CHAPTER 9

Programming with RPi GPIO

In the last chapter, we got introduced to high-level programming languages on Unix-like platforms. We learned to write programs with C and C++ with the GCC compiler. We also learned how to write and execute programs in the Python 3 programming language.

Continuing where we left off at the last chapter, in this chapter, we will explore GPIO programming with RPi and Python 3. The following is the list of topics we will explore in this chapter:

- GPIO pins
- Programming with GPIO

We will be comfortable with GPIO programming and usage of basic electronic components with Raspberry Pi after this chapter.

General-Purpose Input/Output Pins

The RPi board has General-Purpose Input/Output header pins. All the versions of the RPi board have this feature. This feature sets single-board computers apart from other small computers. The GPIO pins give SBCs the ability to directly interface with low-level electronic components and various data transfer buses.

A. Pajankar, *Practical Linux with Raspberry Pi OS*,
https://doi.org/10.1007/978-1-4842-6510-9_9

I am using an RPi 4 B board with 8 GB RAM for this chapter. It is the most recent board in the RPi family. We can see the meanings of pins on any RPi board by running the following RPi OS–specific command:

```
pinout
```

It will show us the layout of the board as shown in Figure 9-1.

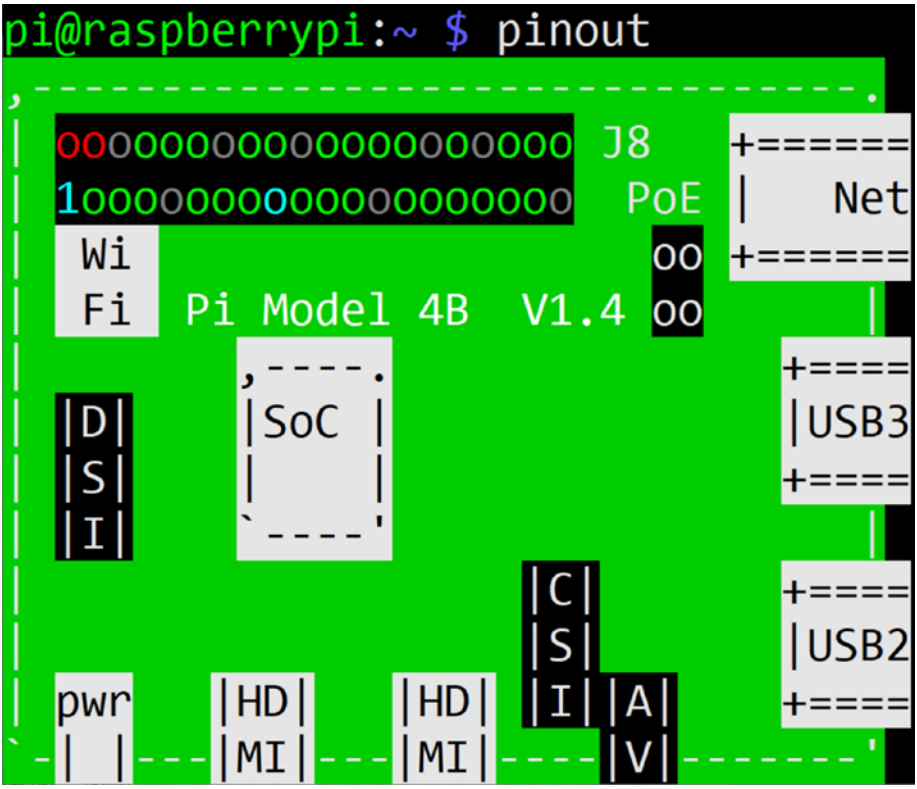

Figure 9-1. *Board layout of RPi 4 B*

In the top-left part of the figure, we can see 40 GPIO pins. A few earlier models of RPi had 26 pins. But they have long been out of production, so we will not discuss them here. The programs we will demonstrate in this

chapter are compatible with all the models of Pi. Figure 9-1 is the first part of the output shown on the command line. If we scroll down, we can see more. The last part of the output shows us the meanings of the pins as shown in Figure 9-2.

