

Raspberry Pi as a Foundation for Boosting Computer and Technology Literacy

Igor Rižnar
Aleksandr Gaisov



Faculty of Management
Monograph Series



Raspberry Pi as a Foundation for Boosting Computer and Technology Literacy

Amplifying Understanding through Projects

Igor Rižnar

Aleksandr Gaisov



*Raspberry Pi as a Foundation for Boosting Computer and Technology Literacy:
Amplifying Understanding through Projects*
Igor Rižnar and Aleksandr Gaisov

Reviewers · Uroš Godnov and Marjan Golob
Layout · Primož Orešnik

Published by · University of Primorska Press
Titov trg 4, 6000 Koper · www.hippocampus.si
Editor in Chief · Jonatan Vinkler
Managing Editor · Alen Ježovnik
Koper, 2024

© 2024 Igor Rižnar and Aleksandr Gaisov

<https://www.hippocampus.si/ISBN/978-961-293-378-4.pdf>
<https://www.hippocampus.si/ISBN/978-961-293-379-1/index.html>
<https://doi.org/10.26493/978-961-293-378-4>



*The publication of the monograph was financially supported
by the Slovenian Research and Innovation Agency
from the state budget funds, tender for the co-financing
of scientific monographs.*

Kataložni zapis o publikaciji (CIP) pripravili
v Narodni in univerzitetni knjižnici v Ljubljani

COBISS.SI-ID 210189059
ISBN 978-961-293-378-4 (pdf)
ISBN 978-961-293-379-1 (html)

Table of contents

- List of figures • 7
- List of tables • 9
- Introduction • 11
- Literacies • 15
- 1 A short history of Raspberry Pi and its models • 17
- 2 Raspberry Pi OS • 23
 - 2.1 Official Raspberry Pi operating system • 24
 - 2.2 Other operating systems • 28
- 3 Fundamentals of the Linux system and terminal usage • 31
 - 3.1 Shell, terminal, and command line interface • 32
 - 3.2 Basics of shell commands • 34
 - 3.3 Overview of shell commands • 37
 - 3.4 Customizing the system and the shell • 44
- 4 Raspberry Pi accessories • 51
 - 4.1 Raspberry Pi official accessories • 53
 - 4.2 Other Raspberry Pi accessories • 58
 - 4.3 Raspberry Pi for embedded applications and industrial applications • 58
 - 4.4 Miscellaneous HATs • 59
 - 4.5 Miscellaneous tools and accessories for DIY electronics experiments and projects • 60
- 5 Raspberry Pi usage • 63
 - 5.1 In research • 65
 - 5.2 In schools • 66
 - 5.3 Machine learning • 69
 - 5.4 Internet of Things • 71
 - 5.5 Home automation • 73
- 6 Programming on the Raspberry Pi • 75
 - 6.1 Selecting a programming language • 75
 - 6.2 Available programming languages • 76

- 6.3 Overview of programming languages · 77
- 6.4 Overview of development environments · 81
- 7 Basic projects for getting to know Raspberry Pi · 85
 - 7.1 A desktop computer · 85
 - 7.2 Christmas tree · 86
 - 7.3 Google Assistant with Google AIY · 87
 - 7.4 Magic Mirror · 90
 - 7.5 Simple music transport · 92
- 8 Advanced projects · 97
 - 8.1 RPi-based NAS · 97
 - 8.2 Advanced RPi-based DAC · 100
- 9 Raspberry Pi Zero, Arduino, and BBC micro:bit · 105
- Conclusion · 109
- References · 111
- Recommended readings · 119
- Recommended web resources · 121

List of figures

- 1.1 Raspberry Pi 5 · 17
- 1.2 Raspberry Pi 5 vs Raspberry Pi 4 · 19
- 1.3 Graphical presentation of Raspberry Pi 5 components · 19
- 2.1 A screenshot of Raspberry Pi Imager UI · 26
- 2.2 A screenshot of Raspberry Pi device menu. · 26
- 2.3 A screenshot of the operating system menu in Raspberry Pi Imager. · 26
- 2.4 A screenshot of storage menu in Raspberry Pi Imager. · 27
- 3.1 Icon of a terminal emulator in the Raspberry Pi OS panel · 32
- 3.2 A window of a running LXTerminal · 33
- 3.4 Example of a shell command executed within a terminal emulator · 33
- 3.3 The command prompt in LXTerminal · 33
- 3.5 Example of a shell command with a flag and an argument · 34
- 3.7 Example of redirecting the command's stderr to a file · 35
- 3.8 Example of using the stdout of one command as the stdin of another command · 35
- 3.6 Example of redirecting the command's stdout to a file · 35
- 3.9 Example of using the semicolon chain operator · 36
- 3.10 Example of using the AND chain operator · 37
- 3.11 Example of using the OR chain operator · 37
- 3.12 Tree of a root directory · 39
- 3.13 Tree of home directory · 39
- 3.14 Checking the CPU used by the user and kernel spaces via the top command · 41
- 3.15 Example of terminating a process in htopmand · 41
- 3.16 Example of the ping command · 42
- 3.17 Example of help flag usage · 43
- 3.18 Example of a man page · 44
- 3.19 Editing the .profile file · 45
- 3.20 Adding a cron job · 46
- 3.21 A working reminder · 46
- 3.22 Finding the PS1 variable in the .bashrc file · 47
- 3.23 Replacing the PS1 variable · 47
- 3.24 Appending an alias to the .bashrc file · 48

- 3.25 The output of the find command is redirected into awk and then into sed · 49
- 3.26 The bash script for fetching random man pages · 50
- 4.1 A basic case with Raspberry Pi 4 · 51
- 4.2 The dearest case for Raspberry Pi 4 · 52
- 4.3 Raspberry Pi with active cooler in Pi-Box Pro5 enclosure · 53
- 4.4 Raspberry Pi 7 touchscreen display · 53
- 4.7 Raspberry Pi-based desk top · 57
- 4.8 Adapters · 61
- 4.9 Soldering set · 61
- 7.1 Raspberry Pi 400 · 85
- 7.2 Raspberry Pi 4 in DiSalvo copper enclosure (right) and in Argon ONE M.2 enclosure (left) · 86
- 7.3 Xmas tree · 87
- 7.4 Assembled Google AIY voice assistant with Raspberry Pi 3 as the central computing unit · 88
- 7.5 MagicMirror · 91
- 7.6 moOde.local screenshot · 94
- 7.7 moO settings in moOde · 95
- 7.10 moOde setting options · 95
- 7.11 A simple music transport · 95
- 8.1 Argon EON NAs enclosure from various angles · 99
- 8.2 Inside of the NAS enclosure · 99
- 8.3 A screenshot of openmediavault UI · 100
- 8.4 Advanced Raspberry Pi-based DAC · 101
- 9.1 BBC micro:bit, Raspberry Pi Pico and Arduino UNO · 107

List of tables

- 1.1 Raspberry Pi family of computers · 18
- 2.1 Raspberry Pi OS versions · 25
- 3.1 Frequently used commands for package management · 39
- 3.2 Frequently used commands for managing files and directories · 40
- 3.3 Frequently used commands for system monitoring · 42
- 3.4 Frequently used commands related to connectivity and networks · 43
- 4.1 DAC HAT features · 57
- 8.1 NAS features · 98

Introduction

One of the authors of this book is a converted Windows-based computer user, who bought his first Mac mini in 2006 and who, a couple of years later (in 2012 or 2013), discovered Raspberry Pi (RPi). This was a life-changing discovery, so much so that after a series of different projects, ranging from a digital Christmas tree with Raspberry Pi Zero to digital-to-analogue converters employing several hardware attached on top (HATs) and power supplies worth ten times more than the heart of the system (RPi 4 with 8 GB of RAM), he decided, first, to co-write this monograph, and second, one day in the near future to start learning coding.

The Raspberry Pi, developed by the Raspberry Pi Foundation, is a series of small, affordable, single-board computers that have revolutionized the world of computing and educational technology since its inception in 2012. Designed to promote the teaching of basic computer science in schools and in developing countries, these versatile devices have found applications far beyond the educational sector, from hobbyist projects and home automation to industrial automation and more.

Raspberry Pi computers boast various models, each offering different specifications. Models vary in terms of processor speed, memory, connectivity options, and IO ports, ensuring there is a Raspberry Pi for virtually every application – from basic to advanced, from learning programming and hardware interaction to managing complex systems.

The Raspberry Pi typically runs on a Linux-based operating system (OS), with the Raspberry Pi OS (formerly known as Raspbian) being the officially supported OS. However, other OS options like Ubuntu, Windows IoT Core, are also compatible. Furthermore, the RPi supports various programming languages such as Python, Java, C++, and Scratch, making it a versatile tool for programmers of different levels of programming proficiency.

Beyond its initial educational purpose, Raspberry Pi has blossomed into a powerhouse for DIY enthusiasts and industries alike, being utilized in numerous applications, including IoT projects, media centres, servers, game consoles, and even in advanced projects like robotics and AI learning platforms.

The Raspberry Pi has a thriving community of developers, educators, and hobbyists who share projects, tutorials, and software through numerous online platforms, making it easy for newcomers to enter the Raspberry Pi ecosystem and effectively navigate their learning and project development journeys.

In a way, this is an exploratory biography of our learning experience as we take the reader through the world of Raspberry Pi, from simple and fun projects for absolute beginners to more advanced projects, all intending to enhance your understanding and skills in the ever-evolving field of computing. We hope this is a journey about discovery, creativity, and the joy of learning by doing that will increase your understanding of what Raspberry Pi is capable of. Our journey spans over ten years, so it is unreasonable to expect that these 100 pages will cover all one needs to know and understand to become a seasoned user. To make it easier for you to continue to learn on your own, we will add some resources both while we write about a certain topic and at the end of the book, where several websites, clubs, tutorials as well as books and magazines will be mentioned.

This monograph has the following layout: after a brief introduction, there is an account of different Raspberry Pis that have existed and can still be purchased. If, at the moment of reading this, you think about buying one, our recommendation would be to go to either a Raspberry Pi 4 (4 or 8GB) or a Raspberry Pi 5 (8GB), which is the latest iteration of the Raspberry Pi single-board computer family (at the time of writing this monograph: late 2023 and early 2024). Then we move to the Raspberry Pi official operating system, a Debian-based distribution specially optimized for the Raspberry Pi hardware that comes with some pre-installed software and utilities to make set-up and use as simple as possible. We continue by describing Raspberry Pi accessories, official and unofficial, and finish this chapter by mentioning some HATs that we find interesting, including some housings that make it possible for a DIY enthusiast to make a Raspberry Pi-based NAS (Argon40's EON), a beautiful desktop computer (solid copper maker block manufactured by DeSalvo Systems), etc. An important chapter of our monograph dis-

cusses Raspberry Pi usage in research, education, DIY projects, and industry. This chapter is logically followed by a longer section where several basic and advanced projects are explained in some detail. The next three chapters are devoted to learning to code with Raspberry Pi, AI and Raspberry Pi, and home automation and Raspberry Pi. Our journey culminates with a conclusion, which encapsulates our main findings and reflects on the journey that is not yet over. Recommended readings and web resources are suggested in the final two chapters, providing the readers with tools for further exploration and a deeper understanding of the topic – we hope these two sections will encourage the readers to continue learning beyond the modest scope of this monograph; we are aware that there are more relevant sources that cover some areas discussed here in much more detail and will be beneficial to our readers while also saving them some time and effort when searching for credible and authoritative sources of information. In a field that is quickly advancing, web resources have an additional added value as they provide up-to-date information.

Literacies

Traditionally, literacy refers to the ability to read and write. In a broader sense, it can also encompass knowledge or understanding in other areas. Among many other important literacies, the following come to mind: financial literacy; cultural literacy; and computer and technology literacy. Understanding environmental issues, sustainability, and the impact of human activities on the environment can be called ‘environmental literacy’. Cultural literacy refers to the knowledge and understanding of the cultural elements, symbols, traditions, and references that are significant within a particular society or community. It involves being familiar with the shared knowledge, values, customs, and expressions that shape the identity of a culture. Cultural literacy is crucial for fostering understanding, tolerance, and effective communication in an increasingly interconnected and diverse world.

The main purpose of this monograph is to show how Raspberry Pi could represent the foundation for boosting computer and technology literacy, i.e. our ability to use, understand, and apply concepts related to computers, software, and digital technologies, together with the understanding of basic programming and computational thinking. We believe this is possible for several reasons. First, Raspberry Pis are affordable, which allows schools and individuals to get hands-on experience with computing technology without unreasonable investments. Second, Raspberry Pi allows for a practical, hands-on approach to learning about computers and technology. Students can assemble and configure the hardware, explore programming languages, and work on various projects, fostering a deeper understanding of how computers work. Third, Raspberry Pi encourages the development of programming skills from an early age. Practically everyone can learn languages such as Python and Scratch to create their own applications and projects. Fourth, Raspberry Pi encourages project-based and research-based learning, whereby users engage in real-world, creative projects, in which there is little room for rote learning and regurgi-

tation. Fifth, Raspberry Pi-based projects not only enhance technical skills but also promote problem-solving, critical thinking, and collaboration.

As will be shown, Raspberry Pi has been widely used in science, technology, engineering, and mathematics (STEM) education. It supports a multidisciplinary approach, integrating various STEM concepts in a practical and engaging manner. We strongly believe that learning with Raspberry Pi leads to a well-rounded computer and technology literacy.

In many countries worldwide, Raspberry Pi in education has been a transformative force for improving computer and technology skills, contributing to digital inclusion in areas where access to traditional computing resources is limited or cost-prohibitive. In other words, Raspberry Pi is a valuable tool for narrowing the technological gap and fostering a more equitable educational landscape in underdeveloped regions.

Last, but not least, Raspberry Pi is green, because it is energy efficient, has a compact design, is known for its longevity and upgradeability, and its design is based on open-source hardware and software.

1 A short history of Raspberry Pi and its models

As there are too many models to mention or describe in detail, a table 1.1 will suffice to see the models still in production and a few models that are no longer being produced.

The latest model (as of January 2024) is Raspberry Pi 5. For practical reasons, our attention will focus on Raspberry Pi 4, Zero 2 W, and Pico, as they still offer very good performance compared to the latest model and, even more importantly, they have received the attention and support of both the Raspberry Pi Foundation and the wider community. There is a myriad of projects, tutorials, and accessories specifically designed for these models that provide a wealth of resources for users.

On the other hand, we have to point out that, due to its increased processing power (faster Broadcom BCM2712 processor) and some extra features (support for OpenGL 3.1 and Vulkan 1.2, faster LPD-DR4X-4267 SDRAM, PCIe slot, twin four-lane MIPI connectors, HEVC hardware video decoding, its ability to add an NVMe SSD), Raspberry Pi 5 will soon become the best choice for several of the most often used applications, ranging from computer vision to machine learning and AI, retro gaming, media streaming, and home automation. For many, Raspberry Pi 4 has been a substitute for a desktop PC or a web and data server. Raspberry Pi 5 brings improvements and superior performance

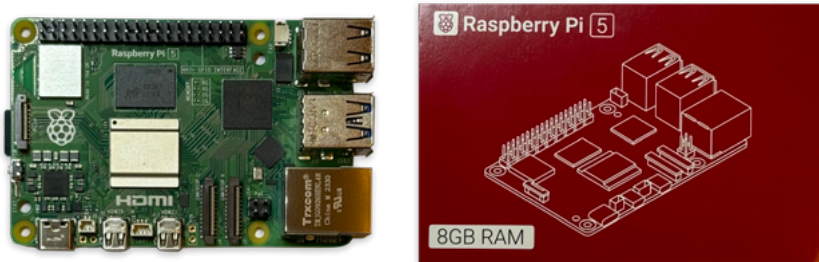


Figure 1.1 Raspberry Pi 5

Table 1.1 Raspberry Pi family of computers

Family	Model	SoC	Memory	Form factor	Ethernet	Wireless	GPIO pins	Released	Discontinued
Pi	B	BCM2835	256 MB	Standard	Yes	No	26	2012	Yes (to be confirmed)
	A		512 MB		No			2012	No
	B+		256 MB		No			2013	
	A+		512 MB	Compact	Yes		40	2014	
Pi 2	B	BCM2836 / 7	1 GB	Standard	Yes	No		2015	
Pi Zero	Zero	BCM2835	512 MB	Ultra-com-	No	No			
	W / WH			pact		Yes		2017	
Pi 3	2 W	BCM2710A1						2021	
	B	BCM2837Ao / Bo1 GB		Standard	Yes	Yes		2016	
	A+	BCM2837Bo	512 MB	Compact	No	Yes		2018	
	B+		1 GB	Standard	Yes			2018	
Pi 4	B	BCM2711Bo / Co1 GB		Standard	Yes	Yes		2019	Yes (2020)
			2 GB					2021	No
Raspberry Pi Pico	400		4 GB					2020	
	Pico	RP2040	264 KB	Keyboard					
	W		4 GB	Pico	No	No	26	2021	
Raspberry Pi 5		BCM2712	4 GB	Standard	Yes	Yes	40	2022	
		8 GB						2023	

SOURCE [Wikipedia](https://en.wikipedia.org/wiki/Raspberry_Pi)  https://en.wikipedia.org/wiki/Raspberry_Pi

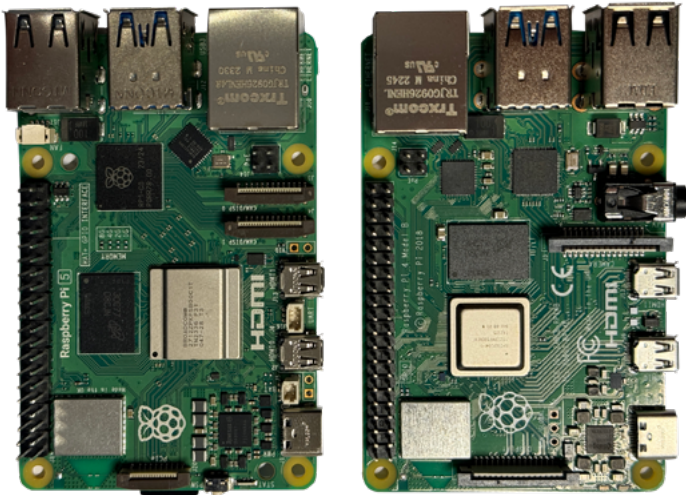


Figure 1.2 Raspberry Pi 5 vs Raspberry Pi 4

for these applications, so it will most likely become the best choice once its production is ramped up and availability improves. Figure 1.2 shows Raspberry Pi 5 on the left side and Raspberry Pi 4 on the right.

A detailed presentation of the Raspberry Pi 5 can be found in the documentation section of the [Raspberry Pi Foundation webpage](https://www.raspberrypi.com/documentation/computers/raspberry-pi-5.html). Here are the most important parts shown in a graphical presentation.

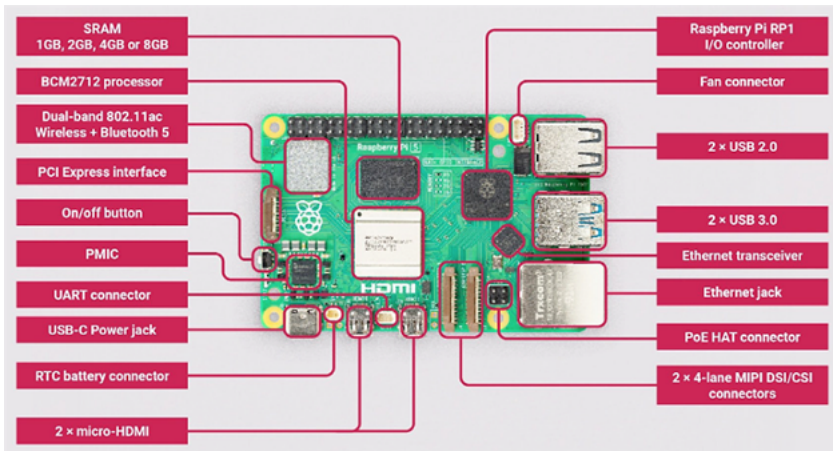


Figure 1.3 Graphical presentation of Raspberry Pi 5 components

Source <https://web.archive.org/web/20240308144927/https://www.raspberrypi.com/documentation/computers/raspberry-pi-5.html>

In accordance with Figure 1.3, key features include (Upton 2023):

- **BCM2712 processor:** 2.4GHz quad-core 64-bit Arm Cortex-A76 CPU with Cryptographic Extension support (AES on hardware), 512KB per-core L2 caches, and 2MB L3 cache
- **VideoCore VII GPU:** OpenGL-ES 3.1, Vulkan 1.2, dual 4kx60 HDMI display output (compared to a maximum of dual 4k30p on the Raspberry Pi 4 model), HDR support, 4kx60 HEVC decoder, and a Raspberry Pi Image Sensor Processor (ISP)
- **4GB or 8GB LPDDR4X-4267 SDRAM:** Synchronous Dynamic RAM with a transfer rate more than twice as fast as that of the previous Raspberry Pi 4 model
- **2x USB 3.0 (capable of simultaneous full throughput), 2x USB 2.0, USB-C power connector**
- **Wi-Fi (802.11ac dual-band), Bluetooth 5.0 (with BLE support), and Gigabit Ethernet**
- **Dual 4-lane MIPI CSI/DSI transceivers:** Camera and display serial interfaces, supporting 2x display; or 2x camera; or 1x display + 1x camera
- **Raspberry Pi 40-pin GPIO header:** General-purpose input/output pins for interfacing with additional electronic circuits
- **UART connector:** A universal asynchronous receiver/transmitter allows for communicating with other devices via a standard serial interface
- **RTC battery connector:** Powering the real-time clock allows the Raspberry Pi to keep track of the time while the main power supply is disconnected
- **Renesas DA9091 “Gilmour” PMIC:** The power management integrated circuit generates the necessary voltages for various board components, ensuring stable operation and power efficiency, which allows for the integration of the RTC and power button
- **PoE support (additional HAT required):** Power over Ethernet allows for power and data transfer through a single connection, enabled by attaching a HAT (Hardware Attached on Top) accessory
- **Raspberry Pi connector for PCIe (1 x 2.0 port; additional HAT required):** The PCI Express 2.0 interface allows for connecting additional accessories (e.g., M.2 HATs)
- **Power button**
- **Fan connector**

If our readers want to get started with their Raspberry Pi, they will need the following:

1. A power supply (official Raspberry Pi power supply is recommended for both Raspberry Pi 4 (5V 3A) and 5 (5.1V 5A),
2. A boot media (a good quality, large enough and fast enough microSD card).

A Raspberry Pi can be set up with:

- A display
- A cable to connect your Raspberry Pi to your display (monitor or TV)
- A keyboard
- A mouse

Alternatively, Raspberry Pi can be set up as a headless computer accessible over the network, in which case you do not need any peripherals. At the time of installing an operating system, you can preconfigure a hostname, user account, network connection, and secure shell (SSH: a method for securely sending commands to a computer over an unsecured network).

2 Raspberry Pi OS

Every computer needs an operating system and a Raspberry Pi is no different. The guidelines below refer to any Raspberry Pi except Raspberry Pi Pico, a microcontroller with a completely different set-up process. There are two ways in which one can install the OS on a Raspberry Pi:

1. A headless install (when you have no monitor or keyboard and mouse connected to a Raspberry Pi)
2. Install Raspberry Pi OS using Raspberry Pi Imager to a microSD card on a computer with an SD card reader – this is the quickest way.

There are many Raspberry Pi operating systems available in addition to the official OS. Raspbian was developed by Mike Thompson and Peter Green, based on the Linux distribution Debian. The initial version was released in June 2012, and there are more than 5,000 packages available for easy installation on the Raspberry Pi, as can be seen here [🔗 Welcome to Raspbian](#).

There is also a Raspberry Pi distribution by [🔗 Kali Linux](#). If Raspbian is especially beginner-friendly, Kali is primarily used for extensive security and penetration testing of computer systems and networks as it contains more than 600 tools. Kali is suitable for computer and network security tests, but less suitable for Linux beginners. Pidora ([🔗 pidora.ca](#)) is another OS, based on the Linux distribution by Fedora. One of its main features is headless mode, which as you already know by now, allows one to access the microcomputer without a monitor. Again, it may not be the best choice for beginners. Microsoft launched its first OS for IoT devices in 2015 (the time when Raspberry Pi 2 and 3 reigned). More about the development and deployment of applications to Windows IoT Core on Raspberry Pi 2 can be found here: [🔗 learn.microsoft.com](#).

Ubuntu is a very popular Linux distribution, because it is highly modifiable and user-friendly. Ubuntu Core (see here: [🔗 Install Ubuntu on a Raspberry Pi](#)) is a good choice and is ready for you and your Raspberry Pi 5.

Then there is RISC OS (find more here: [🔗 RISC OS Open](#)), which is also in use on single-board computers like BeagleBoard and PandaBoard (Raspberry Pi's competition). The system was first released in 1987 by Acorn for ARM-based computer Archimedes. For all who are new to RISC OS, there is a detailed introduction available here: [🔗 Introduction to RISC OS; Internal tutorials and information](#). A minimalist Linux distribution was developed by team members of PlugApps and ArchMobile, called Arch Linux for ARM processors, which developed into what is today known as Arch Linux ARM (see here: [🔗 https://archlinuxarm.org](https://archlinuxarm.org)).

The OS RetroPie (more is available here: [🔗 RetroPie](#)) allows you to turn your Raspberry Pi into a gaming console. It sits on top of a full OS you previously installed on your Raspberry Pi, or you can download an image and add any additional software you wish later.

