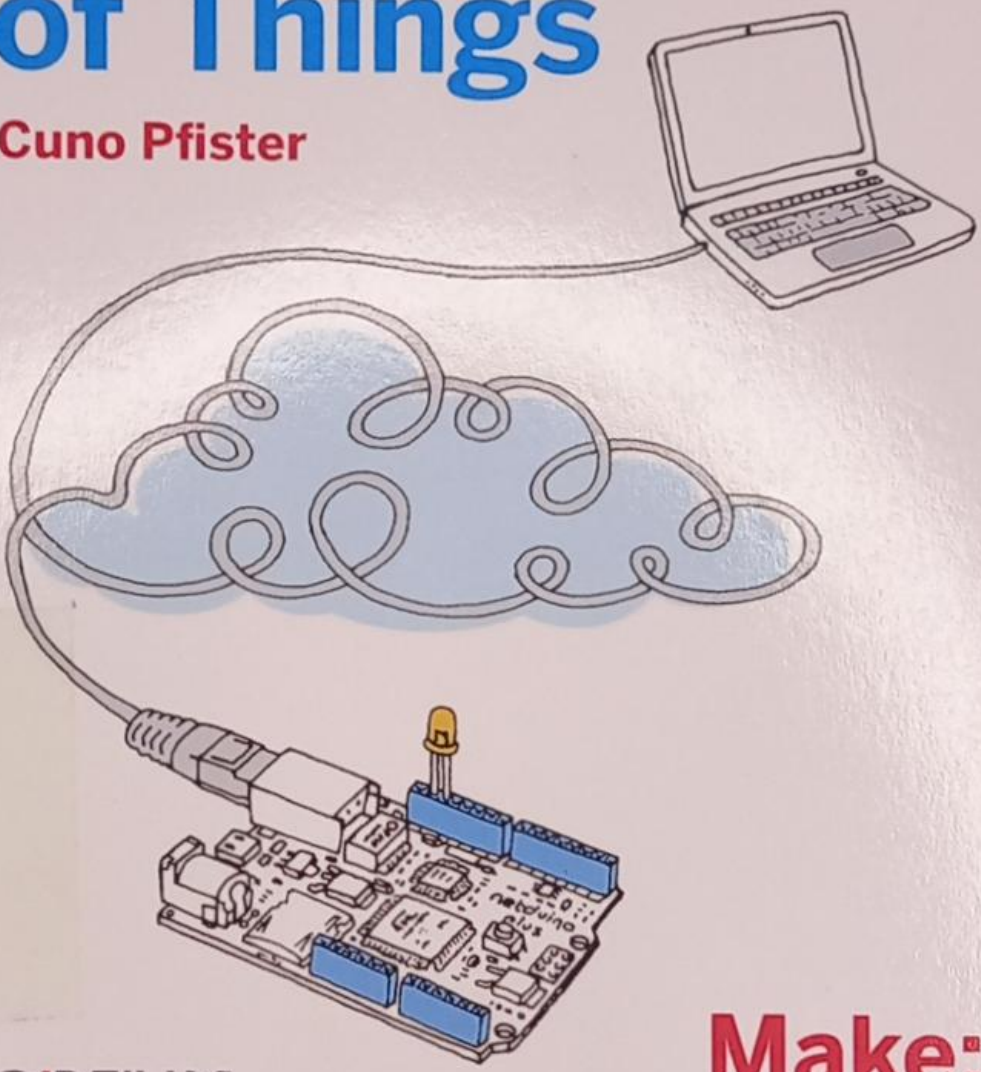
Make: PROJECTS

CONNECTING SENSORS AND MICRO-CONTROLLERS TO THE CLOUD

Getting Started with the Internet of Things

Cuno Pfister

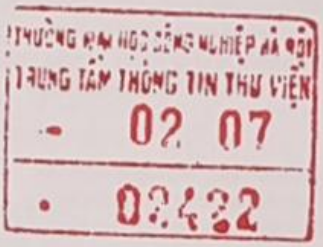


O'REILLY

Make:
makezine.com

Getting Started with the Internet of Things

Cuno Pfister



O'REILLY

BEIJING • CAMBRIDGE • FARNHAM • KÖLN • SEBASTOPOL • TAIPEI • TOKYO

Getting Started with the Internet of Things

by Cuno Pfister

Printed in the United States of America.

Published by O'Reilly Media, Inc.
1005 Gravenstein Highway North, Sebastopol, CA 95472

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Print History: May 2011: First Edition.

Editor: Brian Jepson
Production Editor: Jasmine Perez
Copyeditor: Marlowe Shaeffer
Proofreader: Emily Quill
Compositor: Nancy Wolfe Kotary
Indexer: Angela Howard
Illustrations: Marc de Vinck
Cover Designer: Marc de Vinck

ISBN: 978-1-4493-9357-1

[LSI]

Contents

Preface

One of the most fascinating trends today is the emergence of low-cost *microcontrollers* that are sufficiently powerful to connect to the Internet. They are the key to the *Internet of Things*, where all kinds of devices become the Internet's interface to the physical world.

