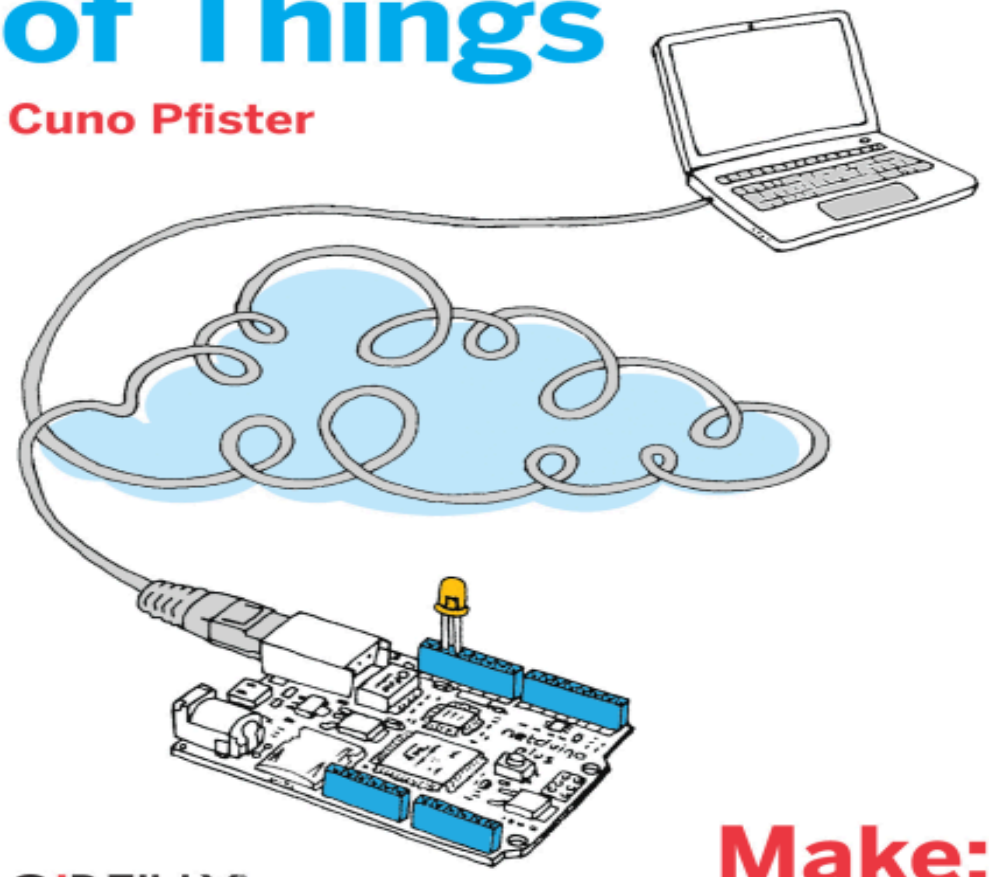
Make: PROJECTS

CONNECTING SENSORS AND MICRO-CONTROLLERS TO THE CLOUD

Getting Started with the Internet of Things

Cuno Pfister



O'REILLY

Make: makezine.com

Engenharia / Computação Em Nuvem

Marca: projectos

Introdução à Internet das coisas

O que é a Internet das coisas? São inúmeros computadores incorporados, sensores e actuadores ligados em linha. Se você tem básico habilidades de programação, você pode usar esses pequenos dispositivos poderosos para criar uma variedade de sistemas úteis-como dispositivos que reagem a eventos do mundo real e tomar medidas. Este guia prático mostra-lhe como começar a construir seus próprios projetos divertidos e fascinantes.

Aprenda a programar dispositivos incorporados usando o. Net Micro Framework e o Conselho Netduino Plus. Em seguida, conecte seus dispositivos ao Internet com Pachube, uma plataforma cloud para partilha de sensores em tempo real dados. Tudo o que você precisa é de um Netduino Plus, um cabo USB, alguns sensores, uma conexão Ethernet com a Internet-e sua imaginação.

- Desenvolver programas com saídas simples (atuadores) e entradas (sensores)
- "Aprenda sobre a Internet das coisas e a teia das coisas
- Construir programas cliente que empurram as leituras do sensor de um dispositivo para um serviço web
- Criar programas de servidor que permitem controlar um dispositivo através da Web
- "Obter programas de exemplo que você pode compilar com o Visual Studio em Windows ou Mono no Mac OS X e Linux

Dr. Cuno Pfister, fundador da Oberon microsystemns, esteve envolvido em uma variedade de projectos industriais, desde a construção de um sistema de monitorização de Centrais Eléctricas até à criação de um- Tempo Sistema Operacional Java. Ele está interessado na reunião muito pequena, a muito grande, tais como microcontroladores que falam para computar nuvens em aplicações Web of Things.

Fazer:
makezine.com

O'REILLY

EUA $ 24.99 CAN $ 28.99

ISBN: 978-1-449 -39357-1

9 781449 393571 52499

Twitter: @ oreillymedia
facebook.com/oreilly

Começando com o Internet das coisas

por Cuno Pflster

Impresso nos Estados Unidos da América.

Publicado pela O'Reilly Media, Inc.
1005 Gravenstein Highway North, Sebastopol, CA95472

Os livros da o'Reilly podem ser adquiridos para fins educacionais, comerciais ou uso promocional de vendas. As edições Online também estão disponíveis para a maioria títulos (http://my.safaribooksonline.com). Formaisinformações, entre em contato com nosso departamento de vendas corporativo/institucional: 800-998-9938 ou corporate@oreilly.com .

História Impressa: Maio De 2011: Primeira Edição.

