesejáveis de solda entre trilhas, poderemos ligar tudo (Figura 9-10). Nós não iremos acondicioná-la na caixa até nos assegurarmos de que tudo está funcionando.

Estamos usando um carregador de celular como fonte de alimentação. Não precisamos de um carregador potente – será suficiente um de 5V capaz de fornecer no mínimo 100mA. Com o carregador desligado, corte o seu cabo próximo do plugue e descasque os fios de modo que você possa ligar um multímetro e medir a tensão. Primeiro, verifique se a tensão é de fato 5V e estável. Segundo, determine qual é o fio positivo. Por enquanto, soldie os fios diretamente na placa perfurada.

A seguir, soldie os terminais do potenciômetro. Observe que as conexões para o potenciômetro na placa perfurada podem não estar na ordem que você poderia esperar. Olhando para a parte detrás do potenciômetro, com os terminais para baixo, o terminal positivo estará à esquerda, o negativo estará à direita e o terminal variável, no centro.

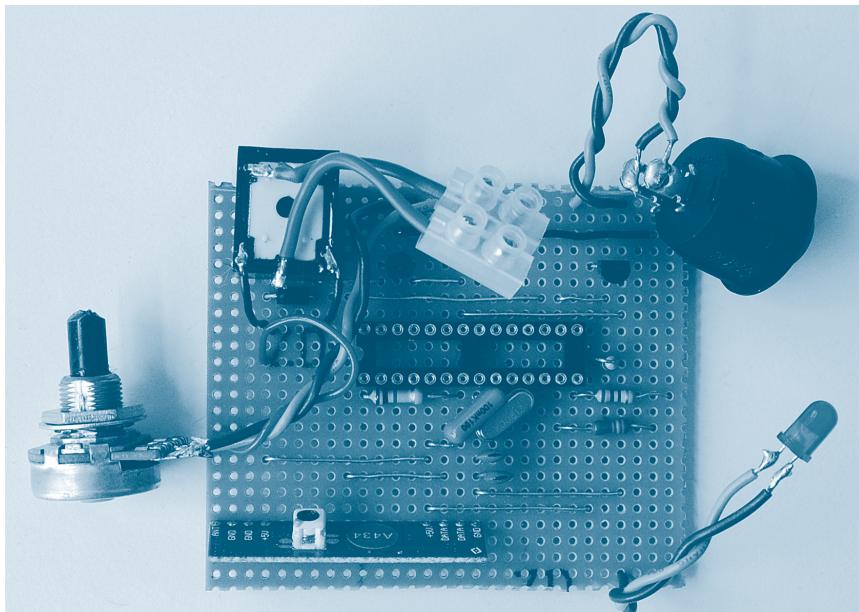


Figura 9-10 Fazendo as conexões na placa.

A chave e o LED devem ambos ser conectados por meio de fios flexíveis coloridos curtos. Para codificar as polaridades, é uma boa ideia utilizar fios coloridos no LED. Desse modo, evita-se a troca acidental de polaridades.

Os terminais da bobina do relé podem ser ligados de qualquer forma. Cole o relé na placa e solde fios nos terminais dos contatos normalmente abertos do relé, ligando-os, em seguida, à barra de bornes de conexão. Muitos relés têm a configuração mostrada na Figura 9-4. Entretanto, você deve verificar qual é a configuração que se aplica ao seu relé.

Coloque a chave em modo manual (fechada) e ligue a fonte de alimentação. Você deve observar que, ao ajustar o potenciômetro no sentido horário, o LED acende e os contatos do relé fecham. Nesse modo, o potenciômetro define a temperatura. Quando o ajustamos para uma temperatura mais elevada do que a medida pelo sensor, o aquecimento é ligado.

Você poderá ajustar lentamente o potenciômetro até que o LED acenda e, em seguida, segurar com os dedos o sensor de temperatura para aquecê-lo.

Quando sua temperatura ultrapassar a temperatura ajustada no potenciômetro, o LED deverá apagar novamente.

Por enquanto, isso é o que faremos para testar a placa. Aplicaremos outros testes depois de modificar o controlador de automação residencial.

» Passo 7: modifique o controlador de automação residencial

Precisamos acrescentar o transmissor de RF à placa do controlador de automação residencial. A Figura 9-11 mostra a placa com as novas modificações.

O módulo do transmissor é menor que o módulo do receptor e está instalado ao lado da placa de interface de áudio. O pino de dados do transmissor está ligado ao pino D12 do Arduino. As demais conexões são GND e +5V, que está ligada a Vcc. Como já fizemos com o receptor, também podemos ligar uma antena na forma de um pedaço de fio conectado ao terminal ANT do transmissor. Para distâncias curtas, não há necessidade dessa conexão.

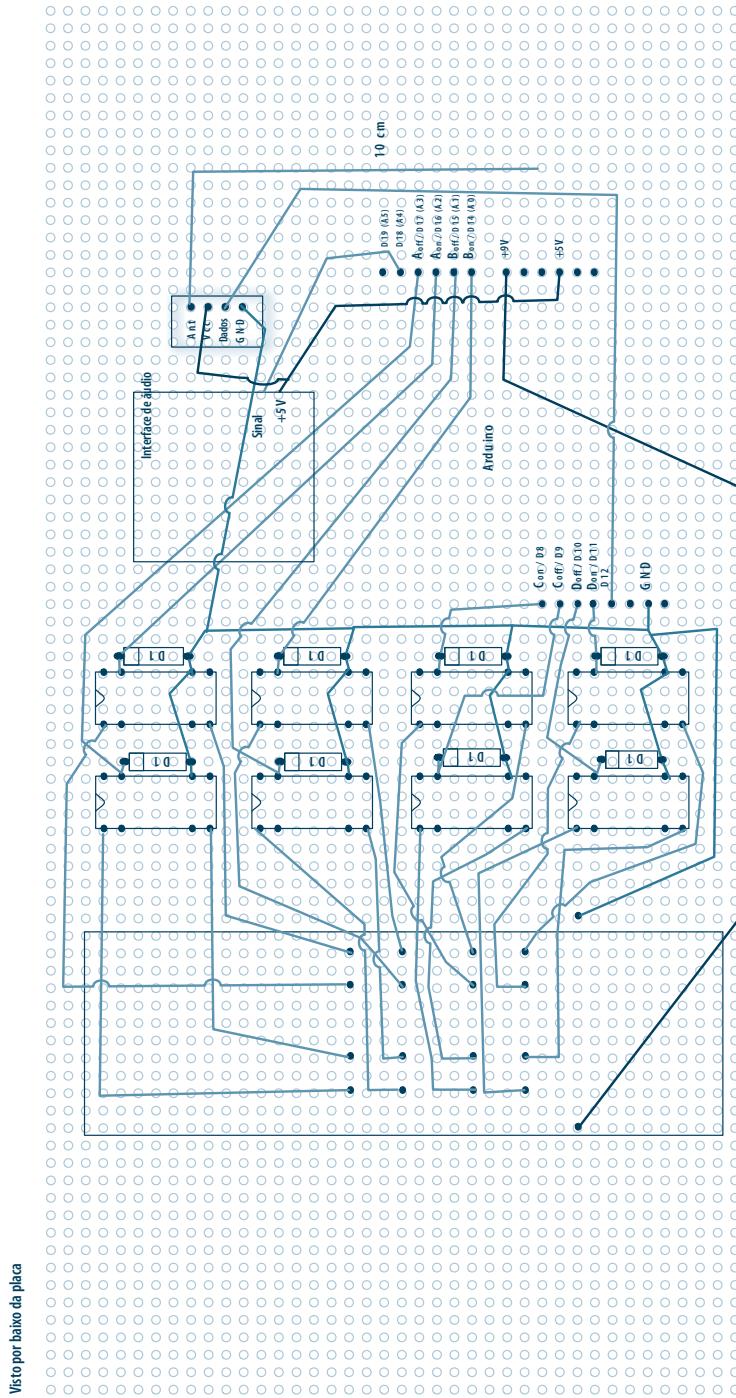


Figura 9-11 A disposição dos componentes na placa modificada.

As partes da frente e detrás da placa perfurada modificada estão mostradas nas Figuras 9-12 e 9-13, respectivamente.

O sketch de Arduino, originalmente instalado na placa, já continha o código necessário para o transmissor. Por isso, não precisamos instalá-lo.

» Passo 8: teste

Agora, podemos testar o projeto utilizando o controlador de automação residencial. Para isso, ligue tudo, incluindo o controlador de automação residencial e coloque a chave de seleção do termostato no modo automático. Depois de uns 10 segundos, o LED deve piscar duas vezes. Isso indica que o controlador de automação residencial enviou um valor de temperatura para o termostato.

A não ser que esteja muito frio, o LED e o relé estarão desligados, porque a temperatura inicial é de 30°F (ou $-1,1^{\circ}\text{C}$).

Vá até a página apropriada “Mon-Fri Heating” (aquecimento de segunda à sexta-feira) no con-

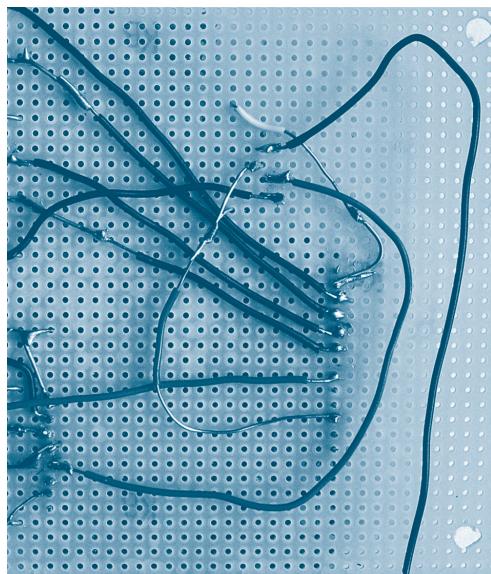


Figura 9-13 A parte detrás da placa modificada.

trolador e, no período de tempo correspondente ao horário corrente (Figura 9-17), ajuste a temperatura para o seu valor máximo. Depois de um momento, o LED deverá piscar no termostato e, em seguida, o LED e o relé deverão ser ativados.

Se o LED não piscar, então volte a conferir as alterações de fiação que você fez no termostato e no controlador de automação residencial, especialmente nas áreas dos módulos de recepção e transmissão. Verifique se eles estão recebendo 5V nos seus pinos de alimentação elétrica.

Para verificar se o relé está funcionando, coloque o multímetro no modo de teste de continuidade e conecte as ponteiras aos bornes de conexão da saída do relé (com nada ligado aos bornes). Quando o relé clicar e o LED acender, então o multímetro deverá indicar a continuidade no mostrador e emitirá um som, se ele tiver esse recurso.

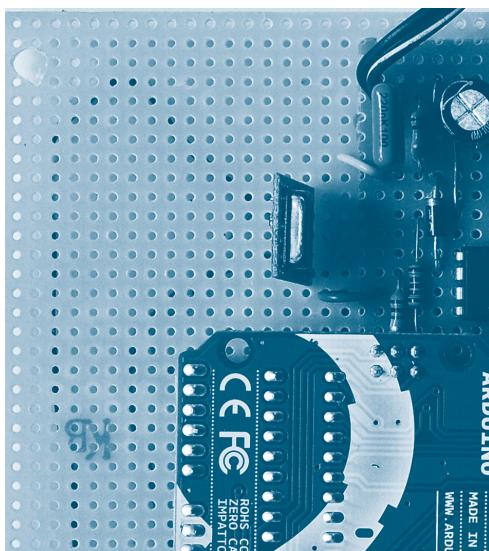


Figura 9-12 A parte da frente da placa modificada, com o pequeno módulo de transmissão. O seu pino GND está na parte debaixo e o ANT, na parte de cima.

» Passo 9: acondicione o projeto

Esse termostato substituirá o que já existia antes. Portanto, o ideal é uma caixa do mesmo tamanho

ou um pouco maior, que possa acomodar a placa perfurada e os demais componentes.

Para a fonte de alimentação, há algumas opções. Se o termostato antigo tinha neutro e fases, então você poderá fazer a fiação da fonte de alimentação no interior da caixa. Caso contrário, será necessário dispor de uma tomada próximo do termostato.

Para acondicionar o projeto, compramos uma caixa suficientemente grande para colocar os componentes do projeto e distribuí-los até conseguir uma boa disposição.

Logo que você decidir que está tudo bem, faça a marcação dos furos na caixa.

A Figura 9-14 mostra os componentes instalados na caixa, e a Figura 9-15 mostra os furos do painel frontal.

O LED é inserido sob pressão em um furo de 5 mm. O encaixe apertado manterá o LED no lugar.

Você também precisará de um orifício para o cabo do adaptador de tensão e também de alguns orifícios na parte detrás da caixa. Isso permitirá a passagem dos fios do sistema de aquecimento e a fixação da caixa na parede.

Para fixar o novo termostato na parede, o ideal seria reutilizar os furos do termostato antigo.

Você precisa incluir também algumas marcações de temperatura no painel frontal do termostato. Se você utilizar graus Fahrenheit, então a temperatura mínima será 30 e a máxima, 100. Marque essas posições e, em seguida, interpole seis pontos entre esses valores (40, 50, 60, 70, 80 e 90). Você pode simplesmente escrever os números na parede frontal da caixa usando uma caneta ou papel transfer. Você também poderá imprimir uma escala, cortá-la e colá-la na caixa. Para escalas em graus Celsius, é só realizar o mesmo procedimento convertendo a escala Fahrenheit em Celsius e interpolando adequadamente os valores numéricos.

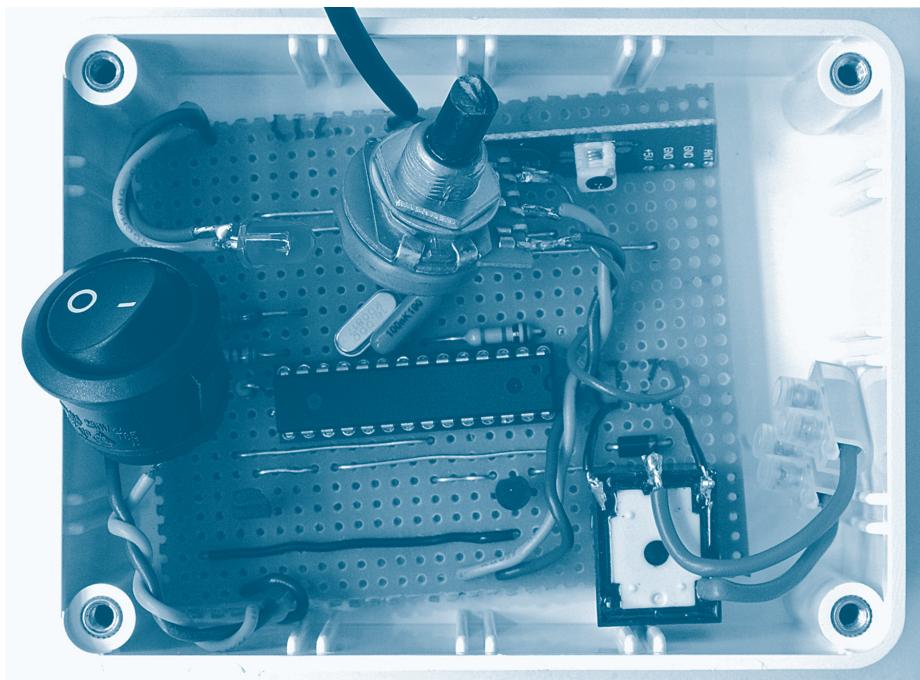


Figura 9-14 Posicionamento dos controles.

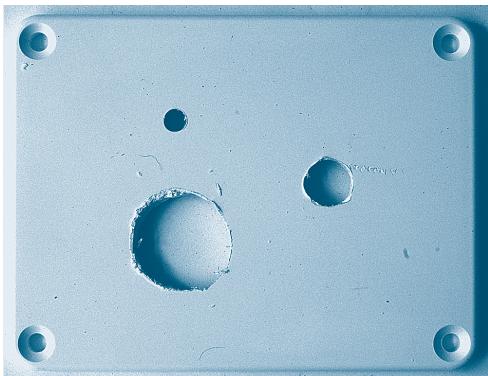


Figura 9-15 O painel frontal com os furos.



Figura 9-16 Definindo a unidade de temperatura.

» Passo 10: instalação

Essencialmente, esse projeto consiste em um interruptor que é fechado quando é necessário aquecer a residência. Isso se dá pelos contatos normalmente abertos do relé. Caberá a você fazer a fiação entre o termostato e o sistema de aquecimento de sua residência.

Eu enfatizo novamente que você deve realizar essa fiação com muito cuidado e, se não estiver absolutamente seguro do que está fazendo, você deverá se valer de algum eletricista qualificado para trabalhar no seu sistema de aquecimento.

» Usando o sistema

A página de configuração do controlador de automação residencial inclui uma opção para definir a unidade de temperatura (Figura 9-16). Se você ativar a caixa “Use degrees C” (use graus C), todas as temperaturas exibidas no controlador de automação residencial serão em graus Celsius.

Você deve fazer a seleção da unidade desejada antes de começar a configurar o seu perfil de temperatura porque, se você alterar a unidade utilizada depois, então será necessário refazer completamente o perfil.

Para otimizar a utilização da energia elétrica, você poderá estabelecer perfis diferentes de temperatura, que sejam adaptados às suas necessidades específicas no fim de semana e nos dias úteis durante a semana. Por isso, o menu de configuração tem opções diferentes para os dias úteis e para o fim de semana.

A Figura 9-17 mostra um típico perfil de temperatura para os dias úteis durante a semana.

A temperatura é ajustada para o mínimo (30°F ou -1,1°C) durante a noite, mas entre 4 e 6 horas da madrugada é aumentada para 50°F (10°C) e, então, para 60°F (15,5°C) entre 6 e 7 horas da manhã. Entre 7 e 9 da manhã, ela é elevada para 70°F (21,1°C).

Das 9 horas da manhã até as 16 horas, a temperatura é baixada para o mínimo, porque não há pessoas em casa. Depois, ao entardecer, ela é mantida elevada até a hora de dormir.

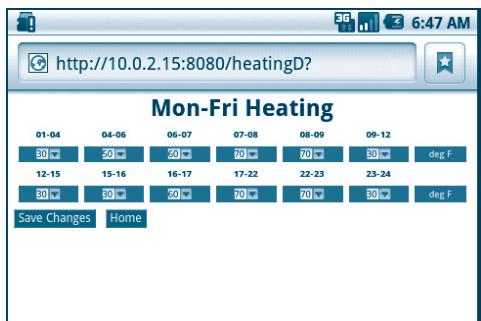


Figura 9-17 Um perfil de temperatura de segunda a sexta-feira (Mon-Fri).

» Teoria

Este é um típico projeto Arduino padrão. Entretanto, utilizamos dois componentes interessantes: chip sensor de temperatura e módulos RF para a comunicação de Arduino para Arduino.

Agora, discutiremos esses componentes e, em seguida, veremos como programar um Arduino para utilizá-los.

» Sensores de um fio

O sensor de temperatura utilizado neste projeto é um dispositivo muito interessante, combinando

um sensor de temperatura e um processador que envia a temperatura em forma digital. Isso permite que ele seja instalado em um local distante do Arduino, sem que o comprimento do fio interfira na exatidão das medidas de temperatura.

Para uma introdução a esse tipo de sensor, acesse um interessante artigo em www.arduino.cc/playground/Learning/OneWire.