Traditionally, programming such tiny *embedded* devices required completely different platforms and tools than those most programmers were used to. Fortunately, some microcontrollers are now capable of supporting modern software platforms like .NET, or at least useful subsets of .NET. This allows you to use the same programming language (C#) and the same development environment (Visual Studio) when creating programs for small embedded devices, smartphones, PCs, enterprise servers, and even cloud services.

So what should you know in order to get started? This book gives one possible answer to this question. It is a *Getting Started* book, so it is neither an extensive collection of recipes (or design patterns for that matter), nor a reference manual, nor a textbook that compares different approaches, use cases, etc. Instead, its approach is "less is more," helping you to start writing Internet of Things applications with minimal hassle.

The Platforms

The *.NET Micro Framework* (NETMF) provides Internet connectivity, is simple and open source (Apache license), has hardware available from several vendors, and benefits from the huge .NET ecosystem and available know-how. Also, you can choose between Visual Studio (including the free Express Edition) on Windows, and the open source Mono toolchain on Linux and Mac OS X.

There is an active community for NETMF at *http://www.netmf.com/Home.aspx*. The project itself is hosted at *http://netmf.codeplex.com/*.

Netduino Plus (*http://www.netduino.com/netduinoplus*) is an inexpensive NETMF board from *Secret Labs* (*http://www.secretlabs.com*). This board makes Ethernet networking available with a price tag of less than $60. It has the following characteristics:

- A 48 MHz Atmel SAM7 microcontroller with 128 KB RAM and 512 KB Flash memory
- USB, Ethernet, and 20 digital I/O pins (six of which can be configured optionally for analog input)
- Micro SD card support
- Onboard LED and pushbutton
- Form factor of the Arduino (*http://www.arduino.cc/*); many Arduino *shields* (add-on boards) can be used
- .NET Micro Framework preprogrammed into Flash memory
- All software and hardware is open source

There is an active community for the Netduino Plus (and NETMF) at *http://forums.netduino.com/*. All the examples in this book use the Netduino Plus.

How This Book Is Organized

The book consists of three parts:

- Part I, Introduction

  The first part tells you how to set up the development environment and write and run a "Hello World" program. It shows how to write to output ports (for triggering so-called *actuators* such as LED lights or motors) and how to read from input ports (for *sensors*). It then introduces the most essential concepts of the Internet of Things: HTTP and the division of labor between clients and servers. In the Internet of Things, devices are programmed as clients if you want them to push sensor data to some service; they are programmed as servers if you want to enable remote control of the device over the Web.

» Part II, Device as HTTP Client

The second part focuses on examples that send HTTP requests to some services—e.g., to push new sensor measurements to the Pachube service (*http://www.pachube.com*) for storage and presentation.

» Part III, Device as HTTP Server

The third part focuses on examples that handle incoming HTTP requests. Such a request may return a fresh measurement from a sensor, or may trigger an actuator. A suitable server-side library is provided in order to make it easier than ever to program a small device as a server.

» Appendix A, Test Server

This contains a simple test server that comes in handy for testing and debugging client programs.

» Appendix B, .NET Classes Used in the Examples

This shows the .NET classes that are needed to implement all examples, and the namespaces and assemblies that contain them.

» Appendix C, Gsiot.Server Library

This summarizes the interface of the helper library `Gsiot.Server` that we use in Part III.

Who This Book Is For

This book is intended for anyone with at least basic programming skills in an object-oriented language, as well as an interest in sensors, microcontrollers, and web technologies. The book's target audience consists of the following groups:

» Artists and designers

You need a prototyping platform that supports Internet connectivity, either to create applications made up of multiple communicating devices, or to integrate the World Wide Web into a project in some way. You want to

turn your ideas into reality quickly, and you value tools that help you get the job done. Perhaps you have experience with the popular 8-bit Arduino platform (*http://www.arduino.cc/*), and might even be able to reuse some of your add-on hardware (such as shields and *breakout boards*) originally designed for Arduino.

» Students and hobbyists

You want your programs to interact with the physical world, using mainstream tools. You are interested in development boards, such as the Netduino Plus, that do not cost an arm and a leg.

» Software developers or their managers

You need to integrate embedded devices with web services and want to learn the basics quickly. You want to build up an intuition that ranges from overall system architecture to real code. Depending on your prior platform investments, you may be able to use the examples in this book as a starting point for feasibility studies, prototyping, or product development. If you already know .NET, C#, and Visual Studio, you can use the same programming language and tools that you are already familiar with, including the Visual Studio debugger.

