elektor books

Raspberry Pi Pico W

Program, build, and master 60+ projects with the **Wireless RP2040**

Dogan Ibrahim

elektor
design > share > earn

Raspberry Pi Pico W

Program, build, and master 60+ projects with the **Wireless RP2040**

Dogan Ibrahim

elektor
design > share > sell

This is an Elektor Publication. Elektor is the media brand of
Elektor International Media B.V.
PO Box 11, NL-6114-ZG Susteren, The Netherlands
Phone: +31 46 4389444

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN 978-3-89576-529-2 Print
ISBN 978-3-89576-530-8 eBook

Editor: Alina Neacsu
Prepress Production: D-Vision, Julian van den Berg

Contents

Preface

A microcontroller is basically a single-chip computer including a CPU, memory, input-output circuitry, timers, interrupt circuitry, clock circuitry, and several other circuits and modules, all housed in a single silicon chip. Early microcontrollers were limited in their capacities and speed and they consumed considerable power. Most of the early microcontrollers were 8-bit processors with clock speeds in the region of several megahertz, and having only hundreds of bytes of program and data memories. These microcontrollers were traditionally programmed using assembly languages of the target processors. Today, 8-bit microcontrollers are still in common use, especially in small projects where large amounts of memory or high speed are not the main requirements. With the advancement of chip technology, we now have 32-bit and 64-bit microcontrollers with speeds in the region of several gigahertz, and several gigabytes of memory. Microcontrollers are nowadays programmed using high-level languages such as C, C#, BASIC, PASCAL, JAVA, etc.

The Raspberry Pi Pico is a high-performance microcontroller designed especially for physical computing. Readers should realize that microcontrollers are very different from single-board computers like the Raspberry Pi 4 (and other family members of the Raspberry Pi). There is no operating system on the Raspberry Pi Pico. Microcontrollers like the Raspberry Pi Pico can be programmed to run a single task and they can be used in fast real-time control and monitoring applications.

The Raspberry Pi Pico is based on the fast and very efficient dual-core ARM Cortex-M0+ RP2040 microcontroller chip, running at up to 133 MHz. The chip incorporates 264 KB of SRAM and 2 MB of Flash memory. What makes the Raspberry Pi Pico very attractive is its large number of GPIO pins, and commonly used peripheral interface modules, such as SPI, I^2C, UART, PWM, and fast and accurate timing modules.

Released in 2022, the Raspberry Pi Pico **W** is the latest model of the Pico family of microcontroller boards. The "Pico W" is identical to the standard "Pico" with one major difference: compared to other Pico family members, the "Pico W" has an on-board Wi-Fi module, thus enabling the board to be used in many communications, control, and especially in IoT based projects. The Pico W board also supports Bluetooth hardware, but the Bluetooth firmware was not ready at the time of writing this book.

Both the standard Raspberry Pi Pico and the Raspberry Pi Pico W can easily be programmed using some of the popular high-level languages, such as MicroPython, or C/C++. There are many application notes, tutorials, and data sheets available on the Internet on using the Pico or the Pico W.

This book is an introduction to using the Raspberry Pi Pico W microcontroller development board with the MicroPython programming language. The Thonny development environment (IDE) is used in all the projects in the book and the readers are recommended to use this IDE. There are many working and tested projects in the book, covering almost all aspects of the Raspberry Pi Pico W. Excepting the Wi-Fi-based subjects, all projects in the book can also be used with the standard Raspberry Pi Pico without any modifications.

The following sub-headings are given for each project where applicable to make the projects easy to follow for the readers:

- Title
- Brief description
- Aim
- Block diagram
- Circuit diagram
- Program listing with full description

I hope your next microcontroller-based projects make use of the Raspberry Pi Pico W, and that this book becomes useful in the development of your projects.

Dr Dogan Ibrahim
London, 2022

Chapter 1 • Raspberry Pi Pico W Hardware

1.1 Overview

The Raspberry Pi Pico W is a single-board microcontroller module developed by the Raspberry Pi Foundation. This module is based on the RP2040 microcontroller chip. In this chapter, you will be looking at the hardware details of the Raspberry Pi Pico W microcontroller module in some detail. From now on, this microcontroller module Pico W will be called "Pico" for short.

1.2 The Pico hardware module

Pico is a very low-cost $6 microcontroller module based on the RP2040 microcontroller chip with a dual-core Cortex-M0+ processor. Figure 1.1 shows the front view of the Pico hardware module which is basically a small board. In the middle of the board is the tiny 7 × 7 mm RP2040 microcontroller chip housed in a QFN-56 package. At the two edges of the board, there are 40 gold-coloured metal GPIO (General-Input-Output) pins with holes. You should solder pins to these holes so that external connections can be made easily to the board. The holes are marked starting with number 1 at the top left corner of the board and the numbers increase downwards up to number 40 which is at the top right-hand corner of the board. The board is breadboard compatible (i.e., 0.1-inch pin spacing), and after soldering the pins, the board can be plugged on a breadboard for easy connection to the GPIO pins using jumper wires. Next to these holes, you will see bumpy circular cut-outs which can be plugged in on top of other modules without having any physical pins fitted.

