The Pragmatic Programmers

Automate Your Home Using Go

Build a Personal Data Center with Raspberry Pi, Docker, Prometheus, and Grafana

Ricardo Gerardi and Mike Riley
edited by Jacquelyn Carter

This extract shows the online version of this title, and may contain features (such as hyperlinks and colors) that are not available in the print version.

For more information, or to purchase a paperback or ebook copy, please visit https://www.pragprog.com.

Welcome to the fascinating world of home automation. Using home automation, you can improve your home's energy efficiency and security. You can automate repetitive tasks that are often forgotten, like ensuring your garage door is closed at night. You can also control your home remotely, and increase your convenience and comfort.

While there are ready-to-use home automation solutions available in the market, the goal of this book is to teach you how to develop your own solutions. This is a perfect opportunity to practice your Go coding skills while developing something useful, tailored to your specific needs.

In this book, you'll use a small device, the Raspberry Pi, and the Go programming language to develop four home automation projects. These projects are fully functional and fun to build. We've tried to keep the source code examples relatively short; nevertheless, you'll apply different aspects of the Go language that allow you to build robust programs. By working on these projects, we expect that you'll have some immediate benefits, but we also expect that you'll learn more about the concepts and tools behind them, so you can expand and adapt them to your own needs.

Go[1] is a fast, statically typed, and compiled language that allows you to build efficient programs with relative ease. Go provides some of the flexibility provided by dynamic languages—such as Python—but the compilation process ensures that you build reliable software that runs efficiently, even on low-powered hardware such as the Raspberry Pi. Go is a versatile language that allows you to build robust software not only for large enterprise projects, but also for small applications, making it a great choice for home automation projects.

Nearly every modern home automation solution requires both hardware and software to make it fully functional. The projects presented in this book are no different. Sensors and controllers need to connect to a computer running code that knows how to interact with those devices. Fortunately, the hardware requirements that are used in this book's projects are inexpensive, and the software we'll use in the book is free, as in open source free. Without the contributions of these dedicated open source contributors, much of what is presented in this book would not be possible. A deep debt of gratitude to all those developers who have contributed their time and expertise into building great software so we can build great projects.

1. https://go.dev/

Your Personal Data Center

Before you can start coding the projects, you first need to set up the infrastructure that the Go code will rely upon. In other words, you'll build your own personal data center that will provide the foundation for running many Go-related projects, not just those related to home automation. Your code will rely on these data center components that help send alerts or notifications when thresholds are exceeded, and/or when actions are taken.

Many of the personal data center services of this infrastructure are already written in Go, and are the same ones used in large data centers. These services are ideal because they use industry-standard protocols, granting you a lot of flexibility when building the projects in the book as well as your own future projects. And since these servers and projects are written in Go, your projects will be able to scale to a high degree of activity as an extra benefit of using the language.

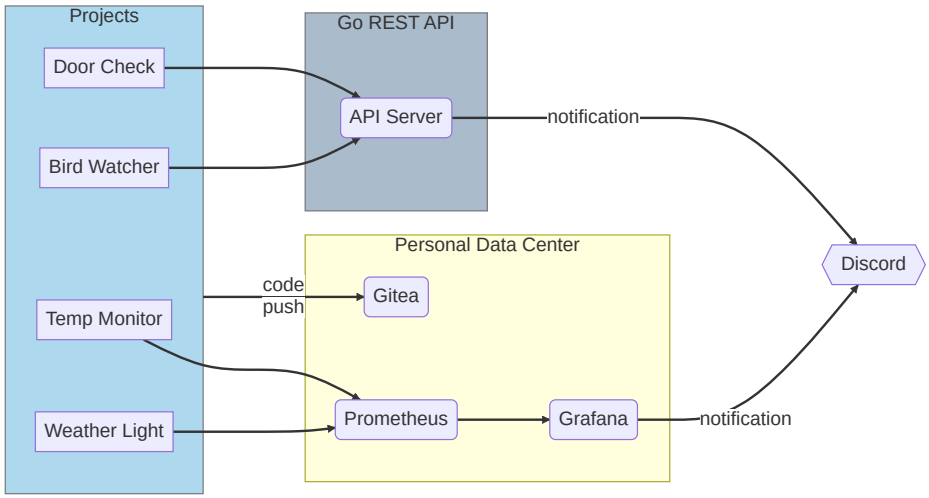
Managing your own home automation infrastructure also gives you a deeper understanding of what commercial systems do behind the scenes, including all the potential data collection that may occur without your knowledge. You'll know what the needs of the project are, and what data needs to be captured and evaluated. If you want to capture and manipulate more details, that's your choice, not the choice of the equipment manufacturer. Take a look at the diagram on page 5 to see how these software components and services interoperate to deliver our robust home automation solutions.

