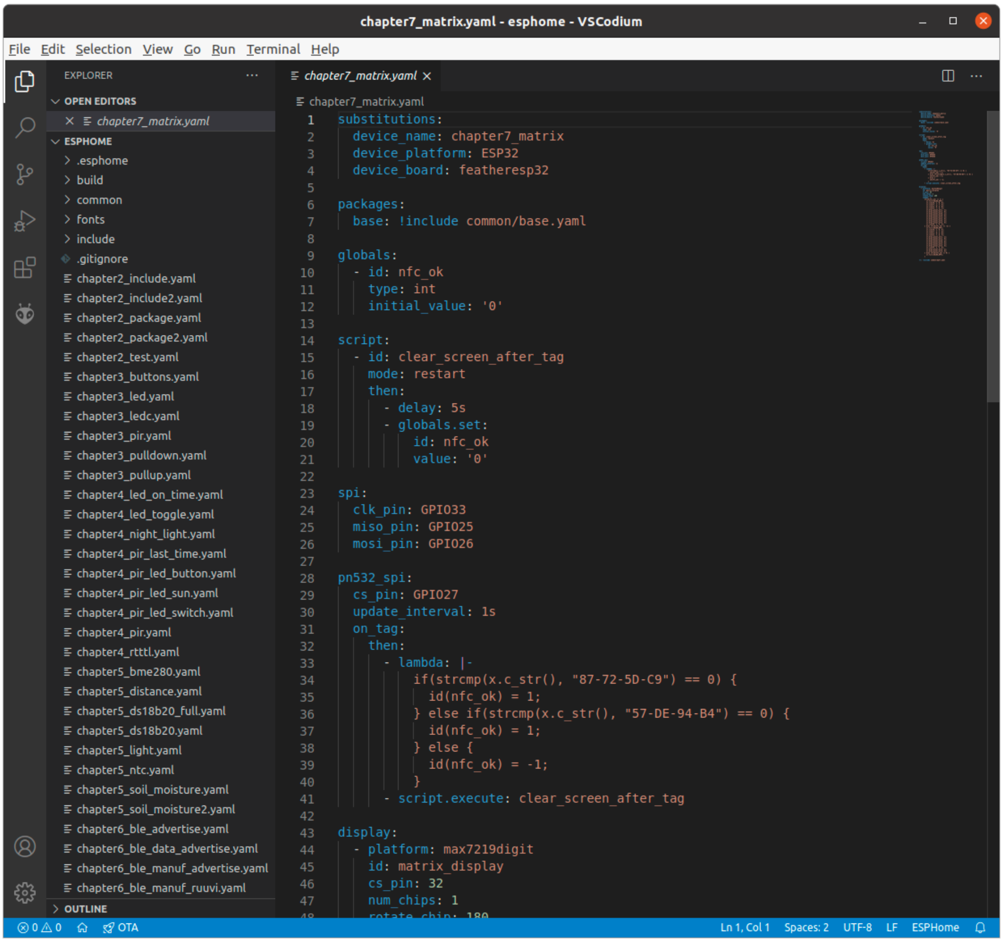
chapter7_matrix.yaml - esphome - VSCodium
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
chapter7_matrix.yaml
ESPHOME
.esphome
build
common
fonts
include
.gitignore
chapter2_include.yaml
chapter2_include2.yaml
chapter2_package.yaml
chapter2_package2.yaml
chapter2_test.yaml
chapter3_buttons.yaml
chapter3_led.yaml
chapter3_ledc.yaml
chapter3_pir.yaml
chapter3_pulldown.yaml
chapter3_pullup.yaml
chapter4_led_on_time.yaml
chapter4_led_toggle.yaml
chapter4_night_light.yaml
chapter4_pir_last_time.yaml
chapter4_pir_led_button.yaml
chapter4_pir_led_sun.yaml
chapter4_pir_led_switch.yaml
chapter4_pir.yaml
chapter4_rtttl.yaml
chapter5_bme280.yaml
chapter5_distance.yaml
chapter5_ds18b20_full.yaml
chapter5_ds18b20.yaml
chapter5_light.yaml
chapter5_ntc.yaml
chapter5_soil_moisture.yaml
chapter5_soil_moisture2.yaml
chapter6_ble_advertise.yaml
chapter6_ble_data_advertise.yaml
chapter6_ble_manuf_advertise.yaml
chapter6_ble_manuf_ruuvi.yaml
OUTLINE
0 0 OTA
Ln 1, Col 1 Spaces: 2 UTF-8 LF ESPHome

```yaml
substitutions:
  device_name: chapter7_matrix
  device_platform: ESP32
  device_board: featheresp32

packages:
  base: !include common/base.yaml

globals:
  - id: nfc_ok
    type: int
    initial_value: '0'

script:
  - id: clear_screen_after_tag
    mode: restart
    then:
      - delay: 5s
      - globals.set:
          id: nfc_ok
          value: '0'

spi:
  clk_pin: GPIO33
  miso_pin: GPIO25
  mosi_pin: GPIO26

pn532_spi:
  cs_pin: GPIO27
  update_interval: 1s
  on_tag:
    then:
      - lambda: |-
          if(strcmp(x.c_str(), "87-72-5D-C9") == 0) {
            id(nfc_ok) = 1;
          } else if(strcmp(x.c_str(), "57-DE-94-B4") == 0) {
            id(nfc_ok) = 1;
          } else {
            id(nfc_ok) = -1;
          }
      - script.execute: clear_screen_after_tag

display:
  - platform: max7219digit
    id: matrix_display
    cs_pin: 32
    num_chips: 1
    rotate_chip: 180
```