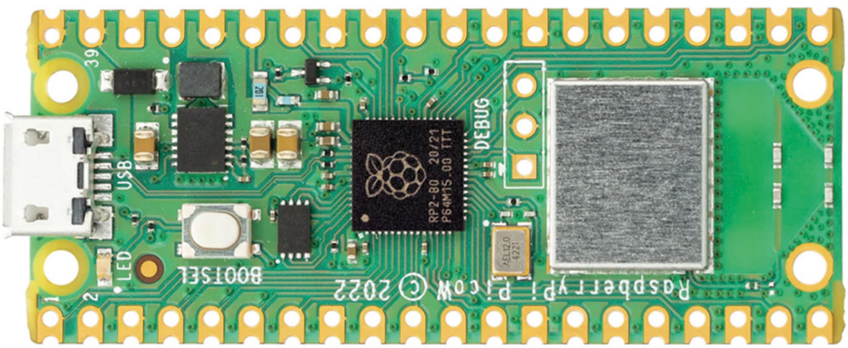

Figure 1.1: Front view of the Pico hardware module.

At one edge of the board, there is the micro-USB B port for providing power to the board and for programming the board. Next to the USB port, there is an on-board user LED that can be used during program development. Next to this LED, there is a button named BOOTSEL that is used during the programming of the microcontroller as you will see in the next chapters. Next to the processor chip, there are 3 holes where external connections can be made to. These are used to debug your programs using Serial Wire Debug (SWD). At the other edge of the board, is the single-band 2.4 GHz Wi-Fi module (802.11n). next to the Wi-Fi module is the on-board antenna.

Figure 1.2 shows the back view of the Pico hardware module. Here, all the GPIO pins are identified with letters and numbers. You will notice the following types of letters and numbers:

GND	- power supply ground (digital ground)
AGND	- power supply ground (analog ground)
3V3	- +3.3 V power supply (output)
GP0 – GP22	- digital GPIO
GP26_A0 – GP28_A2	- analog inputs
ADC_VREF	- ADC reference voltage
TP1 – TP6	- test points
SWDIO, GND, SWCLK	- debug interface
RUN	- default RUN pin. Connect LOW to reset the RP2040
3V3_EN	- this pin by default enables the +3.3 V power supply. +3.3 V can be disabled by connecting this pin LOW
VSYS	- system input voltage (1.8 V to 5.5 V) used by the on-board SMPS to generate +3.3 V supply for the board
VBUS	- micro-USB input voltage (+5 V)

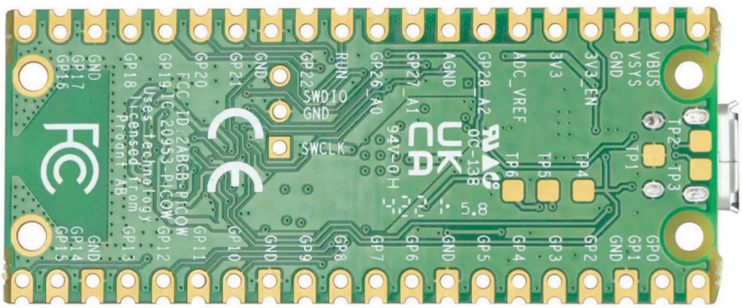

Figure 1.2: Back view of the Pico hardware module.

Some of the GPIO pins are used for internal board functions. These are:

GP29 (input)	- used in ADC mode (ADC3) to measure VSYS/3
GP24 (input)	- VBUS sense - HIGH if VBUS is present, else LOW
GP23 (output)	- Controls the on-board SMPS Power Save pin

The specifications of the Pico hardware module are as follows:

- 32-bit RP2040 Cortex-M0+ dual-core processor operating at 133 MHz
- 2 MByte Q-SPI Flash memory
- 264 KByte SRAM memory
- 26 GPIO (+3.3 V compatible)
- 3× 12-bit ADC pins
- Accelerated floating point libraries on-chip
-

- On-board single-band Infineon CYW43439 wireless chip, 2.4 GHz wireless interface (802.11b/g/n) and Bluetooth 5.2 (not supported at the time of writing)
- Serial Wire Debug (SWD) port
- Micro-USB port (USB 1.1) for power (+5 V) and data (programming)
- 2× UART, 2 × I^2C, 2 × SPI bus interface
- 16 PWM channels
- 1× Timer (with 4 alarms), 1× Real Time Counter
- On-board temperature sensor
- On-board LED at GPIO0, controlled by the 43439 module
- Castellated module allowing soldering direct to carrier boards
- 8× Programmable IO (PIO) state machines for custom peripheral support
- MicroPython, C, C++ programming
- Drag & drop programming using mass storage over USB

Notice that on the Raspberry Pi Pico board, the on-board LED is connected to GP25 and is available to the user. On the Raspberry Pi Pico W board, the on-board LED is controlled by the 43439 Wi-Fi module.

Pico GPIO hardware is +3.3 V compatible, and it is, therefore, important to be careful not to exceed this voltage when interfacing external input devices to the GPIO pins. +5 V to +3.3 V logic converter circuits or resistive potential divider circuits must be used if it is required to interface devices with +5 V outputs to the Pico GPIO pins.

Figure 1.3 shows a resistive potential divider circuit that can be used to lower +5 V to +3.3 V.

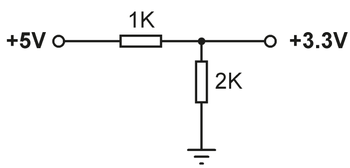

Figure 1.3: Resistive potential divider circuit.