A webpage called [🔗 Raspberrmy Tips](#) recommends OS according to your intended use, so if your goal is to play games, they recommend Batocera OS, for servers they recommend DietPi, and LibreElec is their choice for a media-centre set-up. To delve deeper, see their other recommendations, like Kano OS, Gentoo, RecalBox, OSMC, OpenMediaVault, Pop!_OS, etc., some of which will be mentioned later in our book.

Raspberry Pi OS versions are given in Table 2.1.

2.1 Official Raspberry Pi operating system

Here we shall first describe the second OS installation option. Let's go through the steps.

1. Download Raspberry Pi Imager (for macOS, Windows or Ubuntu for x86) from [🔗 Raspberry Pi OS](#)
2. Install Raspberry Pi Imager to your computer with an SD card reader

Table 2.1 Raspberry Pi OS versions

Release date	Debian version	Linux kernel	GCC	APT	X Server	Pi 1/1+Pi2	Pi 3	Pi Zero W	Pi 3+	Pi 4	Pi Zero 2 W	Pi 5
2013-09-27	7 (Wheezy)	3.6	4.7.2	0.9.7	7.7	✓	✗	✗	✗	✗	✗	✗
2015-05-12						✓	✗	✗	✗	✗	✗	✗
2015-09-28	8 (Jessie)	4.1	4.9	1.0.9.8.1		✓	✗	✗	✗	✗	✗	✗
2017-07-05						✓	✓	✓	✗	✗	✗	✗
2017-08-17	9 (Stretch)		6.3	1.4.6		✓	✓	✓	✗	✗	✗	✗
2019-04-08				1.4.9		✓	✓	✓	✓	✗	✗	✗
2019-06-24	10 (Buster)	4.19	8.3	1.8.2		✓	✓	✓	✓	✓	✗	✗
2021-10-30		5.10.63				✓	✓	✓	✓	✓	✓	✗
2021-12-03	11 (Bullseye)		10.2.1	2.2.4	1.20.11	✓	✓	✓	✓	✓	✓	✗
2023-12-05						✓	✓	✓	✓	✓	✓	✗
2023-10-10	12 (Bookworm)		12.2.0	2.6.1	1.21.1	✓	✓	✓	✓	✓	✓	✓
2023-12-05		6.1.69	12.2.4			✓	✓	✓	✓	✓	✓	✓

Source Adapted from https://en.wikipedia.org/wiki/Raspberry_Pi_OS

3. Run Raspberry Pi Imager on your computer

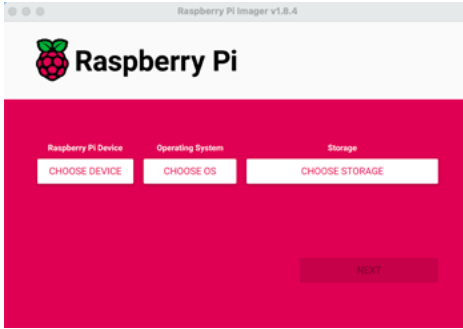


Figure 2.1
A screenshot of Raspberry Pi Imager UI

After clicking on “Choose Device” the following window opens

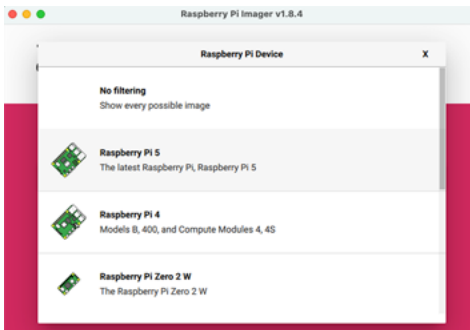


Figure 2.2
A screenshot of Raspberry Pi device menu.

Select the device you would like to prepare the OS for. Next, click the “Operating System” menu:

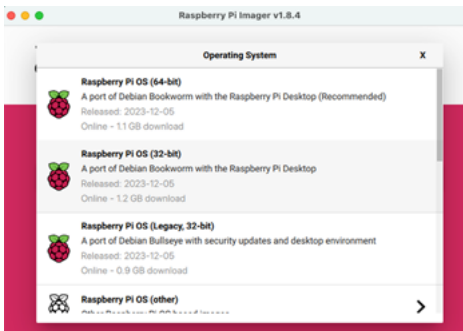


Figure 2.3
A screenshot of the operating system menu in Raspberry Pi Imager.

At the top, you will find the recommended version of Raspberry Pi OS. Feel free to scroll down to see other available options. If you need

a media player distribution for entertainment, click on “Media Player OS” and select LibreELEC; if you need an operating system for streaming and playing music, you can choose Volumio or do some research to see if there are better distributions for playback and the streaming of music files (say MoOde, Ropieee, Vitos, etc).

Next, you have to choose storage. Storage is the microSD you have put in your reader for the OS to be installed on.

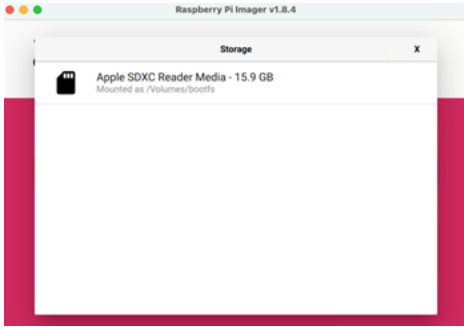


Figure 2.4

A screenshot of storage menu in Raspberry Pi Imager.

As you can see, our microSD card is 64 GB, mainly because bigger is sometimes also faster. On the other hand, 8GB is more than enough space for the OS. You must click “Next” at the bottom right and wait a couple of minutes for the image to copy to the microSD. Be careful to choose the right storage device connected to your computer otherwise you may delete the wrong storage device.

Next, you can customize the OS by clicking on “Edit Settings” in the pop-up menu, but it is often OK, if you do this when you boot for the first time. In customization, you set up your Raspberry Pi and preconfigure a username and password, Wi-Fi credentials, the device hostname, the time zone, your keyboard layout, and remote connectivity. Needless to say, if you decided to customize the OS before inserting the microSD card in your Raspberry Pi, you have to click Save, then Yes to apply settings, and finally, Yes again to continue with data writing to the microSD card.

Soon, you will see a pop-up menu saying, “Write Successful”; click Continue and try booting your Raspberry Pi with the image. If you skipped OS customization in Raspberry Pi Imager, you will have to go through the configuration on the first boot. You now need a monitor and a keyboard. We usually use the original Raspberry Pi keyboard and mouse, but almost any keyboard and mouse will work.

Bookworm is the latest iteration of Raspberry Pi OS (remember that Raspberry Pi 5 is only compatible with this version). In previous versions, it was possible to install Python libraries system-wide by using the pip package management tool, which has become impossible with Bookworm. Now, packages must be installed into a virtual environment so that they do not interfere with the system version of Python. For example, if you have a project that you have named “gpt-project”, the Python virtual environment is created by typing:

```
python -m venv gpt-project
```

Then you have to change directory to the newly created environment folder, which contains a full Python distribution, and activate it:

```
cd gpt-project  
source bin/activate
```

The virtual environment is now ready to use. From within the virtual environment you can install any package you need using a pip command – for example, openai – by typing:

```
pip install openai
```

This will install the module, along with any independencies it requires, within the Python virtual environment. The advantage of this extra step is that it allows you to work on Python projects with specific dependencies in isolation from your system-wide Python installation.

If you reboot the Raspberry Pi, you will need to reactivate the Python virtual environment to use it again.

2.2 Other operating systems

Some operating systems were mentioned in the previous subchapter, but we were far from reaching the end of the list. Raspberry Pi OS is widely accepted for its compatibility and ease of use with Raspberry Pi hardware. As seen above, different operating systems cater to different needs: they are either lightweight (say, for good performance on older models) or meant for systems used for media centres, IoT, gaming, home automation, network attached storage, or security testing. It has become obvious by now that Raspberry Pi OS has been designed with one main purpose in mind: educational use. The pre-installed software (Scratch, Thonny, Phyton IDE) are all suitable for beginners who wish to learn coding and digital creation in an interactive and user-friendly

manner. Through its OS, the Raspberry Pi Foundation also provides easy access to a vast collection of tutorials. An equally important role is that of the community of learners, educators, and enthusiasts who contribute by creating and sharing educational content, offering troubleshooting support and who develop new tools and resources. There is always something for you to learn cost-effectively, as projects range from AI to robotics to serving, playing, synthesizing and composing (with SonicPi software, see [🔗 sonic-pi.net](https://sonic-pi.net)) music.

In the chapter on advanced Raspberry Pi projects, we will become familiar with operating systems like Volumio, MoOde, piCorePlayer (pCP), Ropieee and Vitos. These are used for music playback or streaming, either through a popular, albeit expensive, application Roon (see [🔗 roon.app/en/](https://roon.app/en/) for details) on a rock server (for which Ropieee is among the best options), or as a standalone Raspberry Pi-based music player solution. All are operating systems that do only one thing – make it possible for you to stream or play music through web-based interfaces that allow users to control the music player from a web browser on different devices (phones, tablets, and computers).

If you are using Raspberry Pi Imager v1.8.4 you can choose to install Ubuntu Desktop 23.10 (64-bit) for Raspberry Pi 4/400/5 models with 4 or more GB RAM. For Raspberry Pi Zero 2W/2/3/4/400 you can install Ubuntu Server 23.10 (32-bit version). There are also options for installing Apertis or RISC OS PI for many Raspberry Pi models. For media player OSes RPi recommends LibreELEC (which is a Kodi entertainment distribution), OSMC (a fast and feature-filled open-source media centre), as well as Volumio and moOde. There are also three emulation and game operating systems on offer in Raspberry Pi Imager – RetroPie; Recalbox; Lakka – and a number of specific-purpose operating systems:

- For 3D printing: OctoPi, OctoKlipperPi, Mainsail OS, and Simply-Print
- For home assistants: Homebridge, Home Assistant, Raspberry-Matic, nymea, Gladys Assistant, and openHAB
- A popular open-source digital signage project called Anthias, Kali Linux
- Freemium and paid-for operating systems: digital signage OS; 3D printing; Android by Emteria; TLXOS.

3 Fundamentals of the Linux system and terminal usage

As previously mentioned, the Raspberry Pi Foundation markets its 32-bit Raspberry Pi OS as the officially supported operating system (Raspberry Pi Foundation, n.d.-a). This OS is built upon Debian ([↗ https://www.debian.org/](https://www.debian.org/)), a major GNU/Linux distribution. A reader might be wondering: ‘I have heard of Linux, but what is GNU?’ Stallman (n.d.) describes it as follows: ‘Linux is the kernel, one of the essential major components of the system. The system as a whole is basically the GNU system, with Linux added.’ The kernel is what interacts with hardware. The GNU system, in turn, encompasses various critical software components, including GNU core utilities ([↗ https://gcc.gnu.org/](https://gcc.gnu.org/)), GNU bash shell (Free Software Foundation, n.d.), GNU Compiler Collection (GNU Project 2020b), which operate on top of the kernel. Therefore, GNU can be built with different kernels, with Linus’s Torvalds Linux being the most popular of them.

Hereinafter, for the sake of simplicity and clarity, GNU/Linux systems are referred to simply as ‘Linux’. However, when specifically discussing the Linux kernel, it is explicitly referred to as such.

The process of learning any OS begins by answering the question: ‘What do I want to accomplish with it?’ This is because Linux is essentially a versatile tool that can be employed to fulfil various needs. Consequently, the process of learning is aligned and optimized for a specific purpose, be it a project or general education. However, the basics of Linux remain similar, regardless of the end-goal. By understanding the fundamentals of process spaces, directory hierarchy, basic commands, package management, and troubleshooting, one could confidently customize and utilize the OS to one’s advantage.

The personal experience of one of the authors has demonstrated that one of the most effective ways to grasp these fundamentals is through learning to utilize the terminal and its commands. Therefore, the following several subsections are dedicated to the fundamentals of Linux systems and the role of the terminal in relation to them. Finally,

we will conclude this section with a couple of guided projects that focus on customization.

3.1 Shell, terminal, and command line interface

At the beginning, it is crucial to distinguish between three concepts that are frequently confused.

A *shell* serves as an interface between the user and the Linux kernel, receiving and interpreting user commands before forwarding them to the kernel for execution. These commands can consist of the shell's built-in features, shell scripts, or other programs (e.g. GNU core utilities).

Many Linux systems, including the Raspberry Pi OS, come with two shells: `sh` and `bash`. Bourne shell, or `sh`, was originally developed for early Unix versions and is required for the proper functioning of a Linux system (Ward 2021, 12) and compatibility. At this time, however, `sh` is outdated and is simply a link to another shell on the system (`dash`, in our case). Commands for the 'sh-type' shells are expressed in the Shell Command Language, ensuring adherence to the POSIX standard.

The Bourne-again shell (GNU Project 2020a), or `bash`, is a default shell compatible with `sh` and is also POSIX-compliant. However, the `bash` shell includes a superset of features compared to `sh`, such as command history, tab completion, arrays, dictionaries, enhanced arithmetic operations, command line editing, and more. In the monograph, we will mostly focus on `bash`, for it is the default and widespread shell.

In Linux, a shell can be accessed in a variety of ways, including through *terminal emulators* (Figure 3.1) and virtual consoles (TTYs).

The primary difference between the two lies in their operating environments. A terminal emulator runs within a graphical environment,

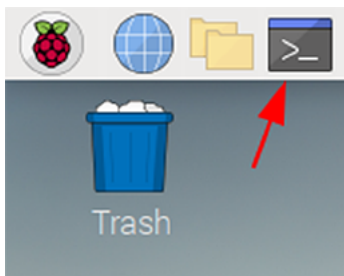


Figure 3.1
Icon of a terminal emulator in the Raspberry Pi OS panel



Figure 3.2 A window of a running LXTerminal

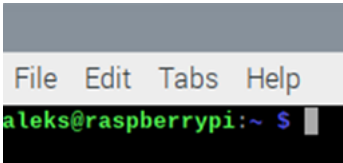


Figure 3.3
The command prompt in LXTerminal

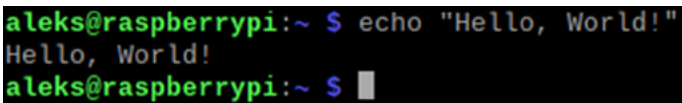


Figure 3.4 Example of a shell command executed within a terminal emulator

essentially functioning as an application with a graphical user interface (GUI). In contrast, virtual consoles are kernel-managed and operate independently of the graphical environment. As a result, terminal emulators provide more features and are generally more convenient to use within a desktop environment. The Raspberry Pi OS comes with LXTerminal (LXDE Blog 2021) as the default terminal emulator (Figure 3.2).

Finally, a terminal emulator offers the user a *command line interface* (CLI) featuring a command prompt (or shell prompt) – an input space where a user can enter commands (Figure 3.3). For instance, let us output a string:

```
echo "Hello, World!"
```

In order to execute this simple `echo` command (Figure 3.4), the reader would need to type it into the aforementioned command prompt and then press ‘Enter’.

```

aleks@raspberrypi:~$ ls -a ~
..          Bookshelf  .greenfoot  mu_code    .python_history  Videos
.           .cache    greenlog    Music      .scratch         .wget-hsts
.bash_history .config   .java       Pictures   Scratch          .Wolfram
.bash_logout Desktop   .lesshtst  .pki      .selected_editor .Xauthority
.bashrc     Documents .local     .profile  .sonic-pi       .xsession-errors
.bluej      Downloads .Mathematica Public     Templates       .xsession-errors.old
aleks@raspberrypi:~$

```

Figure 3.5 Example of a shell command with a flag and an argument

3.2 Basics of shell commands

The shell commands adhere to the following format and accept additional *flags* and *arguments*:

```
command [flags] [arguments]
```

Flags (or attributes) are additional parameters used to enable or disable certain functionalities of a command, thereby modifying its behaviour. They are typically preceded by a hyphen ‘-’ (e.g. `-e`) or two hyphens ‘--’ (e.g. `--version`). When combined with the arguments, which are the specific data that the command operates on, they ultimately control command execution.

For demonstration purposes, we will resort to a useful `ls` command that displays the files and folders within a specified directory. Consider the following `ls` command (Figure 3.5) with the `-a` flag to show hidden files and the tilde character ‘~’ as an argument. The tilde symbol is understood by the shell as a path to the user’s home directory:

```
ls -a ~
```

To clear the terminal window from the command output, one can use the `clear` command or the key combination ‘CTRL+L’.

Input and output streams

Processes on a Linux system interact with each other through *input and output streams*. They receive data from *standard input* (stdin), output data to *standard output* (stdout), and report errors via *standard error* (stderr) (Ward 2021, 14). The stdin of a process, initiated by a shell command, can originate from various sources, such as a file or another shell command. Similarly, a stdout can be generated by the Linux kernel, a file, another shell command, etc.

In the previous example, the `ls` command received a path to a directory as its stdin and returned the list of the directory’s contents as the stdout. In that case, the stdout was captured and displayed by the terminal. However, this does not necessarily have to be the case, as the

```

aleks@raspberrypi:~ $ ls -a 1> file1 2> file2
aleks@raspberrypi:~ $ cat file1
.
..
.bash_history
.bash_logout
.bashrc
.bluej
Bookshelf
.cache

```

Figure 3.6

Example of redirecting the command's stdout to a file

shell allows us to redirect stdout and stderr using file descriptors. The `1>` operator redirects stdout, while `2>` redirects stderr. Consider the following two slightly different commands:

```

ls -a ~ 1> file1 2> file2
ls -a /non-existent/directory/ 1> file1 2> file2

```

In the first case (Figure 3.6), the stdout of the `ls` command is redirected to `file1`, which is created automatically. Concurrently, `file2` is also created but remains empty since the command has not returned an error. It is worth noting that the terminal shows no output because the stdout is sent to a file instead. The contents of `file1` can be inspected using the `cat` command to confirm that it contains the stdout from the `ls` command.

In the second case (Figure 3.7), the `ls` command attempts to access a directory that does not exist on the system. As a result, the stderr is captured and saved into `file2`, whereas `file1` is empty. Similarly, the stderr is not printed out in the terminal because it is redirected to a file. This can be verified by using the `cat` command on `file2`.

Finally, to use the stdout of one command as the stdin of another command, one has to utilize the pipe character `|`. This useful process

```

aleks@raspberrypi:~ $ ls -a /non-existent/directory 1> file1 2> file2
aleks@raspberrypi:~ $ cat file2
ls: cannot access '/non-existent/directory': No such file or directory
aleks@raspberrypi:~ $ █

```

Figure 3.7 Example of redirecting the command's stderr to a file

```

aleks@raspberrypi:~ $ ls -S | head -n 5
Bookshelf
Desktop
Documents
Downloads
mu_code
aleks@raspberrypi:~ $ █

```

Figure 3.8

Example of using the stdout of one command as the stdin of another command

is also frequently denoted by the jargon ‘piping’ in various online educational materials. To see it in action, consider the following command:

```
ls -S | head -n 5
```

The `ls` command is given the flag `-S`, which tells it to sort the current directory contents by size. Next, its stdout is ‘piped’ (i.e. redirected) to the `head` command as its stdin. The `head` command then takes the first five lines (i.e. five largest files and folders) and prints them out in the terminal (Figure 3.8).

Chaining operators

Getting to know a few operators for connecting commands, such as ‘|’ and ‘>’, is certainly useful, and there are many more. However, we will only discuss three more very frequently used chaining operators, specifically the semicolon ‘;’, ‘&&’ (AND), and ‘| |’ (OR).

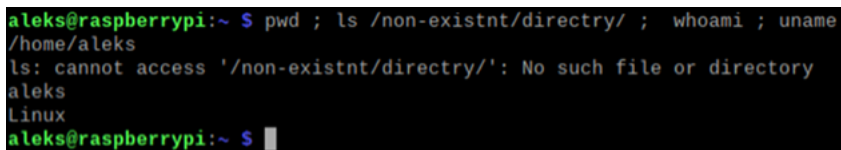
Essentially, *chaining operators* allow us to create sequences of commands by combining them. However, they differ in their execution behaviour. For instance, the semicolon ‘;’ character allows us to combine multiple commands into one line, which will be sequentially executed regardless of whether any of the commands fail or succeed. Consider the following example:

```
pwd ; ls /non-existent/directory/ ; whoami ;
uname
```

The first, third, and fourth commands are successfully executed, and their outputs are displayed in the terminal. The second command fails, with its stderr also displayed, but this does not interrupt the execution flow (Figure 3.9).

Next, consider the following `ping` and `echo` commands chained together:

```
ping -c 1 google.com && echo -e "\nConnected to
the internet!"
```



```
aleks@raspberrypi:~ $ pwd ; ls /non-existent/directory/ ; whoami ; uname
/home/aleks
ls: cannot access '/non-existent/directory/': No such file or directory
aleks
Linux
aleks@raspberrypi:~ $
```

Figure 3.9 Example of using the semicolon chain operator


```

aleks@raspberrypi:~ $ ping -c 1 google.com && echo -e "\nConnected to the internet!"
PING google.com (142.250.201.206) 56(84) bytes of data:
64 bytes from bud02s35-in-f14.1e160.net (142.250.201.206): icmp_seq=1 ttl=114 time=13.1 ms

--- google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 6ms
rtt min/avg/max/mdev = 13.089/13.089/13.089/0.000 ms

Connected to the internet!
aleks@raspberrypi:~ $

```

Figure 3.10 Example of using the AND chain operator

```

aleks@raspberrypi:~ $ ls projects || mkdir projects
ls: cannot access 'projects': No such file or directory
aleks@raspberrypi:~ $ ls
Bookshelf Documents file1 greenlog Music projects Scratch Videos
Desktop Downloads file2 mu_code Pictures Public Templates
aleks@raspberrypi:~ $

```

Figure 3.11 Example of using the OR chain operator

The AND operator means that the next command will run only if the previous command succeeds. Therefore, the string “[Connected to the internet!](#)” (`\n` initiates a new line) will only be displayed if the system was able to reach “google.com”, indicating that we indeed have an Internet connection (Figure 3.10). Otherwise, only the `ping` command will be executed, and its `stderr` will be displayed.

Last, but not least, consider the following example of the OR operator usage:

```
ls projects || mkdir projects
```

The command to the right of the OR operator will only execute if the command to the left fails. As a result, the `mkdir` command will create a “projects” directory only if the `ls` command attempts to access a non-existent directory (Figure 3.11).

Obviously, all chaining operators could be used together to combine various commands and create different clever algorithms.

3.3 Overview of shell commands

Next, let us go over some of the essential operations that a user might want to perform and how they can be accomplished within a terminal. This includes updating the system, installing and removing software, creating and manipulating files, monitoring system resources, and interacting with networks. Hence, the subsequent subsections aim to demonstrate the speed and efficiency of performing these operations using the shell commands, in contrast to using some GUI software.

Package management

Being a Debian derivative, the Raspberry Pi utilizes `apt` as its standard package management tool.

A primary task for users after the installation of a Linux system is to update it, ensuring security, compatibility, and access to the most recent software. The update process consists of two straightforward steps, each requiring elevated privileges. First, execute the following command:

```
sudo apt update
```

In this case, the `apt` command is executed as an argument of `sudo`, which manages user permissions on the system. As a result, the user must input the superuser password. When typing the password in the terminal, it might appear as though nothing is being entered, but the password is simply being hidden for security reasons. After entering the password, press ‘Enter’. This command synchronizes the versions of the packages presently installed on the system with those available in the distribution’s software repository.

At this stage, the user may be interested in knowing which installed packages are outdated and will be upgraded. To check this, one can run the following command, which does not require superuser privileges and, therefore, does not need `sudo`.

```
apt list --upgradable
```

Finally, to upgrade the system, proceed by running the following command:

```
sudo apt upgrade
```

This command will download the new versions of the outdated packages and install them on the system. It will require elevated privileges and, therefore, must be run with `sudo`. When asked, “Do you want to continue? [Y/n]” type ‘Y’ for yes, or ‘N’ for no, and press ‘Enter’.

It is noteworthy that the aforementioned ‘update’ and ‘upgrade’ commands must both be executed in this exact order to perform a full system update. Thus, a user might want to use the AND chain operator (as per the previous section) to combine them:

```
sudo apt update && sudo apt upgrade
```

Below are some of the frequently used `apt` commands for package management (Table 3.1).

Table 3.1 Frequently used commands for package management

Command	Description
<code>sudo apt update</code>	Synchronize the local package versions with the repository
<code>sudo apt upgrade</code>	Install the new versions of packages
<code>apt list --upgradable</code>	List all the packages that can be upgraded
<code>apt list --installed</code>	List all the packages installed on the system
<code>apt search pack- age-name</code>	Search the software repository for a specific package
<code>sudo apt install package-name</code>	Install a specific package
<code>sudo apt remove pack- age-name</code>	Remove a specific package

Directory and file management

The structure of a Linux file system originates from the *root directory* “/”, which subsequently includes all other directories (Figure 3.12). Altering files and folders within the root directory could lead to security vulnerabilities or even completely break the system. Thus, it requires superuser privileges and is generally not advisable.