Editor: Brian Jepson
Editor De Produção: Jasmine Perez
Editor: Marlowe Shaeffer
Revisora: Emily Quill
Composição: Nancy Wolfe Kotary
Indexador: Angela Howard
Ilustrações: Marc De Vinck
Designer da capa: Marc De Vinck

ISBN: 978-1-4493-9357-1

[LSI]

Conteúdo

Prefácio

Uma das tendências mais fascinantes atualmente é o surgimento de microcontroladores de baixo custo que são suficientemente poderosos para se conectar à Internet. Eles são a chave para a Internet das Coisas, onde todos os tipos de dispositivos se tornam a interface da Internet com o mundo físico.

Tradicionalmente, programar esses pequenos dispositivos embarcados exigia plataformas e ferramentas completamente diferentes das que a maioria dos programadores estava acostumada. Felizmente, alguns microcontroladores agora são capazes de suportar plataformas de software modernas como .NET, ou pelo menos subconjuntos úteis do .NET. Isso permite que você use a mesma linguagem de programação (C#) e o mesmo ambiente de desenvolvimento (Visual Studio) ao criar programas para pequenos dispositivos embarcados, smartphones, PCs, servidores corporativos e até mesmo serviços em nuvem.

Então, o que você deve saber para começar? Este livro dá uma possível resposta a essa pergunta. É um livro de Introdução, portanto, não é uma extensa coleção de receitas (ou padrões de design, para dizer a verdade), nem um manual de referência, nem um livro didático que compare diferentes abordagens, casos de uso, etc. Em vez disso, sua abordagem é "menos é mais", ajudando você a começar a escrever aplicativos para a Internet das Coisas com o mínimo de problemas.

As Plataformas

O .NET Micro Framework (NETMF) fornece conectividade com a Internet, é simples e de código aberto (licença Apache), tem hardware disponível de vários fornecedores e se beneficia do enorme ecossistema .NET e do conhecimento disponível. Além disso, você pode escolher entre o Visual Studio (incluindo a edição Express gratuita) no Windows, e a cadeia de ferramentas Mono de código aberto no Linux e no Mac OS X.

Existe uma comunidade ativa para o NETMF em http://www.netmf.com/Home.aspx. O próprio projeto está hospedado em

http://netmf.codeplex.com/.

Netduino Plus (http://www.netduino.com/netduinoplus) é uma placa NETMF de baixo custo da Secret Labs (http://www.secretlabs.com). Esta placa disponibiliza a rede Ethernet com um preço inferior a US$60. Possui as seguintes características:

Um microcontrolador Atmel SAM7 de 48 MHz com 128 KB de RAM e 512 KB de memória Flash
USB, Ethernet e 20 pinos de E/S digital (seis dos quais podem ser configurados opcionalmente para entrada analógica)
Suporte para cartão Micro SD
LED e botão de pressão a bordo
Fator de forma do Arduino (http://www.arduino.cc/); muitos shields Arduino (placas de expansão) podem ser usados
.NET Micro Framework pré-programado na memória Flash
Todo o software e hardware são de código aberto
Existe uma comunidade ativa para o Netduino Plus (e NETMF) em http://forums.netduino.com/. Todos os exemplos neste livro usam o Netduino Plus.

Como Este Livro Está Organizado

O livro é composto por três partes:

» Parte I, Introdução
A primeira parte ensina como configurar o ambiente de desenvolvimento e escrever e executar um programa "Olá, Mundo". Mostra como escrever em portas de saída (para acionar atuadores como luzes LED ou motores) e como ler de portas de entrada (para sensores). Em seguida, apresenta os conceitos mais essenciais da Internet das Coisas: HTTP e a divisão de trabalho entre clientes e servidores. Na Internet das Coisas, os dispositivos são programados como clientes se você deseja que eles enviem dados do sensor para algum serviço; eles são programados como servidores se você deseja permitir o controle remoto do dispositivo pela Web.

» Parte II, Dispositivo como Cliente HTTP
A segunda parte concentra-se em exemplos que enviam solicitações HTTP para

alguns serviços - por exemplo, para enviar novas medições de sensores para o serviço Pachube (http://www.pachube.com) para armazenamento e apresentação.

» Parte III, Dispositivo como Servidor HTTP
A terceira parte concentra-se em exemplos que lidam com solicitações HTTP recebidas. Tal solicitação pode retornar uma medição recente de um sensor ou acionar um atuador. Uma biblioteca do lado do servidor adequada é fornecida para tornar mais fácil do que nunca programar um pequeno dispositivo como servidor.

» Apêndice A, Servidor de Teste
Este apêndice contém um simples servidor de teste que é útil para testar e depurar programas de cliente.

» Apêndice B, Classes .NET Usadas nos Exemplos
Isso mostra as classes .NET necessárias para implementar todos os exemplos, e os namespaces e assemblies que as contêm.

» Apêndice C, Biblioteca Gsi Ot.Server
Isso resume a interface da biblioteca auxiliar Gsi Ot.Server que usamos na Parte III.

Para Quem é Este Livro?

Este livro destina-se a qualquer pessoa com pelo menos habilidades básicas de programação em uma linguagem orientada a objetos, bem como interesse em sensores, microcontroladores e tecnologias da web. O público-alvo do livro consiste nos seguintes grupos:

» Artistas e designers
Você precisa de uma plataforma de prototipagem que suporte conectividade à Internet, seja para criar aplicativos compostos por múltiplos dispositivos de comunicação, ou para integrar a World Wide Web em um projeto de alguma forma. Você deseja transformar suas ideias em realidade rapidamente e valoriza ferramentas que o ajudem a concluir o trabalho. Talvez você tenha experiência com a popular plataforma Arduino de 8 bits (http://www.arduino.cc/), e até mesmo possa reutilizar algum hardware adicional (como shields e breakout boards) originalmente projetado para Arduino.

» Estudantes e entusiastas
Você deseja que seus programas interajam com o mundo físico, usando ferramentas convencionais. Você está interessado em placas de desenvolvimento, como o Netduino Plus, que não custam os olhos da cara.