1.3 Comparison with the Arduino UNO

Arduino UNO is one of the most popular microcontroller development boards used by students, practising engineers, and hobbyists. Table 1.1 shows a comparison of the Raspberry Pi Pico W with the Arduino UNO. It is clear from this table that the Pico W is much faster than the Arduino UNO, it has larger flash and data memories, offers Wi-Fi, provides more digital input-output pins, and has an on-board temperature sensor. Arduino UNO operates with +5 V and its GPIO pins are +5 V compatible. Perhaps some advantages of the Arduino UNO are that it has a built-in EEPROM memory, and its ADC is 6 channels instead of 3.

Feature	Raspberry Pi Pico W	Arduino UNO
Microcontroller	RP2040	Atmega328P
Core and bits	Dual core, 32-bits, Cortex-M0+	Single-core 8-bits
RAM	264Kbyte	2KByte
Flash	2MByte	32KByte
CPU speed	48MHZ to 133MHz	16MHz
EEPROM	None	1KByte
Wi-Fi support	CYW43439 wireless chip	None
Power input	+5V through USB port	+5V through USB port
Alternative power	2 – 5V via VSYS pin	7 – 12V
MCU operating voltage	+3.3V	+5V
GPIO count	26	20
ADC count	3	6
Hardware UART	2	1
Hardware I^2C	2	1
Hardware SPI	2	1
Hardware PWM	16	6
Programming languages	MicroPython, C, C++	C (Arduino IDE)
On-board LED	1	1
Cost	$6	$20

Table 1.1: Comparison of Raspberry Pi Pico W and Arduino UNO.

1.4 Operating conditions and powering the Pico

The recommended operating conditions of the Pico are:

- Operating temperature: -20 ºC to +85 ºC
- VBUS voltage: +5 V ±10%
- VSYS voltage: +1.8 V to +5.5 V

An on-board SMPS is used to generate the +3.3 V to power the RP2040 from a range of input voltages from 1.8 V to +5.5 V. For example, 3 alkaline AA batteries can be used to provide +4.5 V to power Pico.
Pico can be powered in several ways. The simplest method is to plug the micro-USB port to a +5 V power source, such as the USB port of a computer or a +5 V power adapter. This will provide power to the VSYS input (see Figure 1.4) through a Schottky diode. The voltage at the VSYS input is therefore VBUS voltage minus the voltage drop of the Schottky diode (about +0.7 V). VBUS and VSYS pins can be shorted if the board is powered from an external +5 V USB port. This will increase the voltage input slightly and hence reduce ripples on VSYS. VSYS voltage is fed to the SMPS through the RT6150 which generates fixed +3.3 V for the MCU and other parts of the board. VSYS is divided by 3 and is available at analog input port GPIO29 (ADC3) which can easily be monitored. GPIO24 checks the existence of VBUS voltage and is at logic HIGH if VBUS is present.

Another method to power the Pico is by applying external voltage (+1.8 V to +5.5 V) to the VSYS input directly (e.g., using batteries or an external power supply). You can also use the USB input and VSYS inputs together to supply power to Pico, for example, to operate with both batteries and the USB port. If this method is used, then a Schottky diode should be used at the VSYS input to prevent the supplies from interfering with each other. The higher voltages will power VSYS.

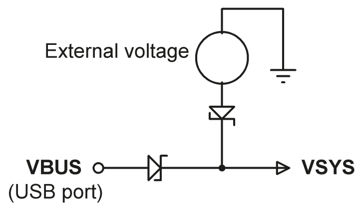

Figure 1.4: Powering the Pico.

1.5 Pinout of the RP2040 microcontroller and Pico module

Figure 1.5 shows the RP2040 microcontroller pinout, which is housed in a 56-pin package. The Pico module pinout is shown in Figure 1.6 in detail. As you can see from the figure, most pins have multiple functions. For example, GPIO0 (pin 1) is also the UART0 TX, I2C0 SDA, and the SPI0 RX pins.

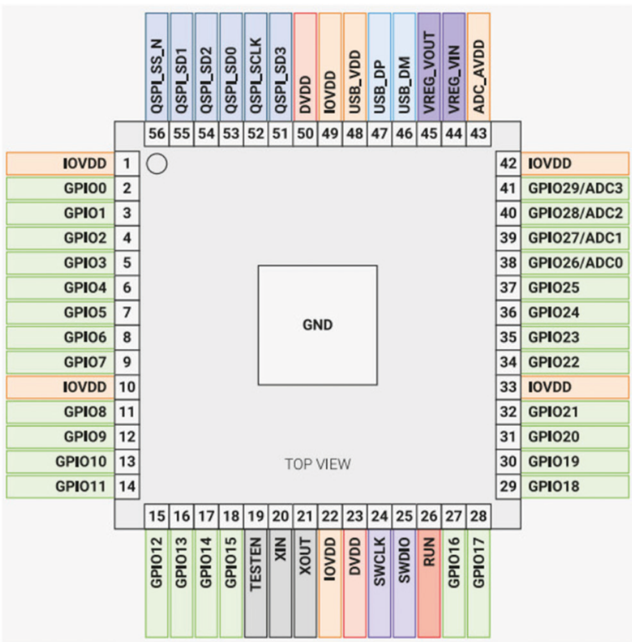

Figure 1.5: RP2040 microcontroller pinout.