Nevertheless, a “/home” directory (Figure 3.12 and Figure 3.13) exists that includes *home folders* (or personal directories) of regular users (located at “/home/user-name” or “~/”). In contrast, any actions performed inside a user’s home folder are safe and do not require `sudo`, as they do not affect the entire system. The terminal emulator

```

aleks@raspberrypi:~ $ tree -d -L 1 /
/
├── bin -> usr/bin
├── boot
├── dev
├── etc
├── home
├── lib -> usr/lib
├── lost+found
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin -> usr/sbin
├── srv
├── sys
├── tmp
├── usr
└── var

```

Figure 3.12 Tree of a root directory

```

aleks@raspberrypi:~ $ tree -d -L 1 ~/
/home/aleks/
├── Bookshelf
├── Desktop
├── Documents
├── Downloads
├── mu_code
├── Music
├── Pictures
├── projects
├── Public
├── Scratch
├── Templates
└── Videos

```

Figure 3.13 Tree of home directory

Table 3.2 Frequently used commands for managing files and directories

Command	Description
<code>touch path-to-file</code>	Create a file
<code>cat path-to-file</code>	Print out the file contents
<code>less path-to-file</code>	Read a large file
<code>rm path-to-file</code>	Remove an existing file
<code>cp /old/path /new/path</code>	Copy a file or directory from one path to another
<code>mv /old/path /new/path</code>	Move a file or directory from one path to another (can be used for renaming)
<code>pwd</code>	Print working directory
<code>cd path-to-directory</code>	Change directory
<code>ls path-to-directory</code>	List the contents inside a directory
<code>mkdir path-to-directory</code>	Make a new directory
<code>rmdir path-to-directory</code>	Remove an existing directory

automatically launches inside a current user's home folder by default (i.e., denoted by '~').

Above are some of the frequently used commands for navigating directories, creating directories and files, as well as copying, moving, and deleting them (Table 3.2).

System resource monitoring

Like every other operating system, Linux obscures its components and inner workings from the user behind various layers of abstraction.

In this case, at the highest level is the *user space*, where *user processes* operate. This space includes various GUI programs (e.g. a web browser), servers (e.g. a display server), and shells (e.g. `bash`). These user processes run in *user mode*, which restricts their RAM access to a limited subset and ensures only safe CPU operations are executed. The data from the user space are received and processed by the kernel, which is responsible for such *kernel processes* as memory and process management as well as handling device drivers. This middle level is referred to as *kernel space*, with the processes running in *kernel mode*, granting them unrestricted access to the CPU and RAM. This distinction in access privileges between kernel and user modes is fundamental for system stability and security. Finally, the CPU starts reading and executing the kernel's instructions from the RAM and interfaces with other hardware devices (Ward 2021, 2–3).

To see this in action, one can run the `top` command to check what processes are currently running on the system as well as how much

```
top - 12:29:20 up 1:32, 2 users, load average: 0.24, 0.11, 0.04
Tasks: 182 total, 1 running, 181 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.3 us, 0.7 sy, 0.0 ni, 97.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 394.3 total, 3063.5 free, 216.7 used, 514.1 buff/cache
MiB Swap: 100.0 total, 100.0 free, 0.0 used, 3469.9 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
  615 root        20   0 184236 57276 37796 S   4.0   1.5   0:31.15 Xorg
  826 aleks     20   0 265520 66684 51988 S   2.0   1.7   0:19.01 mutter
```

Figure 3.14 Checking the CPU used by the user and kernel spaces via the top command

CPU power and RAM are being used. One can also take note of the user and system CPU times (*us* and *sy*, respectively), which represent the percentage of CPU time spent in user and kernel space (Figure 3.14). Like any other running process within the terminal window, the top command can be terminated (i.e. exited in this case) by pressing the key combination ‘CTRL+C’.

Alternatively, one can use the `htop` command, which offers the same functionality as `top` but with greater convenience. To illustrate this, one could execute the `htop` command, navigate to the “lterminal” process within the menu using the arrow keys, press ‘F9’ to kill the process, and finally press ‘Enter’ to confirm (Figure 3.15). This action should result in the terminal window closing as its process has been terminated.

```
0[ 0.0%] Tasks: 63, 79 thr; 1 running
1[| 1.3%] Load average: 0.02 0.01 0.00
2[| 2.0%] Uptime: 01:54:06
3[| 3.8%]
Mem[|||||] 248M/3.71G
Swp[ 0K/100.0M]

  PID USER      PR  NI  VIRT  RES  SHR  S  CPU%vMEM%  TIME+  Command
  615 root        20   0 180M 57280 37800 S  1.3  1.5  0:44.96 /usr/lib/xorg/Xc
 1794 aleks     20   0 119M 32308 27036 S  0.6  0.8  0:01.09 lterminal
 1805 aleks     20   0  8828  3348  2672 R  1.9  0.1  0:00.91 htop
   1 root        20   0 33884  8836  6980 S  0.0  0.2  0:03.16 /sbin/init splas
 142 root        20   0 45712 14132 13344 S  0.0  0.4  0:01.14 /lib/systemd/sys
 175 root        20   0 20644  5144  3448 S  0.0  0.1  0:01.75 /lib/systemd/sys
 320 systemd-t  20   0 22284  5504  4956 S  0.0  0.1  0:00.35 /lib/systemd/sys
 380 systemd-t  20   0 22284  5504  4956 S  0.0  0.1  0:00.01 /lib/systemd/sys
 382 avahi       20   0  6908  2724  2460 S  0.0  0.1  0:00.49 avahi-daemon: ru
 383 root        20   0  8192  2360  2184 S  0.0  0.1  0:00.03 /usr/sbin/cron -
 384 messagebu  20   0  8112  3984  3108 S  0.0  0.1  0:00.85 /usr/bin/dbus-da
 385 avahi       20   0  6748  264  0 S  0.0  0.0  0:00.00 avahi-daemon: ch
 445 root        20   0 40996  7376  5608 S  0.0  0.2  0:00.27 /usr/libexec/pol

F1 Help F2 Setup F3 Search F4 Filter F5 Tree F6 SortBy F7 Nice F8 Nice + F9 Kill F10 Quit
```

Figure 3.15 Example of terminating a process in htopmand

Table 3.3 Frequently used commands for system monitoring

Command	Description
<code>ps</code>	List all running processes
<code>killall</code>	Kill process by name
<code>df</code>	Get info on mounted file systems
<code>free</code>	Get info on free and used memory
<code>top</code>	Display Linux processes
<code>htop</code>	Enhanced version of top

Killing a process without running a CLI program can also be very useful, particularly if it is part of a chain of other commands or within a shell script.

```
ps -u -x
killall lxterminal
```

This can be accomplished by first running a `ps` command with the `-u` and `-x` flags to display all processes belonging to the current user. Then, find the name of the process inside the output (e.g. “lxterminal”) and pass it as an argument to the `killall` command to terminate it.

Above are some of the frequently used commands for managing processes, monitoring CPU and RAM usage, and checking mounted storage (Table 3.3).

Network commands

A very popular way to check if a device is connected to the Internet or if a specific website is currently available is to use the `ping` command. For instance, one could try to ‘ping’ a Google server:

```
ping google.com
```

Thus, if the command returns successful ‘pings’, which are small individual packets of data, then the website is available, and the device has access to the Internet (Figure 3.16).

In table 3.4 are some of the frequently used commands related to connectivity and networks.

```
aleks@raspberrypi:~$ ping google.com
PING google.com (142.251.39.46) 56(84) bytes of data:
64 bytes from bud02s38-in-f14.1e100.net (142.251.39.46): icmp_seq=2 ttl=115 time=144 ms
64 bytes from bud02s38-in-f14.1e100.net (142.251.39.46): icmp_seq=3 ttl=115 time=160 ms
64 bytes from bud02s38-in-f14.1e100.net (142.251.39.46): icmp_seq=4 ttl=115 time=558 ms
```

Figure 3.16 Example of the ping command

Table 3.4 Frequently used commands related to connectivity and networks

Command	Description
<code>ping ip/domain</code>	Check connectivity to a specific server
<code>ifconfig</code>	Check active network interfaces
<code>ssh username@host</code>	Connect to a remote shell

Getting help and manuals

Obviously, it is impossible for this monograph to cover all existing shell commands, let alone their associated flags. There are simply too many, and even more can be downloaded and installed from the software repository via `apt`. Therefore, we highly encourage the reader to do their own research and explore more specialized and advanced resources on Linux systems, and on shells in particular. There is so much more to learn.

Conveniently, a decent place to start is the terminal itself. Almost all shell commands support a `--help` flag, which provides concise information about what a certain command does and its functionality (Figure 3.17).

Alternatively, there is a specialized `man` command that accepts various other commands as arguments and returns an entire manual, explaining everything one needs to know about a command in great detail. These manuals are denoted by the jargon ‘man pages’.

For instance, the following command will open a man page for the `mv` command (Figure 3.18):

```
man mv
```

```
aleks@raspberrypi:~ $ cp --help
Usage: cp [OPTION]... [-T] SOURCE DEST
  or: cp [OPTION]... SOURCE... DIRECTORY
  or: cp [OPTION]... -t DIRECTORY SOURCE...
Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

Mandatory arguments to long options are mandatory for short options too.
-a, --archive                same as -dR --preserve=all
  --attributes-only          don't copy the file data, just the attributes
  --backup[=CONTROL]        make a backup of each existing destination file
-b                            like --backup but does not accept an argument
  --copy-contents            copy contents of special files when recursive
-d                            same as --no-dereference --preserve=links
-f, --force                  if an existing destination file cannot be
                             opened, remove it and try again (this option
                             is ignored when the -n option is also used)
-i, --interactive            prompt before overwrite (overrides a previous -n
                             option)
```

Figure 3.17 Example of help flag usage

```

MV(1)                                User Commands
NAME
  mv - move (rename) files

SYNOPSIS
  mv [OPTION]... [-I] SOURCE DEST
  mv [OPTION]... SOURCE.. DIRECTORY
  mv [OPTION]... -t DIRECTORY SOURCE...

DESCRIPTION
  Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

  Mandatory arguments to long options are mandatory for short options too

  --backup[=]
      make a backup of each existing destination file

  -b      like --backup but does not accept an argument

  -f, --force
      do not prompt before overwriting

Manual page mv(1) line 1 (press h for help or q to quit)

```

Figure 3.18 Example of a man page

3.4 Customizing the system and the shell

Running various shell commands in sequence or combining them using chain operators is already faster and more convenient than doing all these tasks manually, such as through a GUI file manager. However, one might want to take it a step further and automate some of the more frequent and repetitive tasks. Moreover, the `bash` shell itself is highly customizable, both in terms of its appearance and its functionality. In this sense, the following subsections present a few guided projects that can help the reader start experimenting, customizing, and automating their system and the shell.

Launching the terminal at login

In the user’s home folder, there is a file named “.profile” that controls the behaviour of their working environment. Any commands placed in this file are executed at every login session. Thus, for instance, one could add a command to launch the terminal automatically each time they log into the system.

To accomplish this, let us first open the “.profile” file in the Nano (<https://www.nano-editor.org/>) text editor.

```
nano .profile
```

Then, append the command for launching the terminal at the end of the file. To move the cursor around use the arrow keys (Figure 3.19).


```

GNU nano 5.4 .profile
if [ -n "$BASH_VERSION" ]; then
  # include .bashrc if it exists
  if [ -f "$HOME/.bashrc" ]; then
    . "$HOME/.bashrc"
  fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ]; then
  PATH="$HOME/bin:$PATH"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ]; then
  PATH="$HOME/.local/bin:$PATH"
fi

lterminal &

```

Figure 3.19 Editing the `.profile` file

The ‘&’ character is yet another chain operator that sends the process to be executed in the background, preventing it from otherwise freezing the entire session.

```
lterminal &
```

Save the file by pressing ‘CTRL+O’ and then ‘Enter’. Exit the file by pressing ‘CTRL+X’. Finally, log out and log back in, and the terminal window should appear automatically.

Automating tasks using cron jobs

What if a user wants to automate a certain recurring task, such as reminders? There is a built-in tool to do this called `cron`, which will perform a certain job at specified times. To edit the schedule, run the following command and navigate to the end of the document:

```
export EDITOR=nano && crontab -e
```

The first part of the command specifies that we want to use the Nano editor to edit the schedule, as the default editor is Vim (<https://www.vim.org/>). Since we are already somewhat familiar with Nano, we will stick to it.

For instance, we would like to be reminded to take a break from learning shell commands every hour. To do this, append the following command at the end of the document (Figure 3.20), save, and exit (as described in the previous subsection).

```
0 * * * * export DISPLAY=:0 && xmessage -timeout
3 "Take a break now!"
```

```

GNU nano 5.4 /tmp/crontab.DIYhM3/crontab *
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
0 * * * * export DISPLAY=:0 && xmessage -timeout 3 "Hello"

```

Figure 3.20 Adding a cron job

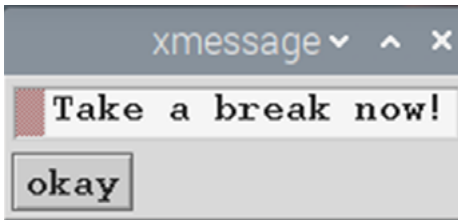


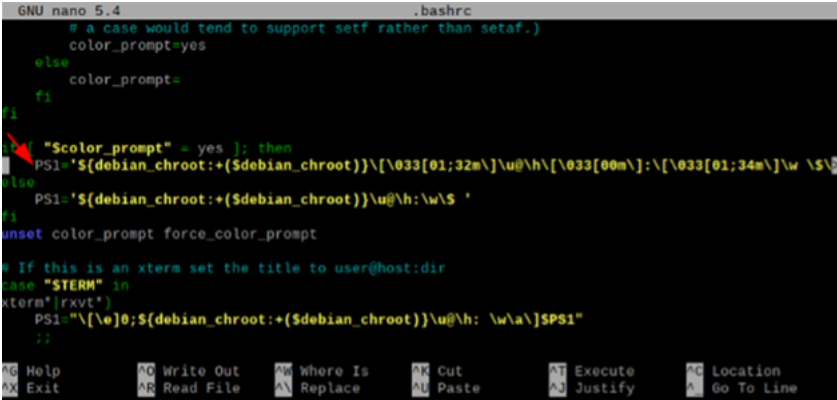
Figure 3.21
A working reminder

The job’s definition adheres to a certain standard. First, there is a schedule at which a command must be executed. It is represented in terms of five numbers that stand for minute, hour, date, month, and weekday. The asterisk “*” character stands for ‘every/any’. Finally, the command is specified.

In our case, we would like the command to execute at “every 0th minute of every hour, every day”, which is translated into “0 * * * *”. As for the command, first, we specify the display on which to show the reminder (i.e. the zeroth display). This is the only display if only one monitor is connected. Then, we call the `xmessage` tool, which will display a window with our message inside, which disappears after three seconds (Figure 2.21).

Modifying the command prompt

What if the default command prompt looks boring, and we would like to change it? The answer lies in the home folder, specifically in the “.bashrc” file. This is a configuration file that is checked and executed every time a new instance of the `bash` shell is launched. The variable



```

GNU nano 5.4 .bashrc
# a case would tend to support setf rather than setaf.)
color_prompt=yes
else
color_prompt=
fi
fi
fi
"Scolor_prompt" = yes ]; then
PS1='$(debian_chroot:+($debian_chroot))\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w \${
else
PS1='$(debian_chroot:+($debian_chroot))\u@\h:\w\$ '
fi
unset color_prompt force_color_prompt

# If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*)
PS1="\[\e]0;$(debian_chroot:+($debian_chroot))\u@\h: \w\a\]$PS1"
;;
*)
;;
esac

Help Write Out Where Is Cut Execute Location
Exit Read File Replace Paste Justify Go To Line

```

Figure 3.22 Finding the PS1 variable in the .bashrc file

we are looking for is “PS1”, which contains the customization options for the current user’s command prompt (Figure 3.22).

```
nano .bashrc
```

Then, locate the first occurrence of the “PS1” variable, comment it out (in case we would like to change it back to the original) as per Figure 3.23. Subsequently, add the following variable definition beneath it (ensuring it remains a single line):

```

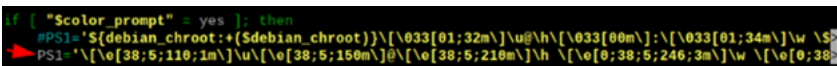
PS1=' \[\e[38;5;110;1m\]\u\[\e[38;5;150m\]@\
[\e[38;5;210m\]\h \[\e[0;38;5;246;3m\]\w \
[\e[0;38;5;150;1m\]> \[\e[0m\]'

```

After saving and exiting the file, close and reopen the terminal or execute:

```
source .bashrc
```

Now, we should be greeted with a more colourful command prompt. The variable may seem like a lot of gibberish, and it can be challenging to modify it by hand, let alone create one from scratch. Therefore, it is a good idea to use online tools, such as a bash prompt generator (<https://bash-prompt-generator.org/>).



```

if [ "Scolor_prompt" = yes ]; then
#PS1='$(debian_chroot:+($debian_chroot))\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w \${
PS1=' \[\e[38;5;110;1m\]\u\[\e[38;5;150m\]@\[\e[38;5;210m\]\h \[\e[0;38;5;246;3m\]\w \[\e[0;38;5;150;1m\]> \[\e[0m\]'
fi

```

Figure 3.23 Replacing the PS1 variable

Creating a command alias

Aliases in `bash` serve as convenient shortcuts that can replace long, complex commands or commands with many flags and arguments. They can save the user a lot of time typing, making their interaction with the terminal more efficient.

Consider, for instance, the `ls` command with many useful flags, such as showing the files' sizes, listing them one per line, enabling colour in the output, and more, such as this one:

```
ls -human-readable -size -l -S -classify -color=auto -group-directories-first
```

To create an alias for it, let us open the `‘.bashrc’` file in Nano:

```
nano .bashrc
```

Then, append the following alias at the end of the file (Figure 3.24):

```
alias ls='ls --human-readable --size -l -S --classify --color=auto --group-directories-first'
```

Finally, save and exit the file, and run the following command for the changes to take effect:

```
source .bashrc
```

Now, instead of running the aforementioned long `ls` command and spelling out all its flags, one can simply run `‘ls’`, and the long version of the command will be executed.

```
GNU nano 5.4 .bashrc
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
  . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

alias ls='ls --human-readabl --size -l -S --classify --color=auto --group-directories-first'
```

Figure 3.24 Appending an alias to the `.bashrc` file

Additionally, a personal recommendation from one of the authors would be to also add the following aliases for the `cp`, `mv`, and `rm` commands:

```
alias cp='cp -v -i'
alias mv='mv -v -i'
alias rm='rm -v -i'
```

These aliases will make these commands more verbose (`-v`) and interactive (`-i`), which can help prevent accidental overwrites and deletions by asking for confirmation before executing.

Fetching a random man page

Last, but not least, let us create a *bash script*, which is essentially a file that can contain multiple commands along with additional programming syntax, such as variables. This allows a user to create their own command line tools that can accept flags and arguments. For demonstration purposes, we will create a small and simple script for fetching and displaying random man pages, so the reader can learn more about various shell commands on their own.

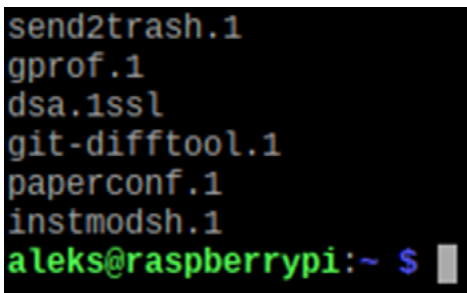
We shall begin by creating an empty file inside the home folder (name it “randman”, for instance) using the `touch` command and then opening it with Nano:

```
touch randman
nano randman
```

The first line of any shell script must be a *shebang*, which informs the system about which interpreter to use for the script. In our case, we are using `bash`, so we must include the following:

```
#!/bin/bash
```

Next, let us design a command that will fetch a random man page:



```
send2trash.1
gprof.1
dsa.1ssl
git-difftool.1
paperconf.1
instmodsh.1
aleks@raspberrypi:~ $
```

Figure 3.25

The output of the `find` command is redirected into `awk` and then into `sed`

```
GNU nano 5.4 randman
~/bin/bash
random_manpage=$(find /usr/share/man/man1 -type f | shuf | awk -F '/' '{print $6}' | sed 's/.gz//g' | head -1)
man $random_manpage
```

Figure 3.26 The bash script for fetching random man pages

```
find /usr/share/man/man1 -type f | shuf | awk -F
 '/' '{print $6}' | sed 's/.gz//g' | head -1
```

First, we utilize the `find` command to search through a directory containing mostly user-focused tools at the path “/usr/share/man/man1” and include only files.

Then, we use `shuf` to mix them in a random order. Next, we clear the output through `awk`, which divides every line into individual fields (using ‘/’ as the separator) and returns the sixth field of every line. Then, we globally remove the unnecessary file extensions “.gz” by substituting them with an empty space using `sed` (Figure 3.25). Finally, we select only the first line using the `head`.

We are using the variable `random_manpage` to store the values generated by the aforementioned command, using the command substitution expression `$()`:

```
random_manpage=$(find /usr/share/man/man1 -type
 f | shuf | awk -F '/' '{print $6}' | sed 's/.
 gz//g' | head -1)
```

To complete our bash script, we will pass the variable to the `man`, which will open the man page of the specified command (Figure 3.26):

```
man $random_manpage
```

Finally, save and exit the file, and make it executable for the current user via the `chmod` command:

```
chmod u+x randman
```

Now, the program can be executed from the home folder by running `./randman` (“.” stands for the current directory).

4 Raspberry Pi accessories

Raspberry Pi accessories can be divided into several categories. The main types are:

- Input/output devices (displays, keyboards and mice, cameras, speakers, and microphones)
- Storage (microSD cards, external hard drives and SSDs, USB flash drives)
- Connectivity (Wi-Fi and Bluetooth dongles, Ethernet cables, USB hubs)
- Expansion boards and HATs (GPIO expansion boards, HATs)
- Power solutions (power supplies, battery packs, UPS HATs)
- Cases and enclosures (basic cases, specialized enclosures, customizable cases)

Figure 4.1 shows the simplest case for Raspberry Pi: two acrylic pieces with standoffs and screws make a case suitable for any Raspberry Pi (3,4, or 5). Figure 4.2. shows the most expensive case we have come

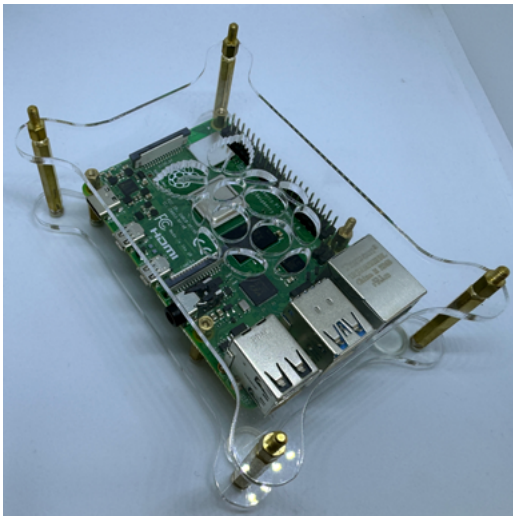


Figure 4.1
A basic case with
Raspberry Pi 4



Figure 4.2
The dearest case
for Raspberry Pi 4

across and we decided that our Raspberry Pi 4 deserves it. It is manufactured by DeSalvo Systems ([🔗 desalvoinc.com](https://desalvoinc.com)) and made from a single block of copper. The CPU, RAM, and USB controller are heat sunk to the housing so passive cooling is guaranteed even when Raspberry Pi 4 is overclocked.

A reasonably priced case for Raspberry Pi 5 is shown in Figure 4.3. This is a rugged industrial enclosure with an extender board with USB-C and two micro HDMI's on the same plane as USBs and Ethernet ports of the Raspberry Pi 5 board.

- Cooling solutions (heat sinks, fans, cooling cases)
- Development and prototyping accessories (breadboards and jumper wires, sensors and modules, robotics kits).

Some of the accessories are official Raspberry Pi gear, many more are products made by third-party manufacturers, ranging from pretty large companies (Adafruit – have a look here for more information: [🔗 www.adafruit.com](https://www.adafruit.com); Pimoroni – [🔗 shop.pimoroni.com](https://shop.pimoroni.com); CanaKit – [🔗 www.canakit.com](https://www.canakit.com); The Pi Hut – [🔗 thepihut.com](https://thepihut.com)) to individuals with a keen interest in electronics and some business acumen. Individuals often sell their accessories through online marketplaces like Etsy, Tindie, or Kickstarter, and some even through Github (e.g. user ianacanada at [🔗 github.com/ianacanada/DocumentDownload](https://github.com/ianacanada/DocumentDownload)).

Vilros (have a look at their About us file here: [🔗 vilros.com/pages/about-us](https://vilros.com/pages/about-us)), a company founded in 2010, was established with the goal of helping to create a more technological savvy society. They promote STEM education and sell components and kits produced by Raspberry Pi, micro:bit (<https://microbit.org>), 8BitDo ([🔗 www.8bitdo.com](https://www.8bitdo.com)), Arduino ([🔗 store.arduino.cc](https://store.arduino.cc)), and many other important manufacturers of Raspberry Pi goodies.

**Figure 4.3**

Raspberry Pi with active cooler in Pi-Box Pro5 enclosure

Source Lincoln Binns

(<https://lincolnbins.com/blog/post/new-pi-box-pro-5-for-the-raspberry-pi-5.html>)

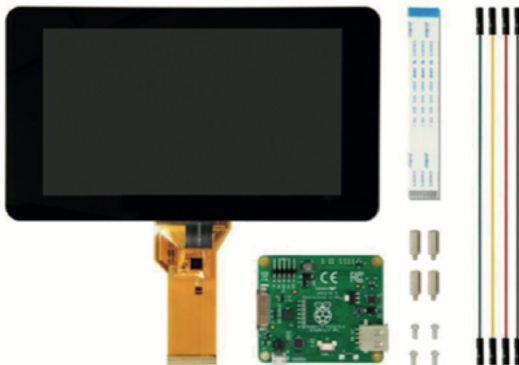
4.1 Raspberry Pi official accessories

The list with detailed descriptions of Raspberry Pi official accessories can be found on the official web page: [Raspberry Pi Documentation; Accessories](#).