» Desenvolvedores de software ou seus gerentes
Você precisa integrar dispositivos embarcados com serviços web e deseja aprender o básico rapidamente. Você deseja construir uma intuição que abrange desde a arquitetura geral do sistema até o código real. Dependendo de seus investimentos anteriores em plataformas, você pode ser capaz de usar os exemplos deste livro como ponto de partida para estudos de viabilidade, prototipagem ou desenvolvimento de produtos. Se você já conhece .NET, C# e Visual Studio, pode usar a mesma linguagem de programação e ferramentas com as quais já está familiarizado, incluindo o depurador do Visual Studio.

Para permanecer flexível, você deseja escolher entre diferentes placas de diferentes fornecedores, permitindo que você passe de protótipos baratos para produtos finais sem ter que alterar a plataforma de software. Para aumentar ainda mais a independência do fornecedor, você provavelmente deseja usar plataformas de código aberto, tanto para hardware quanto para software. Para minimizar os custos, você está interessado em uma plataforma que não exija o pagamento de royalties-alvo, ou seja, custos de licença por dispositivo.

Se sua formação é na programação de PCs ou até mesmo computadores mais poderosos, um aviso justo: programação embarcada para dispositivos de baixo custo significa trabalhar com recursos muito limitados. Isso contrasta chocantemente com a World Wide Web, onde as tecnologias geralmente parecem ser criadas com a máxima eficiência como objetivo. A programação embarcada requer uma consideração mais cuidadosa sobre como os recursos são usados do que o necessário para PCs ou servidores. Plataformas embarcadas fornecem apenas pequenos subconjuntos da funcionalidade de seus primos maiores, o que pode exigir algum engenho e trabalho onde um recurso desejado não está disponível diretamente. Isso pode ser doloroso se você se sentir em casa com "quanto mais, melhor", mas será divertido e gratificante se você ver o atrativo do "pequeno é bonito".

O Que Você Precisa Para Começar

Este livro se concentra na interação entre dispositivos embarcados e outros computadores na Internet, usando protocolos web padrão. Seus exemplos usam principalmente sensores e atuadores básicos, portanto, não é necessário comprar muito hardware adicional além de uma placa de computador de baixo custo. Aqui está uma lista das coisas que você precisa para executar todos os exemplos neste livro:

» Uma placa Netduino Plus (http://www.netduino.com/netduinoplus)
» Um cabo micro USB (plugue USB-A macho normal no lado do PC, plugue micro USB-B macho no lado do Netduino Plus), para ser usado durante o desenvolvimento e para fornecer energia
» Um roteador Ethernet com uma porta Ethernet disponível para o seu Netduino Plus
» Uma conexão com a Internet para o seu roteador Ethernet
» Um cabo Ethernet para a comunicação entre o Netduino Plus e o roteador Ethernet
» Um potenciômetro com uma resistência de cerca de 100 quilohms e conectores through-hole
» Um PC com Windows XP/Vista/7, 32 bits ou 64 bits, para o ambiente de desenvolvimento gratuito do Visual Studio Express 2010 (alternativamente, você pode usar o Windows em uma máquina virtual no Mac OS X ou Linux, ou você pode usar a cadeia de ferramentas Mono no Linux ou Mac OS X)
NOTA: Existem várias fontes onde você pode comprar os componentes de hardware mencionados acima, assumindo que você já tenha um roteador com uma conexão à Internet:

» Maker SHED (http://www.makershed.com/)

Netduino Plus, número de peça MKND02
Potenciômetro, número de peça JM2118791
» **SparkFun** (http://www.sparkfun.com/)

Netduino Plus, número de peça DEV-10186
Esses fornecedores devem ter todos os itens necessários para montar o seu ambiente de desenvolvimento.
Cabo Micro USB, número de peça CAB-10215 (incluído com Netduinos por tempo limitado)
» Cabo Ethernet, número de peça CAB-08916

» Potenciômetro, número de peça COM-09806

Para mais fontes nos EUA e em outras regiões do mundo, por favor, consulte http://www.netduino.com/buy/?pn=netduinoplus. Também é possível adicionar sensores e atuadores adicionais.

Convenções Usadas Neste Livro

As seguintes convenções tipográficas são usadas neste livro:
» Itálico
Indica novos termos, URLs, endereços de e-mail, nomes de arquivos e extensões de arquivo.
» Largura constante
Usado para listagens de programas, bem como dentro de parágrafos para se referir a elementos do programa, como nomes de variáveis ou funções, tipos de dados, declarações e palavras-chave.
» Negrito de largura constante
Mostra comandos ou outro texto que deve ser digitado literalmente pelo usuário.
» Itálico de largura constante
Mostra texto que deve ser substituído por valores fornecidos pelo usuário ou por valores determinados pelo contexto.
NOTA: Este estilo indica uma dica, sugestão ou nota geral.

Usando Exemplos de Código

Este livro está aqui para ajudá-lo a realizar seu trabalho. Em geral, você pode usar o código deste livro em seus programas e documentação. Você não precisa nos contatar para permissão, a menos que esteja reproduzindo uma parte significativa do código. Por exemplo, escrever um programa que usa vários trechos de código deste livro não requer permissão. Vender ou distribuir um CD-ROM de exemplos de livros da O'Reilly requer permissão. Responder a uma pergunta citando este livro e citando código de exemplo não requer permissão. Incorporar uma quantidade significativa de código de exemplo deste livro na documentação do seu produto requer permissão.

Agradecemos, mas não exigimos, atribuição. Uma atribuição geralmente inclui o título, autor, editor e ISBN. Por exemplo:
“Getting Started with the Internet of Things, por Cuno Pfister.
Copyright 2011 Cuno Pfister, 978-1-4493-9357-1.”