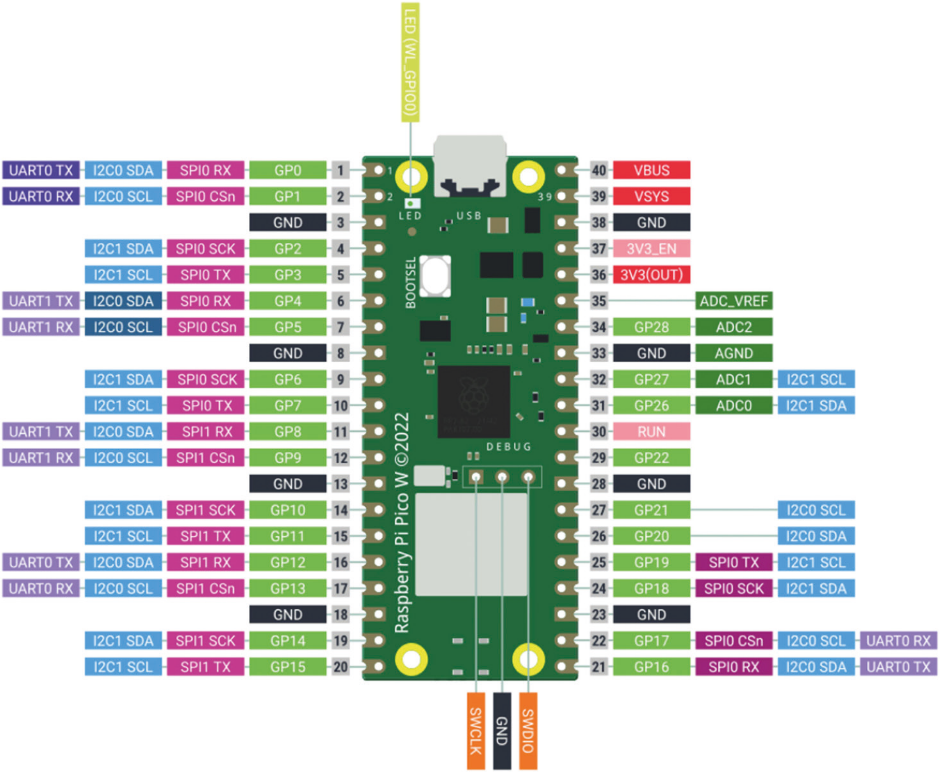

Figure 1.6: Pico pinout.

Figure 1.7 shows a simplified block diagram of the Pico hardware module. Notice that the GPIO pins are directly connected from the microcontroller chip to the GPIO connector. GPIO 26-28 can be used either as digital GPIO or as ADC inputs. ADC inputs GPIO26-29 have reverse diodes to 3Vs and therefore the input voltage must not exceed 3V3+300 mV. Another point to note is that if the RP2040 is not powered, applying voltages to GPIO26-29 pins may leak through the diode to the power supply (there is no problem with the other GPIO pins and voltage can be applied when the RP2040 is not powered).

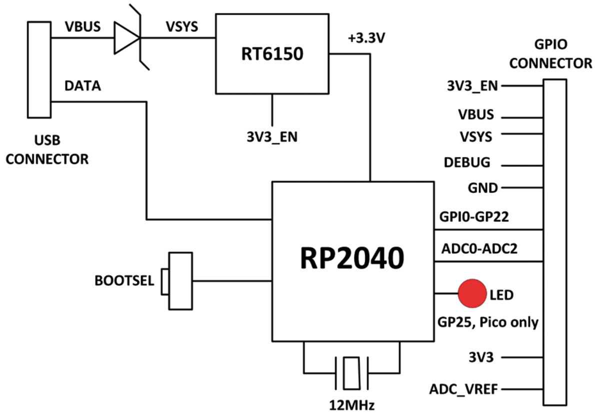

Figure 1.7: Simplified block diagram.

1.6 Other RP2040 microcontroller-based boards

There are many other RP2040 microcontroller-based development boards. At the time of writing this book some of these boards were:

- Adafruit Feather RP2040
- Adafruit ItsyBitsy RP2040
- Adafruit QT Py RP2040
- Pimoroni PicoSystem
- Pimoroni Tufty 2040
- Arduino Nano RP2040 Connect
- SparkFun Thing Plus RP2040
- Pimoroni Pico Explorer Base
- Pimoroni Pico Lipo
- SparkFun MicroMod RP2040 Processor
- SparkFun Pro Micro RP2040
- Pico RGB Keypad Base
- Pico Omnibus
- Pimoroni Pico VGA Demo Base
- Cytron Maker Pi RP2040 development board
- Technoblogy – Minimal RP2040 board
- Pimoroni Tiny 2040
- Seeed Studio Wio RP2040 Mini development board
- Seeed XIAO RP2040 development board
- And many more

Chapter 2 • Raspberry Pi Pico W Programming

2.1 Overview

At the time of writing this book, the Raspberry Pi Pico W accept programming with the following programming languages:

- C/C++
- MicroPython
- Assembly language

Although Pico by default is set up for use with the powerful and popular C/C++ language, many beginners find it easier to use MicroPython, which is a version of the Python programming language developed specifically for microcontrollers.

