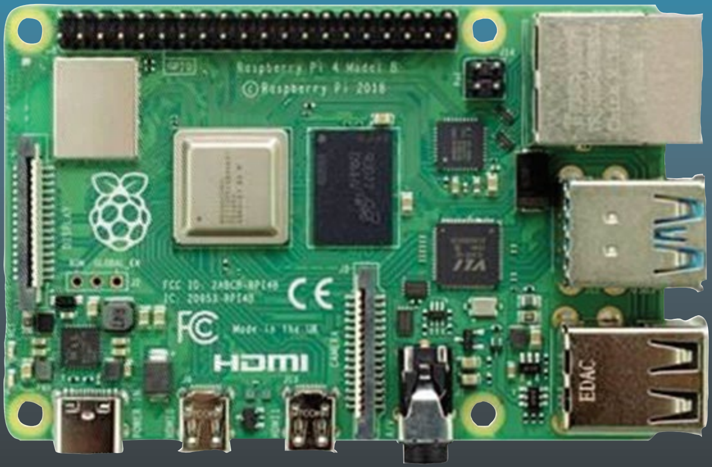
TECHNOLOGY IN ACTION™

Now with Python 3



Raspberry Pi IoT Projects

Prototyping Experiments for Makers

—

Second Edition

—

John Shovic, PhD

Apress®

Raspberry Pi IoT Projects

Prototyping Experiments for Makers

Second Edition

John C. Shovic

Apress®

Raspberry Pi IoT Projects: Prototyping Experiments for Makers

John C. Shovic
Spokane Valley, WA, USA

ISBN-13 (pbk): 978-1-4842-6910-7 ISBN-13 (electronic): 978-1-4842-6911-4
https://doi.org/10.1007/978-1-4842-6911-4

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Aaron Black
Development Editor: James Markham
Coordinating Editor: Jessica Vakili

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 NY Plaza, New York, NY 10014. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-6910-7. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To my best friend Laurie and to my students that inspire me every day.

Table of Contents

About the Author

Dr. John C. Shovic is currently a Professor of Computer Science at the University of Idaho specializing in AI and robotics. He is also Chief Technical Officer of SwitchDoc Labs, a company specializing in technical products for the Maker Movement and the IoT. He was also Chief Technology Strategist at Stratus Global Partners with a focus on supplying expertise in computer security regulatory and technical areas to health-care providers. He has worked in the industry for over 30 years and has founded seven companies: Advanced Hardware Architectures, TriGeo Network Security, Blue Water Technologies, MiloCreek, InstiComm, SwitchDoc Labs, and bankCDA. As a founding member of the bankCDA board of directors, he currently serves as the Chairman of the Technology Committee. He has also served as a Professor of Computer Science at Eastern Washington University and Washington State University. Dr. Shovic has given over 80 invited talks and has published over 70 papers on a variety of topics on Arduinos/Raspberry Pis, HIPAA, GLB, computer security, computer forensics, robotics, AI, and embedded systems.

About the Technical Reviewer

Massimo Nardone has more than 22 years of experience in security, web/mobile development, cloud, and IT architecture. His true IT passions are security and Android.

He has been programming and teaching how to program with Android, Perl, PHP, Java, VB, Python, C/C++, and MySQL for more than 20 years.

He holds a Master of Science degree in Computing Science from the University of Salerno, Italy.

He has worked as a Project Manager, Software Engineer, Research Engineer, Chief Security Architect, Information Security Manager, PCI/SCADA Auditor, and Senior Lead IT Security/Cloud/SCADA Architect for many years.

Acknowledgments

I would like to acknowledge the hard work of the Apress editorial team in putting this book together. I would also like to acknowledge the hard work of the Raspberry Pi Foundation and the Arduino group for putting together products and communities that help to make the Internet of Things more accessible to the general public. Hurray for the democratization of technology! Of course, I have to mention my grandchildren (Lincoln, Hazel, Makenna and Madelyn) that I am constantly building projects to entertain and education them.