Figure 1.3 The ESPHome plugin for Visual Studio Code has some useful features like tooltips and validation and completion of the YAML code.

The next chapter shows you how to install ESPHome standalone. You can also install ESPHome as a dashboard (in a Docker container or as a Home Assistant add-on). The latter also lets you edit your ESPHome configuration files from a web interface. Because I don't want to tie you to Docker or Home Assistant, this book uses the standalone installation. Feel free to use the other methods.

1.4 How to use this book

This book describes a basic set of electronic components you can use to create your own home automation devices with ESPHome. This is by no means meant to be complete. I selected these components to be able to explain as many features of ESPHome as possible with a small set of cheap components.

After completing this book, you will have enough experience with ESPHome to start adding other electronic components. You should be able to create your own personal devices that make your home more comfortable.

Some basic electronic knowledge is recommended to follow the examples in this book. But even without this knowledge, they aren't that difficult, and I've tried to explain most non-trivial electronics. The small electronic circuits in this book don't work with mains electricity and are safe to use. That said, if it's all new to you, I recommend you to buy a basic electronics book.

Here's a short overview of what this book covers:

Chapter 1: Introduction

An introduction to what ESPHome is, why you use it, and what you need to create your own home automation devices.

Chapter 2: Preparing your ESPHome environment

The preparation for this book, where you install ESPHome and its dashboard, flash your first firmware, and learn about over-the-air updates and logs. This chapter also gives some best practices to keep your ESPHome configurations maintainable.

Chapter 3: Simple digital input and output

Your first hardware project with ESPHome, where you learn how to use the power of digital input and output and connect buttons, motion sensors, and LEDs.

Chapter 4: Automations

Automations linking various ESPHome components to each other, which makes your ESP32 device able to do stuff without depending on a home automation gateway.

Chapter 5: Sensors

Various types of sensors, including analog, 1-Wire, I²C, and ultrasonic distance.

Chapter 6: Remote communication

Communication methods over a distance other than Wi-Fi, including NFC, infrared light, and Bluetooth Low Energy.

Chapter 7: Displays

Various types of displays, from simple RGB LED sticks, 4-digit displays to a matrix, OLED, or the TTGO T-Display's built-in display.

Chapter 8: Conclusion

A wrap-up of this book, with some references to more information if you want to improve your ESPHome skills.

Appendix

Specialised tips that could come in handy in various situations.

Note:
All code examples from this book are published on https://github.com/koenvervloesem/Getting-Started-with-ESPHome. Read the instructions in the GitHub repository for more information on how to download them. The repository also lists errors that have been found in the book since its publication, as well as information about changes in ESPHome that impact the examples in this book.

1.5 Summary and further exploration

In this introductory chapter, I merely set the scene of the book so we're on the same page. You've learned the difference between configuring and programming, and by now you know the advantages of ESPHome as a platform to develop your own home automation devices. You've also seen what you need to make the most of this book. These requirements are quite flexible. You're free to change the components listed in the chapter, even the ESP32 board, as long as you know what you're doing.