This diagram shows how the projects you'll develop in this book make use of the home automation infrastructure you'll deploy in Chapter 2, Building a REST API Server, on page ? and Chapter 3, Deploying Your Personal Data Center, on page ?. Projects "Door Check" and "Bird Watcher" use the Go REST API to send notifications out, while projects "Temperature Monitor (Temp Monitor)" and "Weather Light" use services from your personal data center to collect and visualize data. Also, you can use Gitea to manage and version control the source code of all your projects in a central location.

Joe asks:

How did you create this diagram?

The software components diagram was created using Mermaid,[a] an open source JavaScript tool that renders diagrams in different formats from a text-based definition.



We chose this approach because it's easier to create and modify relationship-based diagrams, such as flowcharts and class charts, by defining these relationships as text, without worrying about drawings, positioning, and arrow attachment. Also, because it's text-based, we can version control the diagrams and keep them in the same repository where we keep our code.

a. https://mermaid.js.org

To deploy these services, you'll need hardware to run it on. In this chapter, we'll first take a look at the hardware that we'll use to build the various projects in the book, starting with the Raspberry Pi. After that, we'll briefly summarize the software and services we rely on for the projects to work effectively. Once inventoried and set up, you'll build in the next chapters what we like to call your Personal Data Center.

Selecting a Raspberry Pi

The Raspberry Pi has permanently altered the home automation and technology tinkerer landscape by making rather powerful computers and microcontrollers for a fraction of what their counterparts cost. It's still mind-blowing to realize that a Linux server more powerful than rack-mounted servers from a decade ago can fit in the palm of your hand. In the case of the smallest Pi computer, it even takes up less space than a stick of gum.

For a majority of the projects in the book, you'll need, at the very least, a Raspberry Pi 3 Model B+[2] for roughly $35. Better still, a Raspberry Pi 4B[3] for the same price is an ideal tinkering model. It offers enough CPU, RAM, and ports to deliver a flexible platform to develop these and future projects that you may want to create. If you're willing to spare an additional $20, you can get the latest Raspberry Pi 5[4] that delivers a two to three times CPU performance increase compared with model 4B.

Another Pi model that will support these projects, and may be beneficial when space constraints are a concern, is the Pi Zero 2 W.[5] This remarkable piece of hardware is nearly equivalent to the performance of a Pi 3 for a third of the price. However, because it does not have onboard USB, HDMI, Ethernet ports, or header pins, configuring a Pi Zero takes a bit more effort and additional hardware dongles to properly set up. Once configured, the Pi Zero 2 W is just as manageable and easy to operate and maintain as a more expensive Pi with more options.

One final Pi model that we'll use is not a computer like the previously mentioned Pi's. The Pi Pico W[6] is a fairly new $6 microcontroller that has onboard Wi-Fi. Because it's a microcontroller, it has its own language and is designed for very specific jobs. Unlike a multi-purpose computer, the Pico has limited resources and is best suited for targeted monitoring of defined events. We'll use this inexpensive, wireless microcontroller to transmit monitored values to a REST server that will perform additional processing and make cool things happen. You need this microcontroller to work on the project in Chapter 4, Networking a Temperature Monitor, on page ?.

2. https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/
3. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/
4. https://www.raspberrypi.com/products/raspberry-pi-5/
5. https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/
6. https://www.raspberrypi.com/products/raspberry-pi-pico/?variant=raspberry-pi-pico-w