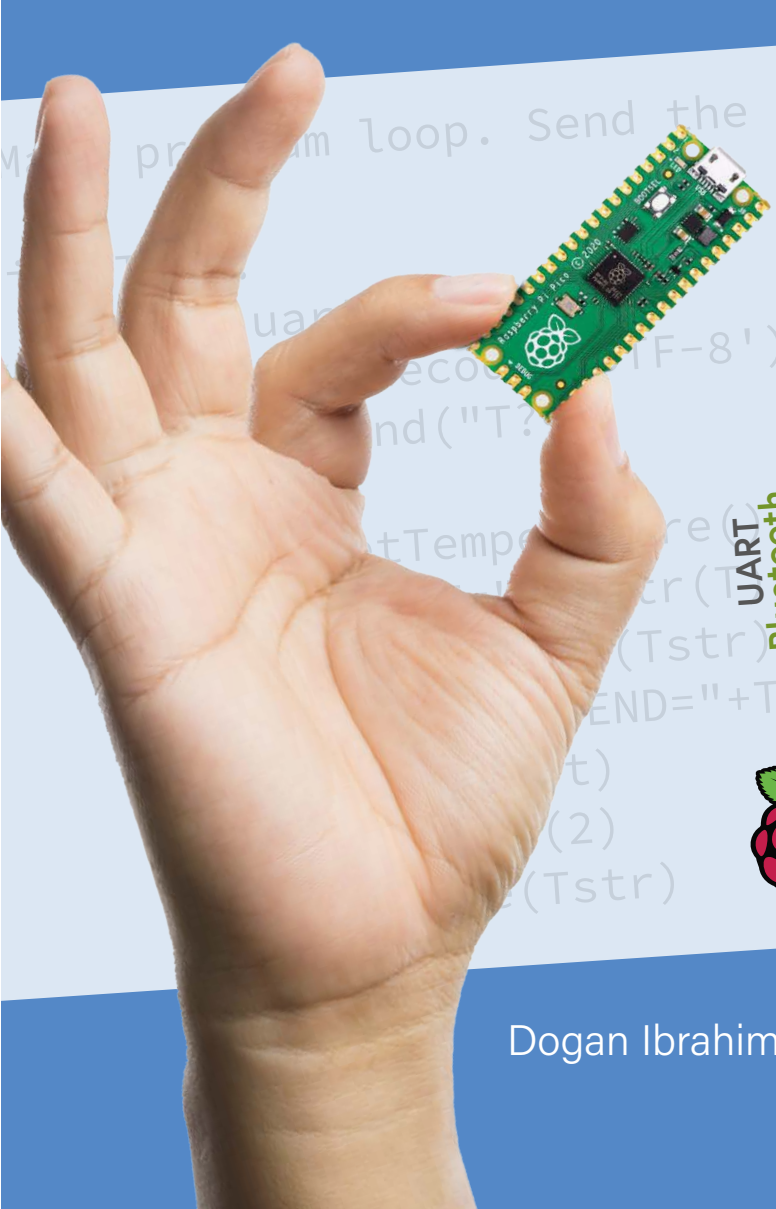
elektor books

Raspberry Pi Pico Essentials

Program, build, and master over 50 projects with MicroPython and the RP2040 microprocessor



Dogan Ibrahim

elektor
design > share > sell

Raspberry Pi Pico Essentials

Program, build, and master over 50 projects with
MicroPython and the RP2040 microprocessor

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

● **Declaration**

The Author and the Publisher have used their best efforts in ensuring the correctness of the information contained in this book. They do not assume, and hereby disclaim, any liability to any party for any loss or damage caused by errors or omissions in this book, whether such errors or omissions result from negligence, accident or any other cause.

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

● **ISBN 978-3-89576-427-1** Print
ISBN 978-3-89576-428-8 eBook
ISBN 978-3-89576-429-5 ePub

Prepress Production: D-Vision, Julian van den Berg

Preface

Traditionally, a computer was built using a microprocessor chip and many external support chips. A microprocessor includes a Central Processing Unit (CPU), an Arithmetic and Logic Unit (ALU), and timing and control circuitry — and as such it is not particularly useful on its own. A microprocessor must be supported by many external chips such as memory, input/output, timers, interrupt circuits etcetera, before it becomes a useful computer. The disadvantage of this type of design was that the chip count was large, resulting in complex design and wiring, and high power consumption.