Se você achar que o uso de exemplos de código está fora do uso justo ou da permissão concedida aqui, sinta-se à vontade para entrar em contato conosco em permissions@oreilly.com.

Como Entrar em Contato Conosco

Por favor, envie comentários e perguntas sobre este livro para o editor:
O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (nos Estados Unidos ou Canadá)
707-829-0515 (internacional ou local)
707-829-0104 (fax)

Temos uma página da web para este livro, onde listamos erratas, exemplos e qualquer informação adicional. Você pode acessar esta página em:
http://oreilly.com/catalog/0636920013037

Para comentários ou perguntas técnicas sobre este livro, envie um e-mail para:
bookquestions@oreilly.com

Para obter mais informações sobre nossos livros, conferências, Centros de Recursos e a Rede O'Reilly, consulte nosso site em: http://oreilly.com

Introdução - Capítulo 1.

Graças ao progresso incansável da indústria de semicondutores, todas as partes digitais de um computador podem ser colocadas em um único chip, chamado microcontrolador. Um chip de microcontrolador de 32 bits custando menos de $10 pode ter mais do que o dobro de memória do que o computador original Apple II de 8 bits, com seus 48 KB de RAM, e pode rodar 100 vezes mais rápido. Uma placa para entusiastas que incorpora tal chip, juntamente com Ethernet e um slot para cartão Micro SD, pode ser adquirida por cerca de $60.

Devido a esse hardware barato e plataformas de desenvolvimento fáceis de usar, agora é possível para entusiastas criar sistemas que interagem com o mundo físico de todas as maneiras concebíveis. Por exemplo, um sensor pode medir a umidade em um vaso de flores, e uma válvula controlada por computador (atuador) permite a passagem de água para o vaso quando a umidade fica muito baixa. Além disso, como o hardware permite o uso de protocolos de Internet padrão, o monitoramento e controle podem ser feitos pela Internet. Vários serviços da Internet podem ser usados para armazenar dados, visualizá-los, compartilhá-los com outras pessoas, etc. Por exemplo, para aprender sobre os efeitos sazonais na umidade, você pode armazenar medições da umidade do seu vaso de flores ao longo de um ano.

Embora essas possibilidades sejam fascinantes e promissoras, também há algo sinistro sobre o potencial para dispositivos espionarem todos os nossos movimentos. Isso fornece mais uma razão pela qual devemos tentar aprender como esses sistemas funcionam. Essa compreensão é, ou pelo menos deveria ser, a base para pensar em políticas de privacidade que se tornarão necessárias mais cedo ou mais tarde.

Na Parte I, vou mostrar como configurar o ambiente de desenvolvimento para que você possa começar a brincar com sensores e atuadores simples. Então, vou preparar o terreno para as Partes II e III, que mostram como você pode programar dispositivos como clientes que enviam solicitações para vários serviços, ou como servidores que lidam com solicitações de clientes, por exemplo, de navegadores da web.

1/Hello World

Para familiarizá-lo com o ambiente de desenvolvimento, seu primeiro programa deve ser um simples Hello World. Como a placa Netduino Plus não possui um display, use a conexão USB entre a placa e o PC de desenvolvimento para escrever a string Hello World na janela de Saída do ambiente de desenvolvimento em execução no PC, conforme ilustrado na Figura 1-1. A conexão USB é usada para implantar e depurar seus programas, e no exemplo HelloWorld, permite que você envie a string Hello World para o seu PC de desenvolvimento.

Configurando o Ambiente de Desenvolvimento

Antes de escrever seu primeiro programa para o .NET Micro Framework, você precisa instalar algumas ferramentas e bibliotecas, incluindo:

» **Microsoft Visual Studio 2010** ou posterior. A versão gratuita Visual Studio Express é suficiente. Versões comerciais completas também podem ser usadas, é claro. Para minhas descrições e capturas de tela, usarei o Visual Studio Express. Se você usar o Visual Studio Express, deve instalar a edição C# em http://www.microsoft.com/express/Downloads.

» **Microsoft .NET Micro Framework 4.1 SDK** ou posterior, disponível em http://www.netduino.com/downloads/MicroFrameworkSDK.msi. (Consulte http://www.netduino.com/downloads/ para obter mais informações sobre SDKs compatíveis.)

» **SDK e drivers da sua placa de desenvolvimento.** O SDK e os drivers para o Netduino Plus podem ser baixados em http://www.netduino.com/downloads/.

A biblioteca **Griot.Pachube Client do lado do cliente e a biblioteca Gsi Ot.Server do lado do servidor**, que são usadas em alguns dos exemplos deste livro. Elas podem ser baixadas em http://www.gsiot.info/download/.

Todos esses pacotes de software são gratuitos. As ferramentas acima requerem Windows XP, Vista ou Windows 7.

NOTA: O suporte para Mac e Linux deve estar disponível quando este livro for impresso. Para as atualizações mais recentes, consulte http://forums.netduino.com/.

HelloWorld

O programa HelloWorld (Exemplo 1-1) contém uma classe HelloWorld com um método estático Main sem parâmetros.

As palavras-chave public, static, void especificam o tipo do método; neste caso, é público (é visível para outras classes), estático (não precisa de uma instância da classe HelloWorld para executar o método) e void (não retorna um valor). Além disso, como os parênteses estão vazios, Main() não espera que você passe argumentos (objetos ou variáveis que seriam referidos dentro do método).

Na verdade, você não chamará Main() por conta própria; o NETMF faz isso por você. Quando o Netduino Plus é reiniciado ou ligado, ele procura pelo método Main() e o executa como o ponto de entrada do seu programa. Este programa escreve a string Hello World em um console de depuração, por exemplo, na janela de saída do Visual Studio.