» O sketch para o termostato

O sketch para o termostato está listado a seguir.

```
#define THERMOSTAT_ID 0x41 // Identificador do termostato
#define MIN_ON_TIME 120L // Tempo ligado mínimo em segundos
#define RADIO_CHECK_PERIOD 10000L // Tempo para verificar se há mensagem
#define MIN_TEMP 30 // Temperatura mínima em F
#define MAX_TEMP 100 // Temperatura máxima em F

#define rfRxPin 11
#define tempRxPin 10
#define potPin 0
#define ledPin 8
#define relayPin 7
#define overrideSwitchPin 9 // Pino da chave de seleção manual de modo

OneWire oneWire(tempRxPin);
DallasTemperature sensors(&oneWire);
DeviceAddress thermometer;

int setTemp = 0; // Temperatura ajustada
int actualTemp = 0; // Temperatura atual corrente
long lastCheckedRadio = 0; // Tempo da última verificação de rádio

void setup(void)
{
    pinMode(ledPin, OUTPUT);
    pinMode(relayPin, OUTPUT);
    pinMode	overrideSwitchPin, INPUT);
    digitalWrite	overrideSwitchPin, HIGH); //Ativa o resistor de pull-up R
    Serial.begin(9600);
    vw_set_ptt_pin(5);
    vw_setup(2000);
    vw_rx_start();
    sensors.getAddress(thermometer, 0);
    sensors.begin();
```

```

        sensors.setResolution(thermometer, 10);
        Serial.println("Ready");
    }

void loop()
{
    if (digitalRead	overrideSwitchPin) == LOW)
    {
        // Modo automático, a temperatura de referência virá do
        // controlador de automação residencial por RF
        if (millis() > (lastCheckedRadio + RADIO_CHECK_PERIOD))
        {
            checkForMessage(); // Verificar se há mensagem
            lastCheckedRadio = millis();
        }
    }
    else
    {
        setTemp = readSetTemperature(); // Leitura da temperatura ajustada
    }
    actualTemp = readTemperature(); // Leitura da temperatura atual
    setPower(); // Energiza ou desenergiza o relé
    delay(500);
}

void checkForMessage()
{
    uint8_t buf[VW_MAX_MESSAGE_LEN];
    uint8_t buflen = VW_MAX_MESSAGE_LEN;
    if (vw_get_message(buf, &buflen))
    {
        // Há algo para receber.
        // Só nos interessam os dois primeiros bytes
        // O primeiro byte identifica a quem se destina a mensagem e
        // o segundo é a temperatura em graus F
        byte receiver = buf[0];
        byte payload = buf[1];
        if (receiver == THERMOSTAT_ID) // A
        {
            // a mensagem é para mim
            setTemp = payload;
            Serial.print("Radio set temp to: "); // Temperatura ajustada via rádio
            Serial.println(setTemp);
            flash(2);
        }
    }
}

int readTemperature()
{
    sensors.requestTemperatures();
    float tempC = sensors.getTempC(thermometer);
}

```

```

        return (int)(DallasTemperature::toFahrenheit(tempC));
    }

    int readSetTemperature()
    {
        int raw = analogRead(potPin);
        int t = map(raw, 0, 1023, MIN_TEMP, MAX_TEMP);
        return t;
    }

    void setPower()
    {
        static boolean lastOnOff = false;
        static long powerLastChanged = 0;
        static int lastSetTemp = 0;
        boolean onOff = (actualTemp < setTemp);
        long t = millis();
        long t2 = powerLastChanged + MIN_ON_TIME * 1000L;
        boolean enoughTimeElapsed = (t > t2); // Tempo decorrido suficiente
        boolean tempSettingChanged = (abs(setTemp - lastSetTemp) > 1);
        digitalWrite(ledPin, onOff);

        if ((onOff!= lastOnOff) && (enoughTimeElapsed || tempSettingChanged))
        {
            Serial.print("set:      "); Serial.println(setTemp);
            Serial.print("temp:     "); Serial.println(actualTemp);
            digitalWrite(relayPin, onOff);
            powerLastChanged = t;
            lastOnOff = onOff;
            lastSetTemp = setTemp;
        }
    }

    void flash(int n)
    {
        for (int i = 0; i <= n; i++)
        {
            digitalWrite(ledPin, HIGH);
            delay(100);
            digitalWrite(ledPin, LOW);
            delay(100);
        }
    }
}

```

Os três comandos “include” carregam as bibliotecas utilizadas pelo sketch.

A constante THERMOSTAT_ID é cópia do código para o controlador de automação residencial usado pelo aplicativo Android. É utilizada para identificar as mensagens destinadas ao termostato.

A seguir, temos quatro constantes que você pode alterar para se adequar às suas próprias necessidades.

MIN_ON_TIME é o tempo mínimo que o sistema de aquecimento permanece ligado ou desligado. Isso evita sequências muito rápidas de ligar e desligar

o sistema de aquecimento em torno do valor programado de temperatura. Isso não seria bom para o sistema de aquecimento. Ajustamos esse valor para dois minutos.

RADIO_CHECK_PERIOD é o tempo entre verificações de se há algum sinal sendo enviado pelo controlador de automação residencial. Ele está ajustado para 10.000 milissegundos ou 10 segundos.

MIN_TEMP e MAX_TEMP definem a faixa de temperatura (em graus Fahrenheit). Mesmo que o sketch assuma graus Fahrenheit como unidade de temperatura, observe que, se você trabalhar com graus Celsius, nada será preciso alterar. Bastará calibrar a escala do botão de ajuste de forma diferente.

A seguir, nós definimos os pinos utilizados pelos diversos dispositivos conectados ao microcontrolador e inicializamos as bibliotecas OneWire e DallasTemperature.

A função “setup” realiza a inicialização usual dos pinos. Um procedimento interessante com o “overrideSwitchPin” (pino de seleção de modo) é defini-lo como INPUT e, em seguida, executar um comando “digitalWrite” aplicando um nível HIGH. Como resultado, ocorre a ativação de um resistor interno de pull-up. Desse modo, nós não precisamos de um resistor separado. Começamos também a verificar se há mensagens chegando via rádio e a ativar o sensor de temperatura.

A função “loop” executa uma entre duas sequências de comandos. A primeira é seguida quando a chave de seleção está no modo automático, e a outra é seguida no caso de modo manual. Se a chave de seleção estiver no modo automático, então verificaremos se alguma mensagem foi recebida pelo receptor de RF, com a condição de que tempo suficiente tenha transcorrido desde a última vez que foi verificado se havia mensagem recebida. Caso contrário (modo manual), o ajuste de temperatura será lido no potenciômetro.

Em seguida, o “loop” (laço) principal lê a temperatura atual e chama a função “setPower” (ativa potência) para determinar se o relé deve ser ativado ou não.

A função “checkForMessage” (verificar se há mensagem) testa se uma mensagem vinda do receptor de RF está no buffer. Se houver mensagem, essa função verificará se o primeiro byte contém 0x41. Isso indica que a mensagem é destinada ao termostato e que o próximo byte contém a temperatura ajustada em graus Fahrenheit. Sempre que isso ocorrer, o LED piscará duas vezes.

A função “readTemperature” (ler temperatura) solicita uma temperatura do sensor e converte a leitura de graus Celsius em Fahrenheit.

A próxima função, “readSetTemperature” (ler temperatura ajustada), converte uma leitura do pino analógico, no qual está ligado o potenciômetro, em uma temperatura dentro da faixa de 30 a 100 usando a função de mapeamento (map). Essa função interna é muito útil para converter leituras de uma faixa de valores para outra. Os seus argumentos são o número a ser convertido, a faixa de valores não convertidos (nesse caso, 0 a 1023 na entrada analógica) e a faixa de números de saída (nesse caso, 30 a 100).

Finalmente, a função “setPower” altera o estado do relé quando é necessário. Essa função usa três variáveis estáticas que memorizam seus valores sempre que a função é chamada. Isso permite que a função determine se o estado do relé mudou, quanto tempo decorreu desde a última alteração e qual é a última temperatura ajustada. Somente quando o estado mudou de ligado para desligado (ou vice-versa) e tempo suficiente decorreu ou quando o ajuste de temperatura foi modificado é que haverá alteração do estado do relé. Essa última condição de verificar se o ajuste de temperatura foi modificado permite uma resposta mais rápida quando o usuário modifica o ajuste de temperatura utilizando o botão de controle no modo manual.

» Resumo

Agora, você poderá permanecer na cama com o telefone Android e ligar ou desligar luminárias e tomadas, além de controlar a temperatura de sua residência.

O próximo e último capítulo de automação residencial mostrará como construir um sistema de fechadura elétrica comandado pelo controlador de automação residencial utilizando etiquetas RFID.



» capítulo 10

Fechadura com RFID

Neste projeto, você aprenderá a comandar a fechadura (elétrica) da porta de sua residência por meio de uma chave RFID ou abrir a porta remotamente usando o controlador de automação residencial.

Objetivos

- » Construir um sistema de fechadura elétrica comandada pelo controlador de automação residencial utilizando etiquetas RFID
- » Examinar o sketch do Arduino deste projeto
- » Examinar como ocorre o armazenamento dos códigos RFID dos cartões na EEPROM
- » Examinar o sketch do Arduino utilizado neste projeto

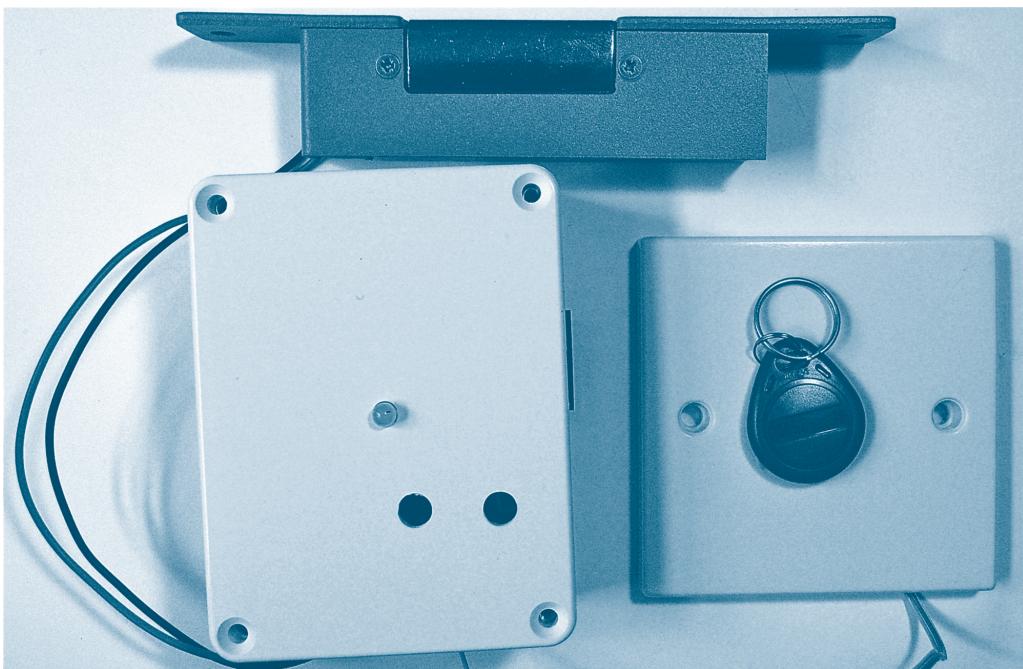


Figura 10-1 A fechadura RFID sem fio.

Se você pretender construir este projeto (Figura 10-1) e instalá-lo de fato em sua casa, será importante considerar que, embora um mecanismo de segurança baseado em RFID* e fechadura elétrica geralmente seja mais seguro do que uma fechadura comum, a capacidade de abrir a porta pela Internet ou mesmo pela sua rede doméstica apresenta alguns pontos fracos. Desse modo, você talvez queira desabilitar a abertura da fechadura pela rede.

Alguém que também tenha este livro poderia facilmente ter acesso à sua casa.

Por exemplo, para abrir a fechadura usando a interface de Web, tudo que é necessário é adivinhar a senha ou, se alguém tiver um Arduino e um transmissor AM de RF, transmitir o código-padrão para abrir a fechadura. Se a senha-padrão não tiver sido mudada, é muito fácil ter acesso. Mesmo quando a senha é mudada, há apenas cerca de 65 mil combinações. Desse modo, todas podem ser testadas automaticamente.

Portanto, esses são os riscos e, tendo isso em mente, você mesmo poderá fazer a sua avaliação e decidir como proteger a sua residência da melhor forma possível.

» Construção

A Figura 10-2 mostra o diagrama esquemático do projeto. Esse é outro projeto em que o microcontrolador é programado em uma placa de Arduino e depois inserido em outra placa. Sob diversos aspectos, é um projeto muito similar ao do termostato. Utiliza o mesmo tipo de receptor de RF, mas, no lugar de um relé, há um transistor de potência MOSFET que comanda a fechadura da porta.

A fechadura requer uma fonte de alimentação de 12V. Para isso, utilizaremos um CI regulador de tensão para gerar a alimentação de 5V do microcontrolador. Também será necessário um LED tricolor. Esses LEDs são, na realidade, dois LEDs – um vermelho e um verde – contidos em um mesmo encapsulo.

* N. de T.: RFID – Radio Frequency Identification (identificação por radiofrequência).

sulamento e com seus catodos (lados negativos) ligados juntos. Você pode acender separadamente o vermelho e o verde e também os dois ao mesmo tempo. Nesse caso, você terá a terceira cor, a laranja.

» O que será necessário

Os componentes listados na tabela *Lista de Componentes*, a seguir, serão necessários para construir a chave RFID e a fechadura da porta.

Você também precisará de um Arduino Uno ou Duemilanove para programar o chip ATMega328 ou o ATMega168. Quando você comprar o ATMega, assegure-se de comprar uma versão com o bootloader do Arduino já instalado.

Este projeto usa o mesmo módulo RF de 433MHz que utilizamos no projeto do termostato. Neste caso, não há necessidade de outro transmissor, porque o mesmo transmissor do controlador de

automação residencial pode enviar mensagens ao termostato e também à fechadura da porta.

O solenoide da fechadura requer uma alimentação de 12V CC com algumas centenas de miliamperes. Para isso, usaremos um adaptador de voltagem CA/CC de 12V com uma saída de corrente de, no máximo, 1A.

O módulo RFID foi comprado no eBay (onde é possível obter bons preços) na forma de um kit que inclui um módulo RFID de 125-kHz, uma bobina de antena e diversos cartões RFID. Os pinos de conexão e o formato do código de saída estão mostrados na Figura 10-3. Se você não encontrar exatamente o mesmo módulo com a mesma disposição de pinos, faça uma transposição de pinos. Para isso, soldue uma barra de pino fêmea (barra de soquetes para CI) na placa perfurada e, com fios e uma barra de pino macho (barra de pinos para CI), faça as ligações e transposições apropriadas para o módulo que você adquiriu. A seguir, insira a barra de pinos na

LISTA DE COMPONENTES

Componente	Quantidade	Descrição	Fornecedor
Microcontrolador	1	ATMega328 com bootloader	Sparkfun: DEV-10524
Fechadura elétrica de porta	1	Fechadura de 12V CC ou equivalente	Farnell: 4333550
Receptor de RF	1	Módulo receptor de 433 MHz	Farnell: 1304026
Soquete para CI	1	Soquete para circuito integrado de 28 pinos	Farnell: 1824463
R1, R2	2	Resistor de filme metálico de $1k\Omega$ e 1/2 W	Farnell: 9339779
R3, R4	2	Resistor de filme metálico de 270Ω e 1/2 W	Farnell: 9340300
C1	1	Capacitor eletrolítico de $100\mu F$ e 16V	Farnell: 1136275
C2	1	Capacitor eletrolítico de $1\mu F$ e 16V	Farnell: 1236655
C3	1	Capacitor cerâmico de $100nF$	Farnell: 1200414
C4, C5	2	Capacitor cerâmico de $22pF$	Farnell: 1600966
D1	1	LED tricolor de catodo comum	Farnell: 1581197
D2	1	1N4001	Farnell: 1458986
T1	1	FQP33N10 – MOSFET de potência de canal n	Farnell: 9845534
CI1	1	Regulador de tensão 7805	Farnell: 1261233
X1	1	Crystal de 16 MHz	Farnell: 1611761
S1, S2	2	Chave táctil	Farnell: 1448152
Placa perfurada trilhada	1	29 trilhas de 24 furos	Farnell: 1201473
Barra de pino fêmea	1	Para os módulos RFID e RF	Farnell: 1218869
Barra de pino macho	1	Uma com dois pinos para a antena RFID	Farnell: 1097954
Fonte de alimentação	1	Adaptador de voltagem CA/CC de 12V e 1A	Farnell: 1279478
Tampa de plástico	1	Para acondicionar a antena RFID	
Caixa para projeto	1		Lojas de eletrônica locais
Jack	1	Jack J4 para fonte de alimentação de 2,1 mm	Farnell: 1217038

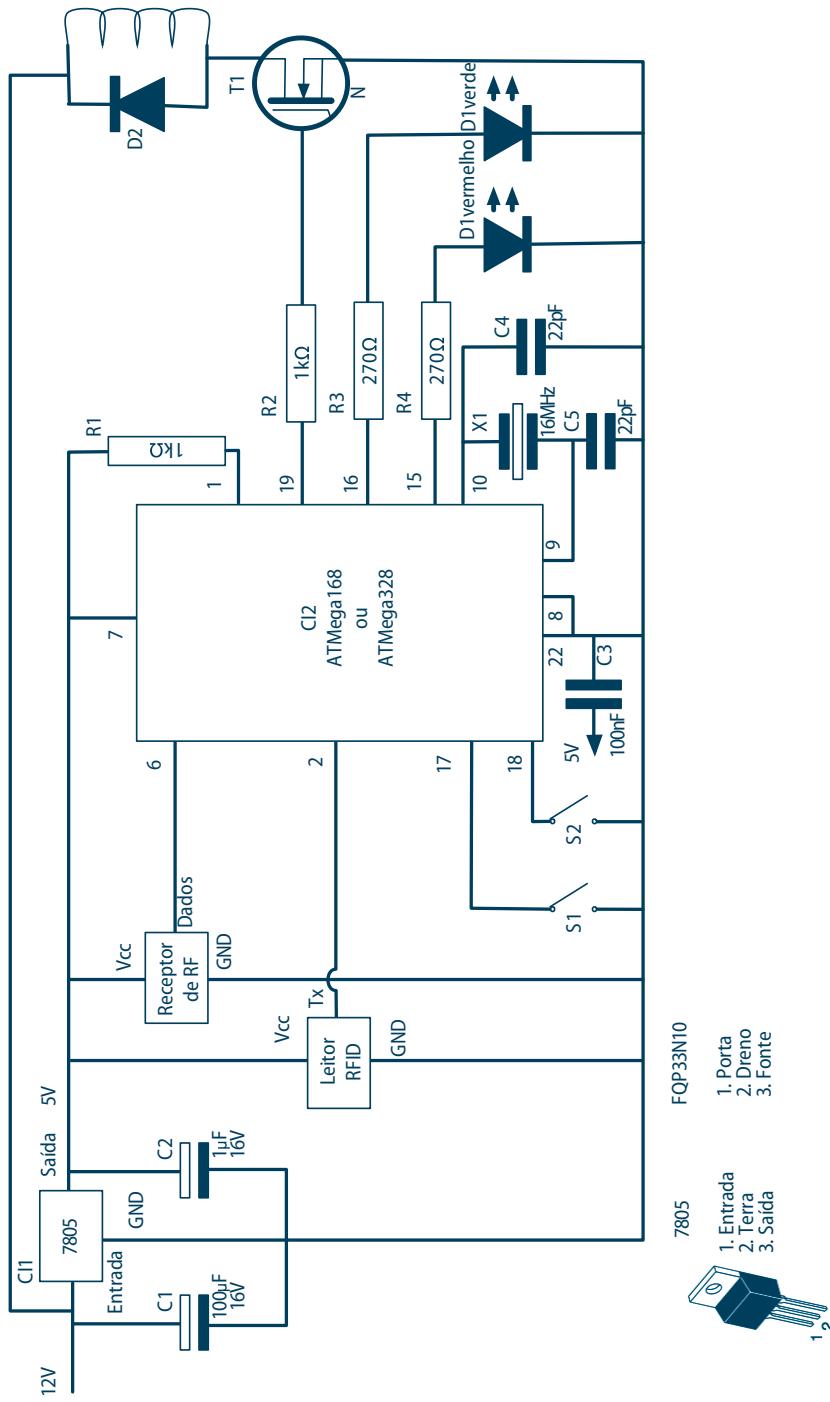


Figura 10-2 O diagrama esquemático.

barra de soquetes, conferindo antes se as conexões estão corretas. Você talvez tenha que alterar o sketch se o código de saída for diferente.

A fechadura elétrica da porta é do tipo que se encaixa no marco da porta. Quando 12V são aplicados a seus terminais, é acionado um solenoide que libera a trava no marco da porta. É um tipo de fechadura que mantém a porta trancada quando falta energia elétrica. Isso significa que, se faltar energia, a porta se manterá fechada a menos que você use uma chave comum para abri-la no lugar da chave RFID. Este projeto trabalha bem com qualquer mecanismo de fechadura elétrica que funcione com 12V. Se você precisar de alguma coisa diferente da fechadura especificada na lista de componentes, faça uma pesquisa e veja o que você consegue achar.

Além desses componentes, você precisará das ferramentas listadas a seguir.

CAIXA DE FERRAMENTAS

- Ferramentas para soldar
- Estilete e régua
- Fios flexíveis de diversas cores
- Uma furadeira elétrica com brocas
- Um multímetro
- Ferramentas de marceneiro para instalar a fechadura

» Passo 1: prepare a placa perfurada

A Figura 10-4 mostra a disposição dos componentes na placa perfurada para a fechadura.

Corte a placa no tamanho correto. A maneira mais simples de fazer isso é riscar fortemente a placa com um estilete e, em seguida, quebrá-la na beira de uma mesa. Você também pode usar uma tesoura, mas o resultado não será tão bom.

A placa assim preparada está mostrada na Figura 10-5.