The Raspberry Pi 7-inch touch display is an LCD display that connects well to the Raspberry Pi (and is, by the way, the only option if you want to use a screen with a music player with Ropieee OS) through a ribbon cable and a DSI connector. Its key feature, instructions for mounting, powering, changing screen orientation, and legacy support (for original Raspberry Pi model A and B) are described in detail here: [Raspberry Pi Documentation; Accessories; Display](#)

Below is a figure of the Raspberry Pi touch display kit (Figure 4.4):

Raspberry Pi Foundation produces several camera modules: Camera Module 3 and 3 wide, 3 NoIR and 3 NoIR wide, a High-Quality Camera with CS or M-12 mount, and a Global Shutter camera. Four new cameras were presented in *The MagPi* magazine, issue 126 (Raspberry Pi 2023). They bring faster autofocus, HDR, and an improved

**Figure 4.4**

Raspberry Pi 7 touchscreen display.

Source (<https://www.raspberrypi.com/documentation/accessories/display.html>)

12-megapixel sensor. All cameras are supported by libcamera software, connect to the camera serial interface (CSI) socket on Raspberry Pi. Camera Module 3 wide has an expanded 120° field of view and features an image sensor from Sony (IMX708), which has a pixel density that produces incredibly detailed images. The latest Raspberry Pi OS also comes with Picamera 2, a Python library that is perfect for snapping images to building advanced image recognition applications. Camera Module 3 can be used for home security, time-lapse photography, wildlife photography, industrial inspection, astrophotography, or art projects. Documentation, product brief, and manuals can be found at [🔗 Raspberry Pi Documentation; Accessories; Camera, camera-module-3-product-brief.pdf](#), and [🔗 picamera2-manual.pdf](#).

Figure 4.5 shows a set-up with a Raspberry Pi 4 (with Bookworm OS installed), Raspberry Pi HQ Camera v1 from 2018 with a C/CS mount, and a 10 megapixels 16 mm lens, to which an old Sony wide conversion lens (VCL-0637H) has been attached. The photo shown was made after we:

1. Made all connections
2. Prepared the software
3. Checked if everything was working by typing:

```
rpicam-hello
```

after which a preview window opened for about five seconds.

4. Finally, we captured a full resolution JPEG image by typing:

```
rpicam-jpeg -o test.jpg
```

in the terminal.

The video capture application for Raspberry Pi is rpicam-vid. For writing a 20-second video to file, we used:

```
rpicam-vid -t 20000 -o test.h264
```

and played the file in VLC application using:

```
vlc test.h264
```

Common projects with Raspberry Pi 4 and an HQ camera include home surveillance systems or remote monitoring, a system for capturing wildlife activity with motion detection features, time-lapse photography, a set-up for a video streaming server for live streaming or



Figure 4-5
Capturing images
with Raspberry Pi 4
and a HQ camera

recording events, employing the HQ camera for astrophotography to capture images of celestial objects, etc.

There is also a keyboard and a mouse produced by Raspberry Pi Foundation. For a reasonable price, one gets a keyboard with three host USB ports to which USB mice, drives, and other USB-controlled devices can be connected (for details, please, see here: [Raspberry Pi Documentation; Accessories; Keyboard and mouse](#)).

Build HAT, Sense HAT, and TV HAT are the next three items worth mentioning. The Raspberry Pi Build HAT is an add-on board that can be used for controlling LEGO® TechniC™ motors and sensors with Raspberry Pi computers by simply connecting the HAT to the 40-pin GPIO header. Detailed instructions can be found here: [Raspberry Pi Documentation; Accessories; Build HAT](#). Sense HAT gives a Raspberry Pi an array of sensing capabilities (pressure, humidity, temperature, colour, orientation, and movement are built in). There is also an 8x8 RGB LED matrix, which allows you to visualize data from the sensors and a five-button joystick that lets you interact with projects. Look here for more detail: [Raspberry Pi Documentation; Accessories; Sense HAT](#). TV HAT allows users to receive digital terrestrial TV broadcast systems through an onboard DVB-T and DVB-T2 tuner. It also makes it possible to create a TV server that allows you to stream received TV over a network to other devices. See here for more information: [raspberrypi Documentation; Accessories; TV HAT](#). A TV HAT fits perfectly on a Raspberry Pi Zero, Zero W, or Zero 2W with soldered headers. One only needs a suitable aerial to receive broadcast TV. After installing the newest version of the Raspbian OS, open a terminal window and run the following two commands to install the TVHeadend software:

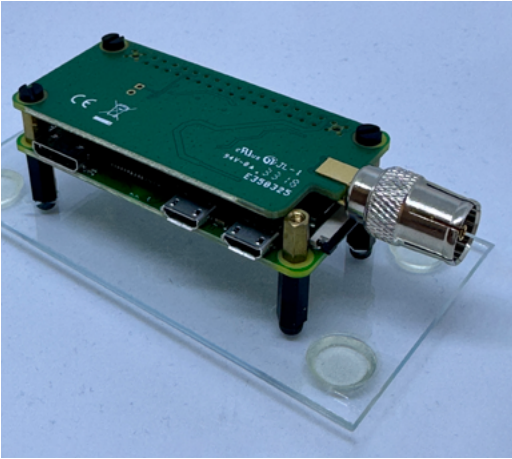


Figure 4.6
TV HAT on Raspberry Pi Zero

```
sudo apt update
sudo apt install tvheadend
```

During the TVHeadend installation you will be asked to choose an administrator account name and password. Put that down on a piece of paper in case you forget it. In a web browser on a different computer, type the following into the address bar:

```
http://raspberrypi.local:9981/extjs.html
```

Depending on your browser, the address above might not work in which case you will need to find out the IP address of the Raspberry Pi, which can be done by opening a terminal window on your Pi and running the command:

```
hostname -I
```

You will see a string of numbers, e.g. 192.168.1.32. Copy this and paste it into the address bar instead of the raspberrypi.local part of the address. When connected, sign in by providing the account name and password; leave Allowed network blank, and enter an asterisk (*) in the username and password fields. Then, choose DVB-T Network for Network type. After selecting your local TV transmitter, Save and click Next. In Service mapping tick all three boxes. Next, you should see all found services, e.g. 86 in a list of TV channels. To watch a channel in a browser, click the little TV icon to the left of the channel listing, which brings up an in-browser media player. To watch a TV channel in a local

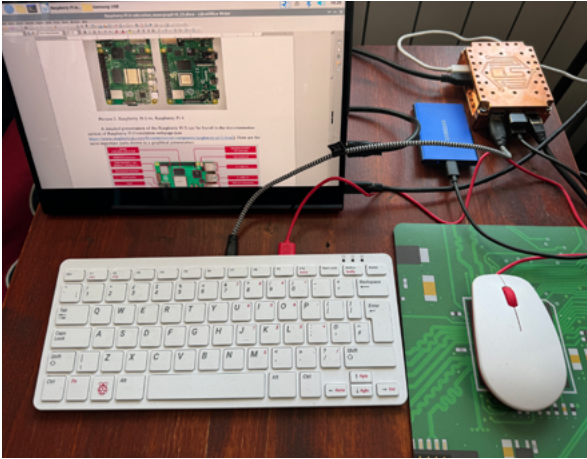


Figure 4.7
Raspberry Pi-based
desk top

Table 4.1 DAC HAT features

	Line out	Balanced out	Stereo speakers	Mono speakers	Headphones	Aux in	Aux out	Ext mic	Built-in mic
DAC Pro	✓	✓			✓				
DAC+	✓				✓				
DigiAmp+			✓						
Codec Zero				✓		✓	✓	✓	✓

Source Raspberry Pi Foundation

media player, download [VLC](#). Figure 4.6 shows our TV HAT on top of a Raspberry Pi Zero.

For music enthusiasts, the Raspberry Pi Foundation, having acquired IQaudio in 2020, offers four DACs. Below is a list of audio HATs with features.

All details on DAC HATs are available from: [Raspberry Pi Audio](#). All HATs are attached to the Raspberry Pi 40-pin header.

There is a product information portal for all items mentioned above here: [Raspberry Pi Product Information Portal](#), with PDF-based documentation available at: [Raspberry Pi Datasheets](#).

Figure 4.7 shows a working desktop set-up, including a Raspberry Pi 4 with the OS installed on a Samsung 1TB external SSD disk, a Raspberry Pi keyboard and mouse, and a high-quality touchscreen produced by DFRobot. The window currently open is LibreOffice Writer with a page of this document displayed on the screen.

4.2 Other Raspberry Pi accessories

There are other accessories a user needs to start a project:

- Raspberry Pi power supply (a 5V 3A USB-C PS for Raspberry Pi 4, and a 5.1V, 5A; 9V, 3A; 12V, 2.25A; 15V, 1.8A (power delivery) PS for Raspberry Pi 5). For older models (Raspberry Pi 3, Zero 2W) a 12.5W (5.1V/2.5A) microUSB PS is needed. There is also a PS for Build HAT (48W, 8V, 6A), which powers both Raspberry Pi (not Pi 400) and connected LEGO® Technic™ devices
- Micro HDMI® to HDMI® cable
- Mini HDMI male to HDMI female adapter
- Mini HDMI male to HDMI female cable
- USB A/male to micro USB/male cable
- USB micro-B to USB-C adapter
- Various Raspberry Pi cases, with or without fans
- Raspberry Pi active cooler (for Raspberry Pi 5)
- Raspberry Pi RTC battery (for Raspberry Pi 5)
- Raspberry Pi debug probe (for debugging).

4.3 Raspberry Pi for embedded applications and industrial applications

Raspberry Pi 4 in a smaller form factor is named Raspberry Pi Compute Module 4. It is suitable for integration into products (specifications are available here: [↗ cm4-product-brief.pdf](#)). The Compute Module 4 is available in 32 variants with different options (in terms of RAM size and eMMC flash sizes). A development platform for the Compute Module 4 is Compute Module 4 IO Board (see here: [↗ Compute Module 4 IO Board](#)). For industrial customers there is Compute Module 4S, which is intended for those migrating from Compute Module 3 or 3+. For previous models of Compute Module please go to the bottom of this page: <https://www.raspberrypi.com/products/>.

The Raspberry Pi Compute Module 4 plugs into a compatible carrier board (e.g. a Compute Module 4 IO Board). The Raspberry Pi Compute Module 4 IO Board is designed to allow connecting NVMe, SATA, networking, or USB cards with all important connectors on one side of the board. There are many alternative carrier boards to the official IO Board: Retro GPi CASE 2 (a GameBoy-inspired gaming handheld console built around the CM4), PiCam Module (a compact board to mount

a Raspberry Pi camera and a Compute Module 4), or Techbase ClusBerry 9500-CM4 (an industrial Raspberry Pi Compute Module 4 cluster).

On 19 October, 2020, Jeff Geerling tested the new Raspberry Pi Compute Module 4. His thorough review can be found here: [🔗 The Raspberry Pi Compute Module 4 Review](#). Raspberry Pi Compute Module 4 was featured in *The MagPi*, issue 99 (Raspberry Pi 2020).

4.4 Miscellaneous HATs

Raspberry Pi is an excellent platform for many projects. Having the right expansion board for a project is a prerequisite for any project you may think of building. In 2014, the HAT standard was introduced to connect HATs to the Raspberry Pi's 40-pin GPIO (btw, for Arduino, HATs are called 'shields'). HATs can be stacked, so you can put a HAT on top of a HAT and often share the same pins of a GPIO underneath. Many HATs on the market come with a Python module and are user-friendly, and many come with documentation and online resources. Let us have a look at some HATs. We have already mentioned the Raspberry Pi HAT for Lego projects, which is relatively cheap at 25 euros, works with all Raspberry Pis, and is certainly a good way to start playing with your child. Another cost-effective and useful HAT is Pimoroni IO Expander (10 euros), which adds an extra 14 GPIO pins. If you are keen on retro gaming, Pimoroni X HAT provides everything one needs for an arcade build and provides power to the Raspberry Pi. It is a good idea to use a GPIO pin extension stacker and a fan or heat sink for the Raspberry Pi, because gaming puts a load on the processor and as the board covers the CPU, Raspberry Pi can become very warm.

Pimoroni Breakout Garden is a HAT that enables multiple sensors to be connected through I2C. There are six slots, two SPI slots for breakouts (e.g. an LCD breakout), and four I2C slots, each with its own address so that different devices can be used at the same time. The Garden HAT comes with standoffs for easy installation on the Raspberry Pi and is recommended for science projects of all types.

Google AIY Voice kit with HAT may be the best HAT for building AI projects. Google assistant will help you find answers to some questions you might have after the project is finished, even if Raspberry Pi Zero or Zero W was your chosen small computer.

Automation HAT from Pimoroni is a monitoring and automation HAT that also belongs to the list of must-haves. So too is Pimoroni Unicorn, which is a HAT with a 64 LED (or 256 LED in an HD model)

matrix display, which, if nothing else, you can use to decorate your Raspberry Pi. Kids enjoy this HAT. If you want to play music with apples, oranges, and bananas, you need Adafruit Capacitive Touch HAT, with 12 capacitive sensors. Buy some alligator clips to connect fruits of your choice to the board and the only limit becomes your creativity. If, like me, you know little about clustering, a Cluster HAT can help you build your first cluster. If you are worried about power outages, the Uptime UPS Hat is ideal for adding a battery-powered UPS to Raspberry Pi. For robotics projects, a Robot HAT (or kit, including a HAT) offered by SunFounder (see: www.sunfounder.com) is a good choice.

Occasionally, when working on a project that requires a display, you will need a HAT to display data, like the Waveshare 2.7-inch E-ink display HAT. E-ink displays are good because they are compact and offer good legibility. The PaPiRus HAT is another option (several model sizes are available here: uk.pi-supply.com).

For mini-CCTV systems, the Pan-Tilt HAT can be sourced from Pimoroni at shop.pimoroni.com. It is equipped with horizontal and vertical motion servos that give a Raspberry Pi camera movement. The Pan-Tilt HAT can be used for face-tracking in a robot project to give the robot eyes. Pimoroni has also put together a Python library to make things super easy for beginners.

4.5 Miscellaneous tools and accessories for DIY electronics experiments and projects

Below is a list of tools and accessories one often needs when embarking even on a simple Raspberry Pi project. A Raspberry Pi Zero, for instance, comes in two varieties: with a 40-pin 2x20 male HAT header soldered to the PCB, or without it, in which case you need a basic soldering iron, some solder, and something to prevent your Raspberry Pi Zero from moving. Electronic hardware projects always require a set of tools, some of which are mentioned below.

1. A decent soldering iron with a soldering stand or, if funds allow, an 80W soldering station for advanced projects.
2. Solder in the form of a roll of 60/40 (tin/lead) solder with a flux core
3. Long nose pliers
4. Wire cutters
5. Wire strippers



Figure 4.8
Adapters

6. Breadboard
7. Helping hand
8. Solder sucker and braided solder wick
9. Multimeter
10. Jumper leads and wires
11. Magnifier lamp
12. Some resistors, capacitors, push buttons, transistors, and LEDs to start basic projects
13. Scalpel
14. Screwdrivers
15. Safety glasses
16. Various sensors (ultrasonic, PIR, etc.)

The list could go on, but we would advise the reader to have a look at *The MagPi*, issue 64, where physical computing is explained in more detail than we can afford to do here (Raspberry Pi 2017a). On the other hand, we must point out once again that one of the reasons for writing



Figure 4.9
Soldering set

this monograph was, from the very beginning, the idea that there are too many users of gadgets and too few makers, hackers, and fixers.

There are some other items that one may need: a Raspberry Pi 400 GPIO adapter (which makes the GPIO pins on Raspberry Pi more accessible); a micro USB to USB-C adapter (the power supply for Raspberry Pi 3 or Raspberry Pi Zero uses a micro USB connector; Raspberry Pi 4, on the other hand, uses a USB-C connector); occasionally a micro USB to USB-A adapter is needed (for some accessories); an HDMI to HDMI mini adapter (for connecting a monitor or TV to a Raspberry Pi Zero family of computers with an HDMI to HDMI cable); and a plug adapter (for when you purchase a power supply with a wrong plug).

Figure 4.9 shows a soldering set consisting of (from upper left corner, upper row) an 85W Weller soldering station with soldering iron connected to it, soldering iron tip cleaning sponge, and a third hand with a magnifying glass. In the middle row there are soldering flux paste, a desoldering wick, and a soldering tip reactivator. The third row shows (from left to right) a solder sucker, syringe solder flux, circuit board holder, a 0.9 mm WBT silver solder, and a 0.5 mm ordinary solder. What is missing from the Figure 4.9 are spare soldering tips.

5 Raspberry Pi usage

It is no wonder that Raspberry Pi, being a low-cost and compact computer, has been so widely used in many areas, ranging from DIY communities to companies and in many scientific fields for a wide range of applications. The small computer can be used, for example, in waste processing facilities for sorting recyclables with image recognition implementation; or in educational settings to get acquainted with computer hardware and software as well as for learning programming languages like Python and Scratch. It can be used in home computing and entertainment, for web browsing, for word processing, and, indeed, is quite often used as a low-cost media centre with software like OSMC or Plex to stream music or videos. Hobbyists and DIY enthusiasts may use it for home automation or robotics, because its GPIO pins allow for hardware interfacing using sensors, motors, and other components. In industry, it is often used for prototyping, where its low-cost and flexibility make it ideal for developing new technologies and testing them before production. Various industries use Raspberry Pi for controlling and monitoring equipment, data collection, and automation tasks.

Raspberry Pi plays an important role in the Internet of Things (IoT) because it enables remote connection to the Internet and interfaces with sensors and various devices, using applications ranging from home automation to industrial solutions. Due to its affordability, versatility, and reasonable price, Raspberry Pi can be found in many home security systems and weather monitoring stations as well as in larger networks of interlinked smart devices in industrial settings. Readers willing to try a DIY IoT project may visit the ALL3DP web page here ([🔗 The Best Raspberry Pi IoT Projects of 2023](#)) to see some of the best small IoT projects that can be connected to an IoT network or IoT device communication platform like Homebridge (available here: [🔗 Homebridge Raspberry Pi Image](#)). Projects range from a smart home hub to a Bluetooth speaker, Picroft voice assistant, child sleep moni-

tor, attendance system, remote healthcare monitoring system, water level sensor, fruit detection system, and IoT fan, to name but a few.

The humble computer has the power to connect the education sector with industry through common projects started at universities and implemented in companies, because prototyping can be started in research labs at educational institutions and later produced by companies. In addition, companies use Raspberry Pi for real-time monitoring of manufacturing processes or tracking of the production line performance, thus helping enterprises to optimize operations and predict maintenance needs. With cameras and sensors installed, Raspberry Pis can be used to inspect products for quality assurance by identifying defects, ensuring products meet set standards and even guiding robots for automated repairs. Smaller businesses may use Raspberry Pis as cost-effective point-of-sale systems (for managing transactions, inventory, and customer data) or implement them in interactive displays that display information (product information, advertisements) aimed at attracting customers.

In the field of smart farming, Raspberry Pis can be used – and indeed are – to monitor soil moisture, weather conditions, crop growth, automate irrigation systems, control greenhouse environments, or track the location of livestock.

In healthcare, Raspberry Pis are used in affordable patient monitoring systems for tracking vital signs and managing patient data. Whitaker (2022) reports that Raspberry Pi is a capable and reliable development platform for medtech worth serious consideration, especially the Compute Module, which offers all the capabilities and development possibilities of similar modules that are positioned as industrial development platforms for medical devices.

There are few fields into which Raspberry Pi has not yet found a way. Some researchers and DIY enthusiasts have built clusters of Raspberry Pis to experiment with parallel computing architectures, distributed systems, and network protocols, others implement Raspberry Pi solutions in robotics and AI, others see its use in transportation and logistics, where companies implement Raspberry Pis for tracking vehicles and route optimization; local authorities in cities use it to monitor and control traffic lights, collect traffic data, and implement intelligent transportation systems, or build surveillance systems that can either monitor premises or record video and alert owners of activities.

It is necessary to point out that there are other alternative single-board computers available on the market (Odroid, BeagleBoard, Banana Pi, Jetson Nano, etc.), but they have limited support and offer poor documentation and tutorials (for more information see www.raspberrypi.org). It is also true that for some applications, alternatives with higher computing power can nevertheless be a preferred solution to Raspberry Pi.

5.1 In research

Balon and Simić (2019) discuss the use of Raspberry Pi in higher education and find the credit card-size computer suitable for improving students' skills in both computer science and for improving their basic knowledge of electronics. They consider Raspberry Pi an ideal platform for acquiring these competencies.

Emani et al. (2019) described the use of Raspberry Pi in the cybersecurity bachelor's degree at Pennsylvania State University, where students created a smart home model to perform reconnaissance and penetration testing using Raspberry Pi computers.

Gooch et al. (2022) described the challenges of teaching parallel and distributed computing using Raspberry Pi clusters to open distance university students. Based on the survey results from almost 500 students, they argued that there are many benefits that remote practical activities had on teaching.

In a conference paper by Ponugumati et al. (2023) the authors described how Raspberry Pi can be used in a smart environment by proposing a design using Raspberry Pi as an edge node (BRIoT) with blockchain implementation in IoT for addressing security issues. Kurkovsky and Williams (2017) also explored the topic of IoT and examined several hardware platforms for IoT and described a student project implementation using a Raspberry Pi with several different sensors.

Soerensen et al. (2016) presented the key steps to design, set-up, and maintain an inexpensive testbed using Raspberry Pi devices for communications and storage networks with network coding capabilities. They described in detail how to set up interconnected Raspberry Pis with memory cards for local storage, a Raspbian Lite image, network connectivity, and proper system administration privileges.

Jolles (2021) gave a thorough account of the use of Raspberry Pi in biology, where Raspberry Pi is used to collect and monitor environmental data, animal behaviour, high-throughput behavioural record-

ing, large-scale plant phenotyping, underwater video surveillance, closed-loop operant learning experiments, and autonomous ecosystem monitoring. The paper also provided guidelines for its use, including the Raspberry Pi set-up, networking, and media recording capabilities. Jolles concluded by saying that Raspberry Pi has been widely used in the biological domain by adding to the above-mentioned possibilities greenhouse monitoring, fluorescent microscopy, virtual reality experiments, and integrated RFID-tag solutions (Jolles, 2575).

Revathi and Sasikaladevi (2022) provide an insight into the Raspberry Pi application for language identification systems (LIDs), where the tiny computer is used for recognizing the language in which the utterance is spoken remotely and recognized through an IoT-based communication system.

5.2 In schools

Raspberry Pi was developed to promote the teaching of basic computer science in schools. In addition, as the last ten years have proved, it is an excellent tool for teaching because the learning curve is short and there is an abundance of online support available for beginners. As Fletcher and Mura (2019) reported, Raspberry Pi can be used not only for teaching the basics of computational science, but also for familiarizing students with scientific processes, i.e. from designing experiments, building instruments, collecting and analysing data, to interpreting results. We are not wrong if we say that there is no other single-board computer available that can provide so much valuable hands-on experience for such a small investment. Our readers can find more teaching resources on [↗ raspberrypi.org](https://www.raspberrypi.org).

Kölling (2016) pointed out that Raspberry Pi's initial purpose was to encourage pre-university students to engage with programming (Pi in the name refers to the programming language Python). Kölling gave a thorough historical account of Scratch, Python, and Java use on Raspberry Pi, and pointed out how Greenfoot and BlueJ environments offered new possibilities, because they are easy to program interactive animated graphical applications by offering multimedia support, which increases student engagement.

Primary and secondary education

On the raspberrypi.org web page ([↗ The Computing Curriculum](https://www.raspberrypi.org/teaching-computing)), teachers can access free resources to help them teach computing to

students aged between 5 and 16 years old. The materials are divided into 77 units subdivided into more than 500 lessons. For each stage, the Raspberry Pi Foundation provides a teacher guide and a curriculum map, together with detailed lesson plans, slides, activity sheets, home assignments, and assessment. The resources are regularly updated based on the latest research and teachers' feedback.

The Raspberry Pi Foundation also provides a range of computer courses for both beginners and experienced programmers to help them learn coding (see [🔗 Learn to program in Python](#)). Courses cover Python, Scratch, AI and machine learning, web design, and cybersecurity, to name but a few.

Another valuable resource made available by the Raspberry Foundation is a free computing and digital making magazine named *Hello World* (available here: [🔗 Hello World](#)). The last issue, for example, is devoted to teaching and AI, bringing readers closer to recent developments in artificial intelligence. This issue is an excellent resource for students wishing to become more AI literate.

Code Club ([🔗 Code Club.org](#)), yet another valuable resource curated by the Raspberry Pi Foundation, provides resources and projects for anyone who would like to run a Code Club for children aged nine and above at any school. Step-by-step project guides (available here: [🔗 Discover our projects and paths](#)) ensure that even those without any prior coding knowledge can learn the necessary skills for coding in Scratch, Python, and HTML/CSS. The global community of Code Club spans from India to the USA. There are 13,000 code clubs around the world, with 180,000 young people learning to code in 160 countries.

The Raspberry Pi Foundation, together with the University of Cambridge, developed Ada Computer Science, a free online platform for teachers and students worldwide. The resources available for teachers and students who sign up are tailored to GCSE and A-level computer science exam specifications. There is an abundance of real code examples in Python, C#, VB, and Java available to support the learning of students.

Quinlan and Baloro (2018) visited 21 primary, secondary, and other schools across England and Scotland, 15 of which employed a teacher who was a Raspberry Pi Certified Educator and another six that had received Raspberry Pi computers as part of a Google giveaway in 2014. They found that Raspberry Pi computers were mostly used for their potential in physical computing in a variety of projects, which tended

to be simple and instruction-led. They also found that the management and tech support of Raspberry Pi computers was often the responsibility of teachers, not technicians.