A microcontroller on the other hand 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 considerably more power. Most of the early microcontrollers were 8-bit processors with clock speeds in the region of several MHz and offered only hundreds of bytes of program and data memories. These microcontrollers were traditionally programmed using the assembly languages of the target processors. 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 GHz and offering several GB of memory space. Microcontrollers are nowadays programmed using a high-level language 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, plus fast and accurate timing modules.

Perhaps the biggest advantage of the Raspberry Pi Pico compared to other many microcontrollers in the marketplace is its very low cost, large memory, and fast and accurate timing modules. At the time of writing this book the cost of a single unit was around $6.

Raspberry Pi Pico 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 datasheets available on the Internet covering the use of the Raspberry Pi Pico.

This book is an introduction to using the Raspberry Pi Pico microcontroller with the Micro-Python programming language. The Thonny development environment (IDE) is used in all the projects in the book, and readers are recommended to use this IDE. There are over 50 working and tested projects in the book, covering almost all aspects of the Raspberry Pi Pico.

The following sub-headings are given for each project to make it easy to follow:

- 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, and this book becomes useful in the development of your projects.

Dr Dogan Ibrahim
London, February, 2021

Chapter 1 • Raspberry Pi Pico Hardware

1.1 Overview

The Raspberry Pi Pico is a single-board microcontroller module developed by the Raspberry Pi Foundation. This module is based on the RP2040 microcontroller chip. In this Chapter we will be looking at the hardware details of the Raspberry Pi Pico microcontroller module in some detail. From here on, we will be calling this microcontroller module "Pico" for short, in for appreciation and recognition though of its official name: Raspberry Pi Pico.

1.2 Pico hardware module

The "Pico" is a very low-cost, $4 microcontroller module based on the RP2040 microcontroller chip having a dual Cortex-M0+ processor. Figure 1.1 shows the front view of the Pico hardware module which is basically a small board. At the centre 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. Soldering pins to these holes enables external connections to be easily made 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 supplying power to the board as well as for programming it. Next to the USB port there is an on-board user LED that can be used during program development. Next to this LED sits a button named as BOOTSEL that is used during programming of the microcontroller as we will see in next Chapters. At the other edge of the board, next to the Raspberry Pi logo, there are 3 connectors that can be used for debugging your programs.
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 (analogue ground)
3V3	-	+3.3 V power supply (output)
GP0 – GP22	-	digital GPIO
GP26_A0 – GP28_A2	-	analogue 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 +3.3V 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)

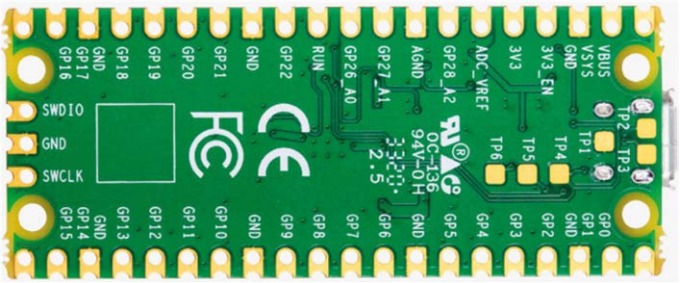


Figure 1.2: Back view of the Pico hardware module.

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

GP29 (input)	-	used in ADC mode (ADC3) to measure VSYS/3
GP25 (output)	-	connected to on-board user LED
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 (+3.3V compatible)
- 3× 12-bit ADC pins
- Serial Wire Debug (SWD) port
- Micro-USB port (USB 1.1) for power (+5V) and data (programming)
- 2× UART, 2 x I^2C, 2 x SPI bus interface
- 16× PWM channels
- 1× Timer (with 4 alarms), 1× Real-Time Counter
- On-board temperature sensor

- On-board LED (on port GP25)
- MicroPython, C, C++ programming
- Drag & drop programming using mass storage over USB

The Pico's 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.

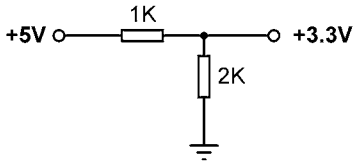


Figure 1.3: resistive potential divider circuit.

1.3 Comparison with the Arduino UNO

The Arduino UNO is one of the most popular microcontroller development boards used by students, practicing engineers, and hobbyists. Table 1.1 shows a comparison of the Raspberry Pi Pico with the Arduino UNO. It is clear from this table that the Pico is much faster than the Arduino UNO, having larger flash and data memories, providing more digital input/output pins, and sporting an on-board temperature sensor. The Arduino UNO operates at +5 V and its GPIO pins are +5 V compatible. Perhaps some advantages of the Arduino UNO include having a built-in EEPROM memory, and having a 6-channel ADC rather than a 3-channel.