Use uma microrretífica para fazer os cortes nos locais marcados com "X" nas trilhas. Isso pode ser feito executando movimentos circulares com a ponta da microrretífica, que você segura entre seu polegar e o dedo indicador. A Figura 10-4 mostra a placa vista do lado de cima (dos componentes), havendo 24 cortes a serem feitos nas trilhas.

» Passo 2: soldé as conexões

O próximo passo é soldar as conexões de fio na placa. Novamente, há diversas conexões para fazer (9). Portanto, tome cuidado para se assegurar de que todas sejam feitas e estejam nos lugares corretos.

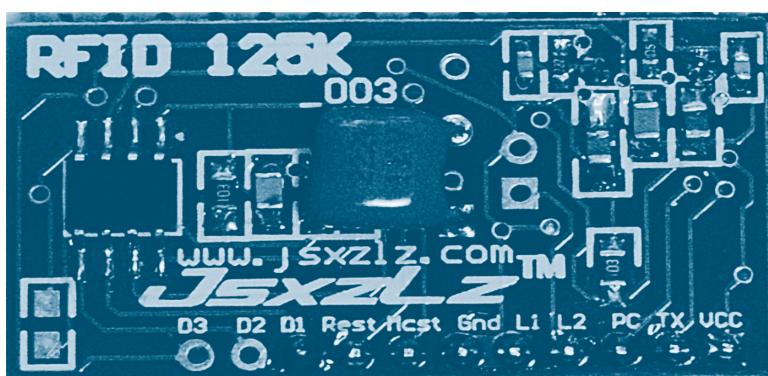


Figura 10-3 O módulo RFID usado no projeto.

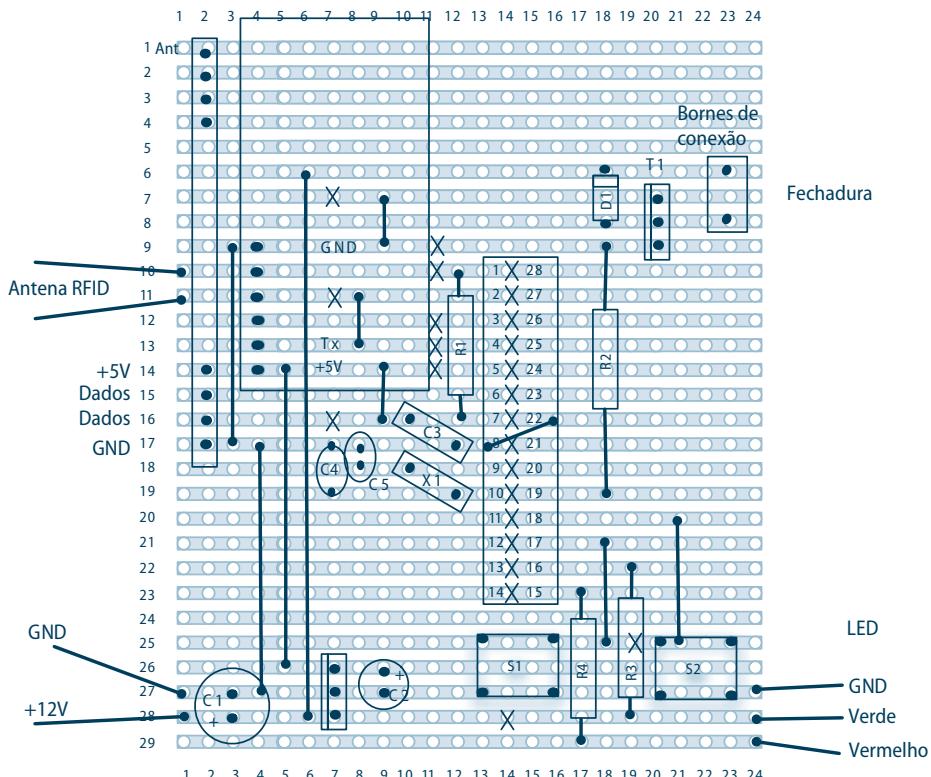


Figura 10-4 A disposição dos componentes na placa perfurada.

No lado esquerdo, há uma conexão longa. Deixe-a encapada para protegê-la de algum curto-circuito acidental.

Quando todas as conexões estiverem soldadas, a placa deverá ser como a mostrada na Figura 10-6.

» Passo 3: solde os resistores e o diodo

Sempre é mais fácil iniciar a soldagem com os componentes baixos. Portanto, soldaremos primeiro os resistores e o diodo.

Verifique se o diodo está na orientação correta, com o lado da faixa voltado para a parte de cima da placa. A distância entre os orifícios para os terminais do diodo é pequena, levando-nos a instalá-lo de lado, como mostrado na Figura 10-7 em cima à direita.

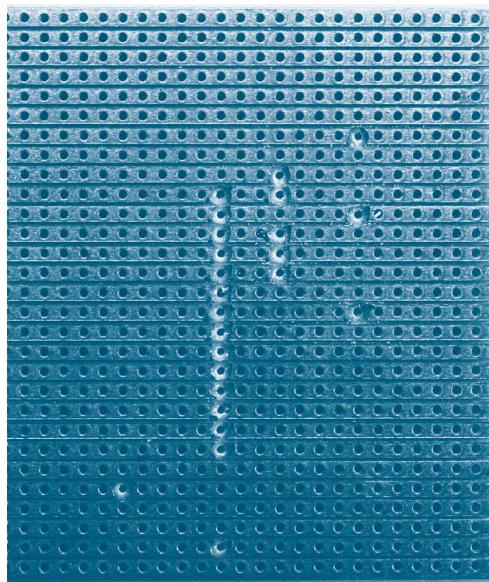


Figura 10-5 A placa perfurada pronta (lado das trilhas).

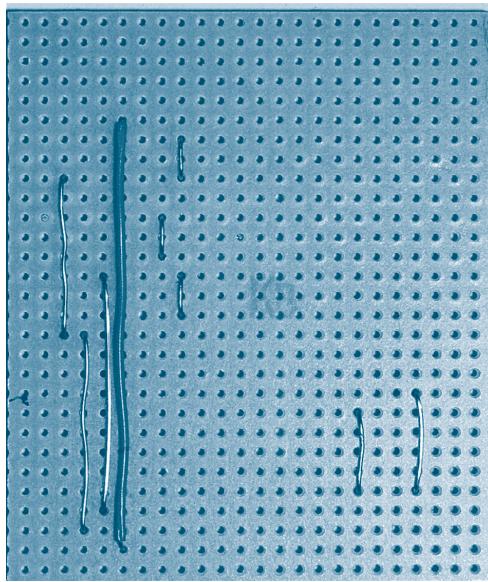


Figura 10-6 A placa perfurada com as conexões de fio.

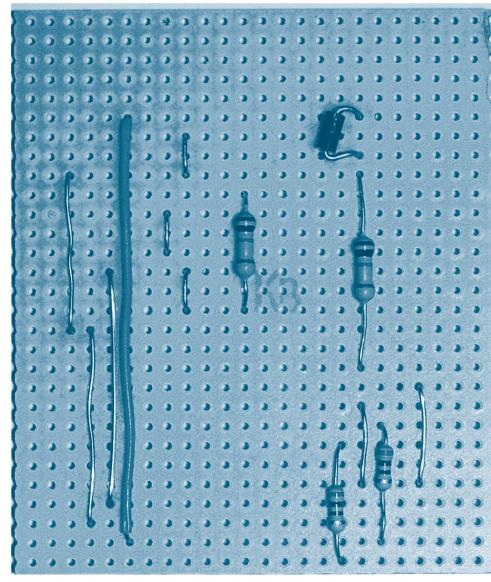


Figura 10-7 A placa perfurada com os resistores e o diodo.

» Passo 4: solde o soquete do Cl e as chaves tácteis

Agora você pode soldar o soquete do Cl e as duas chaves. O soquete do Cl tem uma marca no lado que indica a posição do pino 1 do Cl. É uma boa ideia colocá-lo com a marca para o lado de cima da placa, como mostra a Figura 10-8. Assim, quando chegar o momento de encaixar o Cl do microcontrolador, você saberá como colocá-lo no soquete.

Verifique se você soldou as chaves tácteis com a orientação correta. Elas devem ser orientadas de tal forma que, quando um botão é pressionado, os pinos dos contatos que são feitos estão em cima e embaixo. Você pode usar um multímetro para verificar isso.

» Passo 5: solde os demais componentes

Agora, os demais componentes e soquetes podem ser colocados na placa (Figura 10-9). Observe que ainda não instalamos o circuito integrado nem os módulos RFID e de RF.

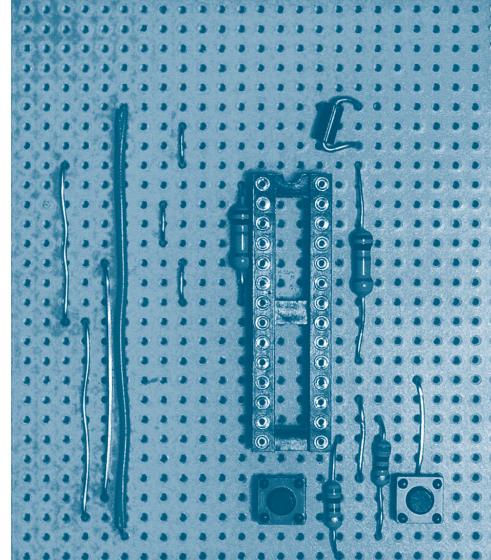


Figura 10-8 A placa perfurada com o soquete do Cl e as chaves tácteis.

Os capacitores pequenos podem ser soldados com qualquer orientação, mas os outros capacitores tubulares eletrolíticos devem ser soldados com a

orientação correta. Os terminais positivos são mais compridos e os negativos normalmente têm uma marca em forma de diamante.

Do mesmo modo, verifique se o transistor e o regulador de tensão estão inseridos corretamente. Tome cuidado para não confundi-los já que têm o mesmo tipo de encapsulamento.

Aqui utilizamos soquetes para os módulos, mas, se você preferir, poderá soldá-los diretamente na placa. Corte a barra de pino fêmea em duas seções para construir dois soquetes, um para os quatro pinos do módulo de RF (você só precisa conectar a metade de baixo) e um para os seis pinos do módulo RFID. Você pode cortar outra seção para os quatro pinos da parte de cima do módulo de RF, mas isso não é essencial.

Um par de pinos é usado para conectar a antena do módulo RFID.

» Passo 6: programe e instale o microcontrolador

Precisamos instalar no microcontrolador o sketch que controla a fechadura. Para isso, utilizaremos uma placa de Arduino.

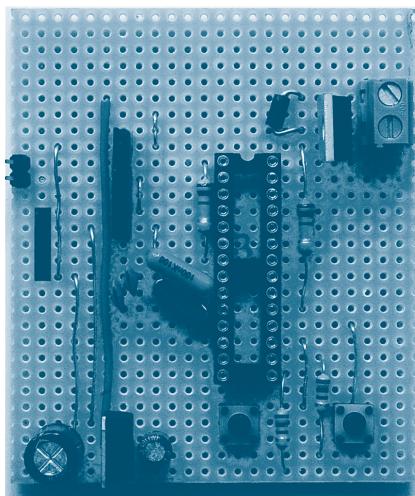


Figura 10-9 A placa perfurada com todos os componentes.

O mais fácil a fazer é programar o microcontrolador que já está na placa do Arduino e, depois de transferi-lo para a placa perfurada, instalar na placa do Arduino o microcontrolador que você comprou.

Antes de instalar o sketch, defina as duas constantes – CODE_1 e CODE_2 – com valores próprios para o seu endereço. Lembre-se de que esses valores representam a chave de acesso à sua residência. O valor de CODE_1 deve estar entre 0x42 e 0xFF e o valor de CODE_2, entre 0x00 e 0xFF. Esse código deve corresponder ao código de quatro dígitos configurado na página de preferências do aplicativo Android de automação residencial. Assim, se você definir CODE_1 como 0xE5 e CODE_2 como 0x77, então o código configurado nas preferências será E577.

Esse projeto utiliza a biblioteca VirtualWire.* Se você não a instalou no Capítulo 9, então siga as instruções do Passo 5 do Capítulo 9.

Instale o sketch atualizado (rfid_door) na placa do Arduino. Então, com a alimentação elétrica desligada, remova cuidadosamente o microcontrolador da placa e encaixe-o no soquete da placa perfurada, verificando se está na orientação correta.

Ao remover esse circuito integrado, tome cuidado porque os pinos podem facilmente se curvar ou quebrar. Levante o circuito integrado pelas extremidades com uma faca, um pouco de cada vez, até se soltar. Você também tem que tomar cuidados com a eletricidade estática, encostando primeiro em alguma coisa aterrada e descarregando a eletricidade do seu corpo antes de tocar nos pinos. Uma pulseira antiestática de aterramento é outra opção.

Neste ponto, uma inspeção completa da placa é uma boa ideia. Verifique, na parte de cima da placa, se todos os componentes e as conexões estão nos lugares indicados pela Figura 10-4. Examine também, na parte debaixo da placa, se todos os cortes de trilha estão nos lugares corretos e se não há pontes acidentais de solda entre as trilhas.

* N. de T.: Mais informações sobre a biblioteca VirtualWire podem ser obtidas em <http://www.airspayce.com/mikem/arduino/> e <http://www.airspayce.com/mikem/arduino/VirtualWire.pdf>.

» Passo 7: ligando tudo

Solde fios flexíveis na placa perfurada para a alimentação de 12V. Eles podem ser soldados na parte debaixo da placa perfurada ou passados através dos orifícios desde o lado de cima da placa. Utilize o método que você achar melhor. Solde as outras extremidades desses fios a um conector de alimentação de 2,1mm (cuidando as polaridades). A seguir, solde o LED tricolor usando fios curtos (veja a Figura 10-10). Consulte os dados de especificação do seu LED. No caso do LED que utilizamos, o terminal mais longo era o catodo comum ligado ao terra (GND). O terminal mais curto era o anodo verde, e o terminal do meio era o anodo vermelho.

Coloque os módulos RFID e de RF no lugar, cuidando para inseri-los com a orientação correta.

» Passo 8: teste

Antes de começar a acondicionar tudo em uma caixa, precisamos conferir o que fizemos e assegurar

que tudo está no lugar certo. Ficará muito mais fácil se você verificar e fizer as correções antes da colocação do projeto em uma caixa.

Comece instalando a antena RFID, o solenoide e o conector de alimentação (Figura 10-11).

Você deve observar que, quando liga a alimentação elétrica, o LED pisca com luz vermelha. Em seguida, pisca com luz verde e, então, com luz laranja duas vezes. Essa é uma pequena verificação inicial para conferir se o microcontrolador está funcionando.

Aproxime da antena um dos cartões RFID que veio com o kit e o LED deverá piscar com luz vermelha. Isso indica que o cartão foi lido mas que não foi reconhecido pelo sistema. Afaste o cartão e pressione o botão táctil esquerdo. O LED deve piscar quatro vezes com cor verde para indicar que o sistema incluiu o cartão que acabou de ser lido na sua lista de cartões conhecidos.

Agora, quando esse cartão for aproximado da antena, você verá o LED verde piscar e ouvirá a fechadura abrir por cinco segundos. Durante esse tempo, você não poderá fazer outro reconhecimento de cartão.

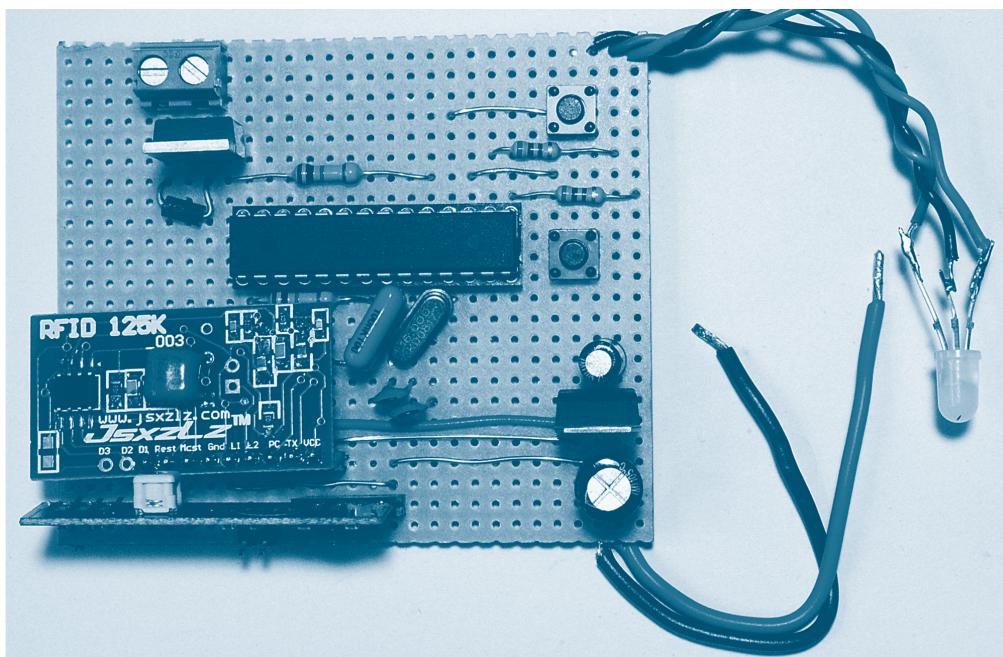


Figura 10-10 Fazendo as conexões com a placa perfurada.

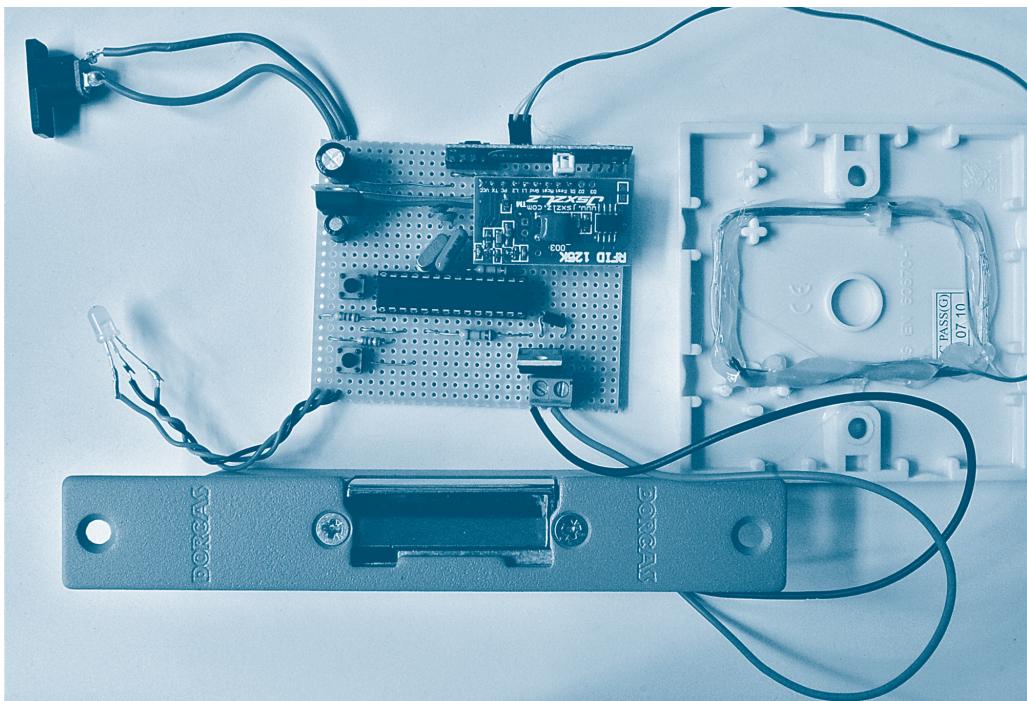


Figura 10-11 Pronto para ser testado com todas as conexões realizadas.

Se o LED verde piscar e a fechadura não abrir, verifique a região da placa perfurada em torno de T1 e as conexões entre a fechadura e a placa perfurada. Use um multímetro para verificar se há 12V nos fios de conexão com a fechadura.

Também precisamos testar se a porta abre remotamente através de RF. Para isso, ative o seu sistema de automação residencial e escolha a opção “Door” (porta) no menu principal (Figura 10-12). A seguir, forneça a senha (“Password”). A senha-padrão adotada pelo sistema é a palavra “password”. Após, clique em “Unlock” (destrancar ou abrir a porta). Você deve ouvir a fechadura abrir por cinco segundos.

Se nada ouvir e se ver o LED piscar com luz laranja, então isso significará que um sinal foi recebido mas que o código estava errado. Portanto, verifique se os valores que você definiu para CODE_1 e CODE_2 no sketch conferem com o valor usado na configuração (Figura 10-13) do controlador de automação residencial (veja o Passo 6 deste capítulo).