In this chapter, you will learn how to install and use the MicroPython programming language. You will be using the Thonny text editor which has been developed specifically for Python programs.

Many working and fully tested projects will be given in the next chapters using MicroPython with your Pico.

2.2 Installing MicroPython on Pico W

MicroPython must be installed on Raspberry Pi Pico W before the board can be used. Once installed MicroPython stays on your Pico unless it is overwritten with something else. Installing the MicroPython requires an Internet connection, and this is required only once. This can be done either by using a Raspberry Pi (e.g., Raspberry Pi 4), or by using a PC. In this section, you will see how to install using a PC (e.g., Windows 10).

The steps are as follows:

- Make sure your PC is connected to the Internet.
- Download the Raspberry Pi Pico W MicroPython UF2 file to a folder (e.g., Downloads) on your PC from the following link. At the time of writing this book the file had the name: **rp2-pico-w-20220909-unstable-v1.19.1-389-g4903e48e3.uf2**.

https://www.raspberrypi.com/documentation/microcontrollers/micropython.html
#drag-and-drop-micropython

- Push and hold down the **BOOTSEL** button on your Pico.
- Connect your Pico to the USB port of your PC using a micro-USB cable while holding down the button.
- Wait a few seconds and let go of the **BOOTSEL** button.
- You should see the Pico appear as a removable drive with the name **RPI-RP2** as shown in Figure 2.1 (drive **E:** in this case).

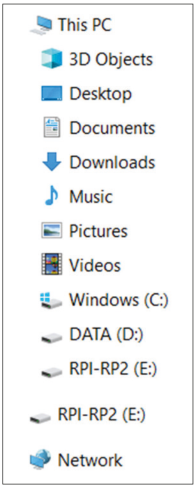

Figure 2.1: Pico as a removable drive RPI-RP2.

- Drag and drop the downloaded MicroPython UF2 file onto the RPI-RP2 volume. Your Pico will reboot, and you are now running MicroPython on your Pico.
- Powering down the Pico will not remove **MicroPython** from its memory.

2.3 Using the Thonny text editor from the PC

In this section, you will be learning how to use the Thonny on the PC to develop and run your programs.

First, you must install Thonny on your PC (if it is not already installed). The steps are:

- Go to the Thonny.org web site:

  https://thonny.org/

- Click on the link at the top right-hand side of the screen to install Thonny (see Figure 2.2).

Figure 2.2: Click to install Thonny.

- You should see an icon on the Desktop (Figure 2.3) of your PC. Double click to start Thonny.

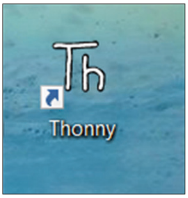

Figure 2.3: Thonny icon on the Desktop.

- The startup screen of Thonny on your PC is shown in Figure 2.4.

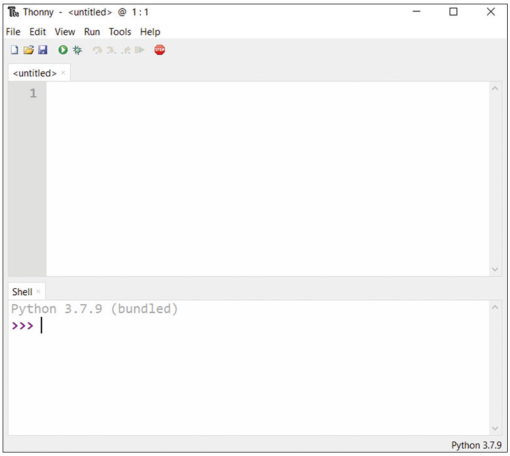

Figure 2.4: Thonny startup screen on the PC.

- Click label **Python** at the bottom right-hand corner of the screen and click to select **MicroPython (Raspberry Pi Pico).**
- You are now ready to write your programs. Let us write a very simple Python statement to test the installation of MicroPython on your Pico.
- Enter the following statement at the lower part of the screen (in **Shell**):

```
print("hello from Thonny on PC")
```

- You should the message **hello from Thonny on PC** is displayed as shown in Figure 2.5.

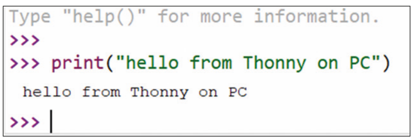

Figure 2.5: Displaying the message.

In this book, you will be using Thonny on the PC to write programs and to execute them on the Raspberry Pi Pico W.

Simple software-only example programs are given in the remaining sections of this chapter. The aim here has been to review the basic Python programming concepts. This book does not aim to teach the Python programming language. Interested readers can find many books and tutorials on the Internet for learning the Python programming language.

2.4 Writing a program using Thonny

In the previous section, you have learned how to execute a statement online using the Thonny **Shell**. Almost in all applications, you must write programs. As an example, the steps to write and run a very simple one-line program to display the message **Hello from program...** are given below:

- Enter the program statements at the upper part of the screen as shown in Figure 2.6.

Figure 2.6: Write the program on the upper part of the screen.

- Click **File** followed by **Save As** and give a name to your program. e.g., **FirstProg.** You have the option of storing the program either on your PC (**This computer**) or on the Pico (**Raspberry Pi Pico**). Click **Raspberry Pi Pico** to save it on the Pico (Figure 2.7). Enter the name of your program (**FirstProg**) and click **OK** (notice that the file is saved with the extension **.py**).