Feature	Raspberry Pi Pico	Arduino UNO
Microcontroller	RP2040	Atmega328P
Core and bits	Dual core, 32-bit, Cortex-M0+	Single-core 8-bit
RAM	264 Kbytes	2 Kbytes
Flash	2 Mbytes	32 Kbytes
CPU speed	48 MHz to 133 MHz	16 MHz
EEPROM	None	1 KByte
Power input	+5 V through USB port	+5V through USB port
Alternative power	2–5 V via VSYS pin	7–12 V
MCU operating voltage	+3.3 V	+5 V
GPIO count	26	20
ADC count	3	6
Hardware UART	2	1
Hardware I2C	2	1
Hardware SPI	2	1
Hardware PWM	16	6
Programming languages	MicroPython, C, C++	C (Arduino IDE)
On-board LED	1	1
Cost	$4	$20

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

1.4 Operating conditions and powering the Pico

The recommended operating conditions for the Pico are:

- Operating temperature: –20ºC to +85ºC
- 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 size-AA batteries can be used to provide +4.5 V to power the Pico.

The Pico can be powered in several ways. The simplest method is to plug the micro-USB port into 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 analogue 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 external power supply). We 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 of the voltages will power VSYS.

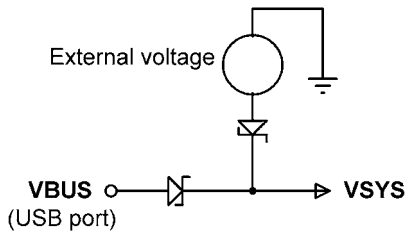


Figure 1.4: Powering the Pico.

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 illustration, most pins have multiple functions. For example, GPIO0 (pin 1) doubles as the UART0 TX, I2C0 SDA, and the SPI0 RX pin.

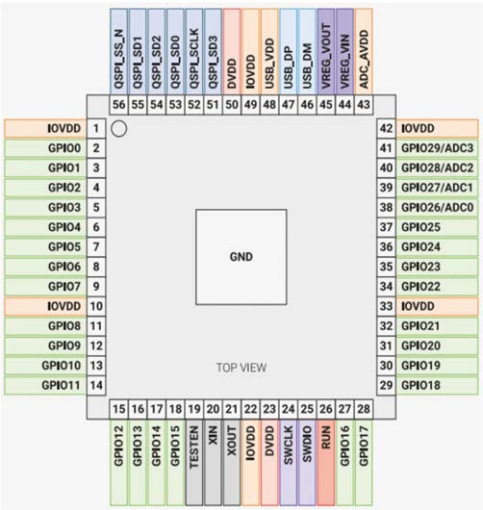


Figure 1.5: RP2040 microcontroller pinout.

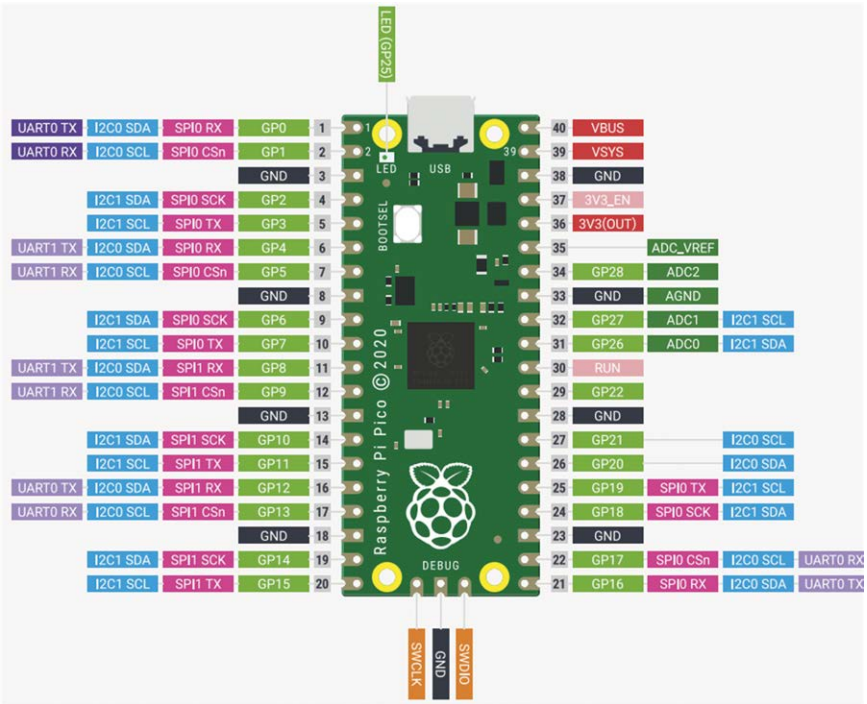


Figure 1.6: Raspberry Pi Pico pinout.