» Passo 9: acondicione o projeto

Agora, podemos ir em frente e acondicionar tudo dentro de uma caixa. Precisaremos de uma caixa para a placa perfurada, o LED e os demais itens que estão instalados no interior da casa. Também será necessário uma tampa de plástico para acondicionar a antena RFID no lado de fora da casa.

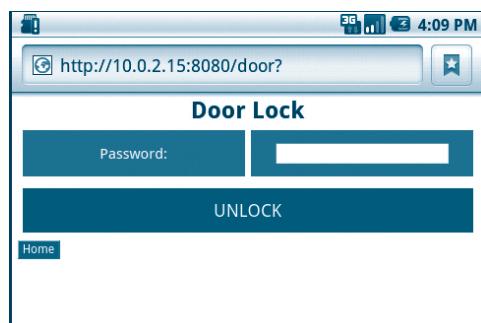


Figura 10-12 Abertura remota da porta.



Figura 10-13 Configurando a senha da porta.

Começando com a antena, ela poderá ser instalada dentro de uma tampa de plástico, como mostrado na Figura 10-14. Simplesmente colocamos a antena na parte interna da tampa que, a seguir, pode ser parafusada na parede que fica próximo à porta.

Essa forma pode ser utilizada quando a porta está debaixo de um pórtico e protegida. Entretanto, se a antena estiver exposta às intempéries, você deverá selá-la ou encontrar um recipiente à prova de mau tempo.

A caixa de comando que fica no interior da casa pode ser simplesmente uma caixa comum de projeto. Você precisará fazer orifícios nas partes laterais e detrás da caixa para:

- O cabo da antena RFID
- O conector de alimentação de 2,1mm
- Os fios de conexão da fechadura
- A fixação da caixa na parede

A parte frontal da tampa da caixa terá um orifício de 5mm para o LED e dois orifícios que ficam imediatamente por cima das duas chaves tácteis. Uma caneta ou outro objeto delgado de plástico pode ser inserido através desses orifícios para pressionar as duas chaves. A Figura 10-15 mostra a tampa e a caixa principal com os orifícios. Distribua os componentes na caixa e faça as marcações dos orifícios antes de começar a fazer os furos.

As chaves tácteis estão escondidas atrás dos orifícios para dificultar o acesso e evitar que cartões não autorizados sejam programados.

» Passo 10: instalação

O procedimento de instalação da fechadura dependerá do tipo de fechadura que você comprou. Se for similar à recomendada na lista de componentes, você provavelmente deverá escavar uma cavidade maior no marco da porta. Você também deve fazer um furo através da parede ou do marco

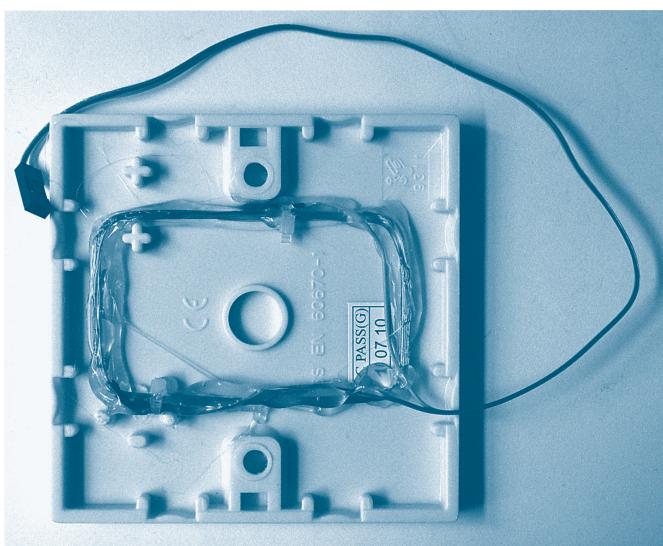


Figura 10-14 Acondicionando a antena RFID.

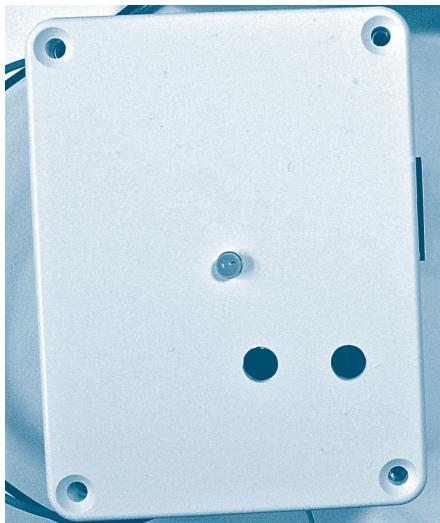


Figura 10-15 A caixa de comando com os orifícios.

da porta para passar os fios de conexão da antena RFID. Esse furo deve ficar atrás da caixa de comando que está no interior da casa.

A caixa de comando deve estar próximo da fechadura para acomodar as conexões entre elas.



Figura 10-16 A caixa de comando.



Figura 10-17 A antena RFID instalada no lado de fora da residência.

A Figura 10-16 mostra a caixa de comando (sem a tampa), e a Figura 10-17 mostra a antena RFID e a fechadura.

» Usando o sistema

O sistema é fácil de usar. Há apenas dois botões. O botão esquerdo é utilizado para incluir o último cartão lido na lista de cartões que abrem a porta. O botão direito é utilizado para limpar a memória e “esquecer” todos os cartões.

O LED tricolor é usado para informar o que o sistema está fazendo no momento. As diversas combinações de flashes luminosos estão resumidas na Tabela 10-1.

TABELA 10.1 Resumo do significado dos flashes do LED

Flashes	Significado
Dois de cor laranja	Foi recebido um sinal do controlador de automação residencial, mas não é destinado à fechadura. Útil para verificar se o controlador de automação residencial está funcionando.
Quatro de cor laranja	Foi recebido um comando vindo do controlador de automação residencial para abrir a fechadura. Será seguido imediatamente de um flash de luz verde.
Um de cor verde	A porta foi aberta.
Um de cor vermelha	Um cartão não válido foi lido.
Dois de cor verde	Após pressionar o botão de acrescentar cartão, isso indica que o cartão já era conhecido.
Quatro de cor verde	Após pressionar o botão de acrescentar cartão, isso indica que o cartão não era conhecido antes e que agora foi incluído na EEPROM.
Cinco de cor vermelha	Após pressionar o botão de acrescentar cartão, isso indica que todas as 16 posições de memória estão ocupadas e que o cartão não foi memorizado.
Dez flashes rápidos de cor vermelha	Isso indica que o botão de limpeza de memória foi pressionado e que todas as 16 posições foram apagadas.

» Teoria

Nesta seção, examinaremos o sketch de Arduino deste projeto, especialmente como ocorre o armazenamento dos códigos RFID dos cartões na EEPROM.

» O sketch para a fechadura

O sketch para a fechadura é relativamente longo e complexo em comparação com os sketches dos de-

```
#include <EEPROM.h>
#include <VirtualWire.h>
#define CODE_1 0xA0
#define CODE_2 0x45
#define doorOpenTime 5000 // 5 segundos
#define numCodes 16
#define codeSize 14
#define redPin 9
#define greenPin 10
#define doorReleasePin 13
#define addButtonPin 11
#define removeButtonPin 12
#define rfRxPin 4

#define RED 1
#define GREEN 2
#define ORANGE 3

char code[codeSize + 1];
char *EMPTY_CODE = "0000000000000000";
```

```
void setup()
{
    Serial.begin(9600);
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(doorReleasePin, OUTPUT);
    pinMode(addButtonPin, INPUT);
    digitalWrite(addButtonPin, HIGH);
    pinMode(removeButtonPin, INPUT);
    digitalWrite(removeButtonPin, HIGH);
    flash(RED, 2, 200);
    flash(GREEN, 2, 200);
    flash(ORANGE, 2, 200);

    vw_setup(2000);
    vw_set_ptt_pin(5);
    vw_set_tx_pin(6);
    vw_set_rx_pin(rfRxPin);
    vw_rx_start();
}

void loop()
{
    if (Serial.available() > codeSize)
    {
        readCard();
        checkCode();
    }

    if (!digitalRead(removeButtonPin))
    {
        clearAllCodes();
```

```

        }
        if (!digitalRead(addButtonPin))
        {
            addCode(code);
        }
        checkForMessage();
    }

void readCard()
{
    int i = 0;
    char ch = Serial.read();
    while (ch != '\n' && i <= codeSize)
    {
        code[i] = ch;
        ch = Serial.read();
        i++;
    }
    Serial.flush();
}

void checkForMessage()
{
    uint8_t buf[VW_MAX_MESSAGE_LEN];
    uint8_t buflen = VW_MAX_MESSAGE_LEN;
    if (vw_get_message(buf, &buflen))
    {
        // Havia algo para receber.
        flash(ORANGE, 1, 200);
        if (buf[0] == CODE_1 && buf[1] == CODE_2)
        {
            flash(ORANGE, 2, 200);
            unlockDoor();
        }
    }
}

void checkCode()
{
    if (isValidCode(code))
    {
        unlockDoor();
    }
    else
    {
        flash(RED, 1, 500);
    }
}

void unlockDoor()
{
    clearLastCode();
    flash(GREEN, 1, 200);
    digitalWrite(doorReleasePin, HIGH);
    delay(doorOpenTime);
    digitalWrite(doorReleasePin, LOW);
    Serial.flush();
}

void clearLastCode()
{
    for (int i = 0; i < codeSize; i++)
    {
        code[i] = 'F';
    }
}

void flash(int color, int times, int duration)
{
    int red = color & 0x01;
    int green = color >> 1;
    for (int i = 0; i < times; i++)
    {
        digitalWrite(redPin, red);
        digitalWrite(greenPin, green);
        delay(duration / 2);
        digitalWrite(redPin, LOW);
        digitalWrite(greenPin, LOW);
        delay(duration / 2);
    }
}

boolean isValidCode(char *code)
{
    return (findCodePosition(code) < numCodes);
}

void addCode(char *code)
{
    if (isValidCode(code))
    {
        // Código já armazenado
        flash(GREEN, 2, 500);
    }
    else
    {
        int pos = findCodePosition
        (EMPTY_CODE);
        if (pos != (codeSize + 1))
        {
            writeCode(pos, code);
            flash(GREEN, 4, 500);
        }
        else
        {
            // Não há espaço para
    }
}

```

```

        // armazenar código
        flash(RED, 5, 500);
    }
}

int findCodePosition(char *code)
{
    int pos = 0;
    while (pos < numCodes &&
           !codesEqual(code, pos))
    {
        pos++;
    }
    return pos;
}

void writeCode(int pos, char *code)
{
    for (int i = 0; i < codeSize; i++)
    {
        EEPROM.write(pos * 16 + i,
                      code[i]);
    }
}

void clearAllCodes()
{
    for (int pos = 0; pos < numCodes;
         pos++)
    {
        writeCode(pos, EMPTY_CODE);
    }
    flash(RED, 10, 50);
}

boolean codesEqual(char *code, int pos)
{
    for (int i = 0; i < codeSize; i++)
    {
        char ch = (char)EEPROM.read(pos
                                     * 16 + i);
        if (code[i] != ch)
        {
            return false;
        }
    }
    return true;
}

```

mais projetos deste livro. A complexidade deve-se em parte à necessidade de treinar o Arduino para reconhecer os códigos de cartão RFID válidos e

também armazená-los de tal forma que não sejam esquecidos quando a energia elétrica é desligada.

Esse é um sketch bem estruturado e um bom exemplo de como uma lógica complexa pode ser simplificada mantendo pequeno o tamanho das funções.

No início, incluímos primeiro as bibliotecas VirtualWire e EEPROM. A biblioteca VirtualWire permite receber mensagens do Arduino que está instalado no controlador de automação residencial. A biblioteca EEPROM permite armazenar os códigos RFID na memória EEPROM para que sejam preservados mesmo quando não houver energia elétrica alimentando a placa do Arduino.

As duas constantes CODE_1 e CODE_2 devem ser definidas com valores próprios para o seu endereço. Lembre-se de que esses valores representam a chave de acesso à sua residência. O valor de CODE_1 deve estar entre 0x42 e 0xFF e o valor de CODE_2, entre 0x00 e 0xFF. Esse código deve corresponder ao código de quatro dígitos configurado na página de preferências do aplicativo Android de automação residencial. Assim, se você definir CODE_1 como 0xE5 e CODE_2 como 0x77, então o código configurado nas preferências deverá ser E577.

A constante “doorOpenTime” (tempo de abertura da porta) define o tempo durante o qual a porta permanece aberta após ser destrancada. O número é em milissegundos. Portanto, 5000 é 5 segundos.

Após a definição usual de pinos, a função “setup” inicializa adequadamente os pinos e, então, põe em execução a biblioteca VirtualWire que fica ouvindo as mensagens pelo rádio.

O leitor RFID está conectado ao pino Rx do Arduino. Assim, sempre que um cartão de identificação estiver suficientemente próximo para ser lido, ele enviará a identificação (ID) ao Arduino. A primeira coisa que fazemos na função “loop” é verificar se recebemos algum dado de cartão utilizando a função Serial.available(). Se for o caso, o código será lido e conferido para ver se é um código RFID válido. Após, a função “loop” também verifica se algum dos botões táctéis foi pressionado e executa as ações necessárias. Finalmente, a função “loop” confere se chegou alguma mensagem pelo rádio.

Essa descrição explica o que está acontece de forma não detalhada. Cada uma dessas ações é implementada por meio de uma série de camadas de funções, que serão discutidas agora.

A função “readCard” (ler cartão) lerá um caractere de cada vez enquanto não atingir o tamanho correto da mensagem, ou nada mais houver para ler, ou ainda enquanto não for lido o caractere “\n” de nova linha. O ID do cartão recém-lido é colocado na variável global “code”. Essa variável é utilizada em todo o sketch para se referir ao último código lido.

A função “checkForMessage” (verifica se há mensagem) testa se foi recebida uma mensagem via rádio. Se for o caso, o código será conferido para verificar se é válido e, em caso afirmativo, a função destrancará a fechadura.

A função “checkCode” (verifica código) simplesmente confere se o último código lido é reconhecido pelo Arduino como sendo válido. Se for o caso, ela destranca a porta.

É óbvia a finalidade da função “unlockDoor” (destrancar a porta). O comando “Serial.flush()” apaga as leituras subsequentes do cartão que ainda estão presentes no buffer. Se isso não fosse feito, a fechadura seria destrancada repetidas vezes. A razão disso é que o leitor de cartão repete a leitura muitas vezes.

A função “clearLastCode” (limpa o último código) apaga o último código que foi lido. Ela é chamada logo que a porta é destrancada. Se não houvesse essa função, a porta seria aberta repetidas vezes até que um cartão desconhecido fosse lido.

A função “flash” é usada para informar o estado do sistema ao usuário. Ela espera receber três argumentos. O primeiro é a cor, que pode ser RED, GREEN ou ORANGE (vermelho, verde ou laranja). O segundo argumento é o número de vezes que o LED deve piscar, e o último argumento é o intervalo de tempo, em milissegundos, entre os flashes. A função pode ser chamada de qualquer lugar do sketch para indicar o que está acontecendo.

A função “isValidCode” (o código é válido) verifica se o código recebido como argumento é um código RFID reconhecido e válido. Ela está acima da função “findCodePosition” (encontre a posição do código), que lê cada um dos 16 códigos possíveis para verificar se o código fornecido confere com algum armazenado na memória. Se conferir, a sua posição na memória será retornada. Se não conferir, o valor 16 será retornado.

A função “addCode” (acrescenta código) é utilizada quando um novo código é incluído. Isso ocorre como resultado de alguém pressionar a chave táctil que acrescenta código. Se o código já existir na memória, a função fará o LED verde piscar duas vezes. Caso contrário, a função procura a primeira posição não usada, entre as 16 posições de códigos, e armazena ali o código. Finalmente, ela faz o LED verde piscar quatro vezes para confirmar a inclusão do código. Se todas as 16 posições estiverem em uso, a função fará o LED vermelho piscar cinco vezes para indicar que não foi possível armazenar o novo código.

A função “writeCode” (escreve código) recebe dois argumentos: o código que será escrito (memorizado) e a posição, de 0 a 15, na qual ele deverá ser armazenado. Todas as leituras e escritas na memória EEPROM devem ser feitas com um byte de cada vez.

Como o nome sugere, a função “clearAllCodes” (limpa todos os códigos) apaga todos os 16 códigos, tornando-os disponíveis novamente para armazenar IDs de cartões.

Finalmente, a função “codesEqual” (código é igual) compara um código com o código presente em uma posição especificada.

» Resumo

Este projeto foi o último a fazer parte do sistema de automação residencial. No próximo capítulo, veremos como um Arduino, com um shield de Ethernet, pode ser utilizado para realizar algumas tarefas simples baseadas na Web.



» capítulo 11

Bandeirolas de sinalização

Neste capítulo, mostraremos como construir um sistema que permite chamar um atendente para trazer toalhas quentes ou cálices de vinho. Esse sistema pode ser usado enquanto você está em sua banheira desfrutando um momento de descanso e relaxamento. Para isso, na cozinha de sua residência, deve haver duas colunas com bandeirolas em seus topo. Essas bandeirolas são erguidas ou baixadas para fazer pedidos aos atendentes.

Objetivos

- » Entender como Arduino pode controlar bandeirolas que podem ser ativadas a partir de qualquer dispositivo conectado à Internet
- » Examinar o que são servomotores
- » Examinar como controlar um servomotor em um sketch de Arduino

Essas bandeirolas de sinalização, que podem ser erguidas ou baixadas por meio de comandos enviados pela Internet, são acionadas por servomotores para dar instruções aos atendentes sobre seus pedidos. Ver Figura 11-1. Além de pedidos de toalhas quentes e vinho, outras solicitações podem ser atribuídas às bandeirolas.

No nosso caso, as bandeirolas são comandadas a partir da página de Web “Minion Summoner” (chamador de atendente) mostrada na Figura 11-2, na qual podemos ver as teclas virtuais “More Wine” (mais vinho), “Cancel Wine” (cancela vinho), “Warm Towels” (toalhas quentes) e “Cancel Towels” (cancela toalhas).



Figura 11-1 As bandeirolas de sinalização comandadas via Internet.



Figura 11-2 A página de Web usada para controlar as bandeirolas.

As bandeirolas são instaladas no lugar conveniente para exercer a sua função. Quando é emitido um comando, a bandeirola gira erguendo-se até ficar na vertical. Quando desativada, ela gira em sentido contrário baixando até a posição de repouso.

Um outro uso para esse projeto foi dado por entusiastas da metodologia de desenvolvimento de software denominada eXtreme Programming. Eles usaram esse projeto para “levantar uma bandeirinha” quando uma integração estivesse sendo feita, conforme sugerido por Kent Beck.

» Construção

Esse projeto é de fácil construção e não exige muitas soldas, fora a ligação de algumas barras de pino macho a alguns fios. Entretanto, é necessário realizar alguns trabalhos simples de marcenaria.

O projeto utiliza um Arduino e um shield de Ethernet. Isso permite que o Arduino funcione como um pequeno servidor de Web. Assim, solicitações (requests) podem ser enviadas ao Arduino a partir de um navegador de Web para erguer ou baixar as bandeirolas.

As bandeirolas são montadas em pequenos servomotores, que por sua vez estão instalados no topo de colunas de madeira. Elas são feitas de papel e fixadas em hastes delgadas de madeira. Quando o servo gira, a bandeirola é erguida.

» O que será necessário

Você precisará dos itens mostrados na tabela *Lista de Componentes*, a seguir, para construir este projeto.

Os servomotores foram obtidos no eBay. Motores pequenos de 9g foram utilizados porque precisam de tão pouca corrente que podem ser acionados diretamente por um pino de saída do Arduino. Procure servomotores com uma especificação de corrente menor que 40 mA.

Como na maioria dos projetos deste livro, você precisará de fios para fazer as conexões. Neste projeto, você precisará de mais fio do que o usual porque você deverá estender fios entre os servos e o shield de Ethernet.

Além desses componentes, você precisará também das ferramentas listadas na Caixa de Ferramentas.

CAIXA DE FERRAMENTAS

- Uma furadeira elétrica com brocas
- Uma serra para madeira
- Um estilete
- Parafusos diversos
- Cola para madeira
- Uma pistola para cola a quente ou cola epóxi
- Um computador para programar o Arduino
- Um cabo de conexão USB do tipo A-B

A placa de Arduino Uno pode ser adquirida em diversos fornecedores. Faça uma pesquisa.

Quando comprar o shield de Ethernet, você deverá adquirir um shield “oficial” baseado no chipset Wiznet. NÃO compre uma placa mais barata e de utilização mais difícil baseada no chip controlador de Ethernet ENC28J60.