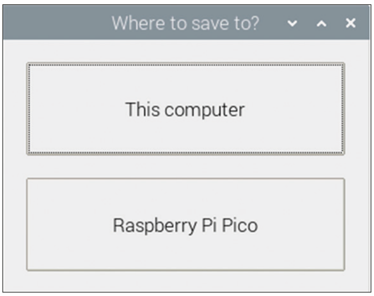

Figure 2.7: Click **Raspberry Pi Pico** *to save your program.*

- Click the green arrow icon at the top of the screen to run your program. The output of the program will be displayed in the lower **Shell** part of the screen as shown in Figure 2.8.

Figure 2.8: Output of the program.

2.5 Software only MicroPython programs using the Raspberry Pi Pico W

Example 1: Average of two numbers read from the keyboard

In this example, two numbers are read from the keyboard and their average is displayed. The aim of this example is to show how data can be read from the keyboard.

Solution 1

The program is named **Average** and the program listing and an example run of the program are shown in Figure 2.9. Function **input** is used to read the numbers in the form of strings from the keyboard. These strings are then converted into floating-point numbers and stored in variables **n1** and **n2**. The average is calculated by adding and then dividing the numbers by two. The result is displayed on the screen.

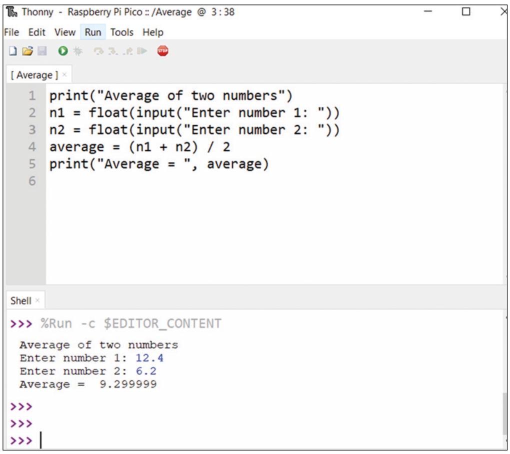

Figure 2.9: Program: ***Average*** *and sample run.*

Example 2: Average of 10 numbers read from the keyboard

In this example, 10 numbers are read from the keyboard and their average is displayed. The aim of this example is to show how a loop can be constructed in Python.

Solution 2

The program is named **Average10,** and the program listing and an example run of the program are shown in Figure 2.10. In this program, a loop is constructed which runs from 0 to 9 (i.e., 10 times). Inside this loop, the numbers are read from the keyboard, added to each other, and stored in a variable **sum**. The average is then calculated and displayed by dividing the **sum** by 10. Notice that a new-line is not printed after the print statements since the option **end = ' '** is used inside the print statement.

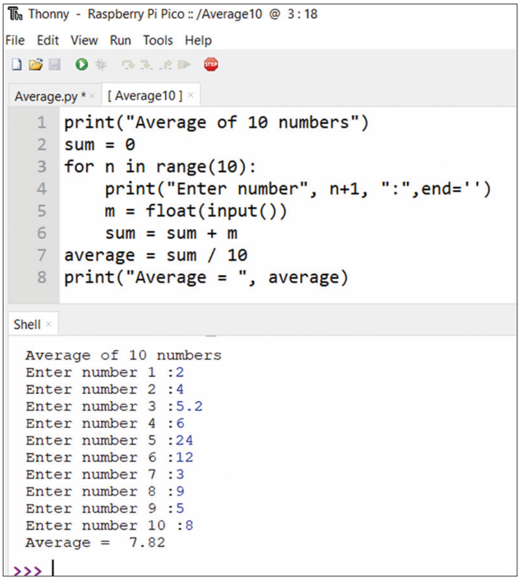

Figure 2.10: Program: **Average10** *and sample run.*

Example 3: Surface area of a cylinder

In this example, the radius and height of a cylinder are read from the keyboard and its surface area is displayed on the screen.

Solution 3

The program is named **CylArea,** and the program listing and an example run of the program are shown in Figure 2.11. The surface area of a cylinder is given by:

Surface area = $2\pi rh$

Where **r** and **h** are the radius and height of the cylinder respectively. In this program the **math** library is imported so that function **Pi** can be used in the program. The surface area of the cylinder is displayed after reading its radius and height.

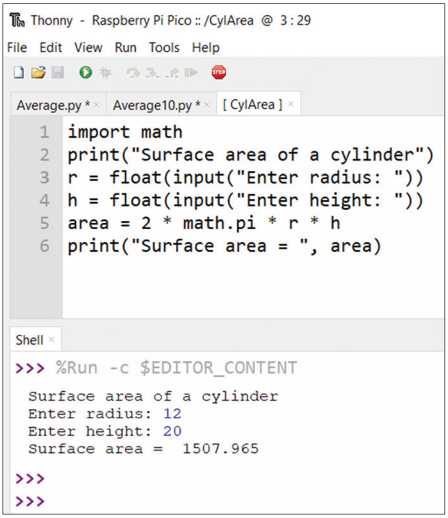

Figure 2.11: Program: ***CylArea*** *and sample run.*

Example 4: ºC to ºF conversion

In this example, the program reads Degrees Celsius from the keyboard and converts and displays the equivalent Degrees Fahrenheit.