Introduction

The Internet of Things (IoT) is a complex concept made up of many computers and many communication paths. Some IoT devices are connected to the Internet and some are not. Some IoT devices form swarms that communicate among themselves. Some are designed for a single purpose, while some are more general-purpose computers. This book is designed to show you the IoT from the inside out. By building IoT devices, the reader will understand the basic concepts and will be able to innovate using the basics to create their own IoT applications.

These included projects will show the reader how to build their own IoT projects and to expand upon the examples shown. The importance of computer security in IoT devices is also discussed as well as various techniques for keeping the IoT safe from unauthorized users or hackers. The most important takeaway from this book is in building the projects yourself.

Chapters at a Glance

In this book, we build examples of all the major parts of simple and complex IoT devices.

In Chapter 1, the basic concepts of IoT are explained in basic terms, and you will learn what parts and tools are needed to start prototyping your own IoT devices.

In Chapter 2, you'll learn how to sense the environment with electronics and that even the behavior of simple LightSwarm type of devices can be very unpredictable.

Chapter 3 introduces important concepts about how to build real systems that can respond to power issues and programming errors by the use of good system design and watchdogs.

Chapter 4 turns a Raspberry Pi into a battery-powered device that senses iBeacons and controls the lighting in a house while reporting your location to a server.

In Chapter 5, you'll do IoT the way the big boys do by connecting to the IBM Bluemix IoT server and sending your biometric pulse rates for storage and display.

In Chapter 6, we'll build a small RFID inventory system and use standard protocols like MQTT to send information to a Raspberry Pi, a complete IoT product.

Chapter 7 shows the dark side of the IoT, computer security. The way you protect your IoT device from hackers and network problems is the most difficult part of IoT device and system design.

Are you totally secure? You will never know. Plan for it.

The reference appendix provides resources for further study and suggestions for other projects.

CHAPTER 1

Introduction to IoT

Chapter Goal: Understand What the IoT Is and How to Prototype IoT Devices

Topics Covered in This Chapter:

- What is IoT
- Choosing a Raspberry Pi Model
- Choosing your IoT device
- Characterization of IoT devices
- Buying the right tools to deal with hardware
- Writing code in Python and in the Arduino IDE

The IoT is a name for the vast collection of "things" that are being networked together in the home and workplace (up to 20 billion by 2022 according to Gardner, a technology consulting firm). That is a very vast collection. And they may be underestimating it.

We all have large numbers of computers in a modern house. I just did a walk-through of my house, ignoring my office (which is filled with about another 100 computers). I found 65 different devices having embedded computers. I'm sure I missed some of them. Now of those computer-based devices, I counted 20 of them that have IP addresses, although I know that I am missing a few (such as the thermostat). So in a real sense, this house has 20 IoT devices. And it is only 2020 as of the writing of this book. With over 100 million households in the United States alone, 20 billion IoT devices somehow don't seem so many.

So what are the three defining characteristics of the IoT?

- Networking – These IoT devices talk to one another (M2M communication) or to servers located in the local network or on the Internet. Being on the network allows the device the common ability to consume and produce data.
- Sensing – IoT devices sense something about their environment.
- Actuators – IoT devices that do something, for example, lock doors, beep, turn lights on, or turn the TV on.

Of course, not every IoT device will have all three, but these are the characteristics of what we will find out there.

Is the IoT valuable? Will it make a difference? Nobody is sure what the killer application will be, but people are betting huge sums of money that there will be a killer application. Reading this book and doing the projects will teach you a lot about the technology and enable you to build your own IoT applications.

Choosing a Raspberry Pi Model

The Raspberry Pi family of single-board computers (see Figure 1-1) is a product of the Raspberry Pi Foundation (RaspberryPi.org). They have sold over 9 million of these small, inexpensive computers. The Raspberry Pi runs a number of different operating systems, the most common of which is the Raspbian release of Ubuntu Linux.