Tertiary education

Many HEIs provide curricula and project repositories that are valuable for education purposes and often incorporate complex and specialized uses of the Raspberry Pi in different academic fields. The [Raspberry Pi Computing Education Research Centre](#) was established as a joint initiative between the University of Cambridge and the Raspberry Pi Foundation with the aim of increasing their understanding of teaching and learning computing, computer science, and associated subjects.

Limbo, Nhinda and Sverdlík (2017) proposed replacing desktop-based instruction in university-level introductory programming classes with the Raspberry Pi, stating that the small and cheap SoC is important as a microprocessor-based embedded system.

Many researchers see the following advantages of using Raspberry Pi in their research over available commercial solutions:

- Large processing power on a compact board
- Large number of dedicated interfaces (UART, I2C, SPI, I2S, CSI, DSI) to connect a wide range of sensors and electrical components
- High connectivity (HDMI, USB, Ethernet, Wi-Fi, Bluetooth)
- Reasonable price, ranging from as little as €5 (RPi Zero; €90 for the RPi 5, 8 GB)
- High ease of use with a huge user community, extensive resources, and easy-to-understand tutorials
- Works both headless and as a full desktop computer
- Is open source
- Is being constantly improved and does not become obsolete
- No extensive programming experience is required and it has a short learning curve, but some trial-and-error can be expected
- Easy to deploy and highly portable due to its size
- Long-term automated image and video recording with high customization
- Built-in HDMI capable graphics (up to two times 4K for latest models)

- Low power consumption (but higher than microcontrollers) and can draw power from a wide range of external sources, ranging from traditional power supplies to batteries to solar panels
- It contains no moving parts and is silent
- Highly customizable and flexible compared to commercial solutions

5.3 Machine learning

Artificial intelligence refers to the data science technique aimed at replicating human intelligence, which encompasses thinking and acting both humanly and rationally (Norvig and Russell 2021, 19–20). As a result, the subject of AI and its derivatives is extraordinarily complex, often leading to significant confusion and misconceptions. This confusion, in turn, frequently results in terms such as ‘artificial neural network’, ‘machine learning’, and ‘deep learning’ being inaccurately interchanged. While these terms are certainly related, they each denote distinct subfields within the vast domain of AI. Thus, AI includes fields such as natural language processing and computer vision, as these fields aim to equip machines with the abilities inherent to human intellect – namely, speaking and seeing.

Thus, *machine learning* is also a subfield of AI since it enables the machine to learn specific tasks by constructing models grounded in observed data and acquiring the ability to predict potential future scenarios (Norvig and Russell 2021, 669). Furthermore, *deep learning* represents a variation of machine learning that utilizes neural networks, encompassing both supervised and unsupervised approaches. A *neural network*, in turn, comprises individual ‘neurons’ that abstractly resemble the structure of neurons within the human brain (Norris 2019, 213). These neurons are interconnected by connections and structured into multiple layers. Thus, the machine’s learning process can be adjusted by changing the number of layers and the weights of the connections (Norvig and Russell 2021, 801–2).

Machine learning algorithms are generally intended to deal with various classification, detection, recognition, and prediction problems (Norris 2019, 214). As a result, they are frequently used in image recognition, text-to-speech, text-to-image processing, and the development of large language models. The rising popularity of such areas has led to the ubiquitous integration of machine learning algorithms

across industries. Thus, numerous tools have been created to simplify and speed up the process of developing and implementing machine learning solutions. When it comes to Raspberry Pi devices, there are two common approaches for delving into machine learning. These approaches involve either utilizing external Python modules or making use of Mathematica's built-in functions.

Python libraries

In order to take advantage of Python's machine learning libraries, one would most probably need to install such libraries as Seaborn (Waskom 2021), Pandas (McKinney 2010), Numpy (Harris et al. 2020), and Matplotlib (Hunter 2007). These libraries are not only common dependencies but also provide useful methods for data processing, manipulation, and visualization.

The subsequent steps depend completely on the particular machine learning model. For instance, if one intends to generally explore and experiment with a wide range of machine learning algorithms, installing Scikit-learn (Pedregosa et al. 2011) could be a viable option. Similarly, the OpenCV (Bradski 2000) library serves as a versatile solution for various computer vision projects (Monk 2022, 225–238). In addition, the Pillow ([↗ https://python-pillow.org](https://python-pillow.org)) library offers a range of general image processing methods that are useful for the pre-processing and augmentation of the image data set.

On the other hand, such libraries as Keras ([↗ https://keras.io](https://keras.io)) and TensorFlow (Abadi et al. 2016) provide a comprehensive solution for deep-learning tasks. TensorFlow offers a range of pre-trained models that enable users to experiment with object and whistle detection, as well as sound identification (Monk 2022, 240–48). Conversely, Keras is a high-level framework that uses TensorFlow as a backend. Thus, it provides modules and methods designed for developing and training neural networks, saving and importing models, applying various activation functions, among others (Norris 2019, 324–325). However, one would need to switch to the 64-bit version of Raspberry Pi OS (Raspberry Pi Foundation, n.d.-a), as all three libraries are incompatible with 32-bit systems.

Wolfram language

Mathematica includes an array of pivotal machine learning algorithms, allowing its users to perform linear regression, logistic regression, and

data clustering tasks. The algorithms are contained within specialized methods and can be effectively implemented through the Wolfram language (Alva 2021, 273). Furthermore, Mathematica has a comprehensive framework for creating custom neural networks. This framework encompasses various functionalities, such as data vectorization, different types of layers, various activation functions, as well as model importing, exporting, and visualization capabilities (Alva 2021, 331). Finally, there is a Wolfram Language neural net framework (Wolfram, n.d.-b), which provides free access to various types of pre-trained machine learning models.

5.4 Internet of Things

The Internet of Things is a collection of interconnected computing devices that are capable of communicating with one another as well as interacting with the physical world (Rose, Eldridge, and Chapin 2015, 11–12). However, to streamline operations and avoid the need for individual communication with each component of an IoT system, this process is centralized and executed on a master node. Thus, the master node receives commands from the user, sends them to the specific IoT appliance, and subsequently relays the appliance's response back to the user. It is noteworthy that Raspberry Pi devices are commonly chosen to take on the role of master nodes due to their affordability and extensibility.

Applications in industries

Raspberry Pi devices have found applications in industries typically involved in production, healthcare, robotics, and cloud computing. Despite the apparent differences, these industries share a common focus on seeking pragmatic solutions in several key areas, which involve real-time monitoring, wireless communications, and remote control. In this sense, they all express interest in the IoT and its potential advantages over manual labour. These benefits encompass increased productivity, greater operating and cost effectiveness, enhanced data analysis capabilities, and the convenience of ubiquitous connectivity (Rose, Eldridge, and Chapin 2015, 9). On the other hand, these advantages might also give rise to certain challenges, depending on the particular application and the surrounding context of the IoT system. These issues are related to security, privacy, standardization, compatibility, and legal implications (Rose, Eldridge, and Chapin 2015, 45).

Nevertheless, healthcare facilities could employ a Raspberry Pi-based IoT system to monitor patients' body temperature, heart rate, respiration, and body movements. Subsequently, the status of multiple patients can be transferred to the server, allowing doctors to remotely review and analyse the data (Kumar and Rajasekaran 2016; Pardeshi et al. 2017).

Similarly, Raspberry Pi devices could be integrated into the existing sensor networks of various industrial companies and manufacturing facilities. By replacing the conventional wired connection between the sensors and the master node with a wireless protocol (e.g. Bluetooth, Wi-Fi, or GPRS), companies can expand their sensor networks rapidly and efficiently. This significantly decreases the number of blind spots by allowing the sensors to be installed on moving parts or in remote places (Senthilkumar et al. 2016). Furthermore, the wireless sensor node (WSN) itself is highly portable, making it easy to maintain and upgrade (Patil et al. 2018).

The aforementioned systems might be employed to monitor and optimize their current energy consumption (Mudaliar and Sivakumar 2020; Zanzmeriya and Panara 2018) along with the factors influencing the production environment. These factors encompass light intensity (N. P. Kumar and Jatoth 2015; Sorte et al. 2021), temperature, humidity, pressure, and air quality (Karankumar D., Shreya B., and Kapil S. 2014; Sunehra 2019).

In addition to mere monitoring, some organizations might want to actively respond to the changing environment detected by sensors. These responses can range from the remote control of motors and cameras to the deployment of full-fledged robots. Similarly, these actions can be performed automatically through an IoT system.

Thus, Raspberry Pi devices could be employed to automate traffic control by analysing traffic density through image processing (B. V. Kumar et al. 2020). Moreover, they could serve as a basis for industrial controllers, offering an alternative to the existing programmable logic controllers (PLCs) (Vieira et al. 2020). The IoT framework may also include surveillance robots designed to operate in manufacturing or hazardous environments (Santhosh Krishna et al. 2016). Furthermore, some potentially useful applications might include a system for controlling the flow rate of pipelines (Suresh et al. 2014), or food storage and colour control in the food industry (Endres et al. 2022).

5.5 Home automation

The application of Raspberry Pi devices in home automation is analogous to their use in the aforementioned industrial settings. In this case, however, rather than controlling industrial machinery, the IoT system includes an array of ‘smart’ home appliances. These electronic devices can be classified into two main categories: ‘two-state’ devices (e.g. lights, doors, curtains) and ‘calling’ devices (e.g. heaters, fridges, boilers, cameras). The term ‘two-state’ refers to devices that can be in one of two states at any given time (i.e. on/off or open/closed) and are capable of only receiving commands. In contrast, the ‘calling’ appliances are equipped with sensors, enabling them to transmit their measurements back to the master node (Stojanoski et al. 2017).

Developers can utilize similar wireless communication protocols, as well as some less formal methods, to send and receive data from their master nodes. Thus, in addition to Wi-Fi (i.e. LAN or WLAN) and Bluetooth, which are limited to local use near the master node, individuals could decide to extend their connectivity via mobile networks (Venkatesh et al. 2018).

One popular way of effectively communicating with an IoT system over the Internet is through a graphical control centre. For instance, it could be a GUI program written in Python (Jamil and Ahmad 2015), a web application (Patchava, Kandala, and Babu 2015; Venkatesh et al. 2018), or a mobile application (Stojanoski et al. 2017). To avoid the development of a custom control centre, one may want to consider utilizing existing solutions, such as an IR or radio remote control. Alternatively, one could send commands to the master node via email, specifying the command in the messages’ subject (Jain, Vaibhav, and Goyal 2014).

Finally, a valuable feature to incorporate into a home automation system is a response mechanism, allowing the master node to send notifications to the owner. For instance, this could be used for fire alarms or motion detection alerts. Such functionality can be achieved through various methods, with the most commonly chosen option being SMS (Bin Bahrudin, Kassim, and Buniyamin 2013; Patchava, Kandala, and Babu 2015).

In DIY projects

The Internet is full of interesting and useful projects, many of which are super simple, while others might be extremely complex. The offi-

cial Raspberry Pi website is a good place to start ([↗ www.raspberrypi.org](http://www.raspberrypi.org)). Also curated by the Raspberry Pi Foundation, is the *MagPi* magazine, available both online ([↗ magpi.raspberrypi.com](http://magpi.raspberrypi.com)) and in print. GitHub ([↗ github.com](http://github.com)) is a platform where many developers share their code and projects. Hackster.io ([↗ www.hackster.io/raspberry-pi](http://www.hackster.io/raspberry-pi)) is a community-driven platform with a section devoted to Raspberry Pi projects, complete with step-by-step instructions, photos, and videos. Project ideas can also be found on YouTube, Raspberry Pi forums, technology blogs and websites, on social media platforms like Twitter, Instagram, and Pinterest or at local maker spaces and tech events in your area.

DIY projects do matter, as they promote a culture of lifelong learning, they can represent a stepping stone for entrepreneurial endeavours, and contribute to a more sustainable lifestyle (remember, a single Raspberry Pi can be reused for all projects described in our monograph). DIY projects are often a cost-effective alternative to purchasing ready-made products (a computer, a NAS, a DAC, etc.). Engaging in DIY projects not only facilitates the acquisition of new skills but also nurtures creativity and innovation within individuals.

6 Programming on the Raspberry Pi

Despite deliberately exposing more of their insides than other computers, the eponymous products of the Raspberry Pi Foundation are still essentially general-purpose computing devices. Consequently, the programming process is not drastically different from that on other platforms (Korom and Illés 2022). Nevertheless, what sets the Raspberry Pi apart is its capability to transcend the boundaries of software-level applications. By incorporating GPIO pins, users are enabled to take their projects into the domain of physical computing, making software interact with the real world through external hardware (Hal-facree 2020, 120–121). This opens up a vast array of possibilities for creative projects that merge both software and hardware. Fortunately, users have the flexibility to choose from a variety of programming languages and related tools.

6.1 Selecting a programming language

The choice of a programming language pivots on the project’s end-goals, specific requirements (e.g. dependence on external libraries, utilization of GPIO pins, etc.), as well as various other factors associated with the given project. In this regard, Spinellis (2006) emphasizes several key factors to consider: ‘programmer productivity, maintainability, efficiency, portability, tool support, and software and hardware interfaces.’ Furthermore, one would also want to consider the code length, development time, memory consumption, and runtime efficiency (Prechelt 2000).

The attributes of programming languages and the projects developed using them substantially depend on the language’s level of abstraction. Levels of abstraction refer to the hierarchical organization of software and programming concepts, which are constructed layer by layer on top of the system’s hardware. Each new layer adds more abstraction, thus disguising the details and complexity of the previous layers, allowing developers to simplify the process of problem-solving

(Kollár, Pietriková, and Chodarev 2012). In practice, the concept of levels of abstraction encompasses a diverse array of elements. These include components such as GUI, object-oriented programming (OOP), application programming interfaces (APIs), external modules, libraries, and various fundamental programming aspects such as functions, variables, and data types.

Thus, although *high-level programming languages* are designed to abstract away the machine code and provide a more user-friendly way of expressing algorithms, they tend to sacrifice intricate control over the hardware and runtime performance. On the other hand, *low-level programming languages* are closely related to the machine's architecture and, therefore, are usually more efficient and optimized at the cost of complexity and human-readability (Winograd 1979).

Moreover, variations in levels of abstraction usually suggest differences in the type of language processor being used, often either a compiler or an interpreter or a combination of both. A *compiler* translates the entire source code of a program into executable machine code while simultaneously detecting and reporting any errors that might arise during this process (e.g. in C and C++). On the other hand, an *interpreter* translates the source code line by line, directly executing the specified operations (e.g. in Python, Ruby, Wolfram language, Perl, and Bash). A hybrid of both approaches would involve initially compiling the source code into an intermediate form, which is subsequently interpreted and executed (e.g. in Java and C#) (Aho et al. 2007). While not an absolute rule, low-level languages are generally compiled, whereas high-level languages tend to be interpreted.

6.2 Available programming languages

Even though the pool of programming languages available on Raspberry Pi devices is almost limitless, the selection is generally determined by the OS that a user decides to install. In this case, the two major competing OS types are Windows (e.g. Windows 10 IoT) and Linux (e.g. Raspberry Pi OS, Ubuntu, Manjaro, etc.). Various OS families exhibit varying degrees of compatibility and support for particular programming languages.

As a result, they have different native, supported, and pre-installed environments and components (such as compilers, interpreters, frameworks, development environments, etc.). For instance, C# development is native to Windows systems, although it is also supported

on Linux. However, the required environments are not available by default on either system. On the other hand, Linux systems typically come with Python interpreter and C compiler already installed as part of the system set-up. However, Windows does not include these tools by default, despite them being fully supported.

As a Debian-based Linux distribution, the 32-bit Raspberry Pi OS comes equipped with a default selection of fully supported programming languages intended for various applications. Among these languages, the most notable ones are Python, C and C++, Perl, and Bash. Moreover, the Full 32-bit version of Raspberry Pi OS (Raspberry Pi Foundation, n.d.-b), which will be our primary focus, incorporates several programming languages that are not typically integrated into the Linux system by default, such as Java and Ruby.

Finally, some users may be drawn to the independent development environments that come with their own interfaces and tools. Thus, the Raspberry Pi OS also encompasses platforms such as Scratch, Mathematica, and Sonic Pi.

In the subsequent sections, we will delve into a more detailed discussion of these languages and development environments, specifically those that are pre-installed on the Raspberry Pi OS Full version. Additional programming languages and their dependencies can be installed (as well as updated) via `apt` package manager.

6.3 Overview of programming languages

Code-based programming implies the creation of software by describing an algorithm using the syntax of a particular programming language. This process typically occurs within an integrated development environment (IDE) or a general-purpose text editor. Raspberry Pi OS offers a range of language-specific IDEs that are aimed at enhancing the software development process by offering various useful features, including syntax highlighting, built-in compilers or interpreters, and library management, among others. Collectively, these features assist users in creating and executing programs. However, their effectiveness, particularly concerning defect injection rates, remains a topic of ongoing debate (Gómez et al. 2017).

Since Raspberry Pi OS is a Linux system, it is equipped with two standard terminal-based (i.e. command line) text editors: Nano and Vim. It is worth noting that on the Raspberry Pi OS, Vim is launched as a link to Vi. Alternatively, some users may prefer Geany

([↗ https://www.geany.org/](https://www.geany.org/)), a versatile text editor with a GUI. Once again, additional IDEs and text editors can be found in the Raspberry Pi OS's repositories and installed via the `apt` package manager.

Python

By default, Raspberry Pi OS includes both versions of Python, namely Python 2 (Python Software Foundation 2020) and Python 3 (Python Software Foundation 2021), to ensure compatibility. However, for our current purpose, we will exclusively focus on Python 3, as it is the most recent version and Python 2 no longer receives support or updates (Python Software Foundation, n.d.). Although Python 2 is still utilized to some extent, users are strongly recommended to primarily rely on Python 3, resorting to Python 2 only when confronted with compatibility issues (Monk 2022, 139–40).

Raspberry Pi OS users are provided with two pre-installed IDEs specifically tailored for Python programming: Mu (Tollervey, n.d.) and Thonny ([↗ https://thonny.org/](https://thonny.org/)). While both programs are decent, Thonny is receiving significantly more attention in the guides and is endorsed more frequently (Halfacree 2020, 93; Monk 2022, 140).

Python is an extremely versatile high-level programming language that is known for its extensibility through a wide selection of libraries. If not in the system's repositories, additional libraries are typically installed through the `pip` (Python Software Foundation 2021) package installer. This flexibility enables developers to create programs capable of interacting with both the software and hardware components of Raspberry Pi devices. The scope of software applications extends significantly beyond mere data input and output (e.g. through the terminal) and the manipulation of user files. Thus, for instance, Python can be employed for developing GUI programs and games, aided by modules such as Pygame (2019) and PyQt5 (Riverbank Computing, n.d.).

Furthermore, the `mcpi` (O'Hanlon, n.d.) Python library empowers users of the Raspberry Pi to engage with the native version of Minecraft Pi (Mojang AB, n.d.), a popular sandbox video game. Once a 'new world' is created in Minecraft Pi, users gain the ability to manipulate it through Python code. This includes a wide array of actions, from printing out messages to in-game chat, changing a player's location, placing single or multiple blocks, as well as monitoring and modifying their states (Philbin 2017, 130–143).

The scope of hardware applications (i.e. physical computing) with Python is even greater than that of software. This is primarily due to the expandability of Raspberry Pi devices through the integration of supplementary electronic components. To harness these extensive capabilities, Raspberry Pi OS comes equipped with an additional Python library, `gpiozero` (Nuttall 2021). The library gives users the ability to control the GPIO pins and, consequently, the hardware using abstract modules and methods. Additionally, the potential can be further expanded by combining `gpiozero` with other built-in Python modules (e.g. `time`, `random`, `os` etc.).

C and C++

When compared to Python, both C (Kernighan and Ritchie 1988) and C++ (Stroustrup 2013) are referred to as low-level programming languages and considered to have higher runtime efficiency. As a result, both of these languages require a compiler for software development. It is noteworthy that the Linux kernel itself is written in C and C++ (Jones 2007), and as a consequence GNU Compiler Collection is pre-installed on the majority of Linux systems, including the Raspberry Pi OS. For editing and executing the code, practitioners suggest using Nano (Ibrahim 2021, 34–35) along with the terminal or configuring Geany by specifying the appropriate compiler options (Tavasalkar 2019, 57–59; Ibrahim 2021, 110–112).

Creating programs in C and C++ on a Raspberry Pi does not require any additional prerequisites unless the user intends to interface with GPIO pins. In that case, one of two commonly used libraries could be utilized: `pigpio` (Pigpio, n.d.) or `wiringPi` (Henderson 2019a). While `wiringPi` remains an older and more popular option that can still be installed (Ibrahim 2021, 107–108) (albeit not through `Apt`), it is worth noting that this library no longer receives support from its developer (Henderson 2019b). In contrast, `pigpio` is currently supported, comes pre-installed on Raspberry Pi OS, and offers similar features to the deprecated `wiringPi`.

Perl

The Perl (Wall, Christiansen, and Orwant 2000) programming language is considerably less popular than Python, C, and C++ for general software development. Alas, it is certainly less popular among the users of the Raspberry Pi OS and is not frequently chosen as a tool

for implementing Raspberry Pi projects. Nevertheless, it comes pre-installed on the majority of Linux distributions, including the Raspberry Pi OS. Being a high-level programming language, Perl is comparable to Python in terms of runtime efficiency, memory consumption, and development time (Prechelt 2000).

Similarly to Python, C, and C++, Perl has an extensive support of libraries, an example being the aforementioned `wiringPi`. As a result, the users can utilize Perl modules to interface with GPIO pins and, therefore, with external hardware (Murray and Bertrand 2018, 29–43). The language also provides modules that support inter-integrated circuit (I2C) and serial peripheral interface (SPI) communication, making it possible to interact with a range of other devices via serial communication channels (Murray and Bertrand 2018, 46–59). Finally, Perl's modules make it simple to interact with other electronic boards, such as Arduino (Murray and Bertrand 2018, 65–69), through the Firmata (Mellis et al., n.d.) protocol, even enabling control over HTTP (Murray and Bertrand 2018, 83–87).

Java

Java (Arnold, Gosling, and Holmes 2005) is another popular high-level programming language that is pre-installed on the Raspberry Pi OS. Nevertheless, this language is significantly centred around OOP and is designed to be independent of any particular OS. As a result, any software written in Java is extremely portable and can be executed on any device with the Java runtime environment (JRE) installed. In turn, the JRE encompasses the Java virtual machine (JVM), which compiles the source code, and basic class libraries. On the other hand, to not only run Java software but also to develop it, one would need to install the Java development kit (JDK), which already includes the aforementioned JRE (Flurry 2021, 44).

The Raspberry Pi OS already includes the JDK, along with two IDEs: Greenfoot (Kölling 2015) and BlueJ (Barnes and Kolling 2016). Greenfoot, offers a graphical environment that enables users to visualize and interact with the objects and classes they create using the built-in editor. On the other hand, BlueJ serves as another IDE specially designed for students and teachers, aiming to simplify the learning and teaching of Java's OOP concepts. Nevertheless, Flurry (2021, 15–17) still advocates for remote development, expressing concerns about perfor-

mance limitations, delays, and the challenges of conducting realistic testing on a Raspberry Pi device.

Ultimately, Java offers a comparable set of possibilities to the aforementioned languages concerning software development and interfacing with external hardware (through GPIO, I2C, and SPI). However, the advantages of Java for Raspberry Pi projects further include modularity and reusability due to its emphasis on OOP, improved safety through static typing, resulting in a lower probability of bugs, the capability to develop and test on separate hardware (due to JVM), a comprehensive array of libraries, greater industry support, and superior performance, at least when compared to Python (Flurry 2021, 6–14).

Bash

Bash scripts can incorporate any of the GNU core utilities and be used to automate various laborious or frequent tasks. For instance, create a project directory, log program output, rename files in bulk, check file existence, and concatenate long commands, among numerous others. The versatility and utility of these scripts have great potential to streamline users' workflows and enhance their efficiency.

6.4 Overview of development environments

In contrast to the typical relationship between a programming language and an IDE, certain development environments are entirely self-sufficient. This implies that all required dependencies are already included and securely isolated within the environment itself. As a result, the entire development process, from writing to executing the source code, can be done within this self-contained framework. Thus, such development environments provide a safe platform for experimentation and learning, significantly reducing the risks of dependency conflicts and system breakdowns.

Scratch

Undoubtedly, Scratch (Maloney et al. 2010) stands out as the most recommended first tool for novice users to explore programming. This is further emphasized by its consistent appearance in the introductory chapters of books dedicated to programming and the Raspberry Pi (Richardson and Wallace 2012, 57; Philbin 2017, 41; Halfacree 2020, 54). In turn, 'Scratch' is a name for both a visual block-based programming language and an IDE. Thus, users utilize blocks as basic elements to

construct more complex programs. As a result, they can undertake a wide array of tasks, ranging from animating sprites, playing sounds – and, thus, creating games – to interfacing with GPIO pins. Moreover, Scratch enables its users to share their projects online, subsequently making them available for import and use by others.

Presently (i.e. in 2024), the most recent version of Scratch is Scratch 3, accessible both online and through an offline desktop client (Scratch Foundation, n.d.-a). The Scratch 3 desktop application is officially available on the majority of systems, with the exception of Linux (Scratch Foundation, n.d.-c). However, due to the collaborative efforts of the Raspberry Pi Foundation and the Scratch Foundation, a version of Scratch 3 has been developed specifically for the Raspberry Pi OS (O’Hanlon 2019). Additionally, Raspberry Pi OS comes with the older Scratch 1.4, which, however, lacks compatibility with the newer version (Scratch Foundation, n.d.-b).

