Real World Instrumentation with Python

Real World Instrumentation with Python

J. M. Hughes

O'REILLY®
Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Real World Instrumentation with Python
by J. M. Hughes

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*.

Editor: Julie Steele
Production Editor: Adam Zaremba
Copyeditor: Rachel Head
Proofreader: Sada Preisch

Indexer: John Bickelhaupt
Cover Designer: Karen Montgomery
Interior Designer: David Futato
Illustrator: J. M. Hughes and Robert Romano

Printing History:
November 2010: First Edition.

This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN: 978-0-596-80956-0

[M]

1289573686

Table of Contents

Preface

This is a book about *automated instrumentation*, and the automated control systems used with automated instrumentation. We will look at how to use the Python programming language to quickly and easily implement automated instrumentation and control systems.

Automated instrumentation can be found in a wide variety of settings, ranging from research laboratories to industrial plants. As soon as people realized that collecting data over time was a useful endeavor, they also realized that they needed some way to capture and record the data. Of course, one could sit with a clock and a pad of paper, staring at thermometers, dials, and gauges, and write down numbers or other information every few minutes or so, but that gets tedious rather quickly. It's much easier—and more reliable—if the process can be automated. Fortunately, technology has advanced significantly since the days of handwritten logbooks and clockwork-driven strip chart recorders.

Nowadays, one can purchase inexpensive instrumentation for a wide variety of physical phenomena and use a computer to capture the data. Once a computer is connected to instrumentation, the possibilities for data collection, analysis, and control begin to expand in all directions, with the only real limitations being the ability to implement the necessary software and the implementer's creativity.

The primary objective of this book is to show you how to create software that you can use to get a capable and user-friendly instrumentation or control application up and running with a minimum of hassle. To this end, we will work through the steps necessary to create applications that incorporate low-level interfaces to the real world via various types of input/output hardware. We will also examine some proven methods for creating programs that are robust and reliable. Special attention will be paid to designing the algorithms necessary to acquire and process the data. Finally, we will see how to display the results to a user and accept command inputs. It is my desire that you will find ideas here that you might take away and creatively apply to meet your own needs in a wide variety of settings.

Who Is This Book For?

This is a hands-on text intended for people who want or need to implement instrumentation systems, also known as *data acquisition and control systems*. You might be a researcher, a software developer, a student, a project lead, an engineer, or a hobbyist. The application might be an automated electronics test system, an analysis process in a laboratory, or some other type of automated instrumentation.

One of the objectives with the software in this book is that it be as platform-independent as possible. I am going to assume that you are comfortable with at least the Windows platform, and Windows XP in particular. With Linux I'll be referring to the Ubuntu distribution, but the discussion should apply to any recent Linux distribution and I will assume that you know how to use either the *csh* or *bash* command-line shells.

Since this is a book about interfacing to the real world via physical hardware, some electronics are involved, but I am not going to assume that you have an extensive background in electrical engineering. Chapter 2 contains an overview of the basics of electronics theory as it relates to instrumentation, for those who might benefit from it. It turns out that it really doesn't take a deep level of electronics knowledge to successfully interface a computer with the physical world. But, as with anything else involving technology, it never hurts to know as much as possible, just on the off chance that things don't quite work out as expected the first time.

Regardless of the type of work you do, or where you do it, the main thing I am assuming that we have in common is a need to capture some data, and perhaps to generate control signals, and to do so through some kind of computer interface. Most importantly, we need the instrumentation and control software we create to be accurate, reliable, and relatively painless to implement.

The Programming Languages

The primary programming language we will use is Python, with a bit of C thrown in. Throughout the book, I will assume that you have some programming experience and are familiar with either Python or C (ideally, both). If that is not the case, experience with Perl or Tcl/Tk or analysis tools such as MatLab or IDL is also a reasonable starting point.

This book explicitly avoids the more esoteric aspects of the Python language, and the examples are profusely documented with comments in the code, diagrams, and screen captures where appropriate. The amount of C involved is minimal; it is used only to illustrate how to create and use low-level extensions for Python applications. Chapter 3 covers the basics of Python, and Chapter 4 provides a summary of the essentials of the C language. Some suggestions for further reading are also provided for those who wish to go deeper into either (or both) of these languages.

Why Python?

Python is an interpreted language developed by Guido van Rossum in the late 1980s. Because of its interpreted nature, there is no compilation step to deal with, and the user can create and execute programs directly from Python's command line. The language itself is also easy to learn and comprehend, so long as one initially avoids the more advanced features (generators, introspection, list comprehension, and such). Thus, Python offers the dual benefits of rapid prototyping and ease of comprehension, which in turn allows for the quick creation of sophisticated tools for a diverse range of instrumentation applications, without the development burdens and learning curve normally associated with conventional compiled languages or a vendor-specific programming environment.