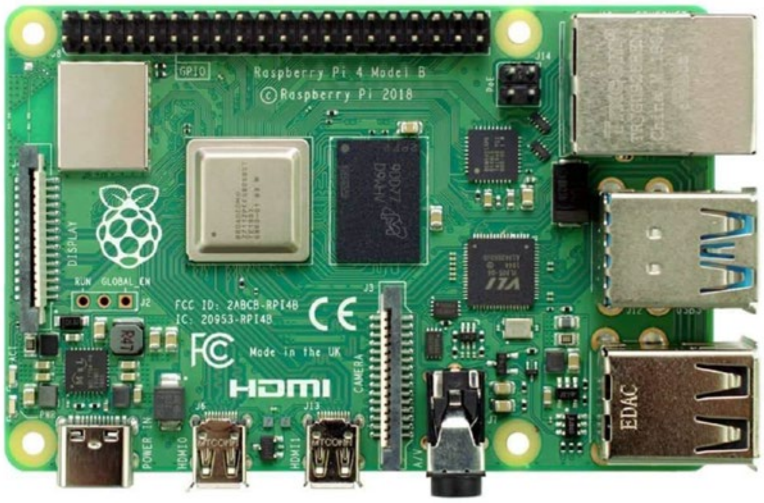


Figure 1-1. *Raspberry Pi 4*

Like Windows, Linux is a multitasking operating system, but unlike Windows, it is an open source system. You can get all the source code and compile it if you wish, but I would not recommend that to a beginner.

One of the best parts of the Raspberry Pi is that there are a huge number of device and sensor drivers available, which makes it a good choice for building IoT projects, especially using it as a server for your IoT project. The Raspberry Pi is not a low-power device, which limits its usage as an IoT device. However, it is still a great prototyping device and a great server.

There is a rather bewildering variety of Raspberry Pi boards available. I suggest for this book that you get a Raspberry PI 3B+ or Raspberry Pi 4. While the $10 Raspberry Pi Zero W is tempting, it takes quite a bit of other hardware to get it to the point where it is usable. While the Raspberry Pi 4 is more expensive ($35), it comes with a WiFi interface built in and extra USB ports.

Note that we are using the Raspberry Pi 4 for building the IoT SkyWeather Weather Station later in this book. The reason for that is computer power! We are doing a lot with the CPU in that project, using cameras and decoding 433MHz signals!

There are many great tutorials on the Web for setting up your Raspberry Pi and getting the operating system software running.

Choosing an IoT Device

If you think the list of Raspberry Pi boards available is bewildering, then wait until you look at the number of IoT devices that are available. While each offering is interesting and has unique features, I am suggesting the following devices for your first projects in the IoT. Note that I selected these based upon the ability to customize the software and to add your own devices without hiding *all* the complexity, hence reducing the learning involved. That is why I am not using Lego-type devices in this book.

We will be using the following:

- ESP8266-based boards (specifically the Adafruit Huzzah ESP8266)
- Arduino Uno and Arduino Mega2560 boards

Characterizing an IoT Project

When looking at a new project, the first thing to do to understand an IoT project is to look at the six different aspects for characterizing an IoT project:

- Communications
- Processor power
- Local storage
- Power consumption
- Functionality
- Cost

When I think about these characteristics, I like to rate each one on a scale from 1 to 10, 1 being the least suitable for IoT and 10 being the most suitable for IoT applications. Scoring each one forces me to think carefully about how a given project falls on the spectrum of suitability.

Communications

Communications are important to IoT projects. In fact, communications are core to the whole genre. There is a trade-off for IoT devices. The more complex the protocols and higher the data rates, the more powerful processor you need and the more electrical power the IoT device will consume.

TCP/IP based communications (think web servers; HTTP-based communication (like REST servers); streams of data; UDP – see Chapter 2) provide the most flexibility and functionality at a cost of processor and electrical power.

Low-power Bluetooth and Zigbee types of connections allow much lower power for connections with the corresponding decrease in bandwidth and functionality.

IoT projects can be all over the map with requirements for communication flexibility and data bandwidth requirements.