Mathematica

The non-exclusive license agreement between the Raspberry Pi Foundation and Wolfram Research Inc. provides all Raspberry Pi users with free access to a range of products for non-commercial purposes. As a result, since 2013 the Raspberry Pi OS has been bundled with the Wolfram programming language and Mathematica, creating a comprehensive development environment (Wolfram 2013). Hence, Mathematica (Wolfram, n.d.-a) serves as both a kernel and an IDE. It interprets and returns the results of expressions in the Wolfram language (Wolfram 2017), a high-level programming language. However, in comparison to all the aforementioned languages and tools, both Mathematica and its language are entirely proprietary.

In general, Mathematica could be used for various technical applications, encompassing ‘neural networks, machine learning, image processing, geometry, data science, visualizations and much more’ (Wolfram, n.d.-b). Additionally, the owners of Raspberry Pi devices are presented with a selection of comprehensive projects available in tutorial format (Raspberry Pi Foundation, n.d.-a).

Sonic Pi

Sonic Pi (Aaron 2020) is a music synthesizer software that, similarly to Scratch and Mathematica, provides a live development environment enabling users to write and execute code in real time. However, rath-

er than interfacing with GPIO pins or performing complex algebraic computations, Sonic Pi, based on the SuperCollider (McCartney 2002) synthesis engine, turns source code into music.

The source code, in this case, is to be written in the Ruby (Usa 2019) programming language. Thus, using Ruby's native data structures, loops, and conditional statements together with Sonic Pi's custom syntax, users can generate a range of musical elements, including individual notes, patterns (i.e. an array of notes), and chords. Furthermore, they can manipulate the synthesizer sound, apply various effects, control the tempo, introduce delays, and incorporate pre-recorded samples, among other functions (Philbin 2017, 151–163). Sonic Pi's support of musical instrument digital interface (MIDI), combined with Raspberry Pi's portability, make it a great tool for creative enthusiasts.

7 Basic projects for getting to know Raspberry Pi

DIY projects are too numerous, so we will focus here only on some projects with which we have had hands-on experience. We have divided them into simple and advanced projects, where simple projects do not require any special knowledge, while advanced projects might require some basic knowledge of soldering.

7.1 A desktop computer

While the Raspberry Pi 4 or 5 may not replace high-end desktops for power users, it excels in providing an affordable and energy-efficient computing solution for a wide range of applications, especially in educational and basic productivity contexts. Its compact form factor and versatility make it a popular choice for DIY enthusiasts and those looking for a cost-effective desktop solution.

A logical decision for using a Raspberry Pi as a desktop computer is the Raspberry Pi 400 (Figure 7.1).

As a standalone unit, it is a complete personal computer built into a keyboard, which means you only need an OS on a microSD card,



Figure 7.1
Raspberry
Pi 400
Source:
amazon.com



Figure 7.2
Raspberry Pi 4
in DiSalvo copper
enclosure (right)
and in Argon ONE
M.2 enclosure (left)

a mouse, a couple of cables, a power supply, and a monitor to start. If you decide to purchase a kit, it comes with a mouse, power supply, a micro HDMI to HDMI cable, and a microSD card preloaded with the Raspberry Pi OS.

Two options are shown in Figure 7.2, each having its advantages.

The Argon ONE M.2 case is worth considering as it provides the following:

- Built-in IR support
- Power management modes
- Integrated M.2 SATA SSD support

The latter allows one to maximize the potential speed of the Raspberry Pi as one can boot via a SATA SSD for faster boot times and larger storage capacity compared to booting from a microSD card.

7.2 Christmas tree

Six years ago, the 3D Xmas Tree was reviewed in *MagPi* magazine by Russell Barnes ([↗ 3D Xmas Tree review](#)). The 3D Xmas Tree is a HAT for the Raspberry Pi available in unsoldered version and a version that comes pre-soldered. The price difference speaks in favour of the pre-soldered version, the more so if you do not have a soldering iron at the time of reading. One has to solder 25 resistors and 25 LEDs to the board, as well as a 40-pin GPIO header. We needed almost an hour to do this, so it is up to you to decide which one to purchase. Programming the tree is a piece of cake as there are code examples available on the PiHut web page ([↗ 3D Xmas Tree for Raspberry Pi](#)). PiHut also suggests using a GPIO Zero as the easiest way to control your Xmas tree board.



Figure 7.3
Xmas tree

By now, you already know that you need a Raspberry Pi Zero (or Zero W or Zero 2 W) and the 3D Xmas Tree HAT. Below are the instructions for the software needed. First, install the necessary software:

```
sudo apt install python-gpiozero python3
-gpiozero
```

This example of a Python code sets all the LEDs flickering randomly:

```
from gpiozero import LEDBoard
from gpiozero.tools import random_values
from signal import pause

tree = LEDBoard(*range(2,28),pwm=True)
for led in tree:
    led.source_delay = 0.1
    led.source = random_values()
    pause()
```

If you decided to purchase the soldering kit, look at the instructions here: [t3D Xmas Tree for Raspberry Pi Assembly Instructions](#). A fully assembled soldering kit on a wooden plate looks like this:

7.3 Google Assistant with Google AIY

In 2017 (following guidelines in issue 57 of *The MagPi*), we built a Google Assistant bot (Raspberry Pi 2017b). Google had launched the Artificial Intelligence You (AIY) Project in collaboration with the Raspberry Pi Foundation to promote DIY AI projects to the maker community. One product within the Google AIY project is the AIY Voice Kit, which

**Figure 7.4**

Assembled Google AIY voice assistant with Raspberry Pi 3 as the central computing unit

was designed to enable users to build their own voice control assistant using a Raspberry Pi.

Here are the key components and steps involved in creating a Google AIY Voice Kit with Raspberry Pi:

- **Hardware Components:** The Raspberry Pi acts as the central computing unit that runs the AIY software, while the Voice hardware attached on top (HAT) is an accessory board that includes necessary hardware for audio input and output. A speaker is provided for audio output of the AI assistant's responses, and a microphone captures user voice commands. Additionally, a button allows users to trigger the AI assistant with a physical button.
- **Software and Libraries:** The Google Assistant SDK is a software development kit provided by Google that enables developers to integrate Google Assistant functionality into their projects. Raspbian OS is the recommended operating system for Raspberry Pi, and Google provides a set of software packages and configurations tailored for the AIY Voice Kit through their AIY Projects software.
- **Assembly and Configuration:** Users need to assemble the hardware components, which includes connecting the Voice HAT to the Raspberry Pi. Once assembled, users can install the required software packages and configure the system to work with the AIY Voice Kit.
- **Integration with Google Assistant:** To integrate with Google Assistant, users must set up a Google Cloud Platform (GCP) project to obtain API credentials. These credentials are then used to configure the AIY kit, allowing it to communicate with Google's servers.

- **Usage:** Once assembled, configured, and integrated with Google Assistant, users can interact with their voice-controlled AI assistant by pressing the button and issuing voice commands. The AI assistant's responses are played through the speaker.

The final product is shown in Figure 7.4.

List of materials of the v1 included:

1. Voice HAT board
2. Voice HAT microphone board
3. 2 plastic standoffs
4. 3-inch speaker
5. Arcade-style push button
6. 4-wire button cable (for connecting
7. 5-wire daughter board cable (for connecting microphone HAT board)
8. External cardboard box
9. Internal cardboard frame

One would also need a Raspberry Pi, needle-nose pliers, a Phillips screwdriver, and two-sided tape for putting parts together. In November 2017, *The MagPi Essentials AIY Projects* was published (Hattersley 2017), so we would advise our readers to consult the text before they start. The magazine can be downloaded from here [↗ Essentials - AIY Voice Projects](#). Chapters cover the following:

- Made by you with Google: the team behind the AIY Projects kit
- Your AIY Projects Voice Kit
- Assemble the kit
- Set up the voice assistant software
- Build a voice recognizer
- Create a voice recognizer
- Control an LED
- Attach a servo
- Control a DC motor

If you have a newer kit, consult the site with vision, voice, and maker kit: [↗ Voice Kit](#). The newer version of voice kit requires a Raspberry Pi Zero WH.

7.4 Magic Mirror

MagicMirror (GitHub Repository for MagicMirror2 is available here: [↗ Github; MagicMirror](#)) is an open-source project that allows you to turn your Raspberry Pi (3 or 4) into a smart mirror. When the display is on, it shows information such as time, date, weather forecast, calendar events, news, and other customizable widgets; when the display is off, the mirror looks like a regular mirror. It was created by Michael Teeuw ([↗ MichaelTeeuw.nl; MgcMirror](#)), a developer from the Netherlands in 2014: the project gained popularity and evolved into an open-source project with contributions from the community.

The project was showcased for the first time in issue 40 of *The MagPi* in 2015, *The MagPi* 54 in 2017, and then again in the same magazine in issue 90 in 2020.

Our build of MagicMirror was made around a Raspberry Pi 3 and an old monitor disassembled and put into a wooden frame. Building a frame took a bit more time than everything else due to our poor carpentry skills.

The ultimate MagicMirror2 requires skills in many areas: carpentry, electronics, programming, and graphic design. Here are the steps you need to follow:

1. Remove the outer casing of a monitor or a TV
2. Use a Raspberry Pi 3 or 4 (we used 3) to power the TV or monitor
3. A power supply for both items above is also a must, as is a HDMI cable to connect the TV or monitor with the Raspberry Pi
4. You'll need some woodworking tools and paint
5. Measure the monitor and cut the wood, then assemble the frame
6. Add the front of the frame, which will cover the bezel of the monitor and keep it (and the mirror) in place. Then, attach the front
7. Drill a couple of holes at the top and the bottom for good ventilation
8. Make a hole at the bottom for cables to be able to run out of the housing
9. Make four small brackets that will keep the monitor from falling out of the back
10. Paint the frame after sanding and smoothing it
11. Put in the mirror and the monitor, connect the Raspberry Pi to the monitor via an HDMI cable, and run the cables through the hole you made



Figure 7.5
MagicMirror

12. Install the software to your Raspberry Pi (`curl -sL http://magpi.cc/MirrorInstall | bash`)
13. Personalize your MagicMirror by modifying `config.js` file. Some modules are pre-installed, others can be added or modified; for example, the weather module will, by default show nothing, because you have to get an API key for openweather by first creating an account at [OpenWweatherMap.org](https://openweathermap.org/api); `api`. If you live in Maribor as one of the authors of this book does, you will also need to change the city. Be careful not to exceed the number of API calls a day, as only 1000 API calls are free.
14. Go to magicmirror.builders for help or additional information. This is the home site for the project with many useful links, updates, and documentation. Another site that may interest you is forum.magicmirror.builders where the community for the MagicMirror resides. Our MagicMirror does not use the so-called 'two-way material', which is also used in police interview rooms and as privacy screening. It was difficult to obtain at the time of building the project and we thought that the magic mirror *sans* mirror also does the job of displaying important information quite well. It took us the whole day to build it and, we admit, we ran out of

energy to take it to the next step. On the other hand, after seven years it still works, and we still admire other makers who managed to invest several days into their projects that resulted in ultimate MagicMirrors. One day, we believe, we may at least update our MagicMirror to version 2. We shall do this by:

15. Accessing the MagicMirror directory:

```
cd ~/MagicMirror
```

16. Fetching the latest changes from the MagicMirror repository:

```
git pull
```

17. Installing the new dependencies or updating existing ones:

```
npm install
```

18. Checking the release notes (it is a good idea to check the release notes before updating as we are updating a specific version)

19. Restarting MagicMirror:

```
npm start
```

7.5 Simple music transport

With product designers in Toulouse, France and production in Bangalore, India, Allo (see: <https://allo.com/company.html>) has become quite popular with the DIY community of music lovers. Here, we are going to present a project on a digital music transport based on the Raspberry Pi 4. Who would build something like this? Imagine, that all your music is either on external drives, USB keys, or on an NAS. If you have a decent hi-fi system that consists of an integrated amplifier with an external or built-in DAC and a pair of speakers, you may also want a decent digital transport. Remember, in a music signal chain, a digital transport is always located before a DAC, so the digital transport that serves the digital data to a DAC is very important for the sound quality of the whole music system.

The music transport described here consists of two HATs and a Raspberry Pi 4, which is why we call it simple. There are simpler builds

available, with only one HAT on top of a Raspberry Pi; and there are many more expensive options available verging on the DAC we describe below. What is good is that all of them are upgradable, meaning a relatively small investment can lead to improved sound quality.

This is what you need:

1. A Raspberry Pi 4 (with at least 2 GB of RAM)
2. A microSD card with an OS installed on it (MoOde, Volumio, Diet Pi, Max2Play, Ropieee)
3. A HAT like Allo's Digione signature ([↗ Allo's Digione signature](#)), which consists of two boards: one for the dirty side (powering Raspberry Pi) and one for the clean side, which sits on the top (clocks, buffers, flip-flops are to be found here, as well as BNC and coaxial outputs).
4. A power supply to power both sides through a USB-C input
 - a. The first option is powering both sides with a 5V 3A switch mode power supply, which is cheap but not recommended.
 - b. The second option is powering the dirty side with a 5V 3A power supply and powering the clean side with a battery power supply, as the clean side requires only 100mA.
 - c. The best option comes last: power the clean and dirty side of the music transport with a Shanti (a dual linear low noise power supply; more here: [↗ Allo's Shanti \(EU\)](#)).

Under 2 above we mentioned the OS for the transport. As you can see, there are many options for the music player software. A good, open-source and community-maintained music streamer and player is moOde 8. After downloading it from [↗ moode™ audio player](#) and preparing the ISO image with Raspberry Pi Imager, one only has to follow the set-up guide, which is available on github ([↗ Github.com; moode-player](#)). Initial installation of moOde is done in a couple of minutes; additional configuration of the moOde audio player is possible through the player's Menu, Quick Help and the (i) information buttons once you have accessed the moOde WebUI using a browser (on phone, tablet, or another computer). Simply type this in the browser:

- a. <http://moode.local> or
- b. http://IP_ADDRESS

The following window (Figure 7.6) will open in a minute or two:

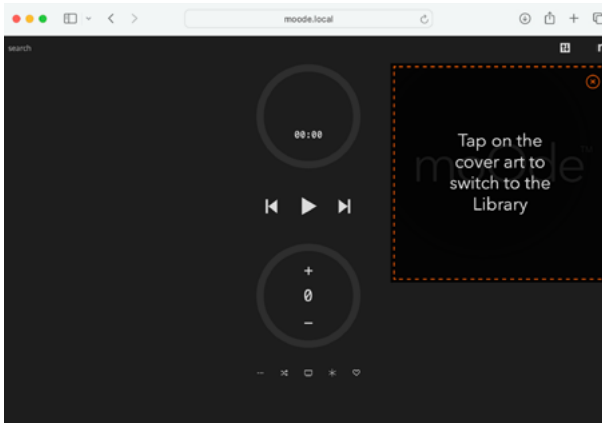


Figure 7.6
moOde.local screen-shot

After clicking on the letter 'm' in the upper right corner, you will see:

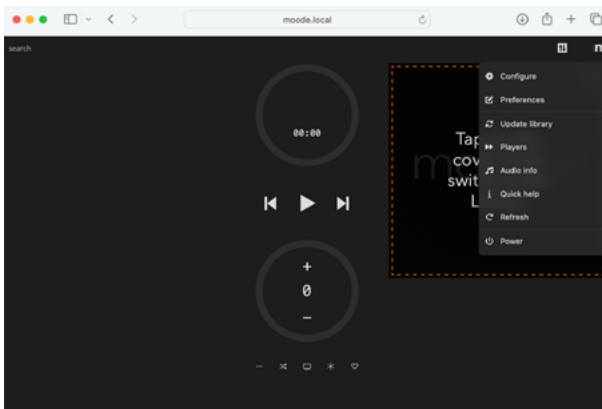


Figure 7.7
moOde menu

Start with setting the configuration settings by clicking on the Configure option. First set the Audio settings.

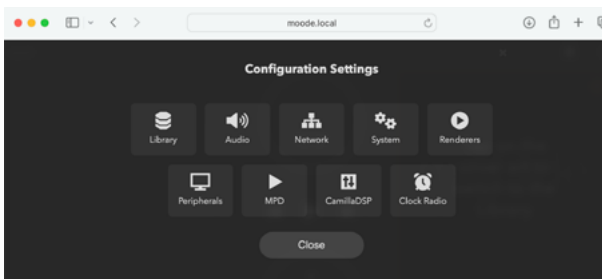


Figure 7.8
Configuration settings in moOde

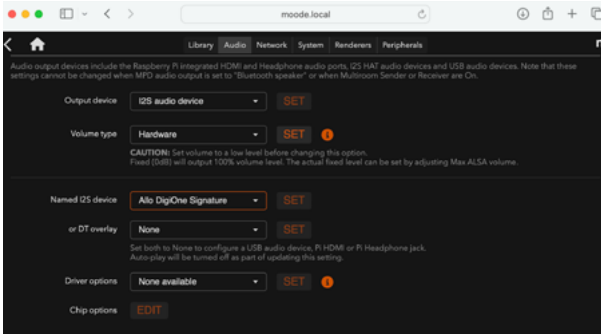


Figure 7.9
Audio settings
in moOde

There are too many setting to mention here, so let us have a look at all options of the moOde audio music transport and player.

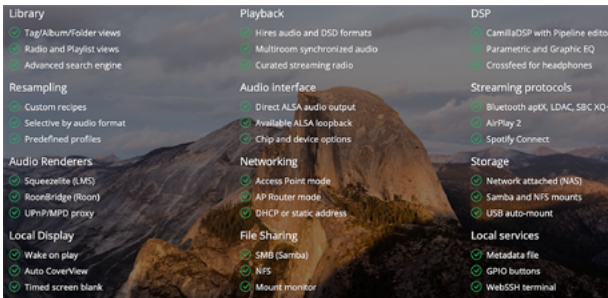


Figure 7.10
moOde setting
options
Source [https://
moodeaudio.org](https://moodeaudio.org)

A figure is worth a thousand words, so have a look at Figure 7.10, in which a digital music transport is shown without any power sources connected to it.



Figure 7.11
A simple music
transport

8 Advanced projects

The two projects described below were dubbed advanced because they require some knowledge of:

1. Linux and command lines
 - a. Understanding of Linux commands and basic system administration
 - b. Knowledge of file systems, permissions, and package management
2. Electronics and hardware
 - a. Basic understanding of electronics, including resistors, capacitors, and sensors
 - b. Familiarity with GPIO and how to interface with external hardware
3. Basic knowledge of networking concepts
 - a. IP addressing, subnetting, routing, and basic networking protocols
 - b. Understanding wireless technologies like Wi-Fi and Bluetooth
4. Knowledge of power management, like knowing power requirements, battery management
5. Understanding project management principles for planning and executing complex projects
6. Basic soldering skills

8.1 RPi-based NAS

In September 2021, Argon Forty successfully crowdfunded Argon EON Pi NAS enclosure ([↗ Argon EON Pi NAS](#)). The project was also backed

by us, and we received the enclosure at the beginning of 2022. There have been many Raspberry Pi-related projects on Kickstarter, a crowd-funding platform based in New York, some of which attracted our attention:

1. Joy-IT Multimedia Case for Raspberry Pi 4
2. Short Circuits: a platform for learning electronics (for building modular devices while learning about circuitry, soldering, and coding)
3. Strawberry4Pi 2: A hardware and software for Raspberry Pi (an IoT solution for Raspberry Pi)
4. Remodo X: a designer remote for Raspberry Pi (a programmable BLE remote that supports Kodi, OpenHab, Home-Assistant, etc.)

Table 8.1 NAS features

Open Source OS	All Raspberry Pi Operating Systems (Raspberry Pi OS, Open Media Vault, Twister OS, Ubuntu for Raspberry Pi, etc.)
Raspberry Pi 4 Model B	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz; 2GB, 4GB, or 8GB LPDDR4-3200 SDRAM
Drive Bays	4
Compatible Drive Type	2.5" SATA HDD (Max 4 drives) 2.5" SATA SSD (Max 4 drives) 3.5" SATA HDD (Max 2 drives)
Maximum Single Volume Size	18TB
Raspberry Pi 4 Model B	RJ-45 1GbE LAN port (1) USB 3.0 (2) USB 2.0 (2) Raspberry Pi 40 pin GPIO header HDMI ports – up to 4kp60 supported (2) 4-pole stereo audio and composite video port (1) Micro-SD card slot (1)
OLED System Display	Multimode (upon installation of script – software controlled)
System Fan	60 mm x 60 mm Brushless Motor Multimode (upon installation of script)
Safe Shutdown Power Button	
Scheduled ON / OFF	Through the built-in RTC (upon installation of script)
Power Supply	60 Watts

Source Adapted from <https://argon40.com/products/argon-eon-pi-nas?variant=39847140851777>



Figure 8.1

Argon EON NAS enclosure from various angles.

Source Argon Forty, <https://argon40.com/products/argon-eon-pi-nas?variant=39847140851777>

5. HackyPi: the hacking tool one can carry in a pocket (for refining coding and programming skills with ethical hacking)
6. 1.54 and 2.7 E-paper HAT display for Raspberry Pi
7. UPS Hat for Raspberry Pi
8. MusicPi: a high-quality stereo audio HAT for Raspberry Pi Pico

Argon EON Pi NAS has several advantages over other similar projects, mainly due to the following characteristics shown in Table 8.1.



Figure 8.2

Inside of the NAS enclosure

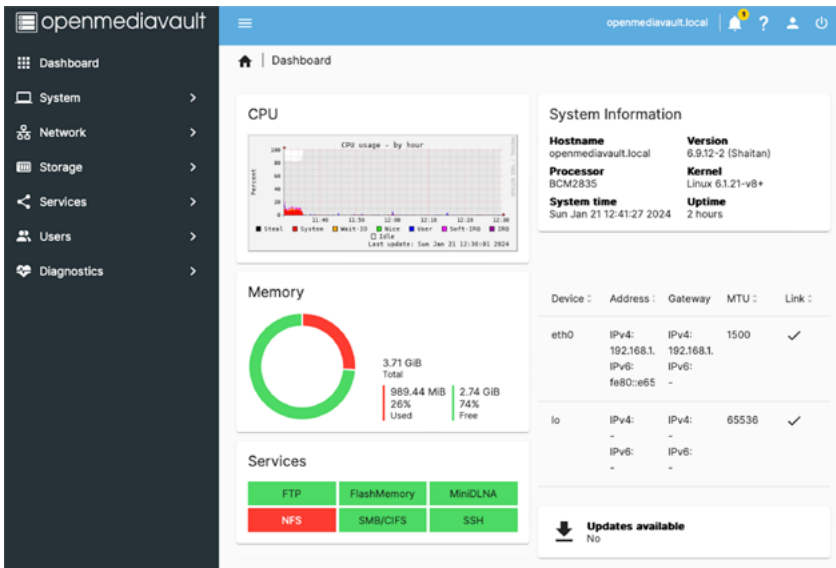


Figure 8.3 A screenshot of openmediavault UI

Figure 8.2 shows one half of the Raspberry Pi-based NAS. Shown are a 1TB SSD and a 4TB HDD, so the capacity of our NAS is 5TB. A user can access the NAS through another computer by simply typing “openmediavault.local” in a browser and logging in to the dashboard. Below is a screenshot of the UI screen.

The instruction manual added to the Argon EON enclosure is good enough to guide you through the installation of Open Media Vault. The installation process was not tricky, the transfer speeds were between 67 and 111 MB/s, temperatures between 33 °C (idle) and 42 °C (when copying a file), which – even without mentioning how good it looks – makes it one of the best Raspberry Pi-powered NAS solutions still available at the time of writing this monograph.

8.2 Advanced RPi-based DAC

It did not come as a surprise that at the end of 2020 some IQaudio products were added to the Raspberry Pi fold. DAC+, DAC Pro, Digi-AMP+, and Codec Zero became available at a competitive price (at \$20, \$25, \$30, and \$20, respectively). 2020 was a busy year for Raspberry Pi Foundation as they launched High Quality Camera, 8GB Raspberry Pi 4, Compute Module 4, and Raspberry Pi 400.

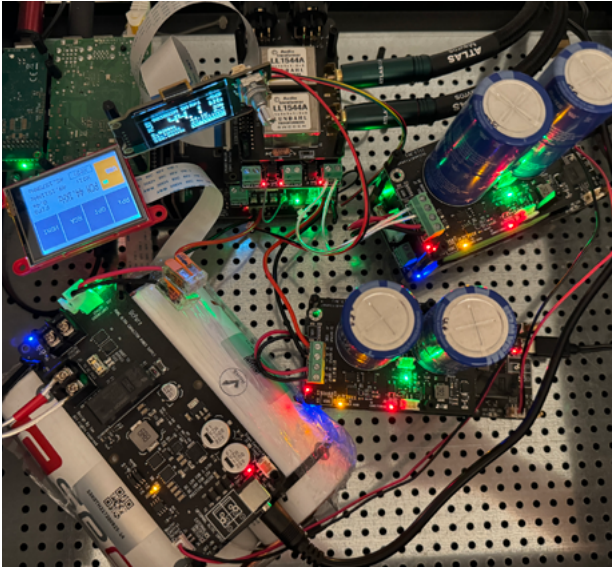


Figure 8.4
Advanced Raspberry
Pi-based DAC

To make a long story short, let us just say that every phone, tablet, laptop, or personal computer has a DAC soldered to its motherboard, including Raspberry Pi (except Pi Zero and Pi Zero W). One can connect headphones directly to the Raspberry Pi headphone output and listen to music. One will soon discover that this is not the best way to enjoy ‘It Never Entered My Mind’ by Ben Webster and Coleman Hawkins ripped and stored on the NAS you have just made with a Raspberry Pi 4. The Raspberry Pi is a general-purpose computer and introduced so much electrical noise that it made listening unbearable. The playback also deteriorated due to poor clock accuracy because the clock on Raspberry Pi has not been optimized for this purpose. The third source of sound deterioration was the poor-quality power supply, which is not suitable for music reproduction. For serious music reproduction in a hi-fi system, dedicated high-quality audio components are preferred. Here, there are two options: one can buy a standalone DAC (start with Schitt Audio, then upgrade by purchasing a Chord Electronics high-quality DAC) or – and this is a much better option, both financially and experience-wise – build one yourself, using a Raspberry Pi 4 with 4 or 8 GB RAM.