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 nos. 26-28 can be used either as digital GPIO or as ADC inputs. ADC inputs GPIO26-29 have reverse-biased diodes to 3 V 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).

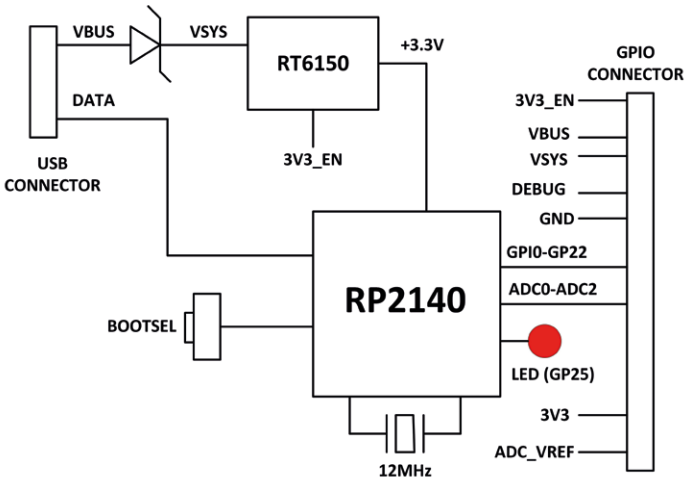


Figure 1.7: Simplified block diagram.

1.6 Other RP2040 microcontroller-based boards

During the writing of this book, some third-party manufacturers have been developing microcontrollers based on the RP2040 chip. Some examples are given in this section.

1.6.1 Adafruit Feather RP2040

This microcontroller board (Figure 1.8) has the following basic specifications:

- RP2040 32-bit Cortex-M0+ running at 125 MHz
- 4 MB Flash memory
- 264 KB RAM
- 4× 12-bit ADC
- 2× I^2C, 2× SPI, 2× UART
- 16× PWM
- 200 mA LiPo charger
- Reset and Bootloader buttons
- 24 MHz crystal
- +3.3 V regulator with 500 mA current output
- USB type-C connector
- on-board red LED
- RGB NeoPixel
- on-board STEMMA QT connector with optional SWD debug port

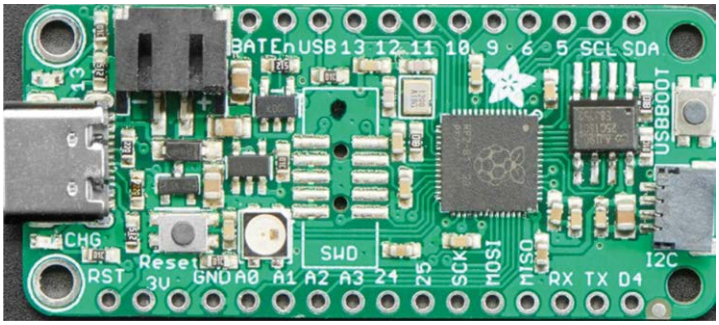


Figure 1.8: Adafruit Feather Rp2040.

1.6.2 Adafruit ItsyBitsy RP2040

The ItsyBitsy RP2040 (Figure 1.9) is another RP2040-based microcontroller board from Adafruit. Its basic features are very similar to Feather RP2040. It has USB-micro B connector and provides +5 V output.

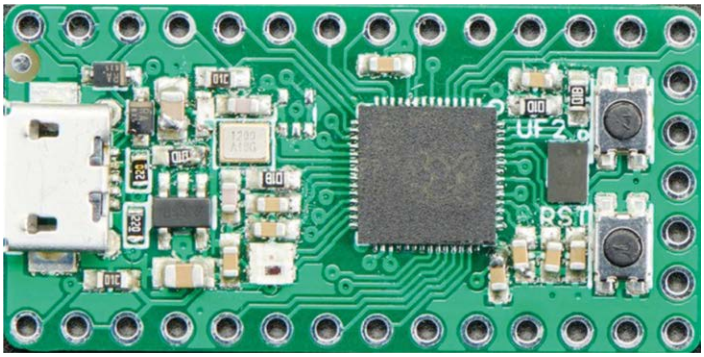


Figure 1.9: Adafruit ItsyBitsy RP2040.