IoT devices having full TCP/IP support are rated the highest in this category, but will probably be marked down in other categories (such as power consumption).

Processor Power

There are a number of different ways of gauging processor power. Processor speed, processor instruction size, and operating system all play in this calculation. For most IoT sensor and device applications, you will not be limited by processor speed as they are all pretty fast. However, there is one exception to this. If you are using encryption and decryption techniques (see Chapter 7), then those operations are computationally expensive and require more processor power to run. The trade-off can be that you have to transmit or receive data much more slowly because of the computational requirements of encrypting/decrypting the data. However, for many IoT projects, this is just fine.

Higher processor power gives you the highest ratings in this category.

Local Storage

Local storage refers to all three of the main types of storage: RAM, EEPROM, and Flash Memory.

RAM (Random Access Memory) is high-data rate, read/writable memory, generally used for data and stack storage during execution of the IoT program. EEPROM (Electrically Erasable Programmable Read-Only Memory) is used for writing small amounts of configuration information for the IoT device to be read on power up. Flash Memory is generally used for the program code itself. Flash is randomly readable (e.g., as the code executes), but can only be written in large blocks and very slowly. Flash is what you are putting your code into with the Arduino IDE (see Chapter 2).

The amount of local storage (especially RAM) will add cost to your IoT device. For prototyping, the more, the merrier. For deployment, less is better as it will reduce your cost.

Power Consumption

Power consumption is the bane of all IoT devices. If you are not plugging your IoT device in the wall, then you are running off of batteries or solar cells and every single milliwatt counts in your design. Reducing power consumption is a complex topic that is well beyond the introductory projects in this book. However, the concepts are well understood by the following:

- Put your processor in sleep mode as much as possible.
- Minimize communication outside of your device.
- Try to be interrupt driven and not polling driven.
- Scour your design looking for every unnecessary amount of current.

The higher the number in this category, the less power the IoT unit uses.

Functionality

This is kind of a catch-all category that is quite subjective. For example, having additional GPIOs (General-Purpose Input/Outputs) available is great for flexibility. I am continuously running into GPIO limitations with the Adafruit Huzzah ESP8266 as there are so few pins available. Having additional serial interfaces is very useful for debugging. Special hardware support for encryption and decryption can make device computer security much simpler. One of the things that I miss in most IoT prototyping system is software debugging support hardware.

I also include the availability of software libraries for a platform into this category. A ten means very high functionality; low numbers mean limited functionality.

Cost

What is an acceptable cost for your IoT device? That depends on the value of the device and the market for your device. A $2.50 price can be great for prototypes, but will be the death of the product in production. You need to size the price to the product and the market. High numbers are low-cost units, and low numbers are higher-cost devices.

The Right Tools to Deal with Hardware

Anything is more difficult without the right tools. When you make the jump from just doing software to doing a software/hardware mix, here is a list of tools you should have:

- 30W adjustable temperature soldering iron – Heating and connecting wires
- Soldering stand – To hold the hot soldering iron
- Solder, rosin-core, 0.031” diameter, 1/4 lb (100g) spool – To solder with
- Solder sucker – Useful in cleaning up mistakes
- Solder wick/braid 5 ft spool – Used along with the solder sucker to clean up soldering messes
- Panavise Jr. – General-purpose 360-degree mini-vise
- Digital multimeter – A good-all-around basic multimeter
- Diagonal cutters – Trimming of wires and leads
- Wire strippers – Tool for taking insulation off wires
- Micro needle-nose pliers – For bending and forming components
- Solid-core wire, 22AWG, 25 ft spools – Black, red, and yellow for bread-boarding and wiring

Adafruit has an excellent beginner’s kit for $100 [www.adafruit.com/products/136]. Figure 1-2 shows the tools that are in it.