Exemplo 1-1. Programa HelloWorld

Código:

```csharp
using Microsoft.SPOT;
public class HelloWorld
{
        public static void Main()
        {
        Debug.Print("Hello World");
    }
}
```

Exemplo com as indentações corrigidas:

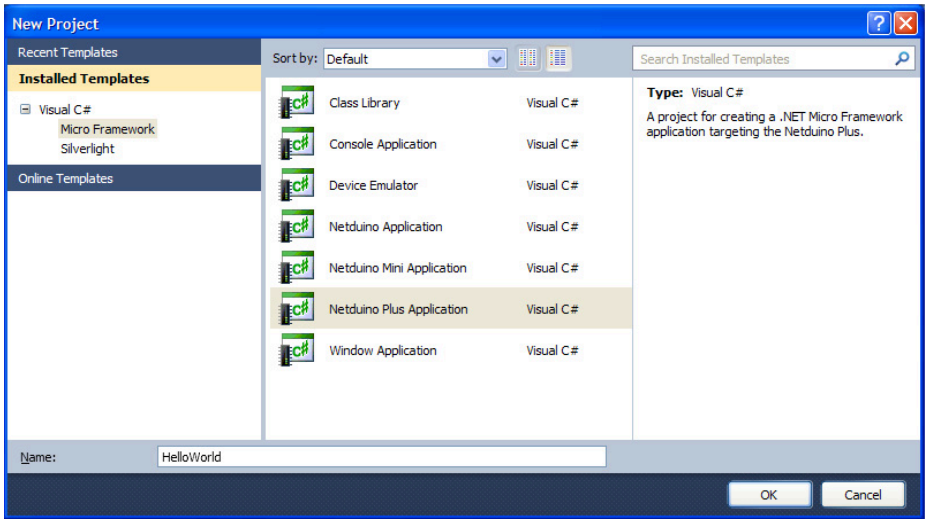
```csharp
using Microsoft.SPOT;

public class HelloWorld
{
    public static void Main()
    {
        Debug.Print("Hello World");
    }
}
```

O NETMF fornece uma classe Debug no namespace Microsoft.SPOT. O método Print do Debug escreve saída de texto diretamente para o ambiente de desenvolvimento via o mesmo transporte (conexão) usado para implantar software no dispositivo e para depurar. Na placa Netduino Plus, é um transporte USB. Outras placas de desenvolvimento podem usar um transporte serial (RS-232) ou um transporte Ethernet.

Compilando o Programa no Visual Studio

Assumindo que você já instalou o SDK do .NET Micro Framework e o SDK do Netduino, existem alguns passos que você deve seguir antes de poder digitar o programa HelloWorld:

1. Inicie o **Visual Studio.**
2. Clique em Arquivo→Novo Projeto....
3. Selecione **Micro Framework no painel Modelos Instalados**, **selecione Netduino Plus Application no painel do meio e digite Hello World** no campo Nome na parte inferior (veja a Figura 1-2). Em seguida, clique em OK.