To remain flexible, you want to choose between different boards from different vendors, allowing you to move from inexpensive prototypes to final products without having to change the software platform. To further increase vendor independence, you probably want to use open source platforms, both for hardware and software. To minimize costs, you are interested in a platform that does not require the payment of target royalties, i.e., per-device license costs.

If your background is in the programming of PCs or even more powerful computers, a fair warning: embedded programming for low-cost devices means working with very limited resources. This is in shocking contrast with the World Wide Web, where technologies usually seem to be created with utmost inefficiency as a goal. Embedded programming requires more careful consideration of how resources are used than what is needed for PCs or servers. Embedded platforms only provide small subsets of the functionality of their larger cousins, which may require some inventiveness and work where a desired feature is not available directly. This can be painful if you feel at home with "the more, the better," but it will be fun and rewarding if you see the allure of "small is beautiful."

What You Need to Get Started

This book focuses on the interaction between embedded devices and other computers on the Internet, using standard web protocols. Its examples mostly use basic sensors and actuators, so it is unnecessary to buy much additional hardware besides an inexpensive computer board. Here is a list of things you need to run all the examples in this book:

- A Netduino Plus board (*http://www.netduino.com/netduinoplus*)
- A micro USB cable (normal male USB-A plug on PC side, male micro USB-B plug on Netduino Plus side), to be used during development and for supplying power
- An Ethernet router with one Ethernet port available for your Netduino Plus
- An Internet connection to your Ethernet router
- An Ethernet cable for the communication between Netduino Plus and the Ethernet router
- A potentiometer with a resistance of about 100 kilohm and through-hole connectors
- A Windows XP/Vista/7 PC, 32 bit or 64 bit, for the free Visual Studio Express 2010 development environment (alternatively, you may use Windows in a virtual machine on Mac OS X or Linux, or you may use the Mono toolchain on Linux or Mac OS X)

NOTE: There are several sources where you can buy the hardware components mentioned above, assuming you already have a router with an Internet connection:

- Maker SHED (*http://www.makershed.com/*)
  - Netduino Plus, part number MKND02
  - Potentiometer, part number JM2118791
- SparkFun (*http://www.sparkfun.com/*)
  - Netduino Plus, part number DEV-10186

- Micro USB cable, part number CAB-10215 (included with Netduinos for a limited time)
- Ethernet cable, part number CAB-08916
- Potentiometer, part number COM-09806

For more sources in the U.S. and in other world regions, please see *http://www.netduino.com/buy/?pn=netduinoplus*.

It is also possible to add further sensors and actuators.

Conventions Used in This Book

The following typographical conventions are used in this book:

- *Italic*

  Indicates new terms, URLs, email addresses, filenames, and file extensions.

- `Constant width`

  Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, data types, statements, and keywords.

- ***`Constant width bold`***

  Shows commands or other text that should be typed literally by the user.

- *`Constant width italic`*

  Shows text that should be replaced with user-supplied values or by values determined by context.

NOTE: This style signifies a tip, suggestion, or general note.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Getting Started with the Internet of Things*, by Cuno Pfister. Copyright 2011 Cuno Pfister, 978-1-4493-9357-1."

If you feel your use of code examples falls outside fair use or the permission given here, feel free to contact us at *permissions@oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

http://oreilly.com/catalog/0636920013037

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our website at:

http://oreilly.com

I/Introduction

Thanks to the unrelenting progress of the semiconductor industry, all the digital parts of a computer can be put onto a single chip, called a *microcontroller*. A 32-bit microcontroller chip costing less than $10 may have more than twice as much memory as the original 8-bit Apple II computer with its 48 KB of RAM, and may run 100 times faster. A hobbyist board that incorporates such a chip, along with Ethernet and a Micro SD card slot, can be purchased for about $60.

Because of such inexpensive hardware and easy-to-use development platforms, it is now possible for hobbyists to create systems that interact with the physical world in every conceivable way. For example, a sensor can measure the humidity in a flowerpot, and a computer-controlled valve (actuator) lets water pass into the pot when the humidity drops too low.

Moreover, since the hardware allows the use of standard Internet protocols, monitoring and controlling can be done over the Internet. Various Internet services can be used for storing data, visualizing it, sharing it with other people, etc. For example, to learn about seasonal effects on humidity, you can store measurements of your flowerpot's humidity over the course of a year.

While these possibilities are fascinating and promising, there is also something creepy about the potential for devices to spy on our every move. This provides another reason why we should try to learn how such systems work. This understanding is, or at least ought to be, the basis for thinking about privacy policies that will become necessary sooner or later.

In Part I, I will show you how to set up the development environment so that you can start playing with simple sensors and actuators. Then I will lay the groundwork for Parts II and III, which show how you can program devices as clients that send requests to various services, or as servers that handle requests from clients, e.g., from web browsers.