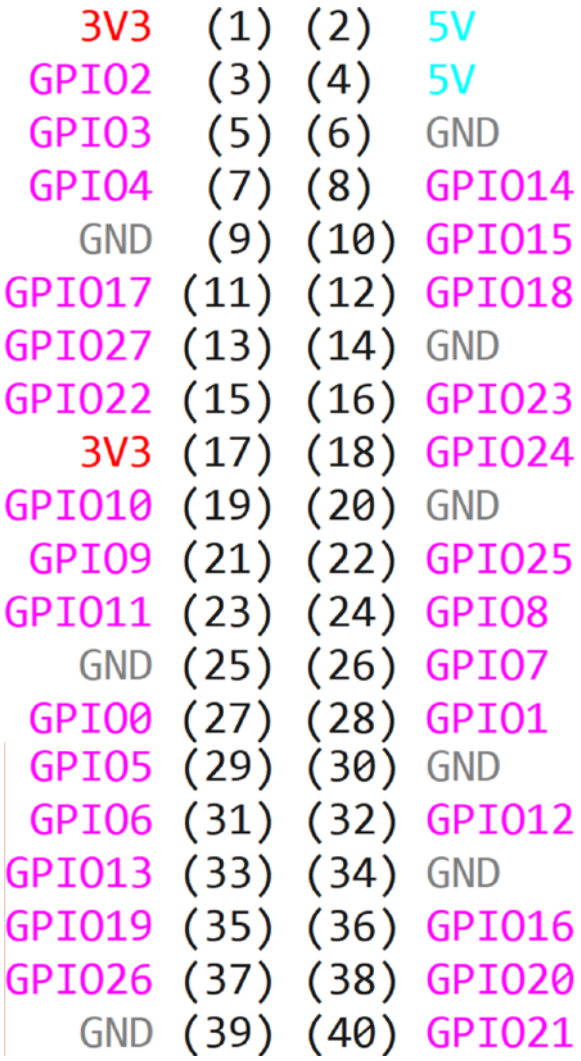

Figure 9-2. *Pinout of RPi 4 B*

This output shows us the power pins (5V, 3V3, and GND) and digital IO pins. GND stands for ground and 3V3 means 3.3 volts. We can see that there are two numbering schemes shown in the output. The physical pin numbers (also known as board pin numbers) are mentioned in the brackets, and BCM names are mentioned outside. For our convenience, we will use board (physical) pin numbering in the programs that we will write.

We can get an idea about the orientation of the pins by comparing both Figures 9-1 and 9-2. The pins are color-coded in both the figures.

Now, take a LED, a resistor, and a few jumper cables and prepare the circuit shown in Figure 9-3.

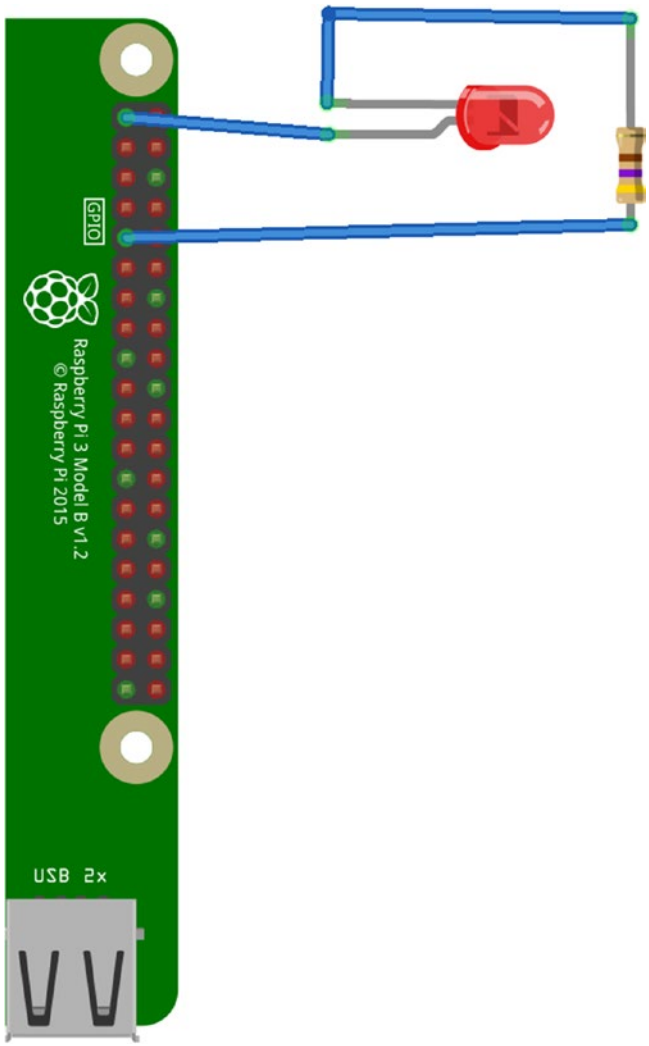

Figure 9-3. *Simple LED circuit*

The LED will glow as long as the RPi board is on. This is because we are connecting the anode of the LED (the longer pin) to the 3V3 pin and the cathode to a GND pin through a resistor of 470 Ohms. This is the simplest LED circuit we can make with this. You may want to try to connect the

anode of the LED to the 5V pin, and it will glow more. We have chosen the resistor with the appropriate value so it will not burn the LED. In the next section, we will see how we can write programs with GPIO.