1.6.3 Pimoroni PicoSystem

This is a miniature gaming board (Figure 1.10) developed around the RP2040 microcontroller. Its basic features are:

- 133 MHz clock
- 264 KB SRAM
- LCD screen
- joypad
- buttons
- LiPo battery
- USB-C power connector

Figure 1.10: Pimoroni PicoSystem.

1.6.4 Arduino Nano RP2040 Connect

This board (Figure 1.11) offers 16 MB flash, 9-axis IMU, and a microphone. It has a very efficient power supply section equipped with Wi-Fi/Bluetooth.

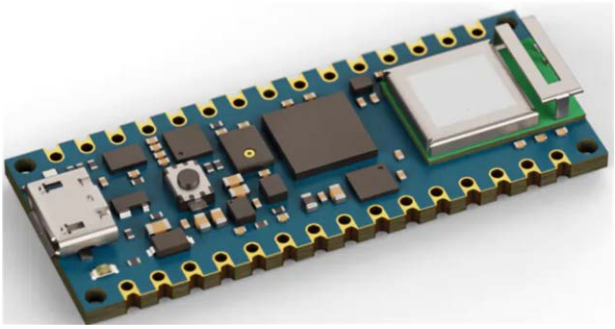


Figure 1.11: Arduino Nano RP2040 Connect.

1.6.5 SparkFun Thing Plus RP2040

This development platform (Figure 1.12) provides an SD card slot, 16 MB flash memory, a JST single-cell battery connector, a WS2812 RGB LED, JTAG pins, and Qwiic connector. Its basic features are:

- 133 MHz speed
- 264 KB SRAM
- 4× 12-bit ADC
- 2× UART, 2× I^2C, 2× SPI
- 16× PWM
- 1× timer with 4 alarms

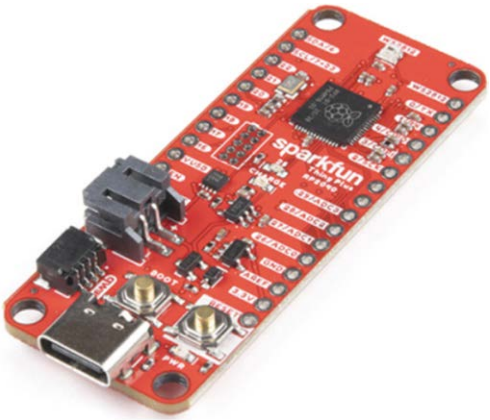


Figure 1.12: SparkFun Thing Plus RP2040.

1.6.6 Pimoroni Pico Explorer Base

This development board (Figure 1.13) includes a small breadboard and a 240 × 240 IPS LC display with 4 tactile buttons. A socket is provided on the board to plug-in a Raspberry Pi Pico board. The basic features of this development board are:

- piezo speaker
- 1.54-inch IPS LCD
- 4× buttons
- 2× half-bridge motor drives
- two breakout I²C sockets
- easy access to GPIO and ADC pins
- mini breadboard
- no soldering required
- Raspberry Pi Pico not supplied

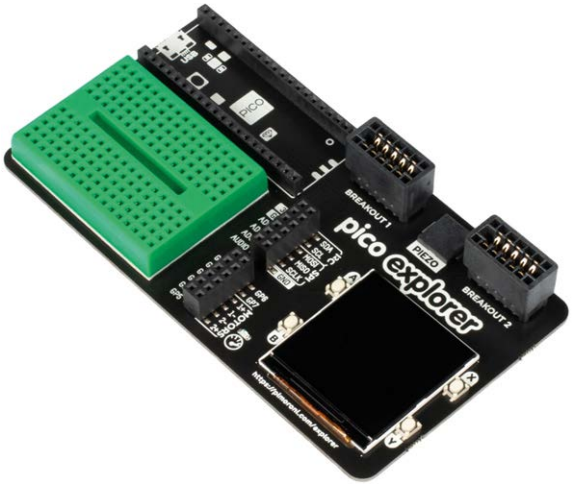


Figure 1.13: Pimoroni Pico Explorer Base.

1.6.7 SparkFun MicroMod RP2040 Processor

This board (Figure 1.14) includes a MicroMod M.2 connector for access to the GPIO pins.

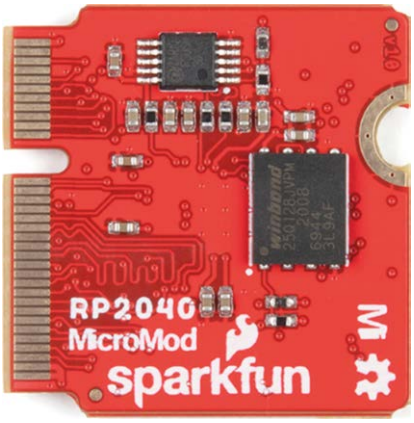


Figure 1.14: SparkFun MicroMod RP 2040 Processor.