» Passo 1: construa a plataforma de madeira

O tamanho e a forma das duas colunas de madeira não são importantes. Você só precisa verificar se es-

LISTA DE COMPONENTES

Componente	Quantidade	Descrição	Fornecedor
Arduino	1	Arduino Uno	Veja a descrição no texto
Shield de Ethernet	1		Sparkfun: DEV-09026
Servomotor	2	Servomotor de 9g	eBay
Barra de pino macho	1	Barra de pino macho para CI	Farnell: 1097954
Barras com bornes de conexão	2	Barra de conexão com dois bornes para 2A	Farnell: 1055837
Base de madeira	1	60 cm por 5 cm por 10 cm	Lojas de ferragem
Colunas de madeira	2	Vara redonda de madeira com comprimento de 1 m e diâmetro de 1 cm (ou similar)	Lojas de ferragem
Hastes de madeira	2	Haste redonda de madeira com comprimento de 25 cm e diâmetro de 3 mm	
Papel	1	Papel tamanho A4 para confeccionar as bandeirolas	
Fonte de alimentação	1	Adaptador de voltagem CA/CC de 12V e 1A	Farnell: 1279478

tão suficientemente afastadas para as bandeirolas não colidirem e se são suficientemente altas para evitar que as bandeirolas batam na base quando estiverem baixadas. Você poderá pensar em uma forma diferente de instalar as bandeirolas, talvez um suporte de fixação em uma parede.

A plataforma usa madeira sólida espessa com cerca de 60 cm, na qual são feitos dois orifícios com o mesmo diâmetro das colunas de madeira. As colunas são então inseridas e coladas nesses orifícios da base.

No topo das colunas, faz-se um corte plano com um estilete (Figura 11-3). Nesse corte, o servo pode ser montado e parafusado usando sua aba inferior (Figura 11-4).

Coloque o Arduino no centro da base e posicione os bornes de conexão em cada lado. Marque as posições para os orifícios do Arduino e dos bornes. Faça furos-guias de pequeno diâmetro para os parafusos de fixação. Fixe as barras de bornes no seus lugares.

» Passo 2: faça a fiação dos servos com os bornes de conexão

O diagrama esquemático do projeto está na Figura 11-5, e o diagrama de fiação está mostrado na Figura 11-6.

Os fios dos servos são soldados a barras de pino fêmea (barra de soquetes) para CI. Para cada servo, temos três ligações. Duas são para a alimentação elétrica (GND e 5V) e uma para o sinal de controle. Corte fios com tamanhos que permitam ir dos servos, no topo das colunas, até os bornes de conexão na base de madeira. Usando o diagrama de fiação da Figura 11-6 como guia, conecte as extremidades inferiores desses fios aos bornes de conexão. Nas extremidades superiores, solde barras de pino macho (barra de pinos) para CI. Essas barras de pinos serão encaixadas nas barras de soquetes que foram soldadas antes nos fios dos servomotores.

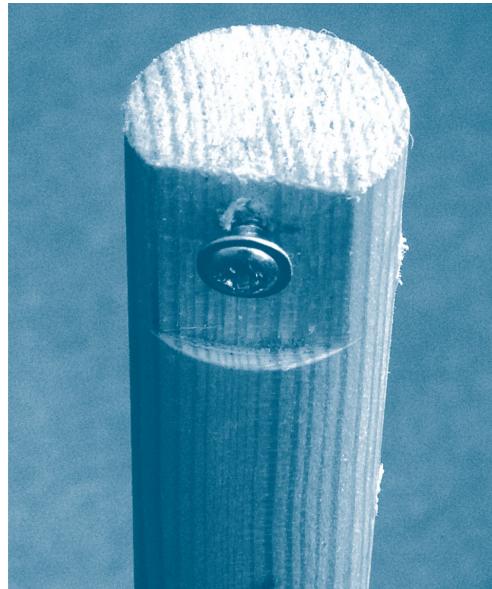


Figura 11-3 Corte plano feito para parafusar o servo.

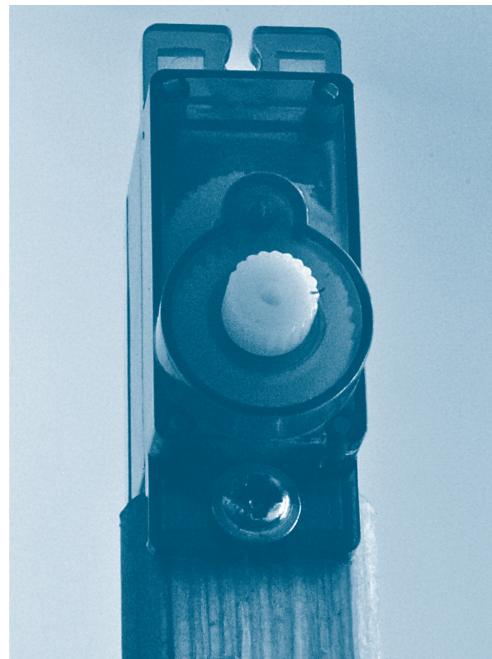


Figura 11-4 O servomotor parafusado na coluna.

» Passo 3: fixe o Arduino na base

Quando a fiação entre os bornes de conexão e os servos estiver toda no lugar, use pequenos parafusos para fixar o Arduino em sua posição e coloque o shield no topo.

A seguir, corte quatro pedaços curtos de fio com uns 5 cm e solde-os em pares a duas barras de pino macho (barras de pinos) de duas vias cada.

Guindo-se pela Figura 11-6, faça a ligação desse fios com os bornes de conexão. A seguir, insira a barra com os pinos de alimentação elétrica nos soquetes de 5V e GND do shield e a barra com os

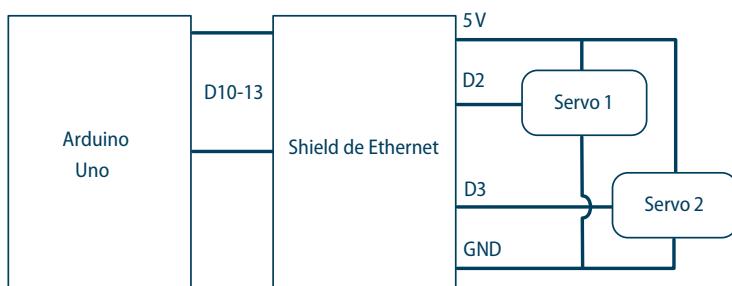


Figura 11-5 O diagrama esquemático do projeto.

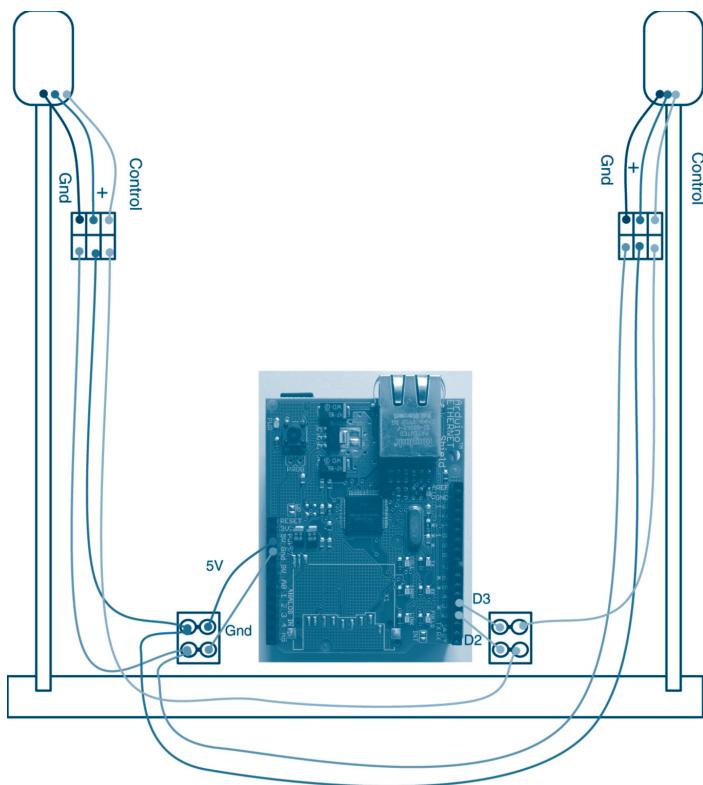


Figura 11-6 O diagrama de fiação do projeto.

pinos de sinal de controle nos soquetes D2 e D3 na borda oposta do shield.

» Passo 4: programe o Arduino

Retire o shield do Arduino antes de programá-lo pela primeira vez. Isso evitará danos devidos aos modos em que ainda possam se encontrar os pinos. Esses modos foram definidos por algum sketch anterior executado na placa do Arduino e que ainda permanece instalado. Programações subsequentes poderão ser realizadas com a placa e o shield no lugar.

Abra o sketch ch11_ethernet_flag. Todos os sketches podem ser baixados na forma de um único arquivo zip do site www.duinodroid.com.

Antes de instalar o sketch, há duas alterações que devemos fazer. Se você olhar no início do sketch, você verá as linhas:

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF,
    0xFE, 0xED };
byte ip[] = { 192, 168, 1, 30 };
```

A primeira delas é o endereço mac, que deve ser único entre todos os dispositivos conectados à sua rede. A segunda linha é o endereço IP. A maioria dos dispositivos conectados à sua rede doméstica tem endereços IP que são atribuídos automaticamente por um processo denominado DHCP. Entretanto, no caso do shield Ethernet, isso não é verdadeiro. O endereço IP desse dispositivo deve ser atribuído manualmente. Esse endereço não é qualquer conjunto de quatro números. Devem ser números habilitados como endereços internos de IP, dentro do intervalo de endereço aceito pelo seu roteador doméstico. Geralmente, os três primeiros números serão algo como 10.0.1.x ou 192.168.1.x, em que x é algum número entre 0 e 255. Alguns desses endereços IP estarão sendo usados por outros dispositivos da sua rede. Para encontrar um

número válido, ainda não usado, abra a página de administração do seu roteador e procure uma opção que diz DHCP. Em algum lugar, você deverá encontrar uma lista de dispositivos com seus endereços IP, semelhante ao mostrado na Figura 11-7. Selecione um número final para usar como seu endereço IP. Nesse caso, 192.168.1.30 pareceu ser uma boa tentativa que, na realidade, veio a funcionar muito bem.

Conecte o Arduino ao seu computador através do cabo USB e transfira o sketch.

» Passo 5: teste

Por fim, estamos prontos para realizar um teste. Coloque o shield no lugar, ligue os fios aos bornes de conexão na base e conecte a fonte de alimentação. Os servos deverão funcionar colocando as bandeirolas em suas posições iniciais.

No seu navegador, abra uma página para o endereço IP que você atribuiu ao shield de Ethernet. A seguir, toque nos quatro botões virtuais da página da Web. Você deverá ouvir os servos funcionando.

» Passo 6: construa e instale as bandeirolas

As bandeirolas são apenas folhas de papel fixadas nas hastes de madeira. Por sua vez, essas hastes são coladas a um dos braços fornecidos juntamente com o servo. Esses braços têm um orifício cilíndrico denteadado que permite a fixação em qualquer posição.

Para colocar os braços na posição correta, toque nos botões “Warm Towels” (toalhas quentes) e “More Wine” (mais vinho) para que as bandeirolas fiquem na posição vertical. A seguir, ajuste as hastes das bandeirolas colocando-as o mais próximo possível da vertical. Teste as bandeirolas mais um pouco e aperte os parafusos de fixação dos braços quando você estiver seguro de que tudo está funcionando corretamente.



Figura 11-7 Procurando um endereço de IP não usado.

» Teoria

Nesta seção, examinaremos rapidamente o funcionamento dos servomotores.

» Servos

Usualmente, os servomotores são usados em modelos de veículos controlados por rádio.

Diferentemente dos motores comuns, um servo não gira continuamente sem parar. Limita-se a fazer giros de cerca de 180 graus. São comandados por pulsos de tensão aplicados à conexão de controle do servo. A duração do pulso controla o ângulo que será assumido pelo servo.

A Figura 11-8 mostra um exemplo de forma de onda. Pode-se ver como a largura do pulso determina o ângulo do servo.

Um pulso de 1,5 milissegundo coloca o servo em sua posição central. Um pulso mais curto de 1,25 milissegundo coloca-o em sua posição mais à esquerda, e um pulso de 1,75 milissegundo, em sua posição mais à direita.

O servomotor espera que haja ao menos um pulso a cada 20 milissegundos para manter estável a sua posição.

Felizmente, há uma biblioteca de Arduino que é utilizada para acionar servos e que permite o comando seguinte:

```
servo1.write(150);
```

Esse comando faz o servo assumir uma posição entre 0 e 180 graus.

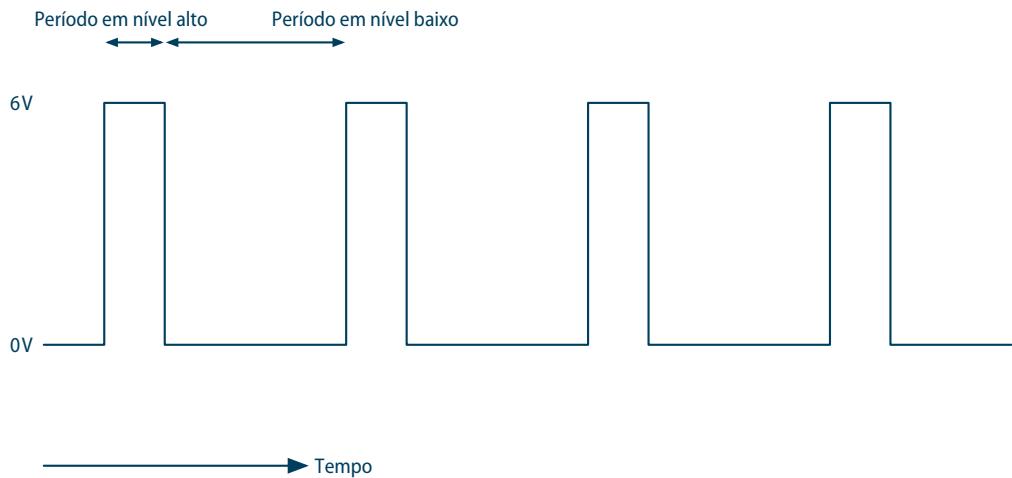


Figura 11-8 O servomotor.

» Resumo

Agora, enquanto descansa em sua banheira, você poderá chamar seus atendentes quando for necessário. No próximo capítulo, voltaremos ao problema de automatizar sua lavanderia.



» capítulo 12

Temporizador

Neste capítulo, mostraremos um dispositivo que atua como temporizador ou marcador de tempo, permitindo ligar sua secadora de roupas durante a noite.

Objetivos

- » Construir um temporizador baseado em Arduino
- » Examinar o sketch do Arduino para esse projeto
- » Entender como um diagrama de transição de estado pode ser de ajuda neste projeto
- » Examinar o sketch do Arduino utilizado neste projeto, com ênfase na implementação do diagrama de transição de estado

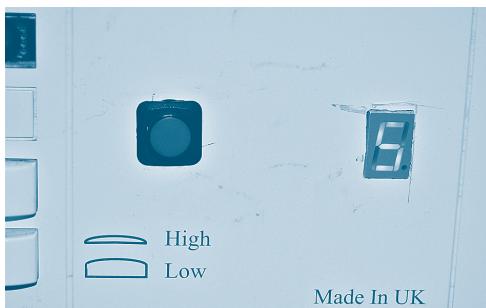


Figura 12-1 O temporizador.

Esse módulo-temporizador (Figura 12-1) pode ser utilizado de diversas formas. No nosso caso, modificamos uma antiga secadora de roupas e incorporamos o temporizador diretamente nela. Entretanto, se você assim o desejar, poderá acondicioná-lo em uma caixa separada.

Após decorrer o número ajustado de horas, o temporizador fechará os contatos do relé durante um décimo de segundo. Por sua vez, os contatos do relé estão ligados aos contatos da chave que comanda a secadora.

ALERTA Se você pretende conectar este projeto a um dispositivo energizado pela rede elétrica de sua residência, tenha em mente que você poderá fazer isso somente se estiver absolutamente seguro do que está fazendo e qualificado para trabalhar com equipamentos de alta tensão.

Você também deve levar em conta que, ao modificar o seu eletrodoméstico, você estará perdendo a respectiva garantia.

» Construção

Este é outro projeto de Arduino construído em uma placa diferente de sua placa original. É um recurso extra que pode ser útil para os projetos de automação residencial que construímos nos capítulos anteriores.

A Figura 12-2 mostra o diagrama esquemático do projeto.

O circuito usa um display de sete segmentos com catodo comum para exibir o número de horas que faltam para ativar o evento programado – neste caso, ligar a secadora de roupas.

Um único botão faz o número de horas avançar. Se o botão não for pressionado novamente dentro de cinco segundos, então a contagem de tempo até a ativação do relé será iniciada.

» O que será necessário

Para construir o projeto, você precisará dos itens listados na tabela *Lista de Componentes*, que está adiante.

Este projeto usa um Arduino Uno para fazer a programação do microcontrolador. O site oficial do Arduino (www.arduino.cc) contém uma lista de fornecedores do Uno. Entretanto, se quiser economizar, você também poderá usar um clone do Arduino Uno.

A fonte de alimentação é o carregador de um telefone celular fora de uso. Precisa ser de 5V. Verifique a tensão com um multímetro e também determine qual fio é o positivo.

O relé utilizado precisa ter uma bobina de 5 ou 6V.

Além desses componentes, você também precisará das ferramentas listadas na Caixa de Ferramentas.

CAIXA DE FERRAMENTAS

- Uma furadeira elétrica com brocas
- Uma serra de arco ou uma ferramenta Dremel
- Um multímetro
- Uma pistola para cola a quente ou cola epóxi
- Um computador para programar o Arduino
- Um cabo de conexão USB do tipo A-B

» Passo 1: prepare a placa perfurada

A placa perfurada é do tipo que apresenta furos distanciados de 1/10 de polegada (2,5 mm). Atrás

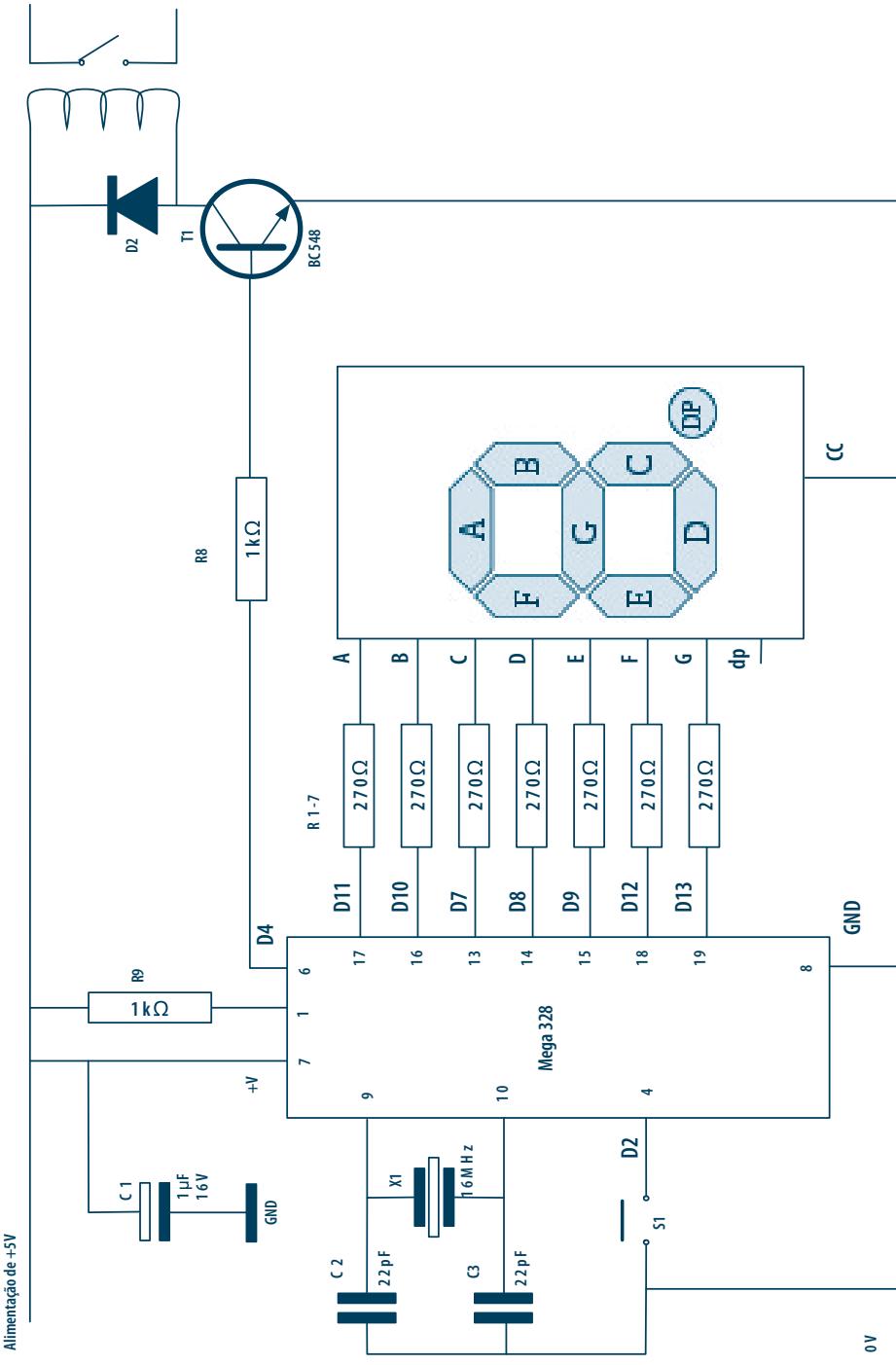


Figura 12-2 O diagrama esquemático.

LISTA DE COMPONENTES

Componente	Quantidade	Descrição	Fornecedor
Microcontrolador	1	ATMega328 com bootloader	Sparkfun: DEV-10524
Relé	1	Relé de 5 ou 6V	Farnell: 1455502
T1	1	BC548	Farnell: 1467872
D1	1	Display de sete segmentos com catodo comum – tipo de meia polegada	Farnell: 1142439
D2	1	1N4001	Farnell: 1458986
C1	1	Capacitor eletrolítico de $1\mu F$	Farnell: 1236655
C2, C3	2	Capacitor cerâmico de $22pF$	Farnell: 1600966
R1-R7	7	Resistor de filme metálico de 270Ω e $1/2 W$	Farnell: 9340300
R8, R9	2	Resistor de filme metálico de $1k\Omega$ e $1/2 W$	Farnell: 9339779
X1	1	Cristal de 16 MHz	Farnell: 1611761
Fonte de alimentação	1	Fonte de alimentação de 5V	<i>Ver descrição no texto</i>
Placa perfurada com trilhas	1	Placa perfurada com 28 trilhas de 18 furos	Farnell: 1201473
Barra de pino fêmea	1	Duas seções de cinco soquetes cortadas de uma barra de pino fêmea (barra de soquetes para CI)	Farnell: 1218869
Soquete para CI	1	Soquete para circuito integrado de 28 pinos	Farnell: 1824463
Interruptor	1	Chave de pressão normalmente aberta	Farnell: 1634627

dos furos, há trilhas de cobre. Os terminais dos componentes entram pelo lado sem cobre e saem pelo outro lado, sendo soldados nas trilhas. A Figura 12-3 mostra a placa perfurada do temporizador vista do lado das trilhas.

Observe que algumas das trilhas de cobre estão cortadas no lugar onde o CI (circuito integrado) deve ser instalado. Esses locais estão marcados com um "X" na placa perfurada da Figura 12-4. Os cortes nas trilhas podem ser feitos usando uma microrretífica com ponta de desbaste. A ponta é

encostada no furo em que será feito o corte e, em seguida, fazendo movimentos circulares com a microrretífica, a trilha de cobre é desbastada sem aumentar muito o diâmetro do furo.

Comece cortando um pedaço de placa perfurada com 28 trilhas de 18 furos cada. Você pode fazer isso com uma tesoura grande, mas nesse caso as bordas poderão ficar irregulares. Um acabamento melhor pode ser conseguido riscando fortemente a placa com um estilete e quebrando-a na beira de uma mesa. Tome cuidado ao quebrar a placa, pois podem surgir arestas afiadas.

Usando as Figuras 12-3 e 12-4 como guia, faça quatro cortes nas trilhas.

» Passo 2: solde os soquetes

O próximo passo é soldar o soquete do CI e as duas seções de pino fêmea (barras de soquetes para CI) de cinco vias. Usando um estilete, as seções poderão ser cortadas de uma barra de pino fêmea comprida. Cada seção terá cinco soquetes (Figura 12-5).

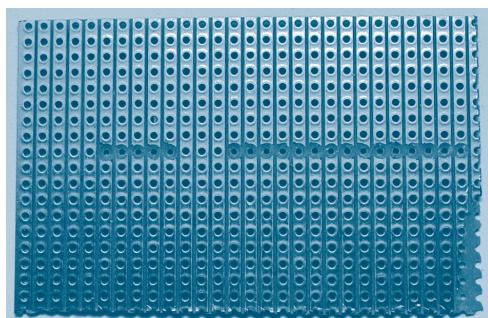


Figura 12-3 A placa perfurada pronta.

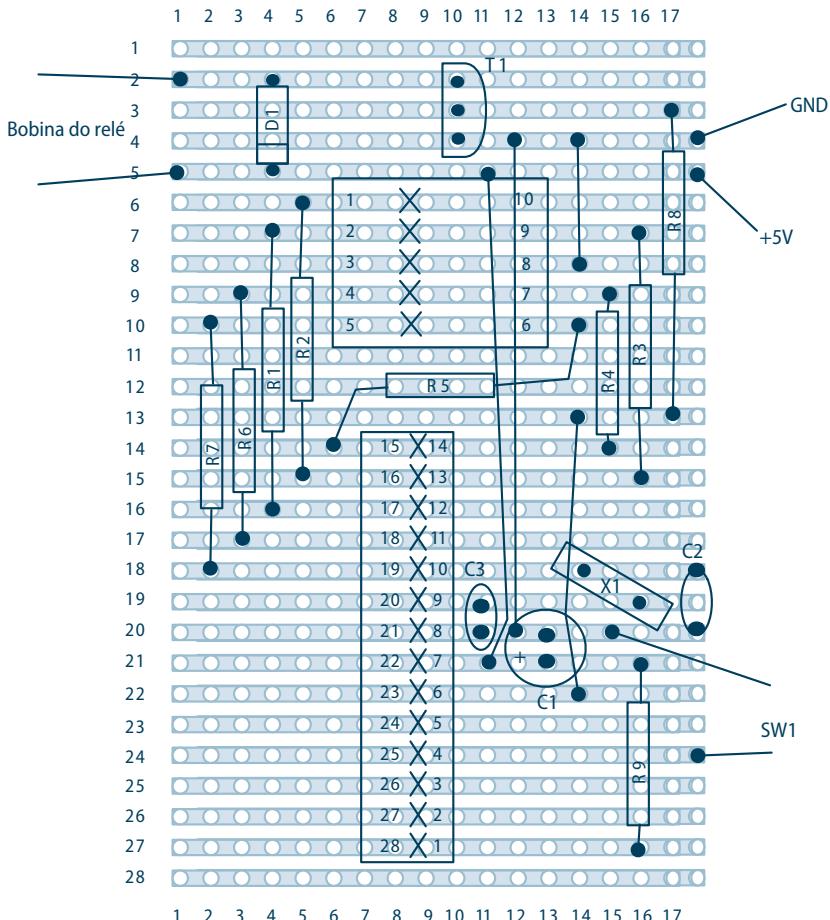


Figura 12-4 A disposição dos componentes na placa perfurada.

O soquete do CI deve ter o lado em que está a reentrância (indicação do pino 1) junto à borda da placa.

(C1) está com a orientação correta (veja a Figura 12-4). Normalmente, o terminal positivo do capa-

» Passo 3: solde as conexões

Há apenas três conexões para serem feitas (Figura 12-6). Como você pode ver, as conexões longas estão encapadas para evitar curtos-circuitos acidentais.

» Passo 4: solde os demais componentes

Agora, podemos soldar os demais componentes (Figura 12-7). Verifique se o capacitor eletrolítico

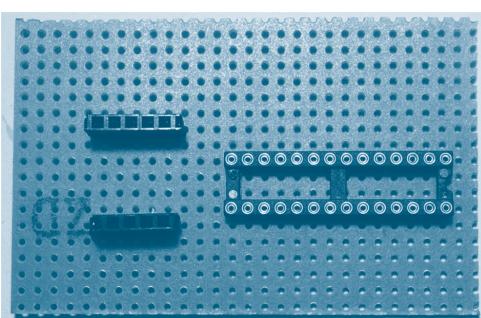


Figura 12-5 Soldando os soquetes no lugar.

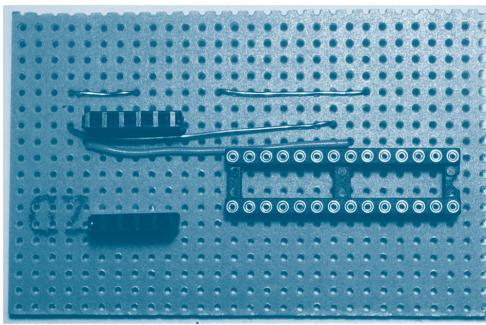


Figura 12-6 Soldando as conexões no lugar.

citor é mais longo, e costuma haver um símbolo na forma de diamante próximo do terminal negativo.

De modo semelhante, confira se o diodo e o transistor estão ambos na orientação correta.

» Passo 5: instale o sketch de Arduino

Se você construiu alguns dos projetos anteriores deste livro, então você já deve ter baixado de

www.duinodroid.com o arquivo zip contendo todos os sketches usados neste livro. Se você ainda não o fez, faça isso agora. Descompacte o arquivo zip e mova toda a pasta Arduino Android para a pasta de sketches. No Windows, a sua pasta de sketches estará em Meus Documentos/Arduino. No Mac, você a encontrará em Documents/Arduino/ e, no Linux, estará no diretório Sketchbook.

Você precisará reiniciar o software Arduino para que ele inclua os novos sketches.

A seguir, no menu File (arquivo) do software Arduino, selecione sketches (ou SketchBook), seguido de Arduino Android, e então o sketch ch12_delay_timer.

Conecte a sua placa de Arduino (sem o shield instalado) ao computador via USB.

Agora, estamos prontos para transferir o sketch para a placa. Para isso, devemos clicar no ícone Upload (segundo a partir da direita na barra de ferramentas). Se aparecer uma mensagem de erro, verifique os tipos de placa e conexão que você está usando.

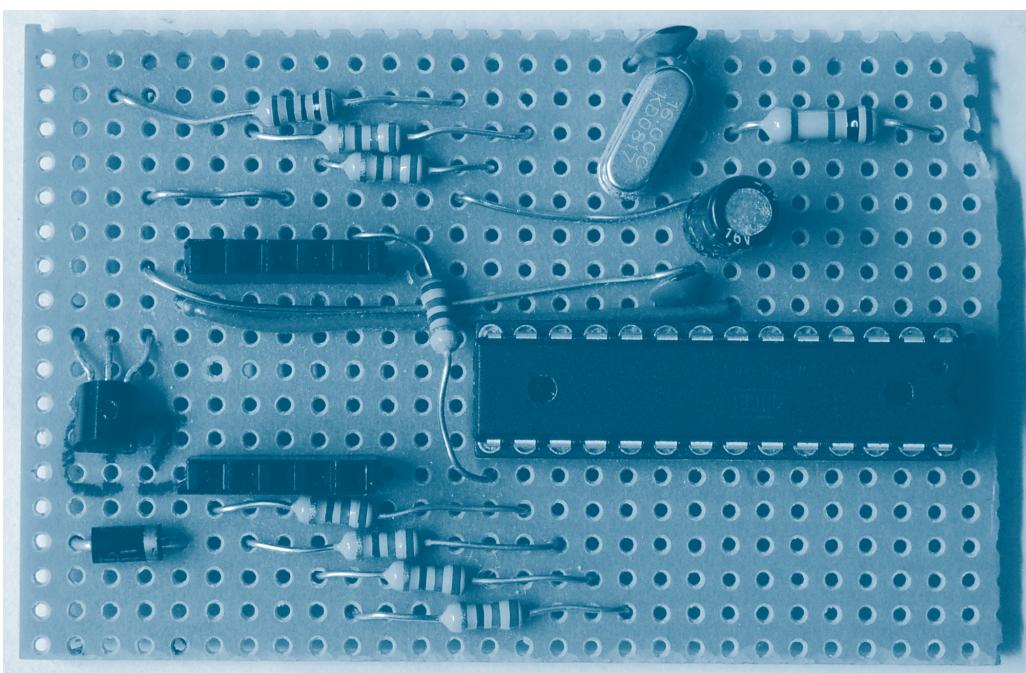


Figura 12-7 Soldando os demais componentes.

Quando o microcontrolador estiver programado, remova-o cuidadosamente e insira-o no soquete do circuito integrado (CI) da placa perfurada. Verifique se ele está com a orientação correta. A reentrância no soquete deve estar alinhada com a reentrância do circuito integrado, a qual indica o lado do chip em que se encontra o pino 1.

» Passo 6: teste

Agora que completamos a construção da placa, poderemos testá-la antes de utilizá-la em alguma aplicação.

Solde fios entre a placa e a bobina do relé e acrescente fios flexíveis aos contatos normalmente abertos do relé. Solde também pedaços curtos de fio flexível (SW1 na Figura 12-4), que serão usados mais adiante para conectar a chave. Finalmente, solde o cabo do carregador na placa, prestando atenção às polaridades dos fios (GND e +5V na Figura 12-4). Veja Figura 12-8.

Ligue a fonte de alimentação. Inicialmente, o display de sete segmentos ficará apagado. Encoste momentaneamente os fios de terminação da chave. Deverá aparecer o número 3. Este é o valor inicial em horas que é mostrado no display do temporizador. Escolhemos 3 para valor inicial por considerá-lo mais apropriado do que o valor 1. Cada vez que os terminais forem encostados, a contagem sobe sucessivamente até alcançar 9, quando deve retornar a 1.

Uma boa maneira de verificar se o relé está funcionando é ajustar o temporizador para uma hora. Imediatamente, o display começará a piscar exibindo "0" e indicando que a contagem de tempo iniciou e que o relé será ativado em 0 hora e 59 minutos.

Coloque o seu multímetro no modo de teste de continuidade (com indicação sonora, se possível) e conecte suas ponteiras aos terminais em aberto do relé. Após transcorrer o tempo ajustado no temporizador, o relé deverá ser ativado por um instante e o sinalizador sonoro do multímetro deverá soar.

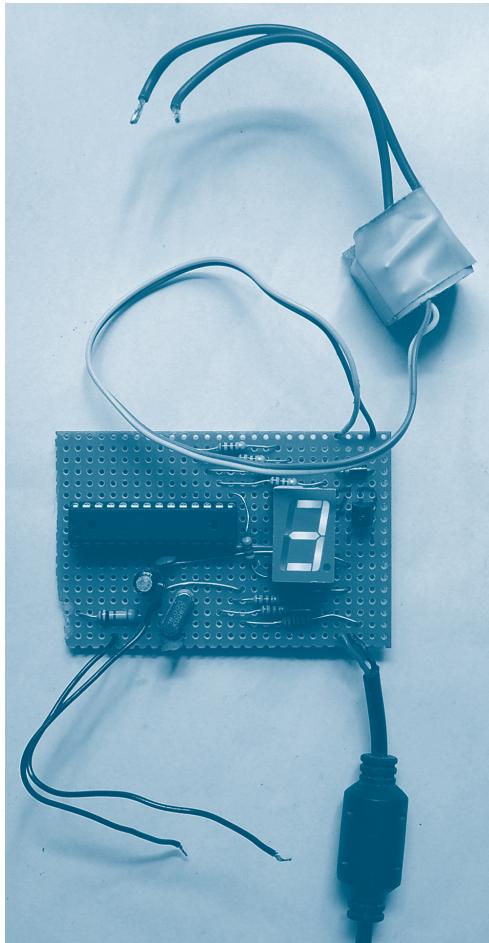


Figura 12-8 Testando o temporizador.

» Passo 7: instalação

Por favor, leia o alerta que está no início deste capítulo. A eletricidade da rede elétrica de sua casa pode matá-lo. Portanto, não instale esse equipamento a menos que você esteja absolutamente seguro do que está fazendo. Nunca trabalhe em um eletrodoméstico quando ele está conectado à rede de energia elétrica. Se for o caso, peça auxílio a alguém qualificado.

Você pode colocar a fonte de alimentação (carregador de celular) em uma tomada separada ou ligá-la convenientemente ao circuito interno do eletrodo-

méstico que será comandado pelo temporizador, como fizemos com a nossa secadora de roupa. Para isso, você começa soldando fios flexíveis aos terminais do carregador, como mostrado na Figura 12-9.

Os fios são enrolados fortemente em torno dos pinos do carregador e soldados. A seguir, devem ficar bem cobertos com fita isolante. Finalmente, as outras extremidades desses fios poderão ser ligadas a um local conveniente no interior do eletrôdoméstico em que a tensão CA necessária esteja disponível.

No nosso caso, fizemos um orifício no painel frontal da nossa secadora para a chave e abrimos um retângulo para o display. A seguir, a placa perfurada foi fixada no lugar, atrás do painel do secador (Figura 12-1).

Verifique se todos os cabos estão presos firmemente utilizando braçadeiras plásticas e se estão bem distantes de qualquer coisa que se move ou esquente.

» Teoria

Nesta seção, examinaremos o software. Veremos como este sistema pode ser projetado por meio de uma técnica útil denominada *diagrama de transição de estado*. Finalmente, analisaremos o sketch com algum detalhe.

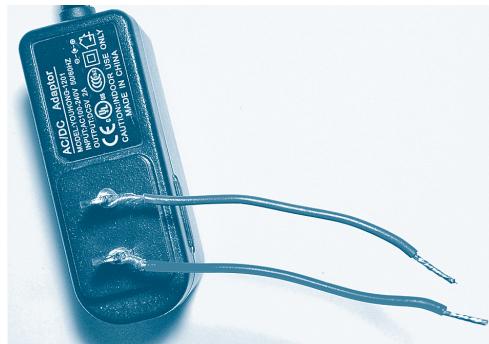


Figura 12-9 Soldando fios flexíveis à fonte de alimentação (carregador de celular fora de uso).

» Diagramas de transição de estado

Embora esse temporizador seja fácil de operar, é surpreendentemente difícil criar um sketch que faz coisas como iniciar a contagem de tempo três segundos depois que o botão foi pressionado pela última vez. Uma técnica útil de projeto, que sempre poupa uma grande dose de aflição em muitas situações, é denominada *diagrama de transição de estado*. Preferimos não criar uma sigla para essa técnica.

A Figura 12-10 mostra o diagrama de transição de estado do temporizador.

Os círculos no diagrama representam estados. Vemos que o temporizador está sempre em um de três estados: aguardando, ajuste de tempo e contagem de tempo. O que acontece quando você pressiona um botão depende do estado em que você se encontra.

Passamos de um estado para outro quando ocorre algum evento. Nesse caso, começamos no estado “Aguardando”, mas, se ocorrer a condição de “Botão pressionado” (conforme especificado na parte acima da linha no rótulo da transição), então executaremos as ações indicadas abaixo da linha e passaremos para o estado indicado pela seta. No caso, a ação consiste em atribuir o valor 3 à variável “horas” (ou “hours”, no sketch) e fazer o display piscar enquanto exibe um 3. Quando estamos no estado “Ajuste de tempo” e pressionamos novamente o botão, a variável “horas” (hours) é incrementada sem ocorrer uma transição para outro estado.

Passaremos para o estado de “Contagem de tempo” somente quando ocorrer a condição de “Botão não pressionado por 3 segundos”. Após passarmos para o estado de “Contagem de tempo”, permaneceremos nele até que a contagem de tempo termine (quando então retornaremos ao estado de “Aguardando”) ou até que o botão seja pressionado (quando então retornaremos ao estado de “Ajuste de tempo”).

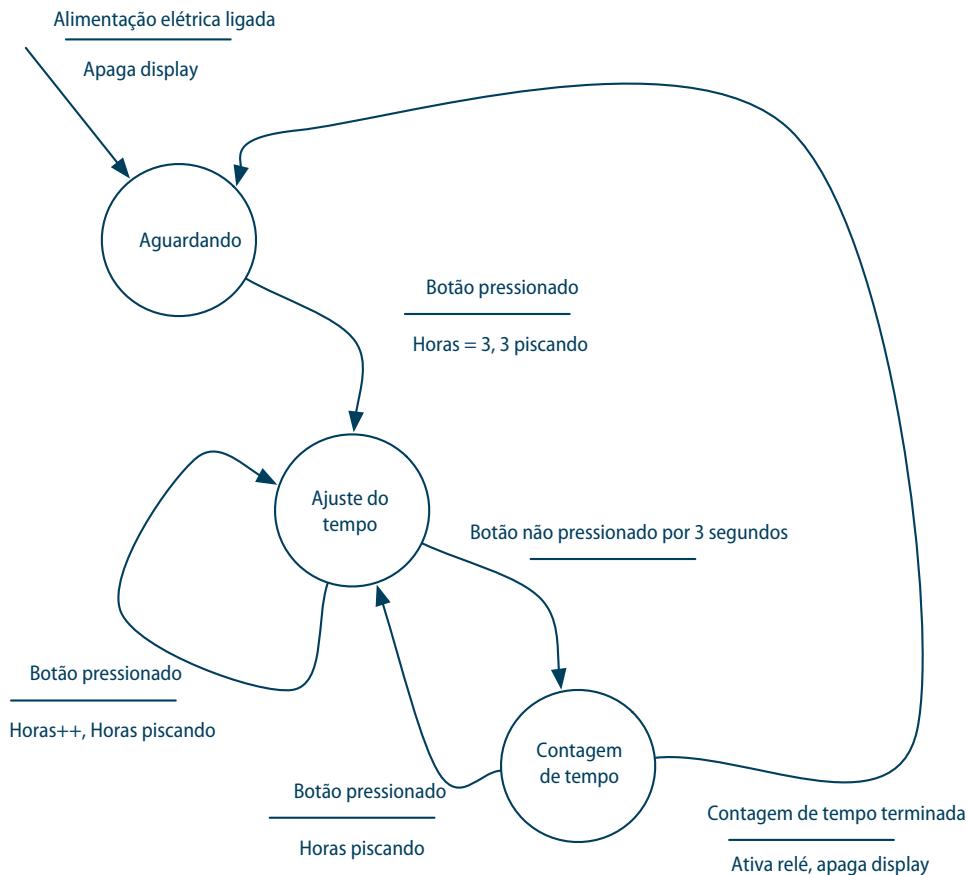


Figura 12-10 O diagrama de transição de estado.

» O sketch do Arduino

À medida que você ler o sketch seguinte, você verá como ele reflete de perto o diagrama de transição de estado que acabamos de discutir.