Python is highly portable, and it is available for almost every modern computing platform. So long as a project sticks to using commonly available interface methods, an application written initially on a PC running Windows will most likely work without change on a machine running Linux. The odds are good that the application will also run on a Sun Solaris machine or an Apple OS X system, although these systems are not specifically covered in this text. It is only when Python is used in conjunction with platform-specific extensions or drivers that it loses its portability, so in these cases I will offer alternatives for both Windows and Linux wherever feasible.

The text includes example code snippets, block diagrams and flow charts to illustrate key points, and some complete examples utilizing readily available and low-cost interface hardware.

The Systems

The types of instrumentation systems we will examine might be utilized for laboratory research, or they might be used in industrial settings. An instrumentation system might be used in an electronics lab, in a wind tunnel, or to collect meteorological data. The systems may be as simple as a temperature data logger or as complex as a thermal vacuum chamber control system.

Generally, just about anything that can be interfaced to a PC is a potential candidate for the techniques described in this book. There are, of course, some devices with closed proprietary interfaces, but I will not address those, nor will I delve into complex data collection and process control scenarios such as oil refineries, nuclear power plants, or robotic spacecraft. Systems in those domains are usually best served with sophisticated and complex custom control hardware, and equally sophisticated and complex software. I will focus instead on those instruments, devices, and systems that can be easily programmed using any of a number of common interface methods.

Methodology

Using a step-by-step approach and real-world examples, we will examine the processes necessary to define the instrumentation application, select the appropriate interfaces and hardware, and create the low-level extension modules needed (if any) to interface Python with instrumentation hardware. We will also investigate the use of TkInter, wxPython, and `curses` for graphical and text-based user interfaces.

The book includes sections describing what is involved in writing an extension for Python in order to encapsulate a hardware vendor's DLL; how to communicate with USB-based I/O devices; and how to use industry-standard interfaces such as RS-232, RS-485, and GPIB, along with a survey of what types of hardware one might expect to find using these interfaces. It also provides references to readily available open source tools and libraries to reduce, as much as possible, the amount of time spent implementing functionality from scratch.

How This Book Is Organized

This book is organized into 14 chapters and 2 appendixes. The first 12 chapters set the stage for the implementation examples described in Chapter 14. Chapters 1 through 6 introduce basic concepts that the advanced reader may elect to skip over. Here's a closer look at what you'll find in each chapter:

Chapter 1, *Introduction to Instrumentation*
: Chapter 1 provides an overview of what instrumentation is, how control systems work, and how these concepts are used in the real world. The examples covered include automatic outdoor lights, test instrumentation in an electronics engineering environment, control of a thermal chamber in a laboratory, and batch chemical processing.

Chapter 2, *Essential Electronics*
: Because this is a hands-on book, we will need to know something about the physical hardware we want to interface to and have at least a general idea of how it works. This chapter starts off with an introduction to the basic concepts of electricity and electronics. It then explores the functional building blocks for data acquisition and control, including discrete digital interfaces, analog interfaces, and counters and timers. Lastly, it reviews the basic concepts behind serial and parallel interfaces. If you are already familiar with electric circuit theory and devices, you could skip this chapter. However, I would recommend that you still at least skim through the material, on the off chance that there might be something unique here that you can make use of later.

Chapter 3, *The Python Programming Language*
: Although this book is not a tutorial on Python, this chapter provides an introduction to the core concepts of Python and summarizes the basics of the language. The primary emphasis is on the features of Python that will be used frequently in this

book. This chapter also provides a brief overview of the tools available to make life easier for the person doing the programming, and where to go about finding them.

Chapter 4, *The C Programming Language*
: Here, the C programming language is introduced in a high-level overview. The objective is to provide enough information to enable you to understand the examples in this book, without delving into the arcane details. Fortunately, C is a relatively simple language, and the information in this chapter should be sufficient to get you started on creating your own extensions for Python.

Chapter 5, *Python Extensions*
: This chapter describes how a Python extension is created, and what extensions are typically used for. Examples are provided, both in this chapter and in later chapters, for you to use as templates for your own efforts.

Chapter 6, *Hardware: Tools and Supplies*
: Although is it possible that one could implement an instrumentation system and never touch a soldering iron, there is a high probability that some screwdrivers, wire cutters, and a digital multimeter (DMM) will come in handy. In this chapter I provide a list of what I would consider to be a basic toolkit for doing instrumentation work. It isn't much and could all easily fit in a small box on a shelf somewhere. However, there could very well come a time when you really need to see what's going on in your system. To this end, I've included a discussion of the two pieces of test equipment that can help you eliminate the guesswork and quickly get to the root of an interface or control problem: the oscilloscope and the logic analyzer. This chapter also covers what types of instruments are available and provides some suggestions for deciding between buying new equipment or picking up something used.