1.6.8 SparkFun Pro Micro RP2040

This board (Figure 1.15) includes a ES2812B addressable LED, boot button, reset button, Qwiic connector, USB-C power interface, PTC fuse, and castellated GPIO pads.

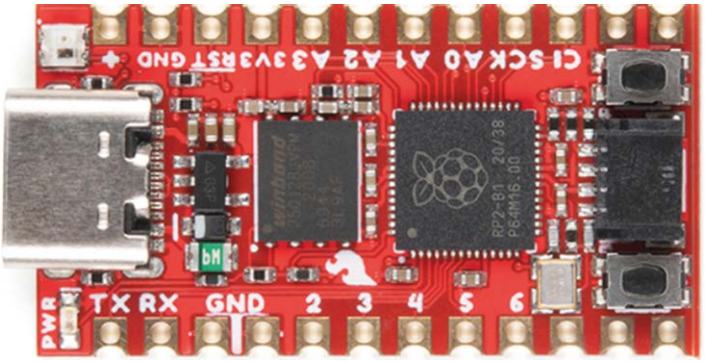


Figure 1.15: SparkFun Pro Micro RP2040.

1.6.9 Pico RGB Keypad Base

This board is equipped with 4×4 rainbow-illuminated keypad (Figure 1.16) with APA102 LEDs. The basic features are:

- 4×4 keypad
- 16× APA102 RGB LEDs
- keypad connected via I^2C I/O expander
- GPIO pins labelled

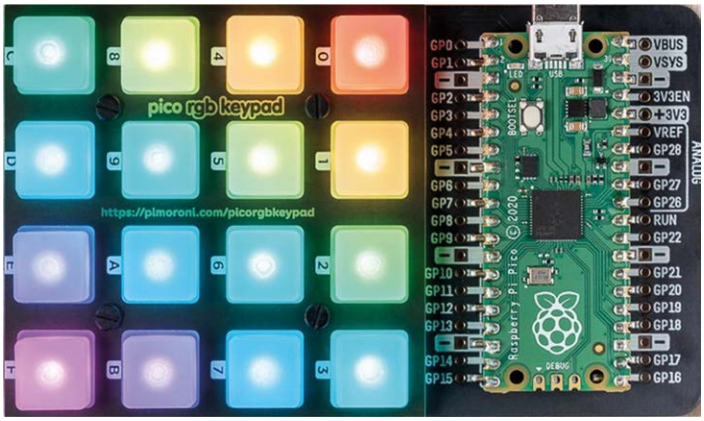


Figure 1.16 Pico RGB Keypad Base.

1.6.10 Pico Omnibus

This is an expansion board (Figure 1.17) for the Pico. The basic features of this board include:

- GPIO pins labelled
- two landing areas with labelled (mirrored) male headers for attaching add-ons
- 4× rubber feet
- compatible with Raspberry Pi Pico
- fully assembled
- dimensions approx. 94 × 52 × 12 mm

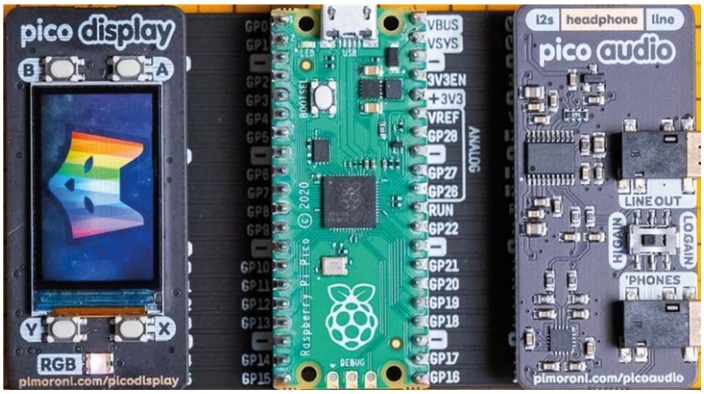


Figure 1.17: Pico Omnibus.