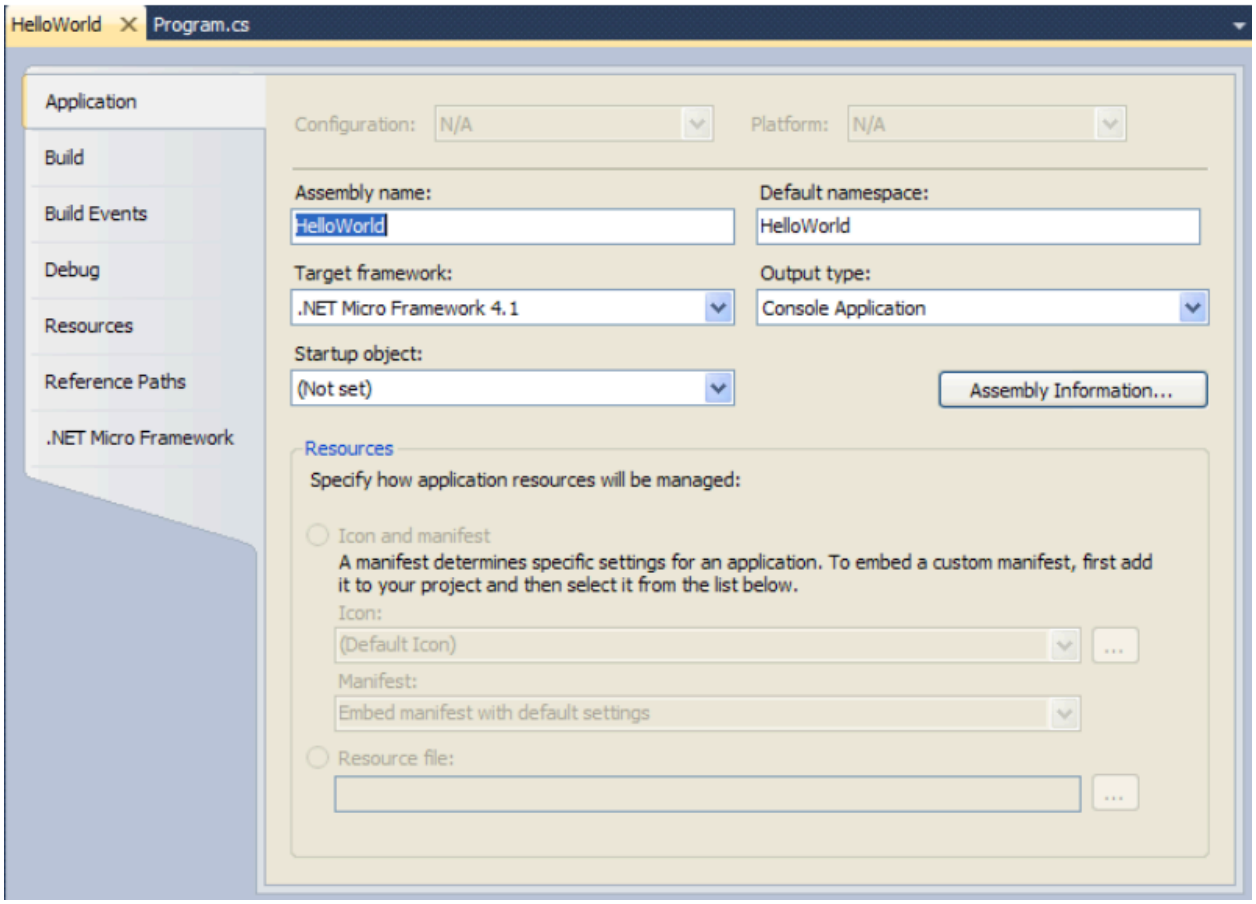


4. **No Solution Explorer no lado direito**, clique duas vezes em *Program.cs.* Uma aba com o título Program.cs será aberta, contendo algum texto básico.
5. Substitua o texto pelo programa **HelloWorld** do Exemplo 1-1.
6. Selecione Depurar→Compilar Solução para compilar a solução. No canto inferior esquerdo do Visual Studio, agora deverá dizer "Compilação bem-sucedida".

Implantando no Dispositivo

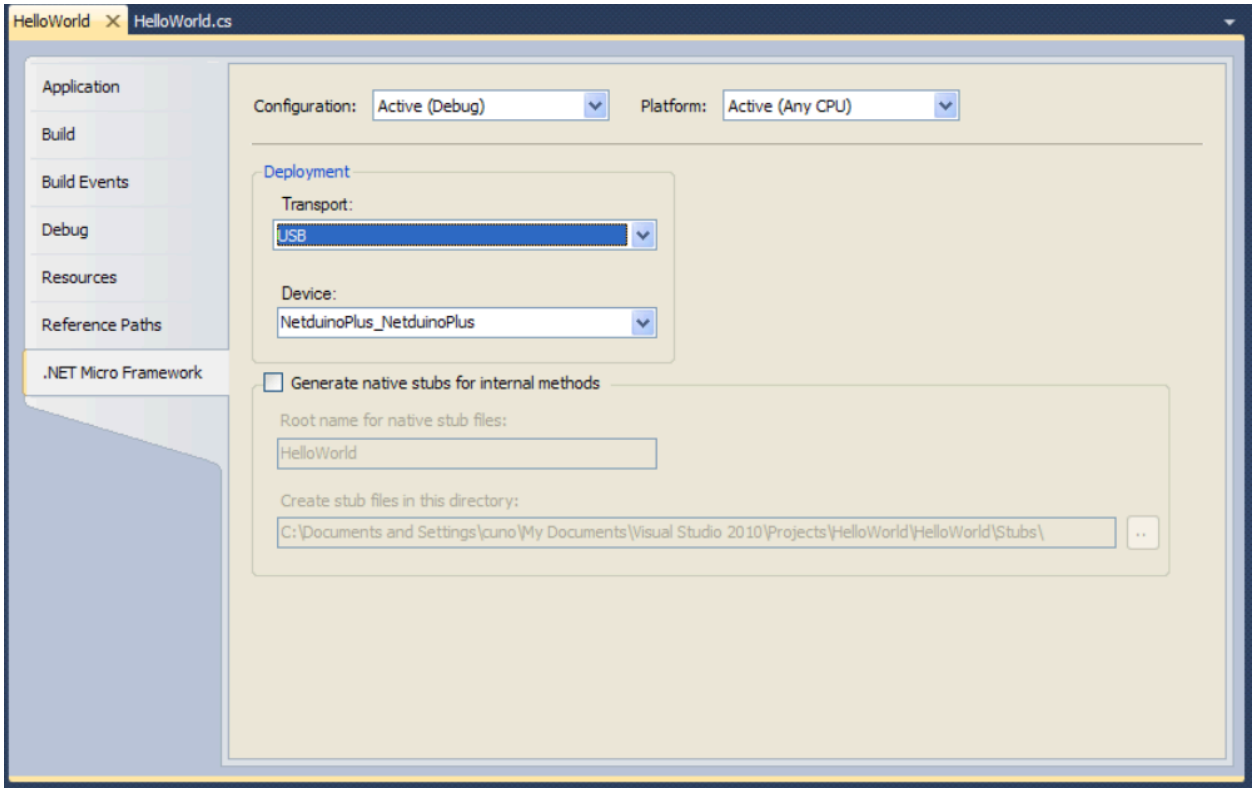
Depois de construir o exemplo, você pode implantá-lo no seu hardware. Primeiro, você precisa garantir que as propriedades de implantação estejam configuradas conforme mostrado na Figura 1-3. Para fazer isso, siga as seguintes etapas:

1. **No Solution Explorer**, clique com o botão direito do mouse no projeto HelloWorld (logo abaixo do texto "Solução 'HelloWorld' (1 projeto)"), em seguida, selecione Propriedades no menu. A guia mostrada na Figura 1-3 será aberta.