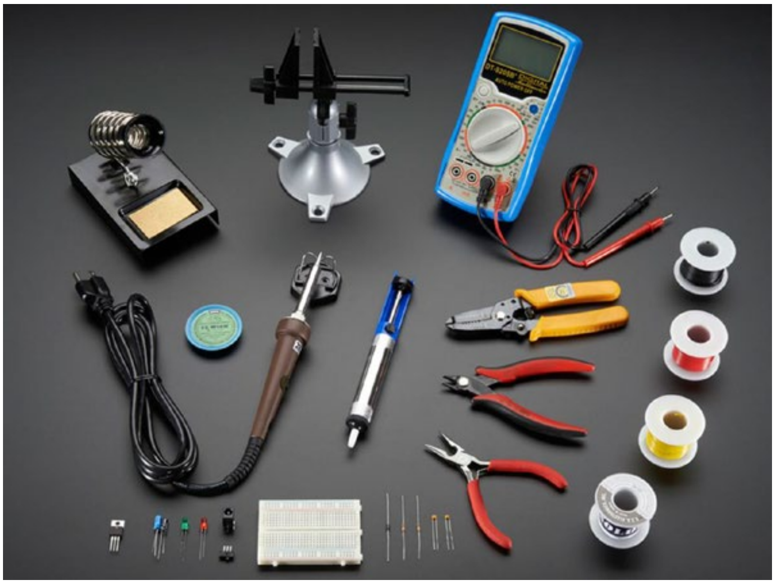


Figure 1-2. *Adafruit Electronics Toolkit*

Writing Code in Python and the Arduino IDE

All the code in this book is in two languages. Specifically, Python is used for the Raspberry Pi and C/C++ (don't be scared, there are many examples and resources) for the Arduino IDE.

Python is a high-level, general-purpose programming language. It is designed to emphasize code readability, and it especially keeps you out of having loose pointers (a curse of all C/C++ programmers) and does the memory management for you. This is the programming language of choice for the Raspberry Pi. Python has the largest set of libraries for IoT and embedded system devices of any language for the Raspberry Pi. All of the examples in this book that use the Raspberry Pi use Python. I am using Python 2.7 in this book, but it is relatively easy to convert to Python 3.5. However, it is not so trivial to find all the libraries for Python 3.5, so I suggest staying with Python 2.7.

Why are we using C++ for the majority of the IoT devices? There are four reasons for this:

- C programs are compiled into native code for these small devices, giving you much better control over size and timing. Python requires an interpreter, which is a large amount of code that would not fit on small IoT devices, such as the Arduino. On a Raspberry Pi, you may have a gigabyte (GB) of RAM and 8GB of SD card storage. On an IoT device, you might only have 2000 bytes (2K) and 32KB of code storage. That is a ratio of 500,000 to 1. That is why you need efficient code on IoT devices. Yes, there is MicroPython, but it is very limited and still uses more memory than most Arduino boards.
- When you program in C/C++, you are closer to the hardware and have better control of the timing of operations. This can be very important in some situations. One of the issues of Python is that of the memory garbage collector. Sometimes, your program will run out of memory and Python will invoke the garbage collector to clean up memory and set it up for reuse. This can cause your program to not execute in the time you expected. An interesting note is that the ESP8266 used in several chapters of this book also has a memory garbage collector, which can cause some issues in critical timing sequences. None of these problems are known to exist in the code used in this book. Keeping fingers crossed, however.
- Libraries, libraries, libraries. You can find Arduino C/C++ libraries for almost any device and application you can imagine for IoT applications. The Arduino

library itself is filled with large amounts of functionality, making it much easier to get your IoT application up and running.

- Finally, the Arduino IDE (Integrated Development Environment) is a good (but not great) environment for writing code for small devices. It has its quirks and some disadvantages. The number one disadvantage of the Arduino IDE is that it does not have a built-in debugger (although the good folks at Arduino have one in beta testing). Even with this significant disadvantage, it runs on Linux, Windows, and Mac, and we will use it in this book. The Arduino IDE is widely available, and there are many resources for learning and libraries designed for this. Other alternatives include Visual Micro (runs on Windows, built on Microsoft Visual Studio) and Eclipse (runs on Linux, Windows, and Mac). Eclipse can be a nightmare to set up and update, but they have made improvements in the past few years.