1.6.11 Pimoroni Pico VGA Demo Base

This board (Figure 1.18) has VGA output and SD card slot. The basic features are:

- powered by Raspberry Pi Pico
- 15-pin VGA connector
- I^2S DAC for line out audio

- PWM audio output
- SD card slot
- Reset button
- headers to install your Raspberry Pi Pico
- three user switches
- no soldering required

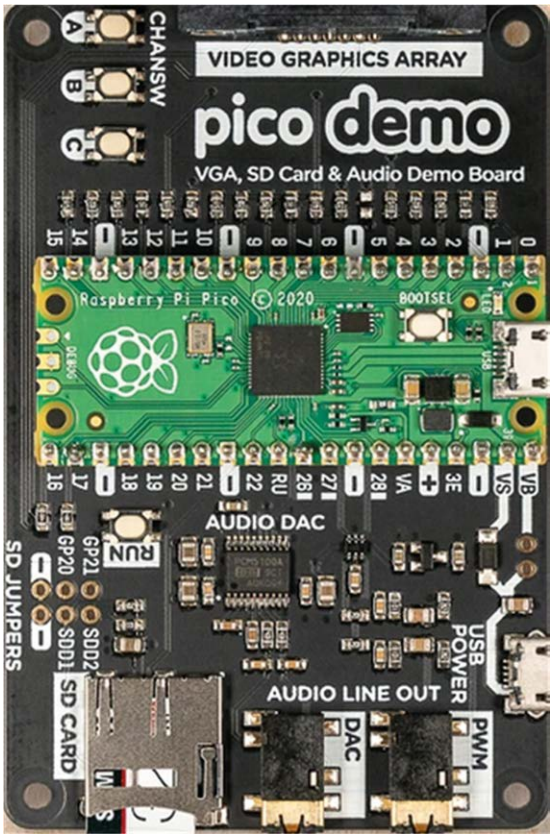


Figure 1.18: Pimoroni Pico VGA Demo Base.

Chapter 2 • Raspberry Pi Pico Programming

2.1 Overview

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

- C/C++
- MicroPython
- assembly language

Although the 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 we will learn how to install and use the MicroPython programming language. We 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 our Pico. Use of the C language will also be discussed in later Chapters with some projects.

2.2 Installing MicroPython on the Pico

MicroPython must be installed on the Pico before the board can be used. Once installed, MicroPython stays on your Pico, unless it is overwritten with something else. Installing MicroPython requires an Internet connection, and this is required only once. Since the Pico has no Wi-Fi connectivity, we will need to use a computer with Internet access. This can be done either by using a Raspberry Pi (e.g. Raspberry Pi 4), or by using a PC. In this section we will see how to install using both methods.

2.2.1 Using a Raspberry Pi 4 to aid installing MicroPython on the Pico

The steps are as follows.

- Boot your Raspberry Pi 4 and log in to Desktop.
- Make sure your Raspberry Pi is connected to the Internet.
- Hold down the **BOOTSEL** button on your Pico.
- Connect your Pico to one of the USB ports of the Raspberry Pi 4 using a micro-USB cable while holding down the button.
- Wait a few seconds and release the **BOOTSEL** button.
- You should see the Pico appear as a removable drive. Click **OK** in the **Removable medium is inserted** window (Figure 2.1).

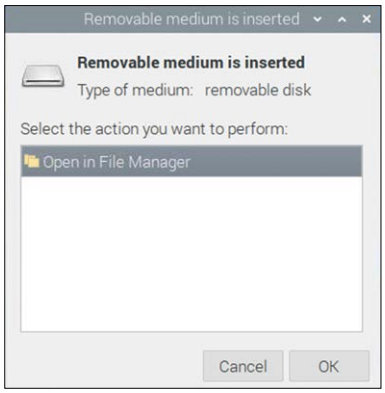


Figure 2.1: Click OK.

- In the **File Manager** window, you will see two files with the names **INDEX.HTM** and **INFO_UF2.TXT** (Figure 2.2).

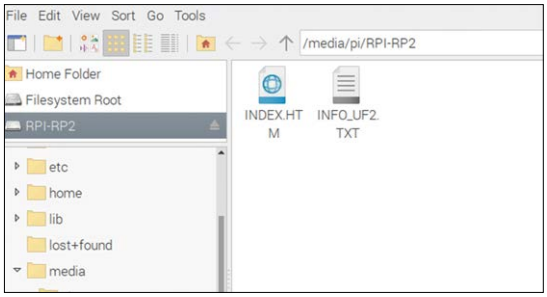


Figure 2.2: Notice two files.

- Double-click on file **INDEX.HTM** and scroll down.
- You should see the message **Welcome to your Raspberry Pi Pico** displayed in a web page (Figure 2.3).