2. No lado esquerdo, clique na guia **.NET Micro Framework**, o que resulta na caixa de diálogo mostrada na Figura 1-4. Certifique-se de que as propriedades estejam configuradas da seguinte forma:

- Configuração: Ativa (Depurar)
- Plataforma: Ativa (Qualquer CPU)
- Transporte: USB
- Dispositivo: selecione o seu Netduino na lista suspensa.
- Gerar stubs nativos para métodos internos: desmarcado



3. Se a **caixa de lista Dispositivo** mostrar <nenhum>, você precisa conectar o seu ***Netduino Plus***. Na primeira vez que você o conectar, o driver deve ser instalado automaticamente. O nome dele deve aparecer quando você clicar na caixa de lista Dispositivo.
4. Para abrir a janela de Saída, que mostrará a saída de depuração, use o atalho de teclado **Ctrl-W**, seguido de O.
5. Em seguida, selecione **Depurar→Iniciar Depuração**, e o programa HelloWorld será enviado para sua placa, carregado pelo **.NET Micro Framework**, após o qual o método Main é executado. O programa então termina imediatamente. Você pode ver a saída de depuração no Visual Studio. O final da saída deve se parecer com isso:

Retorno:
```
O thread '<Nenhum Nome>' (0x2) saiu com código 0 (0x0).
Hello World
O thread '<Nenhum Nome>' (0x1) saiu com código 0 (0x0).
O programa '[1] Aplicativo Micro Framework: Gerenciado' saiu com código 0 (0x0).
```

Agora você implantou com sucesso seu primeiro programa em um dispositivo real! Certamente ainda não é um aplicativo da Internet das Coisas, pois não envolve nenhuma comunicação pela Internet. Também não é um aplicativo

embarcado, pois não usa nenhuma das entradas ou saídas típicas embarcadas (que veremos nos próximos capítulos).

NOTA: Se houver algum problema durante a implantação, desconecte o cabo USB do seu PC. Se uma caixa de diálogo com o texto "Houve erros de implantação. Continuar?" aparecer, clique no botão Não. Reconstrua o programa. Em seguida, conecte o cabo USB novamente e clique imediatamente em Depurar→Iniciar Depuração. Em algumas circunstâncias raras (geralmente envolvendo programas complicados), o dispositivo parece ficar realmente travado, e um ciclo de energia não ajuda. Nestes casos, pode ajudar a apagar seu programa do Netduino Plus seguindo as seguintes etapas:

1. Inicie a ferramenta **MFDeploy** (descrita no Capítulo 6) e certifique-se de que USB está selecionado.
2. Desconecte seu **Netduino Plus**, em seguida, conecte-o novamente enquanto mantém pressionado o botão onboard.
3. Solte o botão e então pressione o botão Apagar na ferramenta **MFDeploy**.

2/Controlando Atuadores

Agora você pode escrever seu primeiro programa verdadeiramente embarcado. Em uma tradição venerável, este programa, BlinkingLed, que é o equivalente embarcado do HelloWorld, faz um LED piscar.

Na **Figura 2-1**, a caixa grande indica um Netduino Plus, que tem um LED azul—rotulado como LED na placa—que pode ser controlado a partir de um programa de aplicação. Este LED está conectado a um pino de entrada/saída de propósito geral (GPIO) do microcontrolador. A maioria dos microcontroladores tem vários desses pinos GPIO, cada um dos quais pode ser configurado como entrada digital ou saída digital. Uma saída digital pode estar conectada a um LED, como no nosso exemplo; uma entrada digital pode estar conectada a um interruptor ou botão.

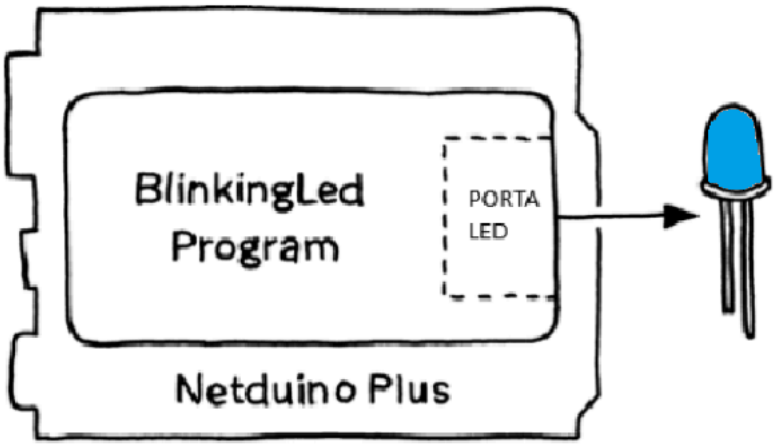


Figure 2-1. Architecture of BlinkingLed

BlinkingLed

O programa BlinkingLed, mostrado no Exemplo 2-1, contém um loop infinito simples que acende o LED, espera meio segundo, desliga o LED novamente, espera mais meio segundo e então começa tudo de novo.

Exemplo 2-1. BlinkingLed

```csharp
using System.Threading;
using Microsoft.SPOT.Hardware;
using SecretLabs.NETMF.Hardware.NetduinoPlus;

public class BlinkingLed
{
    public static void Main()
    {
        var ledPort = new Output Port(Pins.ONBOARD LED, false);

        while (true)
        {
            led Port.Write(true);    // acende o LED
            Thread.Sleep(500);      // espera 500 ms
            led Port.Write(false);   // desliga o LED
            Thread.Sleep(500);      // espera 500 ms
        }
    }
}
```

Código real:

```
using System.Threading;
using Microsoft.SPOT.Hardware;
using SecretLabs.NETMF.Hardware.NetduinoPlus;

public class BlinkingLed
{
    public static void Main()
    {
        var ledPort = new OutputPort(Pins.ONBOARD_LED, false);

        while (true)
        {
            ledPort.Write(true);    // turn on LED
            Thread.Sleep(500);      // wait 500 ms

            ledPort.Write(false);   // turn off LED
            Thread.Sleep(500);      // wait 500 ms
        }
    }
}
```

Este programa cria um *loop* infinito que alterna o estado do LED a cada meio segundo. Ele usa a classe `*Output Port*` do namespace `Microsoft.SPOT.Hardware` para controlar o LED conectado ao pino; `Pinos.ONBOARD LED`. O método `Write(true)` liga o LED e `Write(false)` ;desliga o LED. O método `*Thread.Sleep(500)*` pausa a execução do programa por 500 milissegundos (meio segundo) entre cada alteração de estado do LED.