In This Book

What will we be doing in future chapters? We will be building real IoT projects that actually have a lot of functionality. Yes, it is fun to blink an LED, but it is only the first step to really doing interesting and useful things with all this new technology. Build an IoT Weather Station. Build an IoT LightSwarm. Build your own IoT device with your own sensors. It is all accessible and inexpensive and within your ability whether you are an engineer or not.

CHAPTER 2

Sensing Your IoT Environment

Chapter Goal: Build Your First IoT Device

Topics Covered in This Chapter:

- Building an inexpensive IoT sensor device based on the ESP8266 and Arduino IDE
- Setting up a simple self-organizing IoT sensor net
- Reading I2C sensor (light and color) on the Arduino devices
- Reading data from remote IoT sensors on the Raspberry Pi
- Using the Raspberry Pi for monitoring and debugging
- Displaying your data on the screen and on an iPad

In this chapter, we build our first IoT device. This simple design, a light-sensor swarm, is easy to build and illustrates a number of IoT principles including the following:

- Distributed control
- Self-organization
- Passing information to the Internet
- Swarm behavior

The LightSwarm architecture is a simple and flexible scheme for understanding the idea of an IoT project using many simple small computers and sensors with shared responsibility for control and reporting. Note that, in this swarm, communication with the Internet is handled by a Raspberry Pi. The swarm devices talk to each other, but not with the Internet. The Raspberry Pi is located on the same WiFi access point as the swarm, but could be located far away with some clever forwarding of packets through your WiFi router. In this case, since we have no computer security at all in this design (see Chapter 7 for information on making your IoT swarm and device more secure), we are sticking with the local network.

IoT Sensor Nets

One of the major uses of the IoT will be building nets and groups of sensors. Inexpensive sensing is just as big of a driver for the IoT as the development of inexpensive computers. The ability for a computer to sense its environment is the key to gathering information for analysis, action, or transmittal to the Internet. Sensor nets have been around in academia for many years, but now the dropping prices and availability of development tools are quickly moving sensor nets out to the mainstream. Whole industrial and academic conferences are now held on sensor nets [`www.sensornets.org`]. The market is exploding for these devices, and opportunities are huge for the creative person or group that can figure out how to make the sensor net that will truly drive consumer sales.

IoT Characterization of This Project

As I discussed in Chapter 1, the first thing to do to understand an IoT project is to look at our six different aspects of IoT. LightSwarm is a pretty simple project and can be broken down into the six components listed in Table 2-1.

***Table 2-1.** LightSwarm Characterization (CPLPFC)*

Aspect	Rating	Comments
Communications	9	WiFi connection to the Internet – can do ad hoc mesh-type communication
Processor power	7	80MHz XTensa Harvard Architecture CPU, ~80KB data RAM/~35KB of instruction RAM/200K ROM
Local storage	6	4MB Flash (or 3MB file system!)
Power consumption	7	~200mA transmitting, ~60mA receiving, no WiFi ~15mA, standby ~1mA
Functionality	7	Partial Arduino support (limited GPIO/analog inputs)
Cost	9	< $10 and getting cheaper

Ratings are from 1 to 10, 1 being the least suitable for IoT and 10 being the most suitable for IoT applications.

This gives us a CPLPFC rating of 7.2. This is calculated by averaging all six values together with equal weighting. This way is great for learning and experimenting and could be deployed for some applications.

The ESP8266 is an impressive device having a built-in WiFi connection, a powerful CPU, and quite a bit of RAM and access to the Arduino libraries. It is inexpensive and will get cheaper as time goes on. It is a powerful device for prototyping IoT applications requiring medium levels of functionality.

How Does This Device Hook Up to the IoT?

The ESP8266 provides a WiFi transmitter/receiver, a TCP/IP stack, and firmware to support direction connections to a local WiFi access point that then can connect to the Internet. In this project, the ESP8266 will only be talking to devices on the local wireless network. This is an amazing amount