Programming with GPIO

In this section, we will see simple circuits with LEDs and pushbuttons. We will use the Python 3 GPIO library for that. It comes preinstalled with the RPi OS. If it is not preinstalled, then install it with the following command:

```
sudo apt-get install python3-rpi.gpio -y
```

Prepare a circuit as shown in Figure 9-4.

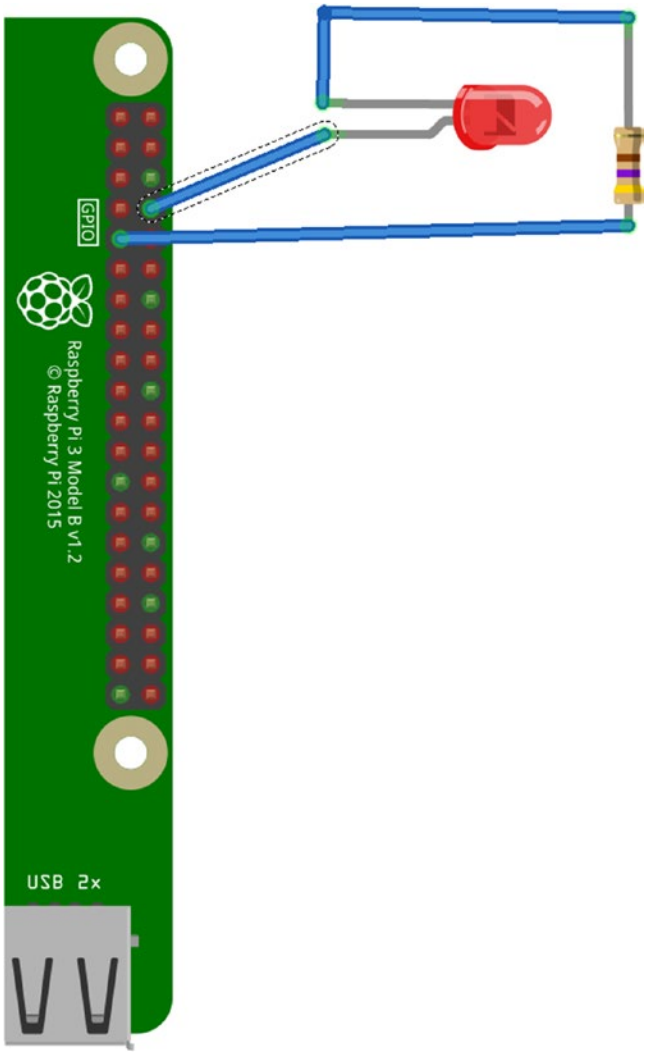

Figure 9-4. *Programmable LED circuit*

We are connecting the anode of the LED to the pin which is physically numbered as 8 in Figure 9-2. That is the only change we have from the earlier circuit. Check out Listing 9-1.

Listing 9-1. LED_Blink.py

```
from time import sleep
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW)

while True:
    GPIO.output(8, GPIO.HIGH)
    sleep(1)
    GPIO.output(8, GPIO.LOW)
    sleep(1)
```

Let us discuss it in detail. The first two lines import required libraries. Then we are instructing the RPi board to use the board (also called as, as we have seen earlier, physical) pin numbering with the statement `GPIO.setmode()`. Then we are disabling warnings. We are using the function `GPIO.setup()` to set board pin 8 to output mode. We are also setting its initial state as `LOW`. Then in an indefinite loop, we are alternately sending `HIGH` and `LOW` signals to pin 8. The call to the function `sleep()` adds an interval of 1 second between the statements. When pin 8 is `HIGH`, the LED is on; and when pin 8 is `LOW`, the LED is off. Run the script with the following command:

```
python3 LED_Blink.py
```

The LED will start blinking. To terminate the program, press Ctrl+C on the keyboard.

This is the basic GPIO programming. We can use this creatively to create various patterns of blinking LEDs if we use a breadboard to connect multiple LEDs to the digital GPIO pins.

Summary

We have continued our journey of Python 3 programming in this chapter and studied a program that makes a LED blink. We have also learned how to build a basic circuit with a LED and a resistor. Raspberry Pi has many more things to offer through its GPIO programming. You can explore this vast topic further at your own convenience.

In the next chapter, we will study the GUI of the RPi OS in detail. We will also see how to install various desktop environments for the RPi OS in detail.