NOTA: Em muitos programas .NET, você verá o desenvolvedor especificar um nome de tipo (como OutputPort) para declarações de variáveis. Para simplificar as coisas, neste livro, uso a palavra-chave var para todas as declarações de variáveis onde o tipo é óbvio, como:

- Se a variável for inicializada com um valor literal (número, string).
- Para um objeto criado através de seu construtor (porque você sempre verá o nome de sua classe no lado direito da expressão, como é o caso deste exemplo).
- Quando um tipo de conversão é usado (veja a seção “C# Protegendo Você de Conversões Perigosas” no Capítulo 12).

Em todos os outros casos, uso o nome do tipo para tornar o tipo da variável inequívoco e óbvio—mesmo que você esteja lendo este livro "em papel".

Espaços de Nomes C#

Em C#, classes relacionadas são agrupadas em chamados espaços de nomes (namespaces). No programa BlinkingLed, o namespace **Microsoft.SPOT.Hardware** fornece a classe *OutputPort*. Seu nome completo é **Microsoft.SPOT.Hardware.Output Port.** Claro, você poderia soletrar o nome completo da classe toda vez que a usasse, mas por uma questão de legibilidade e conveniência, muitas vezes é preferível usar uma diretiva using. Se você especificar a diretiva using Microsoft.SPOT.Hardware; (como eu fiz em BlinkingLed) no início do seu programa, você pode usar o nome curto OutputPort, em vez do nome completo. Eu usarei nomes curtos neste livro; por favor, consulte as tabelas no Apêndice B para encontrar o namespace apropriado para cada classe usada nesses exemplos.

NOTA: O **"SPOT"** em vários namespaces do *NETMF* significa Smart Personal Object Technology, originalmente desenvolvido para dispositivos pessoais programáveis como relógios. O .NET Micro Framework cresceu a partir dessas atividades.

Executando o Programa

Para executar o programa, crie um novo projeto de Aplicativo Netduino Plus no Visual Studio e substitua o conteúdo do Program.cs pelo código fornecido no Exemplo 2-1. Em seguida, construa e implante para o seu Netduino Plus, conforme descrito na seção "Implantando no Dispositivo" no Capítulo 1.

Saídas Digitais

No **.NET Micro Framework**, usar um pino físico como saída é representado por um objeto de porta de saída, que é uma instância da classe OutputPort. Uma porta de saída fornece o método Write que recebe o estado alvo do pino de saída como um parâmetro Booleano (true ou false). Usando uma porta de saída, chamada ledPort no Exemplo 2-1, o LED pode ser ligado escrevendo o valor true e desligado escrevendo o valor false. Quando eu defini a porta de saída ledPort, especifiquei o pino do microcontrolador que está conectado ao LED. Neste caso, quero usar o LED embutido (onboard).

Os pinos são representados pelo tipo `Cpu.Pin`, mas você não especifica o número do pino que deseja usar. Em vez disso, os fabricantes fornecem constantes para os pinos em suas placas. Em um Netduino Plus, você deve

especificar `**Pins.ONBOARD LED**` para o pino do LED embutido. Neste livro, estamos principalmente interessados nas constantes mostradas na Tabela 2-1, onde também incluo alguns portos de entrada a serem usados em capítulos posteriores. Esses pinos são definidos no namespace `**SecretLabs.NETMF.Hardware.Netduino Plus**`, que é fornecido como parte do SDK do Netduino. Quando você digita `Pins.`, o Visual Studio convenientemente exibe uma lista de todos os pinos disponíveis em um Netduino Plus.

Os pinos são representados pelo tipo `**Cpu.Pin**`, mas você não especifica o número do pino que deseja usar. Em vez disso, os fabricantes fornecem constantes para os pinos em suas placas. Em um Netduino Plus, você deve especificar `**Pins.ONBOARD LED**` para o pino do LED embutido. Neste livro, estamos principalmente interessados nas constantes mostradas na Tabela 2-1, onde também incluo alguns portos de entrada a serem usados em capítulos posteriores. Esses pinos são definidos no namespace `**SecretLabs.NETMF.Hardware.Netduino Plus**`, que é fornecido como parte do SDK do Netduino. Quando você digita `**Pins**.`, o Visual Studio convenientemente exibe uma lista de todos os pinos disponíveis em um Netduino Plus.

Hardware Conectado:	Uso do Pino:	Constante:
LED embutido (azul)	Saída digital	`Pins.ONBOARD_LED`
Botão embutido	Entrada digital	`Pins.ONBOARD_SW1`
Pinos D0 a D13	Entrada digital ou saída digital	`Pins.GPIO_PIN_D0` a `Pins.GPIO_PIN_D13`
Pinos A0 a A5	Entrada analógica	`Pins.GPIO_PIN_A0` a `Pins.GPIO_PIN_A5`

OBSERVAÇÃO: Muitos LEDs azuis e brancos super brilhantes podem tolerar os GPIOs de 3,3V que o Netduino Plus usa. Você pode conectar tal LED a qualquer um dos pinos GPIO: o terminal longo (positivo) vai para o pino GPIO, e o terminal curto (negativo) vai para o pino de terra da placa. No entanto, se estiver usando um LED de outra cor, observe que ele prefere uma tensão mais baixa; portanto, você deve colocar um resistor de 220 ohms entre um dos terminais do LED (qualquer um está OK) e sua placa.

O segundo parâmetro do construtor Output Port mostrado no Exemplo 2-1 indica se o LED deve ser inicialmente ligado ou desligado. No nosso caso, falso indica que ele deve estar desligado no início.

Um pino pode ser usado com no máximo uma porta de saída (ou entrada) ao mesmo tempo — ou seja, criar um objeto de porta reserva esse pino. Tentativas de reservar um pino várias vezes levarão a uma exceção, que é um evento de software acionado por uma condição de erro. A menos que você crie manipuladores que capturem e resolvam exceções, elas normalmente causarão a parada do seu programa Netduino Plus.