Chapter 7, *Physical Interfaces*
: Chapter 7 examines the types of interfaces one is most likely to encounter when attempting to interface Python to data acquisition or control instrumentation. RS-232 and RS-485, the two most commonly encountered types of serial interfaces, are examined from an instrument interface perspective. This chapter also covers the basics of USB and GPIB/IEEE-488 interfaces, along with a discussion of where one might expect to encounter them. Finally, we turn our attention to I/O hardware designed to be plugged into the bus of a PC, typically PCI-type circuit boards, and what one can typically expect in terms of API support from the hardware vendor.

Chapter 8, *Getting Started*
: This chapter contains a description of a proven approach to software development. It is included here because, when implementing an instrument system in any language, it is essential to plan and define what is to be implemented, and then to test the result against the expectations captured in a set of requirements. By extending the reach of Python into the real world, we open the door for the uncertainties and vagueness of the real world to wander back in and impact—sometimes severely—the instrumentation software.

Chapter 9, *Control System Concepts*
: A book on real-world data acquisition and control would be incomplete without a discussion of control systems and the theory behind them. Chapter 9 expands on the concepts introduced in Chapter 1 with detailed examinations of common control system concepts and models, including topics such as feedback, "bang-bang" controllers, and Proportional-Integral-Derivative (PID) controls. It also provides an introduction to basic control system analysis and provides some guidelines for choosing an appropriate model. Lastly, we'll look at how the mathematics of control systems translates into actual Python code.

Chapter 10, *Building and Using Simulators*
: Chapter 10 examines simulators and how they can be leveraged to speed up the development process, provide a safe environment in which to test out ideas, and provide some invaluable (and otherwise unattainable) insights into the behavior of not only the instrumentation software, but also the device or system being simulated. Whether because the instrumentation hardware just isn't available yet or because the target system is too valuable to risk damaging, a simulation can be a quick and easy way to get the software running, test it, and have a high degree of confidence that it will work correctly in the real world.

Chapter 11, *Instrumentation Data I/O*
: In this chapter we'll look at how to use the interfaces that were introduced in Chapter 7 to move data between the real world and your applications. We'll start with a discussion of interface formats and protocols in order to define the basic concepts we will need for the upcoming software examples, and then we'll take a quick tour of some packages that are available for interface support in Python with the *pySerial*, *pyParallel*, and *PyVISA* packages. Lastly, I'll show you some techniques to read and write instrumentation data. We'll take a look at blocking versus nonblocking I/O, asynchronous input and output events, and how to manage potential data I/O errors to help make your applications more robust.

Chapter 12, *Reading and Writing Data Files*
: Chapter 12 examines some of the implementation considerations and techniques for saving instrumentation data in a variety of file formats, from plain ASCII and CSV files to binary files and databases. We'll also examine Python's configuration data file capabilities, and see how easy it is to store and retrieve configuration parameters using Python's library methods.

Chapter 13, *User Interfaces*
: Unless an application is deeply embedded or specifically designed to run as a background process, it will probably need some type of user interface. Chapter 13 examines what one can do with just the command line and the `curses` screen control package for Python, and how to use an ANSI-capable terminal emulator program to display data and accept input. The chapter wraps up with a look at the TkInter GUI toolkit provided with the standard Python distribution, and also provides an overview of the wxPython GUI package.

Chapter 14, *Real World Examples*
: In Chapter 14 we look at several different types of devices used for data acquisition and control applications. This chapter starts with an example of capturing the continuous data output from a digital multimeter. We then examine a common type of data acquisition device that uses a serial interface for command and data exchanges. Lastly, we wrap up with a detailed look at a data I/O device with a USB interface and its associated API DLL provided by the vendor. The selected devices illustrate key concepts shared by almost all instrumentation components, and the examples draw on earlier chapters to show how the theory is put into practice.

Two appendixes provide additional useful information:

Appendix A, *Free and Open Source Software Resources*
Appendix B, *Instrument Sources*

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 Python modules and 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 values determined by context

This icon signifies a tip, suggestion, or general note.

This icon indicates a warning or caution.

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: "*Real World Instrumentation with Python* by J. M. Hughes. Copyright 2011 John M. Hughes, 978-0-596-80956-0."

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

Safari® Books Online

Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at *http://my.safaribooksonline.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://www.oreilly.com/catalog/9780596809560/

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://www.oreilly.com

Acknowledgments

I would like to acknowledge some of the people who helped make this book possible: my wife, Carol, and daughter, Seren, for their patience and understanding when I needed to disappear into my office for extended periods; my friend and co-worker Michael North-Morris for his perpetual optimism; my acquisition editor, Julie Steele, for her willingness to take a chance and extend to me the opportunity to write for O'Reilly; Rachel Head, the diligent copyeditor, for catching my abuses of the English language; and all of the helpful and friendly staff at O'Reilly.

I would also like to thank the folks at LabJack Corporation for providing me with real hardware to work with and graciously offering their time and support to help make sure I got it working correctly. Thanks also to Janet Smith of Agilent for providing me with high-quality photographs of some of their products.

—John Hughes
Tucson, Arizona, 2010