If this is your first time working with ESPHome, ESP8266, ESP32, or a home automation gateway, I recommend sticking as closely as possible to the book's recommendations initially. I hope this book will give you the confidence to explore different paths, and the inspiration to create original home automation devices that no one has developed before.

Chapter 2 ● Preparing your ESPHome environment

In this chapter, you will install ESPHome, create your first ESPHome configuration, and flash your first firmware to your ESP32 device.

You also learn how to integrate ESPHome with your MQTT broker or Home Assistant. Pushing over-the-air updates to your devices and managing them from a web-based dashboard are other important tasks.

I also take some time to give some best practices to make your ESPHome configurations more maintainable. After a while, you maybe have ten ESPHome devices with different configurations in your house. If you want to add a status sensor to all of them, you don't want to copy and paste YAML code to ten files. Luckily ESPHome offers some interesting features to modularise your code and improve maintainability.

At the end of the chapter, you have a full-blown ESPHome environment and will be ready to experiment with all sorts of configurations described in the rest of this book.

2.1 ● Installing ESPHome

ESPHome is a Python program, so you first have to install Python. If you're running Linux, you probably already have Python installed by default. Just make sure that it's Python 3 and not Python 2.

If you're using Windows or macOS, download the latest Python version from the website (https://www.python.org). At the moment of writing this book, this is Python 3.9.2. For Windows, choose the **Windows installer**, and not the **Windows embeddable package**. Make sure to select the correct version for your architecture (32-bit or 64-bit, probably the latter).

After starting the installer, enable **Add Python 3.x to PATH**. This way you'll be able to run Python just by calling the `python` command on the command line.

After finishing the installation, open a command line (in Windows: Click **Start**, type `cmd` in the **Search** or **Run** line, and press **Enter**). Then check whether you can run Python by asking its version:[1]

```
$ python3 --version
Python 3.9.2
```

The Windows installer also installs Python's package manager, `pip`. This allows you to install extra Python packages, such as ESPHome. If you're using Linux, you probably have to install it yourself. On Ubuntu, this goes like this:

1 I'm a Linux user, and all examples in this book use Linux. The command prompt is `$` and not `C:\WINDOWS\system32`. Paths are delineated by `/` and not `\`. Make sure to change these examples to your own operating system.

```
$ sudo apt install python3-pip
```

Most Linux distributions allow you to install Python 2 and 3 at the same time, as well as pip and Python packages for Python 2 and Python 3. The commands for Python and pip are then generally called `python` and `pip` for Python 2, and `python3` and `pip3` for Python 3. In this book, I'll use `python3` and `pip3` to make sure you're using the correct version if both versions are installed.

Check whether you have pip installed and if it's the version for Python 3:

```
$ pip3 --version
pip 20.0.2 from /usr/lib/python3/dist-packages/pip (python 3.9)
```

Then, finally, install ESPHome as a Python package:

```
$ pip3 install esphome
Collecting esphome
...
```

This installs a lot of extra packages too because ESPHome needs them. At the end, you should see a line that starts with **Successfully installed** and then the list of all installed packages and their versions.

You're now ready for your first ESPHome project.

2.2 • Creating your first ESPHome configuration

It's best to create one directory where you store all your ESPHome configurations. For instance, I have created a directory `esphome` in my home directory. Now enter this directory and run the following command:

```
$ esphome chapter2_test.yaml wizard
```

This starts a wizard that guides you through creating a basic ESPHome configuration file in four steps:

1. Give your device a name (only small letters, numbers, and underscores are allowed).
2. Choose a platform (`ESP32` in this case) and then board name. Have a look at PlatformIO's website (https://docs.platformio.org/en/latest/platforms/espressif32.html#boards) if you're not sure what the name of your board is. For the TTGO T-Display ESP32, this is `featheresp32`.[2]
3. Set up Wi-Fi. Enter the SSID and password for your Wi-Fi network.
4. Choose a password for OTA (over-the-air) updates and the native API.

2 The TTGO T-Display doesn't have its own board name yet in PlatformIO, but it's compatible with the Adafruit ESP32 Feather.