```
int segmentPins[] = {11, 10, 7, 8, 9,
                     12, 13};
int switchPin = 2;
int relayPin = 4;

byte digits[11][7] = {
// a b c d e f g
{ 1, 1, 1, 1, 0, 0, 1}, // 0
{ 0, 1, 1, 0, 0, 1, 1}, // 1
{ 1, 0, 1, 1, 0, 1, 1}, // 2
{ 1, 0, 1, 1, 1, 1, 1}, // 3
{ 1, 1, 1, 0, 0, 0, 0}, // 4
{ 1, 1, 1, 1, 1, 1, 1}, // 5
{ 1, 0, 1, 1, 1, 1, 1}, // 6
{ 1, 1, 1, 0, 0, 0, 0}, // 7
{ 1, 1, 1, 1, 1, 1, 1}, // 8
{ 1, 1, 1, 1, 0, 1, 1}, // 9
{ 0, 0, 0, 0, 0, 0, 0} // Display apagado
};
```

```
#define BLANK 10

#define STANDBY 0
#define SETTING_DELAY 1
#define WAITING 2
int state;

(continua)
```

```

long hours;
void setup()
{
    for (int i = 0; i < 7; i++)
    {
        pinMode(segmentPins[i], OUTPUT);
    }
    pinMode(switchPin, INPUT);
    digitalWrite(switchPin, HIGH);
    // Ativa resistor de pull-up
    pinMode(relayPin, OUTPUT);
    state = STANDBY;
    show(BLANK);
}

void loop()
{
    static long lastButtonPress = 0;
    static long startTime = 0;
    static long endTime = 0;
    boolean buttonPressed = !digitalRead(switchPin);
    if (state == STANDBY)
    {
        if (buttonPressed)
        {
            lastButtonPress = millis();
            state = SETTING_DELAY;
            hours = 3;
            flash(hours);
        }
    }
    else if (state == SETTING_DELAY)
    {
        if (buttonPressed)
        {
            lastButtonPress = millis();
            hours++;
            if (hours == 10)
            {
                hours = 1;
            }
            flash(hours);
        }
        else if ((millis() - lastButtonPress) > 3000)
        {
            startTime = millis();
            endTime = startTime + (hours * 60 * 60 * 1000);
            state = WAITING;
        }
    }
}

void show(int n)
{
    for (int i=0; i < 7; i++)
    {
        digitalWrite(segmentPins[i],
        digits[n][i]);
    }
}

void flash(int n)
{
    show(BLANK);
    show(n);
    delay(200);
    show(BLANK);
    delay(200);
    show(n);
}

void toggleRelay()
{
    digitalWrite(relayPin, HIGH);
    delay(100);
    digitalWrite(relayPin, LOW);
}

```

Começamos definindo um array de “ints” para conter os pinos usados pelos segmentos do display. A seguir, definimos os pinos que serão usados com a chave e o relé.

Para exibir as diferentes configurações de segmentos correspondentes aos diversos dígitos, utilizaremos um array bidimensional. Acrescentamos uma fila extra para representar o caso em que todos os LEDs estão apagados. Veremos depois como isso simplifica o sketch. A constante BLANK (em branco, vazio) faz referência a essa fila.

As três constantes seguintes STANDBY (aguardando), SETTING_DELAY (ajuste de tempo) e WAITING (contagem de tempo) representam os três estados que descrevemos no nosso diagrama de transição de estado. Podem ser números quaisquer, desde que sejam diferentes entre si.

A seguir, declaramos as duas variáveis “hours” (horas) e “state” (estado). A variável “state” especifica o estado atual em que se encontra o temporizador, e a variável “hours” é o número de horas ajustado.

A função “setup” usa um laço para definir que todos os pinos de segmento serão de saída (OUTPUT) e define os modos dos demais pinos. Quando aplicamos um comando “digitalWrite HIGH” ao pino da chave (switchPin), o resistor interno de pull-up desse pino é ativado.

Agora chegamos à função “loop”. É aqui que modelamos o diagrama de transição de estado. Primeiro, definimos algumas variáveis estáticas que não serão inicializadas a cada vez que a função é executada. Essas variáveis são utilizadas para registrar os tempos em que ocorrem os eventos,

como a última vez em que o botão foi pressionado.

Em seguida, temos seções para cada um dos três estados. Assim, se estivermos no estado de STANDBY (aguardando), a primeira coisa que faremos é registrar o tempo em que o botão foi pressionado e, em seguida, atribuir o valor SETTING_DELAY (ajuste de tempo) à variável “state”, atribuir 3 à variável “hours” e fazer o display piscar.

Você consegue analisar os códigos similares dos outros dois estados relacionando-os com o diagrama de transição de estado.

Há também três outras funções utilitárias. A função “show” (exibir) acende os segmentos do display correspondentes ao dígito fornecido como argumento. A função “flash” (piscar) faz o mesmo, exibindo o dígito por apenas 200 milissegundos, e a função “toggleRelay” faz pulsar o relé que, de desligado, fica 100 milissegundos ligado e retorna em seguida à condição de desligado.

» Resumo

Este é um pequeno projeto que pode ser bem utilizado em diversas situações. É também o último capítulo baseado em projeto deste livro.

Espero que você tenha desfrutado a construção desses projetos e que tenha se inspirado para combinar dispositivos Android e Arduino de maneira criativa.

O apêndice a seguir oferece um ponto de partida para o aprendizado do desenvolvimento de projetos baseado no Open Accessory para Android.



» apêndice

Fundamentos de Open Accessory

Tornar-se um conhecedor da programação Android constitui um livro por si só. Sobre o protocolo de programação Open Accessory, há à disposição apenas a documentação concisa do Google, além de pouca coisa mais.

Neste apêndice, assumiremos que você já tem alguma familiaridade com a programação Arduino e Android, mas que deseja entender como trabalham os aplicativos Open Accessory utilizados em muitos projetos deste livro.

Objetivos

- » Entender os fundamentos de funcionamento dos aplicativos Open Accessory utilizados neste livro
- » Examinar um exemplo de utilização de Open Accessory do lado Arduino e do Android, detalhando as diversas seções Android necessárias

» Aprendendo a programação Android

Há muitos livros bons para aprender a programar com Android. Compre um deles e estude pelo menos os primeiros capítulos para compreender conceitos-chaves, tais como *Activities* e *Intents*, assim como para aprender a criar um ambiente de desenvolvimento e usar a linguagem Java de programação.

Para começar, na Internet há muitos recursos e tutoriais disponíveis. Entretanto, a vantagem de um livro bem estruturado e específico consiste em uma abordagem única e coerente, ao passo que os recursos obtidos na Internet podem ser confusos para o iniciante, já que cada autor faz as coisas de um modo diferente.

O desenvolvimento de aplicativos com Android é bem mais difícil do que a programação para Arduino – ou melhor, há muito mais para aprender.

Aqui utilizamos o plug-in Eclipse para Android. Esse plug-in é popular e parece funcionar muito bem. O Eclipse é um IDE* como o IDE do Arduino, mas consideravelmente mais complexo e potente.

» Programando o Arduino

De um modo geral, o aprendizado de Arduino é uma tarefa muito mais fácil. Novamente, há muitos bons livros à disposição e outros surgindo a todo momento. Consulte as listas e comentários de livros disponíveis em sua livraria preferida. Se você apreciou o estilo deste livro, você gostará de outros dois livros de nossa autoria: *Programação com Arduino: começando com sketches* (publicado pela Bookman Editora) e *30 Projetos com Arduino***.

» O exemplo

A documentação do Google vem acompanhada de um exemplo bem abrangente que mostra como usar a interface Open Accessory com uma placa, denominada ADK***, baseada em Arduino. Essa placa foi desenvolvida especialmente para demonstrar as características do Open Accessory. Baseia-se no Arduino Mega e contém recursos de USB host. Essas placas não são placas Arduino oficiais e são mais caras e menos comuns do que as placas padronizadas de Arduino, como a Uno.

O exemplo fornecido pelo Google (usando a placa ADK) é abrangente demais. São literalmente milhares de linhas de código divididas entre 18 classes. Desse modo, fica muito difícil entender como funcionam as partes e o todo.

No exemplo dado neste apêndice, reduzimos tudo ao essencial. Mesmo assim, ainda temos 300 linhas de código divididas em apenas duas classes.

O propósito do exemplo (mostrado na Figura A-1) é permitir entrar com um número que será enviado ao Arduino. Por sua vez, o Arduino incrementa o número e o envia de volta. É um exemplo muito simples, mas demonstra bem a comunicação em ambos os sentidos.

O aplicativo também tem a vantagem de nos informar tudo a respeito do que acontece. Para isso, em uma área da tela ("Log"), é exibido um registro de todas as operações realizadas. Isso é útil porque, diferentemente de um aplicativo comum, que pode ser conectado ao debugger do Eclipse para mostrar o que está acontecendo, o aplicativo Open Accessory será conectado a um Arduino e não estará disponível para debugging através da conexão USB.

* N. de T.: Integrated Development Environment, ou Ambiente Integrado de Desenvolvimento.

** Nota: Livro em produção pela Bookman Editora. Logo estará à disposição do leitor.

*** N. de T.: Accessory Development Kit, ou Kit de Desenvolvimento para Accessory.



Figura A-1 O aplicativo que será usado como exemplo.

» A parte do Arduino

Neste exemplo, a parte relativa ao Arduino é muito simples. O seu tamanho é aproximadamente um décimo da parte relativa ao Android.

» Instalando as bibliotecas

O Google Open Accessory precisa que duas bibliotecas sejam instaladas no seu ambiente Arduino. A primeira é uma versão da biblioteca USB host, que foi preparada para trabalhar com o hardware padrão de um Arduino. Ela pode ser baixada de microbridge.googlecode.com/files/usb_host_patched.zip.

Se, por alguma razão, você não conseguir algum software utilizado nos projetos deste livro, acesse o site do livro (www.duinodroid.com), no qual haverá instruções para obter o software em outro lugar.

Para instalar a biblioteca, baixe o arquivo zip, descompacte-o e mova toda a pasta descompactada para a sua pasta de bibliotecas (libraries) do Arduino. No Windows, a pasta de bibliotecas estará em Meus Documentos/Arduino. No Mac, você a encontrará no diretório-raiz, em Documents/Arduino/ e, no Linux, estará no diretório sketch-

book do diretório-raiz. Se a pasta “libraries” não existir no seu ambiente Arduino, você deverá criá-la. Depois de instalar o software, inicie novamente o software do Arduino.

A segunda biblioteca, a própria biblioteca AndroidAccessory, é baixada como parte do pacote ADK, e pode ser encontrada em: <http://developer.android.com/guide/topics/usb/adk.html>.

Clique em “Adk package download”. Com isso, o arquivo zip será baixado. Descompacte-o e você verá que, dentro de uma pasta de nome ADK_release_0512, há dois arquivos e três pastas. A única pasta que nos interessa é denominada firmware. Por sua vez, essa pasta contém uma pasta de nome arduino_libs, e dentro dela há duas pastas, cada uma contendo uma biblioteca Arduino. Uma é a USB_Host_Shield, que não precisa ser instalada porque acabamos de instalar uma versão dela, e a outra é a biblioteca AndroidAccessory, que precisa ser instalada.

Para fazer isso, mova a pasta inteira AndroidAccessory para a sua pasta “libraries”, do mesmo modo que fez com a biblioteca USB host. Você precisará iniciar novamente o software do Arduino para ativar as novas bibliotecas.

» O sketch

O sketch completo está listado aqui. Ele também está no arquivo zip com os sketches de projeto que podem ser baixados do site do livro (www.duinodroid.com). O sketch é denominado apA_open_accessory_test.

```
#include <Max3421e.h>
#include <Usb.h>
#include <AndroidAccessory.h>

AndroidAccessory acc("Simon Monk",
    "OpenAccessoryTest",
    "DemoKit Arduino Board",
    "1.0",
    "http://www.duinodroid.com",
    "0000000012345678");
```

```

void setup()
{
    acc.powerOn();
}

void loop()
{
    byte msg[1];
    if (acc.isConnected())
    {
        int len = acc.read(msg,
                           sizeof(msg), 1);
        if (len >= 1)
        {
            byte value = msg[0];
            sendMessage(value + 1);
        }
    }
}

void sendMessage(int value)
{
    if (acc.isConnected())
    {
        byte msg[2];
        msg[0] = value >> 8;
        msg[1] = value & 0xff;
        acc.write(msg, 2);
    }
}

```

Após os primeiros comandos `#include`, é criada uma nova instância da classe `Accessory`. Os argumentos para o “constructor” informam ao seu dispositivo Android tudo a respeito do acessório que acabou de ser conectado.

Os três primeiros parâmetros são “vendor” (vendedor ou fornecedor), “application name” (nome da aplicação) e “version” (versão) – neste caso, temos “Simon Monk”, “OpenAccessoryTest” (teste de Open Accessory), e “1.0”. Quando o acessório for conectado, esses parâmetros serão usados pelo lado Android da interface para que o aplicativo apropriado no telefone Android seja executado automaticamente.

O próximo argumento é um endereço URL. Se o aplicativo apropriado não estiver instalado, o telefone Android usará um navegador para fazer

uma conexão direta com esse endereço URL, em que encontrará o aplicativo necessário. Agora, esse aplicativo pode ser baixado, instalado e executado.

O argumento final é um número de série para o aplicativo.

A função “setup” precisa apenas chamar o método “powerOn” para dar início à comunicação USB.

Na função “loop”, usamos “isConnected” (está conectado) para determinar se há alguma conexão com o dispositivo Android. Observe que, se não houver conexão, será feita uma tentativa de estabelecer a conexão. Desse modo, se os dispositivos se desconectarem, não haverá necessidade de novamente executar “powerOn” ou inicializar o Arduino.

O método “read” determina se há uma mensagem esperando para ser lida. Se houver, então será retornado o tamanho da mensagem em bytes. Assumiremos que será utilizado um único byte para conter o número. Depois de lido, o byte é incrementado e então entregue para a função “sendMessage” que o enviará para o dispositivo Android.

A função “sendMessage” (envia mensagem) primeiro verifica se há uma conexão. Em caso afirmativo ou se uma puder ser estabelecida quando “isConnected” é chamada, então uma mensagem é construída em um array de dois bytes. Isso enviará um valor “int” de volta ao dispositivo Android. Se você quiser enviar outros dados, eles devem ser organizados dessa forma em um array de bytes.

Finalmente, o método “write” recebe o array de bytes e o comprimento da mensagem que será enviada ao dispositivo Android.

» A parte do Android

O código dessa parte é grande demais para ser listado aqui por inteiro. Iremos mostrá-lo em partes, mas essa explicação ficará mais fácil se você baixar o código-fonte de www.duinodroid.com e colocá-lo em um editor de texto ou em um IDE enquanto o analisamos.

» Execução automática e Download

Um dos melhores atributos do padrão Open Accessory é que o seu Arduino pode dizer efetivamente ao seu telefone Android qual é o aplicativo necessário e iniciar automaticamente a execução desse aplicativo quando o dispositivo é conectado. Pode inclusive orientar o telefone conectando-o a um endereço URL, do qual ele poderá baixar o aplicativo.

A parte correspondente no Arduino é definida no “constructor” do `AndroidAccessory`.

```
AndroidAccessory acc("Simon Monk",
    "OpenAccessoryTest",
    "DemoKit Arduino Board",
    "1.0",
    "http://www.duinodroid.com",
    "0000000012345678");
```

Na solução Android, as partes correspondentes estão distribuídas em dois arquivos.

Primeiro, você deve incluir as linhas seguintes no “manifest” Android do projeto. Dentro do tag “activity” acrescente o tag:

```
<intent-filter>
    <action android:name="android
        .hardware.usb.action.USB_ACCESSORY
        _ATTACHED"/>
</intent-filter>
```

Dentro do tag “application,” acrescente os tags seguintes:

```
<uses-library android:name="com.android
    .future.usb.accessory"/>
<meta-data android:name="android.
    hardware
        .usb.action.USB_ACCESSORY_ATTACHED"
    android:resource="@xml/accessory
        _filter"/>
```

O segundo tag define o arquivo `xml/accessory_filter`. É o arquivo usado para decidir se este aplicativo deve ser executado.

```
<?xml version="1.0" encoding="utf-8"?>

<resources>
    <usb-accessory manufacturer=
        "Simon Monk"
        model="OpenAccessoryTest"
        version="1.0" />
</resources>
```

» Ciclo de vida

O aplicativo Android deve lidar com diversas situações e eventos relativos ao acessório e o ciclo de vida genérico da atividade (Activity) principal do aplicativo.

A Figura A-2 mostra o ciclo de vida de uma Activity Android, juntamente com os métodos que são chamados a cada vez que a Activity passa de um estado para outro.

» Abrindo o acessório

Inicialmente, quando uma Activity é criada, três funções são chamadas: `onCreate`, `onStart`, e `onResume`. No nosso exemplo de Activity, implementamos `onCreate` e `onResume`, como está mostrado adiante.

O método `onCreate` é muito semelhante ao `onCreate` de qualquer Activity. Ele define variáveis-membros para os diversos controles da Activity e cria um listener que é disparado sempre que o botão Send (enviar) é pressionado. Ele também chama `setupAccessory`.

O método `setupAccessory` cria uma instância de `UsbManager` e a associa à Activity. A seguir, registra a ação `ACTION_USB_ACCESSORY_DETACHED` com o “broadcast receiver” de nome `mUsbReceiver`. Assim, poderá invocar o método `closeAccessory` do “broadcast receiver” quando o acessório for desconectado.

A função `setupAccessory` sempre será chamada depois que `onResume` é chamada. Assim, será possível

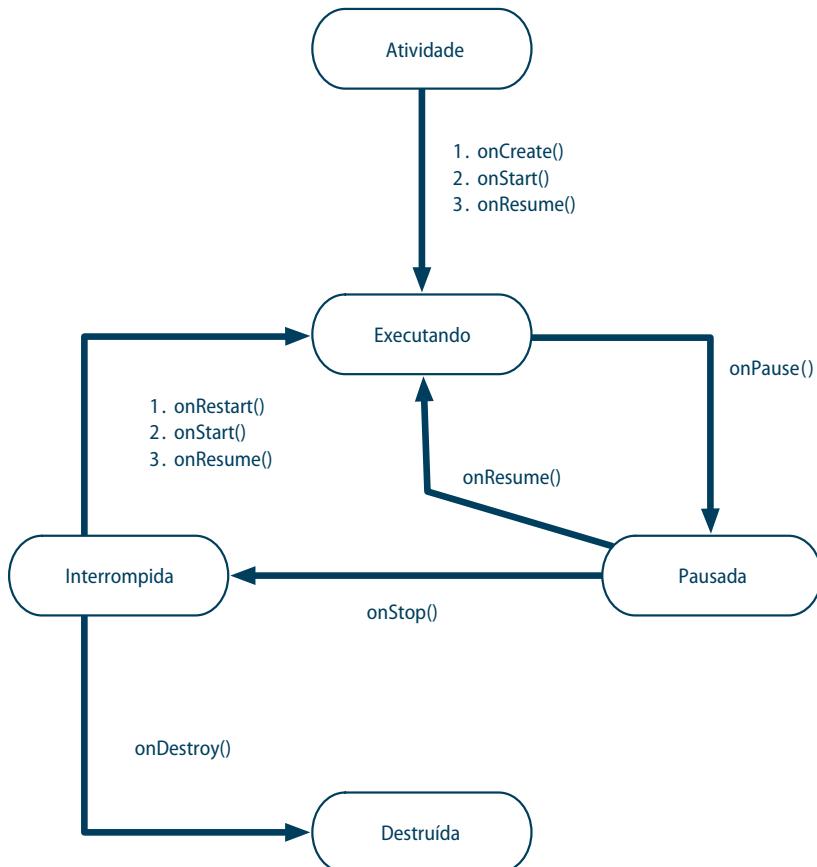


Figura A-2 O ciclo de vida de uma Activity Android (Atividade Android).