A glimpse at Figure 8.4 will give an idea of the final product of your building efforts. Raspberry Pi 4, the heart of the build, is in the upper left corner, sitting upside down and next to the smaller green PCB

called Amanero Combo386 module (more here: [↗ amanero.com](https://amanero.com)), a USB class 2 to I2S 32bit and DSD output adapter.

Below is the list of parts you will need for the project:

1. A Raspberry Pi 4 with 4 or 8GB RAM
2. A microSD card with a special OS on it (Volumio, moOde audio or Ropieee)
3. Amanero Combo386 HAT
4. DAC HAT (e.g. Ian Canada's ES90382M dual mono II DAC HAT; more here [↗ ES9038Q2MP iDual Mono II Manual.pdf](#))
5. A HAT for enabling music playback from multiple external sources, like a CD player, a DVD player (through HDMI input), or a TV through SPDIF inputs (coaxial or optical input) (e.g. Ian Canada's ReceiverPi Pro HAT; more here: [↗ ReceiverPi ProUserS Manual.pdf](#))
6. A reclocker HAT (e.g. Ian Canada's FiFoPi Q7, a flagship synchronous mode I2S/DSD/DoP FIFO reclocker: more details are available here: [↗ FifoPi Q7 Manual.pdf](#))
7. Better clocks for the reclocker HAT (e.g. Ian Canada's SC-Pure 45.1584 MHz and SC-Pure 49.1520 MHz clocks; see here: [↗ ScPure Datasheet.pdf](#))
8. Output stage HAT (e.g. Ian Canada's LL1544A transformer I/V HAT kit; see here for assembly: [↗ LL1544a Transformer I/V Guidance.pdf](#))
9. A pair of good audio transformers to solder on output stage HAT (e.g. Lundahl LL1544A transformers)
10. A Raspberry Pi HAT adapter station to further decrease EMI noise and to separate Raspberry Pi and AUDIO HATs in two-stack configuration with multiple separate power supplies for HATs (e.g. Ian Canada's StationPi Pro; see here for detailed assembly instructions: [↗ StationPi Pro Users Manual.pdf](#))
11. A power supply for powering the Raspberry Pi side of the StationPi Pro (5V; we used a MCRU linear power supply; there are other options, so feel free to do some research)
12. A power supply to power the audio side of the StationPi Pro (5V; we used an ultracapacitor power supply designed by Ian Canada named UcPure. As the name suggests you will need a pair of 3000F capacitance ultracapacitors. The set-up is capable of delivering up to 1000A dynamic current, so be very careful and do not just skim

- the manual, but read it very carefully; see here: [↗ UcPure Manual.pdf](#)
13. Two capacitors for the above power supply (e.g. Maxwell BCAP3000 P270 or BCAP3000 P300; other 3000F/2.7V or higher capacitance/voltage ultracapacitors are also good for UcPure)
 14. A balancer/protection board for ultracapacitors (e.g. Ian Canada's UcBalancer protection board kit; see here: [↗ UcBalancer Kit Manual.pdf](#))
 15. A power supply for the DAC. 9038Q2MPi dual mono II DAC HAT can be powered by one, two, or three 3.3V power supplies. For our project, we decided to use only one and upgrade at a later time. We used a battery power supply designed by Ian Canada called LifePO₄ Mini 3.3V, which uses two 26650 LifePO₄ battery cells (a good choice is an A123 ANR26650M1B cell). See here for more detail: [↗ LifePO₄ Mini 3.3V Manual.pdf](#). The LiFePO₄ Mini 3.3V power supply needs two battery holders, which have to be soldered to the board. Battery holders can be sourced from Audio-phonics or Amazon. In addition, LiFePO₄ must be connected to either a DC power supply (with a standard 5V USB-C cable or an AC power source (5V-6V)). We chose a 5V DC power supply.
 16. We further upgraded the above power supply by adding to it an UcConditioner 3.3V board designed by Ian Canada ([↗ UcConditioner 3.3V Users Manual.pdf](#)) The UcConditioner 3.3V requires two ultracapacitors that have to be soldered to the board (recommended capacitors are BCAP0325 P270 S19, or BCAP0450 P270 S18, or BCAP0350 P270 S18, or XV3560-2R7407-R). For a good connection, you will need a soldering iron with 80W or higher power.
 17. For the power supply for the clean side of the FiFoPi Q7 you need to assemble another set described under points 15 and 16 above.
 18. You also need something to control your DAC set-up, ideally something with a display to see what is going on. Here, again, MonitoPi Pro is the best solution. See here, for more information: [↗ MonitorPi Pro Manual.pdf](#). You simply plug it into the GPIO port, and voila, MonitorPi Pro detects the digital audio format and the I2S/DSD signal status in real time and shows the information on the LED display. It is recommended to install the MonitorPi Pro into the non-isolated GPIO of FiFoPi Q7. It works with Apple remote control A1294, A1156 or any other remote control with the NEC IR protocol, and can also be moved to the front pan-

el of your set-up housing with an FFC/FPC cable and a universal extension kit (Universal Raspberry Pi GPIO extension kit can be seen here: [↗ GPIO extensionKit.pdf](#))

19. A StationPi Pro touchscreen display and controller is also a useful add-on for the set-up. You need it if you installed the HAT described in point 5 (ReceiverPi Pro). Ian Canada's instructions are here: [↗ Open source StationPi Pro touch screen display/controller](#), but you also have to source the display (model GEN4-ULCD-28PT, produced by 4Dsystems ([↗ GEN4-ULCD-28PT-PI](#))). A starter kit includes 4D-UPA programmer, which is needed for programming the display.

As for the OS, one can install moOde, or, alternatively, if the preferred music server is Roon, install RoPieee, a RoonBridge image for Raspberry Pi. Roon can be checked here: [↗ roon.app](#), and the RoPieee image is downloadable from here: [↗ ropiee.org](#). The microSD card can be flashed using balenaEtcher ([↗ balenaEtcher](#)), which is yet another piece of software for writing flash images to SD cards and USB drives. Once the OS has been on the microSD card, we would advise the reader to check the installation guide for RoPieee, which is available here: [↗ ropieee installation guide.pdf](#).

9 Raspberry Pi Zero, Arduino, and BBC micro:bit

The landscape of single-board computers and microcontrollers would be incomplete without mentioning Arduino and BBC micro:bit boards in addition to Raspberry Pi Zero. Both BBC micro:bit and Arduino are similar to the Raspberry Pi Pico and are designed to introduce programming and electronics to students. Raspberry Pi Pico uses a microcontroller (RP2040) with dual ARM Cortex-M0+ cores, which makes it more capable than a traditional microcontroller. BBC micro:bit uses Nordic Semiconductor nRF51822) but is designed to be more beginner-friendly with a simpler architecture; Arduino boards use various microcontrollers, depending on the model. The Arduino Uno, for example, uses the ATmega328P microcontroller. As for the programming language, Pico can be programmed using MicroPython or C/C++; BBC micro:bit uses a block-based language called MakeCode, but it also supports MicroPython and other languages; Arduino boards are most often programmed using the Arduino IDE with a simplified version of C/C++. Raspberry Pi Pico has 26 multifunction GPIO pins, providing flexibility for various projects.

The BBC micro:bit has 25 LED lights that can be used as individual LEDs and also includes buttons, sensors, and edge connectors for additional functionality. The number of I/O pins varies depending on the Arduino model. For example, the Arduino Uno has 14 digital I/O pins. All boards support USB for programming and power. Raspberry Pi Pico also has UART, I2C, SPI, and PWM capabilities; BBC micro:bit has Bluetooth, and Arduino typically has UART, I2C, and SPI capabilities. Each board serves its purpose and has strengths in different areas. The choice between Raspberry Pi Pico, BBC micro:bit, and Arduino depends on factors such as project requirements, programming preferences, and community support.

Examples of projects for the three boards include:

Raspberry Pi Pico

- Simple Projects
 - LED Blinking: Write a simple program to blink an LED connected to one of the GPIO pins.
 - Button Input: Create a program that responds to a button press by turning an LED on or off.
 - Temperature and Humidity Monitoring: Use a DHT sensor to measure temperature and humidity and display the readings.
- Advanced Projects
 - Weather Station: connect sensors for temperature, humidity, and pressure to create a weather station that logs data and displays it on a website.
 - Robotics: build a small robot using motors and sensors, and control it with the Raspberry Pi Pico.
 - Home Automation System: create a home automation system using Wi-Fi or Bluetooth connectivity to control lights, appliances, and other devices.

BBC micro:bit

- Simple Projects
 - LED Patterns: program the micro:bit to display different LED patterns and animations.
 - Dice Simulator: create a simple program to simulate a dice roll and display the result on the LEDs.
 - Reaction Timer Game: build a reaction timer game using the micro:bit's buttons and display.
- Advanced Projects
 - Gesture-controlled Robot: use the micro:bit's accelerometer to control the movement of a robot based on hand gestures.
 - Wearable Fitness Tracker: develop a fitness tracker using the micro:bit to monitor steps, distance, and heart rate.

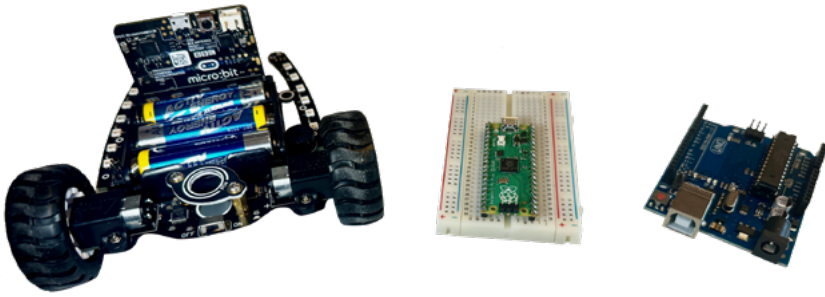


Figure 9.1 BBC micro:bit, Raspberry Pi Pico and Arduino UNO

- **Bluetooth Communication:** use the micro:bit's built-in Bluetooth capabilities to communicate wirelessly with other micro:bits or devices.

Arduino

- **Simple Projects**
 - **LED Blinking:** start with the classic 'Hello World' of electronics – blinking an LED using an Arduino.
 - **Traffic Light Controller:** simulate a traffic light system with LEDs to control the flow of 'traffic'.
 - **Temperature Display:** interface a temperature sensor and display the temperature on an LCD screen.
- **Advanced Projects**
 - **Home Automation System:** build a sophisticated home automation system using Arduino to control lights, HVAC systems, and security.
 - **Quadcopter/Drone:** create a drone using Arduino for flight control and stabilization.
 - **Gesture Recognition System:** implement a gesture recognition system using sensors and machine learning algorithms with Arduino.

Figure 9.1 shows a BBC micro:bit (on the far left side) plugged into a 4Tronix bit:bot XL robot and programmed to control the movement, follow lines or – by adding an ultrasonic distance sensor board – avoid obstacles. The robot can be remote-controlled wirelessly if an addition-

al BBC micro:bit has been programmed or by using the bitty controller application on a mobile phone. Next to it is a Raspberry Pi Pico on a breadboard, ready to have an LED and a resistor inserted, and on the right is a popular member of the Arduino family: Arduino UNO.

Conclusion

The monograph covered a broad range of topics, including the history and development of Raspberry Pi, its operating systems, accessories, and applications in various fields such as education, DIY projects, and industrial applications. We also delved into programming on Raspberry Pi, including language selection and development environments, and showcased both basic and advanced projects to demonstrate its versatility and capabilities. Importantly, the monograph emphasized the role of Raspberry Pi in enhancing computer and technology literacy, showcasing its utility in educational contexts and beyond.

We believe that the Raspberry Pi family of computers has a transformative role, not only because of its reasonable price but also due to its versatility of use across various domains. We can think of no other way to enhance computer and technology literacy and cannot emphasize enough the ongoing journey of discovery, creativity, and learning than the one facilitated by Raspberry Pi. The tiny computer emerged from a backdrop of declining computer and technology literacy, with the founders aiming to provide an accessible platform that would reintroduce the basics of computing to schools and hobbyists. Raspberry Pi's impact on education has been profound, integrating practical computing education into classrooms worldwide. It has enabled students to engage hands-on with programming, digital making, and problem-solving, effectively demystifying computer science and encouraging a deeper interest in STEM subjects. In addition, the platform has become synonymous with DIY culture, empowering hobbyists to create a wide range of projects from home automation systems to weather stations. This flexibility has not only showcased the ingenuity of users but also demonstrated the potential of Raspberry Pi to solve real-world problems creatively.

Beyond education and hobbies, Raspberry Pi has found a niche in professional and industrial applications. Its cost effectiveness and adaptability make it ideal for prototyping, digital signage, and even

serving as the brain for commercial products and industrial systems. Raspberry Pi supports a variety of programming languages and development environments, making it a versatile tool for learning coding and software development. This inclusivity ensures that individuals at different learning stages can find resources and community support to develop their skills.

The journey of Raspberry Pi reflects a remarkable blend of vision, community, and technology. It underlines a collective commitment to making computing accessible, engaging, and fun. As the platform continues to evolve, it promises to inspire new generations of innovators and thinkers to explore the limitless possibilities of technology. The authors hope that this monograph will bring a deeper understanding and appreciation of Raspberry Pi's capabilities to audiences across the globe, and especially in countries like Slovenia, where interest should be sparked and the knowledge base around Raspberry Pi expanded. The potential of Raspberry Pi for educational innovation and technological advancement is vast.

References

- Aaron, Sam. 2020. 'Sonic Pi 3.2.2.' *GitHub*, 7 April. <https://github.com/sonic-pi-net/sonic-pi/releases/tag/v3.2.2>.
- Abadi, Martin, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin et al. 2016. 'TensorFlow: A System for Large-Scale Machine Learning.' In *12th USENIX Symposium on Operating Systems Design and Implementation*, 265–283. Savannah, GA: USENIX.
- Aho, Alfred V., Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. 2007. *Compilers: Principles, Techniques, and Tools*. 2nd ed. Boston, MA: Addison-Wesley.
- Alva, Jilil Villalobos. 2021. *Beginning Mathematica and Wolfram for Data Science: Applications in Data Analysis, Machine Learning, and Neural Networks*. New York: Apress.
- Arnold, Ken, James Gosling, and David Holmes. 2005. *The Java Programming Language*. 4th ed. Upper Saddle River, NJ: Addison-Wesley.
- Balon, Branko, and Milan Simić. 2019. 'Using Raspberry Pi Computers in Education.' In *Proceedings of the International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, edited by Karolj Skala, 671–676. Opatija: Croatian Society for Information and Communication Technology, Electronics and Microelectronics.
- Barnes, David, and Michael Kölling. 2016. *Objects First with Java: A Practical Introduction Using BlueJ*. 6th ed. Boston: Pearson.
- Bin Bahrudin, Saifudaullah, Rosni Abu Kassim, and Norlida Buniyamin. 2013. 'Development of Fire Alarm System Using Raspberry Pi and Arduino Uno.' In *2013 International Conference on Electrical, Electronics and System Engineering (ICEESE)*, 43–48. Kuala Lumpur: Institute of Electrical and Electronics Engineers.
- Bradbury, Alex, and Ben Everard. 2014. *Learning Python with Raspberry Pi*. New York: McGraw-Hill.
- Bradski, G. 2000. 'The OpenCV Library.' *Dr. Dobb's Journal of Software Tools* 120:122–125.
- Debian. 2022. 'Apt.' *Debian*, 13 January. <https://wiki.debian.org/Apt>.
- Emani, Rahul, Edward Glantz, Chris Gamrat, and Michael K. Hills. 2019. 'Using the Raspberry Pi in IT Education.' In *SIGITE 2019 Proceedings of*

- the 20th Annual SIG Conference on Information Technology, 153–153. New York: Association for Computing Machinery.
- Endres, Creciana Maria, Crivian Pelisser, Doglas André Finco, Maristela Schlicher Silveira, and Valério Junior Piana. 2022. 'IoT and Raspberry Pi Application in the Food Industry: A Systematic Review.' *Research, Society and Development* 11 (1). <https://doi.org/10.33448/rsd-v11i1.24270>.
- Fletcher, Anthony C., and Mura, Cameron. 2019. 'Ten Quick Tips for Using a Raspberry Pi.' *PLoS Computational Biology* 15 (5). <https://doi.org/10.1371/journal.pcbi.1006959>.
- Flurry, Greg. 2021. *Java on the Raspberry Pi: Develop Java Programs to Control Devices for Robotics, IoT, and Beyond*. Austin, TX: Apress.
- Free Software Foundation. n.d. 'GNU Bash.' *GNU Operating System*. <https://www.gnu.org/software/bash/>.
- Gómez, Omar S., Antonio A. Aguilera, Raúl A. Aguilar, Juan P. Uacán, Raúl H. Rosero, and Karen Cortes-Verdin. 2017. 'An Empirical Study on the Impact of an IDE Tool Support in the Pair and Solo Programming.' *IEEE Access* 5 (1): 9175–9187.
- Gooch, Daniel, Jon Rosewell, Douglas Leith, and Mike Richards. 2022. 'Passive or Active Learning: The Challenges of Teaching Distributed Computing Using Raspberry Pi Clusters to Open Distance University Students.' *Open Learning: The Journal of Open, Distance and e-Learning*. <https://doi.org/10.1080/02680513.2022.2118573>.
- Halfacree, Gareth. 2020. *The Official Raspberry Pi Beginner's Guide*. 4th ed. Cambridge: Raspberry Pi.
- Halfacree, Gareth, and Ben Everard. 2021. *Get Started with MicroPython on Raspberry Pi Pico*. Cambridge: Raspberry Pi.
- Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser et al. 2020. 'Array Programming with NumPy.' *Nature* 585:357–362.
- Hattersley, Lucy. 2017. 'The AIY Projects: Create a Voice Kit with Your Raspberry Pi.' *The MagPi Magazine*, November. <https://magpi.raspberrypi.com/books/essentials-aiy-v1>.
- Henderson, Gordon. 2019a. 'wiringPi 2.52.' <http://wiringpi.com>.
- . 2019b. 'Wiring Pi: GPIO Interface Library for the Raspberry Pi.' *Wiring Pi*, 6 August. <https://web.archive.org/web/20220405225008/http://wiringpi.com/wiringpi-deprecated/>.
- Hunter, John D. 2007. 'Matplotlib: A 2D Graphics Environment.' *Computing in Science and Engineering* 9 (3): 90–95.
- Ibrahim, Dogan. 2021. *C Programming on Raspberry Pi*. London: Elektor.
- Jain, Sarthak, Anant Vaibhav, and Lovely Goyal. 2014. 'Raspberry Pi Based Interactive Home Automation System through E-Mail.' In 2014 *International Conference on Reliability Optimization and Information Technology*

- (ICROIT), 277–280. Faridabad: Institute of Electrical and Electronics Engineers.
- Jamil, M. Mahadi Abdul, and M. Shukri Ahmad. 2015. 'A Pilot Study: Development of Home Automation System via Raspberry Pi.' In *2015 2nd International Conference on Biomedical Engineering (ICoBE)*, 1–4. Penang: Institute of Electrical and Electronics Engineers.
- Jolles, Jolle W. 2021. 'Broad-Scale Applications of the Raspberry Pi: A Review and Guide for Biologists.' *Methods in Ecology and Evolution* 12 (9): 1562–1579.
- Jones, M. Tim. 2007. 'Anatomy of the Linux Kernel.' *IBM Developer*, 5 June. <https://developer.ibm.com/articles/l-linux-kernel/>.
- Karankumar, D. Mehta, B. Mehta Shreya, and Raviya S. Kapil. 2014. 'Analysis of TOI (Things of Internet) Industrial Monitoring System on Raspberry Pi Platform.' *International Journal of Computer Science and Mobile Applications* 2 (11): 33–40.
- Kernighan, Brian W., and Dennis M. Ritchie. 1988. *C Programming Language*. 2nd ed. Englewood Cliffs, NJ: Pearson.
- Kollár, Ján, Emília Pietriková, and Sergej Chodarev. 2012. 'Abstraction in Programming Languages According to Domain-Specific Patterns.' *Acta Electrotechnica et Informatica* 12 (2): 9–15.
- Kölling, Michael. 2015. *Introduction to Programming with Greenfoot: Object-Oriented Programming in Java with Games and Simulations*. 2nd ed. Boston: Pearson.
- . 2016. 'Educational Programming on the Raspberry Pi.' *Electronics* 5 (3): 33.
- Korom, Szilárd, and Zoltán Illés. 2022. 'Methods of Teaching Programming with Raspberry Pi.' In *2022 Second International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICA-ECT)*, 1–5. Bhillai: Institute of Electrical and Electronics Engineers.
- Kumar, Bura Vijay, Seena Naik Korra, N. Swathi, D. Kothandaraman, Nagen-der Yamsani, and Yerrolla Chanti. 2020. 'Traffic Control System for Vehicles on Indian Roads Using Raspberry Pi.' *IOP Conference Series: Materials Science and Engineering* 981 (3): 032098.
- Kumar, N. Pradeep, and Ravi Kumar Jatoth. 2015. 'Development of Cloud Based Light Intensity Monitoring System Using Raspberry Pi.' In *2015 International Conference on Industrial Instrumentation and Control (ICIC)*, 1356–1361. Pune: Institute of Electrical and Electronics Engineers.
- Kumar, R., and M. Pallikonda Rajasekaran. 2016. 'An IoT Based Patient Monitoring System Using Raspberry Pi.' In *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*, 1–4. Tamilnadu: Institute of Electrical and Electronics Engineers.
- Kurkovsky, Stan, and Chad Williams. 2017. 'Raspberry Pi as a Platform for the Internet of Things Projects: Experiences and Lessons.' In *Proceedings of*

- the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, 64–69. New York: Association for Computing Machinery.
- Limbo, Anton, Gabriel Tuhafeni Nhinda, and William Sverdlík. 2017. 'Rethinking Tertiary Computer Science Education: Let's Have Pi.' In *IST-Africa 2017 Conference Proceedings*, 1–6. Windhoek: Institute of Electrical and Electronics Engineers.
- LXDE Blog. 2021. 'LXTerminal 0.4.0 Released.' *LXDE Blog*, 24 February. <https://blog.lxde.org/2021/02/24/lxterminal-0-4-0-released/>.
- Maloney, John, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. 'The Scratch Programming Language and Environment.' *ACM Transactions on Computing Education* 10 (4). <https://doi.org/10.1145/1868358.1868363>.
- McCartney, James. 2002. 'Rethinking the Computer Music Language: Super-Collider.' *Computer Music Journal* 26 (4): 61–68.
- McKinney, Wes. 2010. 'Data Structures for Statistical Computing in Python.' In *Proceedings of the 9th Python in Science Conference*, edited by Stefan van der Walt and Jarrod Millman, 56–61. Austin, TX: SciPy.
- Mellis, A. David, Julian Gautier, Norbert Truchsess, Paul Stoffregen, Francis Gulotta, and Jeff Hoefs. n.d. 'Firmata Protocol Documentation.' *GitHub*. <https://github.com/firmata/protocol>.
- Mojang AB. n.d. 'What is Minecraft: Pi Edition?' *Minecraft*. <https://www.minecraft.net/en-us/edition/pi>.
- Molloy, D. 2016. *Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux*. Hoboken, NJ: Wiley.
- Monk, Simon. 2012. *Programming the Raspberry Pi: Getting Started with Python*. New York: McGraw-Hill.
- . 2017. *Hacking Electronics: Learning Electronics with Arduino and Raspberry Pi*. New York: McGraw-Hill.
- . 2022. *Raspberry Pi Cookbook*. 4th ed. Sebastopol, CA: O'Reilly.
- Mudaliar, Mani Dheeraj, and N. Sivakumar. 2020. 'IoT Based Real Time Energy Monitoring System Using Raspberry Pi.' *Internet of Things* 12:100292.
- Murray, Timm, and Steve Bertrand. 2018. *Programming the Raspberry Pi with Perl*. <http://leanpub.com/programmingtheraspberrypiwithperl>.
- Norris, Donald J. 2013. *Raspberry Pi Projects for the Evil Genius*. New York: McGraw-Hill.
- . 2019. *Machine Learning with the Raspberry Pi: Experiments with Data and Computer Vision*. New York: Apress.
- Norvig, Peter, and Stuart Russell. 2021. *Artificial Intelligence: A Modern Approach*. 4th ed. Harlow: Pearson.
- Nuttall, Ben. 2021. 'Gpiozero.' *GitHub*. <https://github.com/gpiozero/gpiozero>.
- O'Hanlon, Martin. 2019. 'Scratch 3 Desktop for Raspberry Pi OS on Raspberry Pi.' *Raspberry Pi Foundation*, 16 August. <https://www.raspberrypi.org/blog/scratch-3-desktop-for-raspbian-on-raspberry-pi/>.