Solution 4

The program is named **CtoF,** and the program listing and an example run of the program are shown in Figure 2.12. The formula to convert ºC to ºF is:

$$F = 1.8 \times C + 32$$

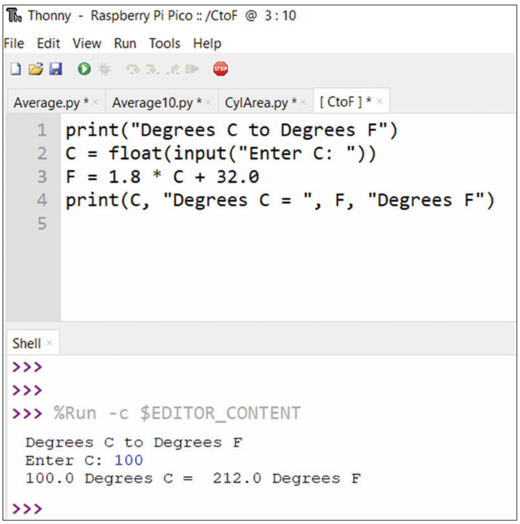

Figure 2.12: Program: ***CtoF*** *and sample run.*

Example 5: Surface area and volume of a cylinder – user function

In this example, the surface area and volume of a cylinder are calculated whose radius and height are given. The program uses a function to calculate and return the surface area and the volume.

Solution 5

The program is named **CylAreaSurf,** and the program listing and an example run of the program are shown in Figure 2.13. The surface area and the volume of a cylinder are given by:

Surface area = $2\pi rh$
Volume = $\pi r^2 h$

Where **r** and **h** are the radius and height of the cylinder respectively. Function **Calc** is used to get the radius and height of the cylinder. The function returns the surface area and volume to the main program which are displayed on the screen.

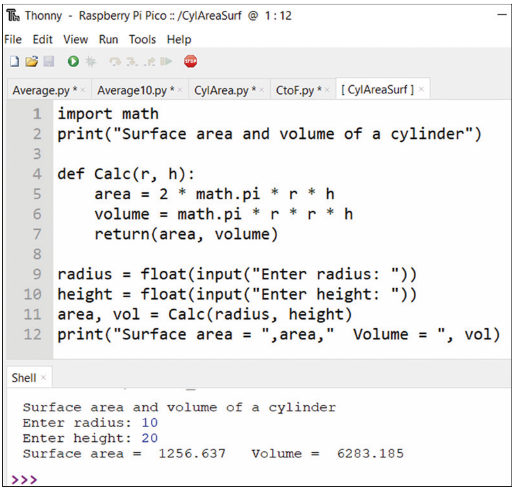

Figure 2.13: Program: ***CylAreaSurf*** *and sample run.*

Example 6: Table of squares of numbers

In this example, the squares of numbers from 1 to 11 are calculated and tabulated.

Solution 6

The program is named **Squares** and the program listing and an example run of the program are shown in Figure 2.14. Notice that **\t** prints a tab so that the data can be tabulated nicely.

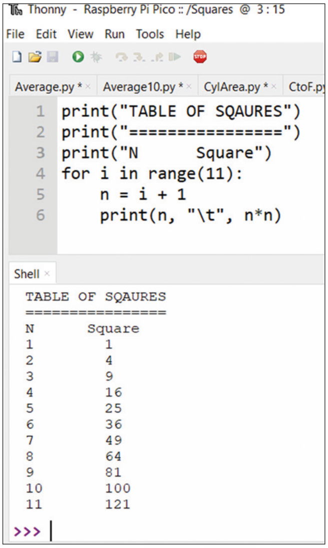

Figure 2.14: Program: ***Squares*** *and sample run.*

Example 7: Table of trigonometric sine

In this example, the trigonometric sine is tabulated from 0 to 45 degrees in steps of 5 degrees.

Solution 7

The program is named **Sines** and the program listing and an example run of the program are shown in Figure 2.15. It is important to notice that the arguments of the trigonometric functions must be in radians and not in degrees.

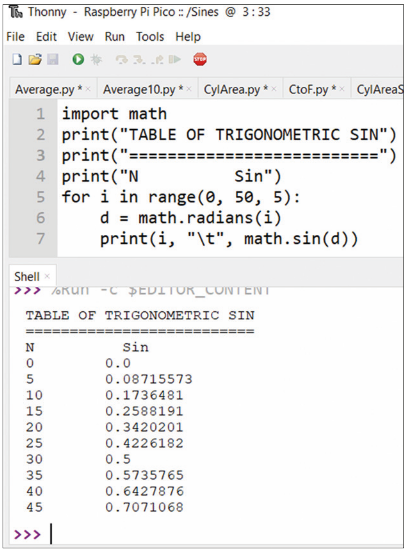

Figure 2.15: Program: ***Sines*** *and sample output.*

Example 8: Table of trigonometric sine, cosine, and tangent

In this example, the trigonometric sine, cosine, and tangent are tabulated from 0 to 45 degrees in steps of 5 degrees.

Solution 8

The program is named **Trig** and the program listing and an example run of the program are shown in Figure 2.16.