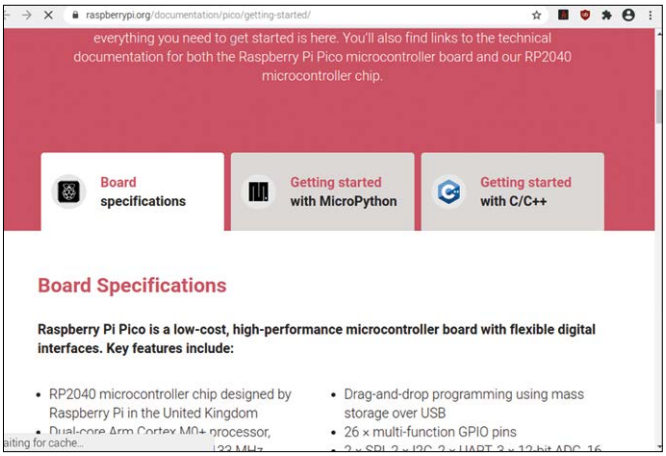


Figure 2.3: Displayed message.

- Click on the **Getting started with MicroPython** tab and click **Download UF2 file** to download the **MicroPython** firmware. You should see the downloaded file at the bottom of the screen. This will take only a few seconds (Figure 2.4).

Figure 2.4: Download the UF2 file.

- Close your browser window by clicking on the cross icon located at the top right corner.
- Open the **File Manager** by clicking on menu, followed by **Accessories**.
- Open the **Downloads** folder (under **/home/pi**) and locate the file with the extension: **.uf2**. This file will have a name similar to: **micropython-20-Jan-2021.uf2** (Figure 2.5)

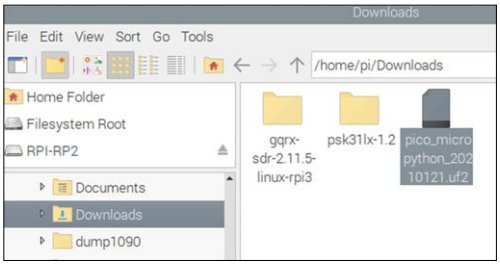


Figure 2.5: Locate the file with extension: ".uf2".

- Drag and drop this file to Raspberry Pi Pico's removable drive which is named: **RPI-RP2** (at the top left side of the screen – see Figure 2.5).
- After a while, the **MicroPython** firmware will be installed onto the internal storage of Pico and the drive will disappear.
- Your Pico is now running **MicroPython**.
- Powering-down the Pico will not erase MicroPython from its memory.

Using the Thonny text editor from Raspberry Pi

Thonny is a free Python Integrated Development Environment (IDE) developed specifically for Python. It has built-in text editor and debugger and a number of other utilities that can be useful during program development.

In this section we will learn how to use Thonny by invoking it from Raspberry Pi. You should leave your Pico connected to Raspberry Pi. We will create a one-line program to display the message **Hello from Raspberry Pi Pico**.

The steps are given below.

- Click Menu, followed by **Programming** on your Raspberry Pi Desktop and then click **Thonny Python IDE** (see Figure 2.6). The author had version 3.3.3 of Thonny installed on his Raspberry Pi 4.

Figure 2.6: Start Thonny on your Raspberry Pi.

- Click on the label **Python** at the bottom right-hand corner of Thonny (Figure 2.7).

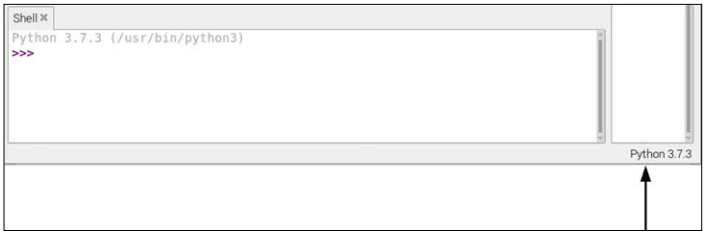


Figure 2.7: Click on Python in the bottom right-hand corner.

- Click to select **MicroPython (Raspberry Pi Pico)** as shown in Figure 2.8.

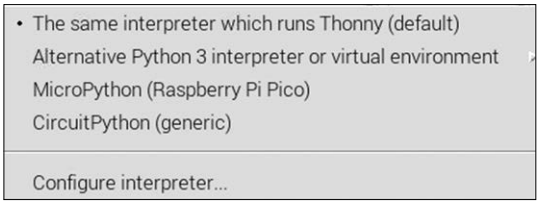


Figure 2.8: Select: Raspberry Pi Pico.

- You should see the version number of your MicroPython displayed in the bottom part of the screen where Shell is located (Figure 2.9).