- . n.d. 'Mcp1 1.2.1.' *GitHub*. <https://github.com/martinohanlon/mcpi>.
- Pardeshi, Vivek, Saurabh Sagar, Swapnil Murmurwar, and Pankaj Hage. 2017. 'Health Monitoring Systems Using IoT and Raspberry Pi: A Review.' In *2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 134–137. Bengaluru: Institute of Electrical and Electronics Engineers.
- Patchava, Vamsikrishna, Hari Babu Kandala, and P. Ravi Babu. 2015. 'A Smart Home Automation Technique with Raspberry Pi Using IoT.' In *2015 International Conference on Smart Sensors and Systems (IC-SSS)*, 1–4. Bangalore: Institute of Electrical and Electronics Engineers.
- Patil, Vishal P., Chanchal Patil, Shreepad Sawant, Swapnali Desai, and M. S. Chavan. 2018. 'Design and Implementing a Secured Wireless Communication System for Industrial Automation by Using Raspberry Pi.' *International Journal of Pure and Applied Mathematics* 118 (16): 59–73.
- Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. 'Scikit-Learn: Machine Learning in Python.' *Journal of Machine Learning Research* 12 (85): 2825–2830.
- Philbin, Carrie Anne. 2017. *Adventures in Raspberry Pi*. 3rd ed. Hoboken: Wiley.
- Pigpio. n.d. 'The Piggpio Library.' <https://abyz.me.uk/rpi/pigpio/index.html>.
- Ponugumati, Sunil Kumar, Kamran Ali, A. Lasebae, Zazid Zahoor, Anun Tanveer Kiyani, Mohammad Ali Khoshkholghi, and Latha Maddu. 2023. 'Efficient Design for Smart Environment Using Raspberry Pi with Blockchain and IoT (BRIoT).' *CCGridW: 4th Workshop on Secure IoT, Edge and Cloud Systems (SioTEC)*, pp. 75–80. Bangalore: Institute of Electrical and Electronics Engineers.
- Prechelt, Lutz. 2000. 'An Empirical Comparison of Seven Programming Languages.' *Computer* 33 (10): 23–29.
- Pygame. 2019. 'Pygame 1.9.6.' <https://www.pygame.org/news>.
- Python Software Foundation. 2020. 'Python 2.7.18.' *Python*, 20 April. <https://www.python.org/downloads/release/python-2718/>.
- . 2021. 'Pip 20.3.4.' <https://pypi.org/project/pip/20.3.4/>.
- . 2021. 'Python 3.9.2.' *Python*, 19 February. <https://www.python.org/downloads/release/python-392/>.
- . n.d. 'Sunsetting Python 2.' *Python*. <https://www.python.org/doc/sunset-python-2/>.
- Quinlan, Oliver, and Samantha Baloro. 2018. 'Raspberry Pi Computers in Schools.' *Raspberry Pi Foundation Research* 5. <https://www.raspberrypi.org/app/uploads/2018/08/Raspberry-Pi-Computers-in-Schools-2018.pdf>.
- Raspberry Pi. 2017a. 'Electronics Starter Guide.' *The MagPi*, December. <https://magpi.raspberrypi.com/issues/64>.

- . 2017b. 'Build Your AIY Projects Kit.' *The MagPi*, May. <https://magpi.raspberrypi.com/issues/57>.
- . 2020. 'Raspberry Pi Compute Module 4.' *The MagPi*, November. <https://magpi.raspberrypi.com/issues/99>.
- . 2023. 'Introducing Camera Module 3.' *The MagPi*, February. <https://magpi.raspberrypi.com/issues/126>.
- Raspberry Pi Foundation. n.d.-a. 'Find a Project.' <https://projects.raspberrypi.org/en/projects?software%5B%5D=wolfram>.
- . n.d.-b. 'Operating System Images.' <https://www.raspberrypi.com/software/operating-systems/>.
- Revathi, A., and N. Sasikaladevi. 2022. 'Real-Time Raspberry Pi-Based System for Linguistic Content Recognition from Speech.' *Social Science Research Network*. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4202297.
- Richardson, Matt, and Shawn Wallace. 2012. *Getting Started with Raspberry Pi*. Sebastopol, CA: O'Reilly Media.
- Riverbank Computing. n.d. 'What is PyQt?' <https://www.riverbankcomputing.com/software/pyqt/>.
- Rose, Karen, Scott Eldridge, and Lyman Chapin. 2015. 'The Internet of Things (IoT): An Overview.' *Internet Society*, 15 October. <https://www.internetsociety.org/resources/doc/2015/iot-overview/>
- Santhosh Krishna, B.V., J. Oviya, S. Gowri, and M. Varshini. 2016. 'Cloud Robotics in Industry Using Raspberry Pi.' In *2016 Second International Conference on Science Technology Engineering and Management (ICON-STEM)*, 543–547. Chennai: Institute of Electrical and Electronics Engineers.
- Santos, Rui, and Sara Santos. 2018. *20 Easy Raspberry Pi Projects: Toys, Tools, Gadgets, and More!* San Francisco, CA: No Starch Press.
- Scratch Foundation. n.d.-a. 'Create Stories, Games, and Animations: Share with Others Around the World.' *Scratch*. <https://scratch.mit.edu/>.
- Scratch Foundation. n.d.-b. 'Scratch 1.4.' *Scratch*. https://scratch.mit.edu/scratch_1.4.
- . n.d.-c. 'Scratch Offline Editor.' *Scratch*. <https://scratch.mit.edu/download>.
- Senthilkumar, Selvaraj, R. Lakshmi Rekha, L. Ramachandran, and S. Dhivya. 2016. 'Design and Implementation of Secured Wireless Communication Using Raspberry Pi.' *International Research Journal of Engineering and Technology (IRJET)* 3 (1): 1015–1018.
- Soerensen, Chres, Hernandez-Marcano Wiant, Javier Nestor, Juan Alberto Cabrera, Simon Wunderlich, Daniel Enrique Lucani, and Frank Hans Paul Fitzek. 2016. 'Easy as Pi: A Network Coding Raspberry Pi Testbed.' *Zenodo*. <https://doi.org/10.5281/zenodo.154328>.

- Sorte, Swati, Wani Patil, Sonali Joshi, and Payal Ghutke. 2021. 'Vision System Check for Authentication of Quality of Industry Automation for Detection of System Parts Using Raspberry Pi.' *Journal of Physics: Conference Series* 1913 (1): 012134.
- Spinellis, Diomidis. 2006. 'Choosing a Programming Language.' *IEEE Software* 23 (4): 62–63.
- Stallman, Richard. n.d. 'Linux and the GNU System.' *GNU Operating System*. <https://www.gnu.org/gnu/linux-and-gnu.en.html>.
- Stojanoski, Hristijan, Dijana Capeska Bogatinoska, Abdel-Badeeh M. Salem, and Vineta Srebrenkoska. 2017. 'Practical, Cheap Smart Home Implementation with General Purpose Embedded Hardware Raspberry Pi.' In *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, 335–341. Cairo: Institute of Electrical and Electronics Engineers.
- Stroustrup, Bjarne. 2013. *The C++ Programming Language*. 4th ed. Westport: Addison-Wesley Professional.
- Sunehra, Dhiraj. 2019. 'Raspberry Pi Based Pollution and Climate Monitoring System Using Internet of Things.' *International Journal of Advanced Research in Engineering and Technology (IJARET)* 10 (2): 44–54.
- Suresh, Natarajan, E. Balaji, K. Jeffry Anto, and J. Jenith. 2014. 'Raspberry Pi Based Liquid Flow Monitoring and Control.' *International Research Journal of Engineering and Technology (IRJET)* 3 (7): 122–125.
- Tavasalkar, Dinesh. 2019. *Hands-On Robotics Programming with C++*. Birmingham: Packt.
- Tollervey, Nicholas H. n.d. 'Code with Mu: A Simple Python Editor for Beginner Programmers.' <https://codewith.mu/en/>.
- Upton, Eben. 2023. 'Introducing: Raspberry Pi 5!' *Raspberry Pi*, 28 September. <https://www.raspberrypi.com/news/introducing-raspberry-pi-5/>.
- Usa. 2019. 'Ruby 2.4.7 Released.' *Ruby*, 28 August. <https://www.ruby-lang.org/en/news/2019/08/28/ruby-2-4-7-released/>.
- Venkatesh, K., S. Hemaswathi, Rajalingam Balakrishnan, and Raj Kumar. 2018. 'IoT Based Home Automation Using Raspberry Pi.' *Journal of Advanced Research in Dynamical and Control Systems* 10 (7): 1721–1728.
- Vieira, Gustavo, José Barbosa, Paulo Leitão, and Lucas Sakurada. 2020. 'Low-Cost Industrial Controller Based on the Raspberry Pi Platform.' In *2020 IEEE International Conference on Industrial Technology (ICIT)*, 292–297. Buenos Aires: Institute of Electrical and Electronics Engineers.
- Wall, Larry, Tom Christiansen, and Jon Orwant. 2000. *Programming Perl*. 3rd ed. Cambridge, MA: O'Reilly Media.
- Ward, Brian. 2021. *How Linux Works: What Every Superuser Should Know*. 3rd ed. San Francisco: No Starch.
- Waskom, Michael L. 2021. 'Seaborn: Statistical Data Visualization.' *Journal of Open Source Software* 6 (60): 3021.

- Whittaker, Ashley. 2022. 'Medical Device Development with Raspberry Pi.' *Raspberry Pi*, 16 November. <https://www.raspberrypi.com/news/medical-device-development-with-raspberry-pi/>.
- Winograd, Terry. 1979. 'Beyond Programming Languages.' *Communications of the ACM* 22 (7): 391–401.
- Wolfram, Stephen. 2013. 'Putting the Wolfram Language (and Mathematica) on Every Raspberry Pi.' *Stephen Wolfram Writings*, 21 November. <https://blog.wolfram.com/2013/11/21/putting-the-wolfram-language-and-mathematica-on-every-raspberry-pi/>.
- Wolfram, Stephen. 2017. *An Elementary Introduction to the Wolfram Language*. 2nd ed. Champaign, IL: Wolfram Media.
- Wolfram. n.d.-a. 'Wolfram Mathematica: The World's Definitive System for Modern Technical Computing.' <https://www.wolfram.com/mathematica/>.
- . n.d.-b. 'Wolfram Neural Net Repository.' <https://resources.wolframcloud.com/NeuralNetRepository/>.
- Zanzmeriya, Divyesh, and Ankita Panara. 2018. 'Implementation of Industrial Automation Systems Using Raspberry Pi by IOT with FIREBASE.' *International Research Journal of Engineering and Technology (IRJET)* 5 (5): 1330–1334.

Recommended readings

- Raspberry Pi User Guide* (Upton and Halfacree 2012). This book is coauthored by the founder of the Raspberry Pi Foundation and provides a thorough introduction to the functionality and usage of the Raspberry Pi.
- Adventures in Raspberry Pi* (Philbin 2017). Aimed at young adults and beginners, this book provides a fun and interactive guide through various Raspberry Pi projects, providing a practical application of computer science concepts.
- Getting Started with Raspberry Pi* (Richardson and Wallace 2012). A great resource for those beginning their journey with the Raspberry Pi, offering insights into set-up, programming, and hardware interaction.
- Raspberry Pi Cookbook: Software and Hardware Problems and Solutions* (Monk 2022). For users looking for a comprehensive guide that walks through various software and hardware projects, Simon Monk's cookbook proves to be an invaluable resource.
- Exploring Raspberry Pi: Interfacing to the Real World with Embedded Linux* (Molloy 2016). This book dives deep into interfacing, controlling, and managing real-world devices with the Raspberry Pi, bridging the gap between embedded systems and the device.
- Programming the Raspberry Pi: Getting Started with Python* (Monk 2012). A wonderful resource for those keen to explore the powerful combination of Python programming and Raspberry Pi hardware, useful for various applications including IoT, sensors, and more.
- Raspberry Pi Projects for the Evil Genius* (Norris 2013). For enthusiasts who enjoy fun, challenging projects, this guide will walk you through several innovative uses for the Raspberry Pi, allowing the exploration of its vast capabilities.
- The Official Raspberry Pi Beginner's Guide* (Halfacree 2020). Created by the Raspberry Pi Foundation, this official guide helps beginners navigate through the initial set-up, exploring the capabilities and features of the Raspberry Pi.
- Get Started with MicroPython on Raspberry Pi Pico* (Halfacree and Everard 2021). RP2040, a microcontroller, is the heart of Raspberry Pi Pico, the smallest board of the Raspberry Pi family, used for controlling connected hardware by writing programs in MicroPython.

20 *Easy Raspberry Pi Projects: Toys, Tools, Gadgets, and More!* (Santos and Santos 2018). A good starting point if you want to learn about LEDs, displays, sensors, cameras, web applications, and games and toys. The book is suitable for beginners as you do not need any previous knowledge, except a keen interest in the world of electronics and programming. Most projects are based on Raspberry Pi 3.

Hacking Electronics: Learning Electronics with Arduino and Raspberry Pi (Monk 2017). For those who want to equip themselves with the skills needed to understand and use Arduino microcontroller boards.

Learning Python with Raspberry Pi (Bradbury and Everard 2014). When the book was published, the authors claimed that Python was one of the world's top programming languages. This is an excellent companion for anyone wishing to learn how to program on the Raspberry Pi. The book is packed with examples and usable source code.

Recommended web resources

Raspberry Pi Foundation Website. The official website and an indispensable resource for all things Raspberry Pi, including the latest news, forums, blogs, and downloadable resources like the Raspberry Pi OS.

Instructables' Raspberry Pi Channel. Instructables offers a wide array of DIY projects, with step-by-step guides on how to accomplish various projects using the Raspberry Pi, suitable for various skill levels.

Pi My Life Up. A fantastic site with a plethora of Raspberry Pi projects and tutorials, ranging from beginner to advanced levels, covering diverse topics from setting up a server to building a gaming console.

Adafruit's Raspberry Pi Section. Adafruit offers numerous learning guides and sells hardware that can be integrated with the Raspberry Pi, making it a great place for project ideas and acquiring compatible components.

The MagPi Magazine. The official Raspberry Pi magazine, which provides the latest news, reviews, features, and tutorials related to Raspberry Pi and its various applications.

Hackaday's Raspberry Pi Section. Hackaday features numerous Raspberry Pi projects and hacks, showcasing innovative and creative applications of the Raspberry Pi in various domains.

Stack Exchange's Raspberry Pi Section. A question-and-answer site where users can post inquiries regarding Raspberry Pi and get answers from a vibrant community of Raspberry Pi enthusiasts and experts.

Element14 Raspberry Pi Community. This is a forum where you can join discussions, find project ideas, and get support on various topics related to Raspberry Pi, making it a crucial site for both newcomers and seasoned users.

Reddit's Raspberry Pi Forum. A thriving subreddit where users discuss news, projects, and challenges related to Raspberry Pi, with members often sharing their own project guides and seeking or offering assistance.

GitHub – Raspberry Pi. The official Raspberry Pi repositories on GitHub, which include source codes and support files for Raspberry Pi products, allowing users to delve into the more technical aspects and even contribute to the development ecosystem.

Arch Linux Wiki. Arch is a Linux distribution that is popular among intermediate and advanced users. Nevertheless, its wiki is highly recommen-

ded even for novice users, as it features a plethora of useful information regarding Linux-compatible hardware and software, an overview of applications and commands, as well as comprehensive tutorials. Many of its articles have been translated into multiple languages. An essential resource for acquainting yourself with Linux systems.

Luke Smith's YouTube channel. Luke Smith has uploaded many useful tutorials, reviews, and discussion videos about open-source software and Linux systems. He has extremely good tutorials on Linux shells, command-line software, system customization, Vim, and LaTeX. Besides the technical content, he also publishes philosophical and political content. Thus, if the reader wants to watch videos selectively, as a starting point, one of the authors highly recommends checking out the playlists about [command-line programs](#) and [the terminal](#). Furthermore, if the reader decides to delve into learning to use the Vim editor, Luke has a [series of videos about Vim](#), showing its practical applications in development and writing documents.

Derek Taylor's YouTube channel. Derek Taylor, also known as “DistroTube”, started out by reviewing various Linux distributions, explaining the installation process and how to set them up. Currently, in addition to that, he uploads reviews of various open-source software, guides on Vim and Emacs editors, tutorials on shell commands and system customization, among other topics. He also occasionally publishes tips on writing useful Bash scripts. It is highly recommended to check out some of his videos, particularly his playlists about the [command line](#), [Linux customization](#), [shell scripting](#), and [“from noob to power user”](#) on various Linux distributions.

Reviews

The monograph *Raspberry Pi as a Foundation for Boosting Computer and Technology Literacy: Amplifying Understanding through Projects* by Igor Rižnar and Aleksandr Gaisov is an extensive guide that explores the potential of Raspberry Pi in enhancing computer and technology literacy. Through a comprehensive layout, the authors cover a wide range of topics ranging from the basics of Raspberry Pi, to its operating systems, and accessories, to its application in education, DIY projects, and even in industrial settings. The book is designed to be a journey of discovery, from simple projects to advanced applications, aiming to deepen the reader's understanding of computing and programming. It emphasizes the Raspberry Pi's role in STEM education, its affordability, and its support for project-based learning. With detailed explanations of basic and advanced projects, programming languages, and the use of Raspberry Pi in various applications, this monograph is a valuable resource for educators, students, and hobbyists looking to explore the capabilities of Raspberry Pi and enhance their computer literacy. Its structure is as follows: first, the authors explain the term computer literacy in the context of other literacies, after which they write a chapter on Raspberry Pi operating system(s). This chapter is followed by an introduction to the fundamentals of Linux and terminal usage, subdividing it further into basic shell commands, an overview of shell commands, and finish it by a subchapter describing the customization of the system and the shell. Their work also includes Raspberry Pi official accessories and several accessories produced by other companies. To help readers who are willing to try any of the described simple and advanced projects they also make suggestions for miscellaneous tools for electronics experiments and projects.

The authors devoted an important chapter to Raspberry Pi usage in educational institutions, where they give examples of its use in primary, secondary, and higher education. A subchapter on Raspberry Pi usage describes machine learning and Internet of Things as well as the appli-

cation of Rpi SOC in various industries. Programming on Raspberry Pi is the main topic of the next chapter, which covers advice on selecting a programming language, available programming languages and an overview of the most important programming languages for Raspberry Pi computer. The final part of this chapter is an overview of development environments, where authors discuss Scratch, Mathematica, and Sonic Pi. For readers who prefer hands-on experience there is a large section of basic and advanced projects with detailed descriptions of the procedures to reach the final goal. In the chapter preceding the Conclusion, the authors briefly compare Raspberry Pi Zero, Arduino, and BBC micro:bit.

This reviewer believes that the monograph written by Rižnar and Gaisov fills in the gap that has remained unfilled for too many years in many countries, especially in Slovenia where Raspberry Pi's potential in education has been underexploited. The authors present a compelling case for the adoption of this affordable and versatile platform in schools to foster hands-on learning. Their work not only highlights the practical applications of Raspberry Pi but also its role in democratizing access to technology education. By providing an easy-to-understand introduction to the Raspberry Pi family of computers for educators, DIYs and learners alike, Rižnar and Gaisov's monograph stands as an important resource for integrating Raspberry Pi into educational curriculums, thereby enhancing technology literacy and preparing students for a future in which digital skills are paramount. For this reason, I warmly recommend their monograph for publication.

Uroš Godnov

Igor Rižnar and Aleksandr Gaisov's monograph *Raspberry Pi as a Foundation for Boosting Computer and Technology Literacy: Amplifying Understanding through Projects* serves as an in-depth manual for utilizing Raspberry Pi to improve understanding of computers and technology. This guide addresses a variety of subjects, starting with Raspberry Pi essentials, its operating systems, and peripherals, extending to its role in educational environments, do-it-yourself initiatives, and professional applications. The text is crafted as an exploratory progression, guiding readers from introductory activities to complex uses, to enhance knowledge in computing and coding. It highlights the significance of Raspberry Pi in STEM learning, noting its cost-effectiveness and its

facilitation of project-oriented education. Offering clear descriptions of both fundamental and intricate projects, coding languages, and Raspberry Pi's diverse uses, the guide stands as a crucial tool for teachers, students, and enthusiasts eager to delve into Raspberry Pi's possibilities and boost their tech proficiency.

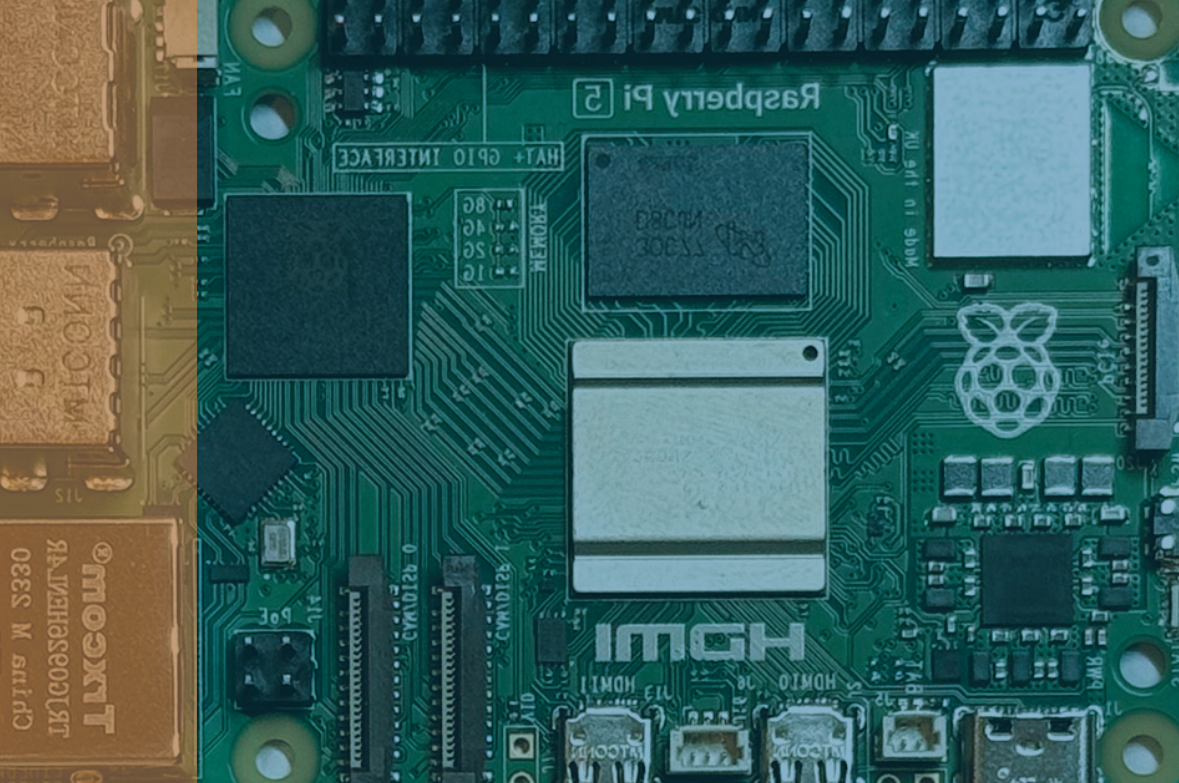
The guide's organization begins with a discussion on computer literacy, is followed by a detailed chapter on Raspberry Pi's operating systems. An introduction to Linux basics and terminal commands is provided, including sections on basic shell commands, an overview of these commands, and customizing the system and shell. The authors also explore official Raspberry Pi accessories alongside those from other manufacturers and recommend various tools for electronic experiments and projects.

A key section is dedicated to the use of Raspberry Pi in educational settings, offering examples from elementary to tertiary levels. Discussions on Raspberry Pi encompass machine learning and the Internet of Things, as well as its application in different industrial sectors. The narrative then shifts to programming on Raspberry Pi, advising on language selection, detailing available languages, and highlighting essential ones for Raspberry Pi computing. This is complemented by a review of development environments, including Scratch, Mathematica, and Sonic Pi. For those seeking practical experience, the guide provides an extensive array of simple to advanced project tutorials. Before concluding, the authors compare the Raspberry Pi Zero with similar devices like the Arduino and BBC micro:bit.

This monograph addresses a long-standing void, particularly in countries like Slovenia where the educational promise of Raspberry Pi has been underutilized. Rižnar and Gaisov make a persuasive argument for integrating this cost-effective and versatile tool into classrooms to promote experiential learning. Their monograph not only showcases Raspberry Pi's practical uses but also its capacity to make tech education more accessible. The main purpose of the authors, which was to advance technology literacy and equip students for a technology-driven future, has been completely achieved. By laying out the capabilities of the Raspberry Pi as both a teaching tool and a versatile platform for innovation, they have opened new avenues for educational enrichment. Their guide not only fosters a deeper understanding of computer science principles but also encourages creative problem-solving and hands-on learning. Through this comprehensive exploration, Rižnar and Gaisov have

not only met their objective but have also set a new benchmark for educational resources, making a significant contribution to the field of technology education. Their work signifies a shift towards more inclusive, engaging, and practical approaches to learning in the digital age. Hence, I highly endorse this monograph for publication.

Marjan Golob



The Raspberry Pi has a thriving community of developers, educators, and hobbyists who share projects, tutorials, and software through numerous online platforms, making it easy for newcomers to enter the Raspberry Pi ecosystem and effectively navigate their learning and project development journeys.

This monograph is an exploratory biography of our learning experience as we take the reader through the world of Raspberry Pi, from simple and fun projects for absolute beginners to more advanced projects, all intending to enhance your understanding and skills in the ever-evolving field of computing. We hope this is a journey about discovery, creativity, and the joy of learning by doing that will increase your understanding of what Raspberry Pi is capable of. Our journey spans over ten years, so it is unreasonable to expect that these 100 pages will cover all one needs to know and understand to become a seasoned user. To make it easier for you to continue to learn on your own, we will add some resources both while we write about a certain topic and at the end of the book, where several websites, clubs, tutorials as well as books and magazines will be mentioned.