Trig.py

```
import math
print("          TABLE OF TRIGONOMETRIC FUNCTIONS")
print("          ================================")
print("N\t   Sin\t\t  Cos\t\t   Tan")
for i in range(0, 50, 5):
    d = math.radians(i)
    s = math.sin(d)
    c = math.cos(d)
    t = math.tan(d)
    print(i, "\t%9.7f\t%9.7f\t%9.7f" %(s,c,t))
```

Shell

```
          TABLE OF TRIGONOMETRIC FUNCTIONS
          ================================
N          Sin            Cos              Tan
0        0.0000000      1.0000000      0.0000000
5        0.0871557      0.9961946      0.0874887
10       0.1736481      0.9848078      0.1763270
15       0.2588191      0.9659258      0.2679492
20       0.3420201      0.9396926      0.3639702
25       0.4226182      0.9063078      0.4663076
30       0.5000000      0.8660254      0.5773502
35       0.5735765      0.8191521      0.7002075
40       0.6427876      0.7660444      0.8390996
45       0.7071068      0.7071068      1.0000000
```

Figure 2.16: Program: ***Trig*** *and sample output.*

Example 9: Trigonometric function of a required angle

In this example, an angle is read from the keyboard. Also, the user specifies whether the sine (s), cosine (c), or the tangent (t) of the angle is required.

Solution 9

The program is named **TrigUser,** and the program listing and an example run of the program are shown in Figure 2.17.

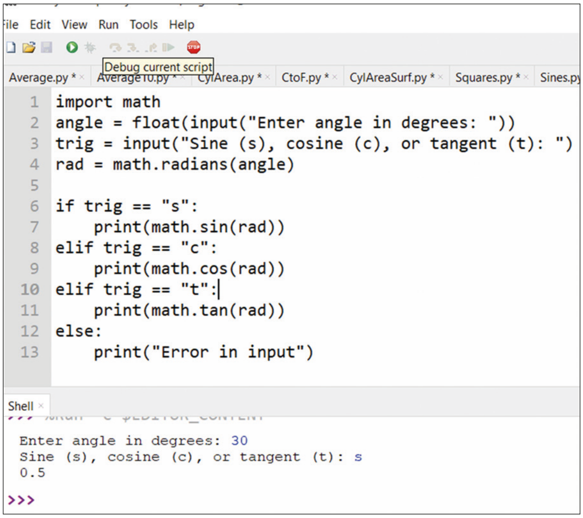

Figure 2.17: Program: ***TrigUser*** *and sample output.*

Example 10: Series and parallel resistors

This program calculates the total resistance of several series or parallel connected resistors. The user specifies whether the connection is in series or in parallel. Additionally, the number of resistors used is also specified at the beginning of the program.

Solution 10

When several resistors are in series then the resultant resistance is the sum of the resistance of each resistor. When the resistors are in parallel then the reciprocal of the resultant resistance is equal to the sum of the reciprocal resistances of each resistor.

Figure 2.18 shows the program listing (program: **Serpal**). At the beginning of the program a heading is displayed, and the program enters a **while** loop. Inside his loop the user is prompted to enter the number of resistors in the circuit and whether they are connected in series or in parallel. Function **str** converts a number into its equivalent string. e.g., number 5 is converted into string "5". If the connection is serial (mode equals to **'s'**) then the value of each resistor is accepted from the keyboard and the resultant is calculated and displayed on the screen. If on the other hand, the connection is parallel (mode equals to **'p'**) then again, the value of each resistor is accepted from the keyboard and the reciprocal of the number is added to the total. When all the resistor values are entered, the resultant resistance is displayed on the screen.

```
print("RESISTORS IN SERIES OR PARALLEL")
print("===============================")
yn = "y"

while yn == 'y':
  N = int(input("\nHow many resistors are there?: "))
  mode = input("Are the resistors series (s) or parallel (p)?: ")
```

```
  mode = mode.lower()
#
# Read the resistor values and calculate the total
#
  resistor = 0.0

  if mode == 's':
    for n in range(0,N):
       s = "Enter resistor " + str(n+1) + " value in Ohms: "
       r = int(input(s))
       resistor = resistor + r
    print("Total resistance = %d Ohms" %(resistor))

  elif mode == 'p':
    for n in range(0,N):
      s = "Enter resistor " + str(n+1) + " value in Ohms: "
      r = float(input(s))
      resistor = resistor + 1 / r
    print("Total resistance = %.2f Ohms" %(1 / resistor))
#
# Check if the user wants to exit
#
  yn = input("\nDo you want to continue?: ")
  yn = yn.lower()
```

Figure 2.18: Program: ***Serpal****.*

Figure 2.19 shows a typical run of the program.

```
RESISTORS IN SERIES OR PARALLEL
===============================

How many resistors are there?: 3
Are the resistors series (s) or parallel (p)?: s
Enter resistor 1 value in Ohms: 100
Enter resistor 2 value in Ohms: 150
Enter resistor 3 value in Ohms: 200
Total resistance = 450 Ohms

Do you want to continue?: y

How many resistors are there?: 2
Are the resistors series (s) or parallel (p)?: p
Enter resistor 1 value in Ohms: 100
Enter resistor 2 value in Ohms: 100
Total resistance = 50.00 Ohms

Do you want to continue?: n
```

Figure 2.19: Typical run of the program.

Example 11: Words in reverse order

Write a program to read a word from the keyboard and then display the letters of this word in reverse order on the screen.