```

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mByteField = (EditText) findViewById(R.id.messagebyte);
    mResponseField = (EditText) findViewById(R.id.arduinoresponse);
    mSendButton = (Button) findViewById(R.id.sendButton);
    mSendButton.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            sendMessageToArduino();
        }
    });
    setupAccessory();
}

```

recuperar um “handle” para mAccessory usando o método getLastNonConfigurationInstance, que então abrirá o acessório. Se mAccessory ainda não tiver

sido instanciada, então a função será criada quando ela for chamada posteriormente por onResume.

```

private void setupAccessory() {
    log("In setupAccessory");
    mUsbManager = UsbManager.getInstance(this);
    mPermissionIntent = PendingIntent.getBroadcast(this, 0, new Intent(
        ACTION_USB), 0);
    IntentFilter filter = new IntentFilter(ACTION_USB);
    filter.addAction(UsbManager.ACTION_USB_ACCESSORY_DETACHED);
    registerReceiver(mUsbReceiver, filter);
    if (getLastNonConfigurationInstance() != null) {
        mAccessory = (UsbAccessory) getLastNonConfigurationInstance();
        openAccessory(mAccessory);
    }
}

```

O método `openAccessory` cria streams de entrada e saída para o acessório, além de iniciar uma thread

separada que será usada para ouvir (listen) as mensagens que estão entrando.

```

private void openAccessory(UsbAccessory accessory) {
    log("In openAccessory");
    mFileDescriptor = mUsbManager.openAccessory(accessory);
    if (mFileDescriptor != null) {
        mAccessory = accessory;
        FileDescriptor fd = mFileDescriptor.getFileDescriptor();
        mInputStream = new FileInputStream(fd);
        mOutputStream = new FileOutputStream(fd);
        Thread thread = new Thread(null, this, "OpenAccessoryTest");
        thread.start();
        alert("openAccessory: Accessory opened");
        log("Attached");
    } else {
        log("openAccessory: accessory open failed");
    }
}

```

Como mencionamos antes, quando uma Activity começa a ser executada, ela chama `onCreate` e `onResume`.

O método `onResume` verifica se os streams são nulos. Se forem, ele chama o método de `establishPermissionsAndOpenAccessory`. Como o nome sugere, essa função assegura que a Activity receba as permissões necessárias e, em seguida, abre os streams para o acessório chamando novamente o método `openAccessory`, mostrado a seguir.

```

public void onResume() {
    log("Resuming");
    super.onResume();

    if (mInputStream != null &&
        mOutputStream != null) {
        log("Resuming: streams were not
            null");
    } else {
        log("Resuming: streams were
            null");
        establishPermissionsAndOpen
        Accessory();
    }
}

```

```

private void establishPermissionsAndOpenAccessory() {
    UsbAccessory[] accessories = mUsbManager.getAccessoryList();
    UsbAccessory accessory = (accessories == null? null: accessories[0]);
    if (accessory!= null) {
        if (mUsbManager.hasPermission(accessory)) {
            openAccessory(accessory);
        } else {
            synchronized (mUsbReceiver) {
                if (!mPermissionRequestPending) {
                    mUsbManager.requestPermission(accessory, mPermissionIntent);
                    mPermissionRequestPending = true;
                }
            }
        } else {
            log("establishPermissionsAndOpenAccessory:mAccessory is null");
        }
    }
}

```

» Broadcast Receiver

Um “broadcast receiver” permite que um aplicativo intercepte notificações feitas pelo sistema.

Neste caso, queremos saber quando alguém desconectou o acessório. Uma instância de “Broadcast Receiver” é criada e atribuída à variável-membro mUsbReceiver. Veja a listagem seguinte do código.

```

private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (UsbManager.ACTION_USB_ACCESSORY_DETACHED.equals(action)) {
            UsbAccessory accessory = UsbManager.getAccessory(intent);
            if (accessory!= null && accessory.equals(mAccessory)) {
                log("Detached");
                closeAccessory();
            }
        }
    }
};

```

Esse “receiver” é registrado como tal no método “setupAccessory”, que é chamado em onCreate.

No caso do acessório ser desconectado do telefone, o método closeAccessory será chamado.

» Enviando dados

Quando se trata de enviar dados do Android para o Arduino, estamos de volta ao território familiar de escrever em um “stream”.

```

private void closeAccessory() {
    log("In closeAccessory");
    try {
        if (mFileDescriptor != null) {
            mFileDescriptor.close();
        }
    } catch (IOException e) {
    } finally {
        mFileDescriptor = null;
        mAccessory = null;
        mInputStream = null;
        mOutputStream = null;
    }
}

```

Neste exemplo, estamos simplesmente enviando um único byte ao Arduino. O método sendMessageToArduino é invocado quando pressionamos o botão “Send to Arduino” (enviar para o arduino). Ele toma o texto do campo de entrada e o valida como sendo um número. Em seguida, passa o valor do byte para sendCommand.

```

public void sendMessageToArduino() {
    String valueStr =
        mByteField.getText().toString();
    byte val;
    try {
        val = Byte.parseByte(valueStr);
        log("Sending to Arduino: " + val);
        sendCommand(val);
    } catch (NumberFormatException e) {
        // TODO Auto-generated catch
        // block
        e.printStackTrace();
        alert("The Byte should be a
              number between 0 and 255");
    }
}

```

O comando `send` (enviar) constrói um array de bytes (de apenas um único byte) e então o coloca (`write`) no stream de saída. Nesse caso, o array de bytes não é necessário, mas, se você for enviar mais dados, você deverá construir um array de bytes com o tamanho necessário, empacotá-lo com os dados e, então, colocá-lo (`write`) no stream.

```

public void sendCommand(byte value) {
    byte[] buffer = new byte[1];
    buffer[0] = (byte) value;
    if (mOutputStream != null) {
        try {
            mOutputStream.write(buffer);
        } catch (IOException e) {
            log("Send failed: " +
                e.getMessage());
        }
    } else {
        log("Send failed: mOutStream was
            null");
    }
}

```

» Recebendo dados

Não podemos permitir que a thread principal da interface de usuário da Activity pare e fique esperando uma resposta do Arduino. Na realidade, é possível que o Arduino comece a comunicação. Por essa razão, o método openAccessory inicia uma thread separada que fica ouvindo as mensagens vindas do Arduino.

```

public void run() {
    int ret = 0;
    byte[] buffer = new byte[16384];
    int i;

    while (true) {
        try {
            ret = mInputStream.
                read(buffer);
        } catch (IOException e) {
            break;
        }

        i = 0;
        while (i < ret) {
            int len = ret - i;
            if (len >= 2) {
                Message m = Message.obtain
                    (mHandler);
                int value = composeInt
                    (buffer[i], buffer[i + 1]);
                m.obj = new ValueMsg('a',
                    value);
            }
        }
    }
}

```

```

        mHandler.sendMessage(m);
    }
    i += 2;
}
}

```

A thread fica indefinidamente se repetindo dentro do laço (loop), esperando ler alguma coisa do stream de entrada. Quando encontra uma mensagem (caso em que o valor de leitura é maior do que zero), ela constrói um inteiro (int) utilizando o método utilitário composeInt e os dois próximos bytes da mensagem.

```

private int composeInt(byte hi, byte lo) {
    int val = (int) hi & 0xff;
    val *= 256;
    val += (int) lo & 0xff;
    return val;
}

```

Agora, após receber uma mensagem do Arduino, precisamos exibi-la. No entanto, para isso, precisamos interagir com a Activity, o que requer o uso de um Handler.

```

Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        ValueMsg t = (ValueMsg) msg.obj;
        log("Arduino sent: " +
            t.getFlag() +
            " " + t.getReading());
    }
};

```

A seguir, uma mensagem é enviada ao handler para que ele possa incluí-la no campo de registros (log). A mensagem é encapsulada em uma classe denominada ValueMsg. É um mecanismo um tanto sofisticado para uma aplicação como essa, mas é bom para o caso em que é necessário receber informações mais estruturadas vindas de um Arduino, tal como as leituras de um sensor. A classe ValueMsg tem membros para leitura de inteiro (int reading) e um flag que pode ser usado para indicar o tipo de leitura.

Em um exemplo mais prático, essa classe poderia ser estendida para incluir outros dados adicionais na mensagem do Arduino.

```

public class ValueMsg {
    private char flag;
    private int reading;

    public ValueMsg(char flag, int reading) {
        this.flag = flag;
        this.reading = reading;
    }

    public int getReading() {
        return reading;
    }

    public char getFlag() {
        return flag;
    }
}

```

Finalmente, o método “log” acrescenta uma linha no topo do campo de registro das operações efetuadas (log) do aplicativo, e “alert” exibe uma mensagem em um diálogo.

» Conclusão

No Open Accessory, o lado do Arduino é bem fácil de utilizar. Entretanto, o lado do Android é consideravelmente mais complexo.

Quando chegar o momento de você construir o seu próprio aplicativo, um bom ponto de partida é começar com o exemplo de projeto mostrado aqui, que pode ser baixado de www.duinodroid.com, e modificado para se adequar às suas necessidades.

Você também poderá encontrar algum projeto deste livro que se aproxima do que você quer fazer. Nesse caso, você poderá alterar o aplicativo e adaptá-lo ao que pretende. Todos esses projetos baseiam-se no aplicativo fornecido pelo Google, que foi utilizado para exemplificar o uso do ADK.



Índice

As referências às figuras estão em itálico.

A

- ADK, 23–25, 189–190
 - Veja também Open Accessory*
- ADK, baixando o pacote, 190–191
- Amarino, 18–20
- Android
 - Activity, ciclo de vida, 192–194
 - broadcast receivers, 196–197
 - enviando dados ao Arduino, 196–198
 - Open Accessory no, 192–198
 - programação, 189–190
 - recebendo dados, 197–198
 - software para o controlador de automação residencial, 105–108
- Android, contador Geiger. *Veja contador Geiger*
- Android, show de luzes. *Veja show de luzes*
- aplicativos Android. *Veja cada projeto individualmente*
- Arduino
 - Duemilanove, 94–96
 - Open Accessory no, 190–193
 - programação, 189–190
 - Uno, 10–11, 94–96
 - Veja também sketches*

B

- bandeirolas de sinalização
 - caixa de ferramentas, 171–172
 - construção, 170–175
 - construção da plataforma de madeira, 171–173
 - construção e instalação das bandeirolas, 174–175
 - diagrama de fiação, 172–173
 - diagrama esquemático, 172–173

fiação dos servos com os bornes de conexão, 172–174

fixação do Arduino à base, 172–174

lista de componentes, 170–171

programando o Arduino, 172–175

servos, 174–176

teoria, 174–176

teste, 172–174

visão geral, 169–171

Beeper, 110–113

bibliotecas, instalando, 190–192

Bluetooth, módulos, 10–14

broadcast receivers, 196–197

C

codificação de dados com som, 109–113

contador Geiger

acondicionando o projeto, 33–36

aplicativo Android, 41–42

caixa de ferramentas, 27–28

construção, 24–36

diagrama esquemático, 26

fiação final, 30–31

instalação do aplicativo Android, 32–33

instalação do sketch de Arduino, 31–32

instalação do tubo GM, 32–33

instalando as bibliotecas Open Accessory, 30–32

lista de componentes, 25, 27

soldando as barras de pino macho no shield, 27–29

soldando fios aos pinos do Arduino, 29–31

soldando os componentes baixos, 28–29

soldando os demais componentes, 28–30

teoria, 36–42

testando a fonte de alta tensão, 31–33

teste, 33–34

visão geral, 23–24

- controlador de automação residencial
acesso à Internet, 108–110
acondicionando o controlador, 103–106
cabos, instalação dos, 98–101
caixa de ferramentas, 96–97
codificação de dados com som, 109–113
decodificação dos sons no Arduino, 114–116
diagrama esquemático, 95, 112–113
eletrônica da interface de áudio, 112–114
integração do controlador de potência com o, 126–130
lista de componentes, 93–96
modificando, 142–145
módulo de conexão de áudio, 93–106
preparando a placa perfurada, 96–98
software Android, 105–108
soldando as conexões, 96–98
soldando o Cl, 97–99
soldando os demais componentes, 97–100
soldando os resistores e o diodo, 96–99
teoria, 109–116
teste, 99–105
visão geral, 91–94
Veja também temporizador; controlador de potência; fechadura com RFID; bandeirolas de sinalização; termostato inteligente
controlador de potência
caixa de ferramentas, 119, 121
configurando a sua casa, 130–131
construindo o módulo de controle de potência, 118–127
desmontando o controle remoto, 119, 121–122
diagrama de fiação com a interface de áudio, 128
diagrama esquemático, 120
dispondo os componentes na placa perfurada, 121, 124, 126
eletrônica, 117–119
integração com o controlador de automação residencial, 126–130
lista de componentes, 119, 121
relés, 130–132
sketches, 131–124
soldando as conexões, 124, 126
soldando fios à placa do controle remoto, 121–121, 124
teoria, 130–124
teste, 124, 126–127
visão geral, 117–118
- controle remoto de TV
acondicionando o projeto, 65–67
caixa de ferramentas, 62–64
construção, 62–67
construção de uma Base para Acessório Droid, 62–64
cortando a placa perfurada no tamanho, 62–64
diagrama esquemático, 62–63
fiação, 64–65
instalação do Aplicativo Android, 65–66
instalação do sketch de Arduino, 64–66
IR (infravermelho), controles remotos por, 66–68
lista de componentes, 62–63
soldando componentes, 62–65
teoria, 66–68
teste, 65–66
usando o projeto, 66–67
visão geral, 61–62

D

diagramas de transição de estado, 184–185

E

Eclipse, plug-in, 189–190
eletrônica da interface de áudio, 112–114

F

fechadura com RFID
acondicionando o projeto, 161–164
caixa de ferramentas, 157–158
construção, 154–164
diagrama esquemático, 156
disposição de componentes, 157–158
flashes do LED, 163–165
instalação, 163–164
ligando tudo junto, 160–161, 161–162
lista de componentes, 154–155, 157
preparando a placa perfurada, 157–159
programando e instalando o microcontrolador, 160–161
soldando as conexões, 157–159
soldando o soquete de Cl e as chaves, 157–160
soldando os demais componentes, 159–161
soldando os resistores e o diodo, 157–159
teoria, 165–168
teste, 160–163

- usando o sistema, 163–165
visão geral, 153–155
- f**
fechadura elétrica. Veja fechadura com RFID
fios, 94–96
- I**
infravermelho, controles remotos por, 66–68
Veja também controle remoto de TV
- M**
marcador de tempo. *Veja temporizador*
medidor de distância ultrassônico
 acondicionando o projeto, 83–85
 caixa de ferramentas, 81–82
 construção, 80–85
 construção de uma Base para Acessório Droid, 81–82
 diagrama esquemático, 80–81
 instalação do aplicativo Android, 82–83
 instalação do sketch de Arduino, 82–83
 lista de componentes, 81–82
 soldando o laser e o resistor, 82–83
 teoria, 84, 86–87
 teste, 83–84
 usando o projeto, 84, 86
 visão geral, 79–81
medidores ultrassônicos de distância, 84, 86–87
modulação por largura de pulso, 55–58
MOSFETs, 55–57
motores com engrenagens, 11–12
- O**
Open Accessory, 24–25
 exemplo, 189–191
 instalando as bibliotecas, 190–192
 no Android, 192–198
 no Arduino, 190–193
 programação com Android e Arduino, 189–190
 sketch, 191–193
Open Accessory, kit de desenvolvimento. *Veja ADK*
- P**
Pachube
 contas, 74–76
 site, 69–71
- placa perfurada, 118–119
 cortando no tamanho, 49–52
- Polulu, 11–12
- programação
 Android, 189–190
 Arduino, 189–190
- PWM. *Veja modulação por largura de pulso*
- R**
registrator de temperatura
 acondicionando o projeto, 73–75
 caixa de ferramentas, 71–72
 construção, 70–75
 construção de uma Base para Acessório Droid, 71–72
 diagrama esquemático, 70–71
 instalação do aplicativo Android, 72–73
 instalação do sketch de Arduino, 72–73
 instalação dos componentes no conector KRE, 71–73
 lista de componentes, 71–72
 teoria, 75, 77–77
 teste, 73–74
 usando o projeto, 74–75, 77
 visão geral, 69–70
relés, 130–132
robô Bluetooth
 aplicativo Android, 21–22
 caixa de ferramentas, 11–12
 construção, 9–21
 corte da parte debaixo da caixa e instalação do rodízio, 14–15
 diagrama esquemático, 10–11
 fiação final, 14–16
 instalação do aplicativo Android, 18–20
 instalação do módulo Bluetooth, 12–14
 instalação do sketch de Arduino, 18–19
 instalação dos conectores KRE no shield, 12–13
 instalação dos motores e do suporte de pilhas na caixa, 13–15
 lista de componentes, 11–12
 pondendo o robô em funcionamento, 19–21
 sketch de Arduino, 20–22
 soldando as barras de pino macho no shield, 11–13
 teoria, 20–22
 testando os motores, 16–19
 visão geral, 9–10

S

- servos, 174–176
- shield de motor, 11–12
- show de luzes
 - acondicionando o projeto, 52–56
 - aplicativo Android, 57–60
 - caixa de ferramentas, 44–46, 49–51
 - conectando tudo, 52–54
 - conexão da chave e os fios de alimentação, 48–49
 - conexão da fonte de 5V, 47–49
 - construção da Base para Acessório Droid, 44–50
 - construção do projeto de show de luzes, 49–56
 - corte da placa perfurada no tamanho, 49–52
 - diagrama esquemático, 45–46, 49–50
 - instalação do aplicativo Android, 52–55
 - instalação do cristal e dos outros componentes, 47–48
 - instalação do sketch de Arduino, 52–55
 - instalação dos fios das demais conexões, 47–48
 - lista de componentes, 45–46, 49–51
 - modulação por largura de pulso (PWM), 55–58
 - MOSFETs, 55–57
 - sketch de Arduino, 57–60
 - soldando a barra de pino fêmea, o conector KRE e o transistor, 51–52
 - soldando o soquete de CI no lugar, 46–47
 - soldando os demais componentes, 51–54
 - soldando os fios de conexão no shield USB, 46–47
 - teoria, 55–60
 - teste, 48–50, 52–55
 - usando o projeto, 55–57
 - visão geral, 43–46
- sketches
 - Open Accessory, 191–193
 - para a fechadura com RFID, 165–168
 - para o contador Geiger, 36–41
 - para o controlador de automação residencial, 114–116
 - para o controlador de potência, 131–124
 - para o robô Bluetooth, 20–22
 - para o show de luzes, 57–60
 - para o temporizador, 184–187
 - para o termostato inteligente, 148–152
- Sparkfun, 11–12

T

- temporizador
 - caixa de ferramentas, 178, 180
 - construção, 177–184
 - diagrama esquemático, 179
 - diagramas de transição de estado, 184–185
 - disposição dos componentes, 178, 180–181
 - instalação, 183–184
 - instalação do sketch de Arduino, 180–183
 - lista de componentes, 178, 180
 - preparando a placa perfurada, 178, 180–182
 - sketch, 184–187
 - soldando as conexões, 180–182
 - soldando os demais componentes, 180–182
 - soldando os soquetes, 180–182
 - teoria, 184–187
 - teste, 182–184
 - visão geral, 177–178
- termostato inteligente
 - acondicionando o projeto, 145–147
 - caixa de ferramentas, 139–140
 - construção, 136–147
 - diagrama esquemático, 138
 - disposição de componentes, 139–140
 - disposição dos componentes na placa modificada, 144
 - instalação, 146–147
 - ligando tudo junto, 141–143
 - lista de componentes, 137, 139
 - modificando o controlador de automação residencial, 142–145
 - preparando a placa perfurada, 139–141
 - programando e instalando o microcontrolador, 141–142
 - sensores de um fio, 148
 - sketch, 148–152
 - soldando as conexões e o soquete de CI, 140–141
 - soldando os demais componentes, 141–142
 - soldando os resistores e o diodo, 140–141
 - teoria, 148–152
 - teste, 145
 - usando o sistema, 146–148
 - visão geral, 135–137
- tubo Geiger-Müller (GM), 23–24, 32–33, 36–37