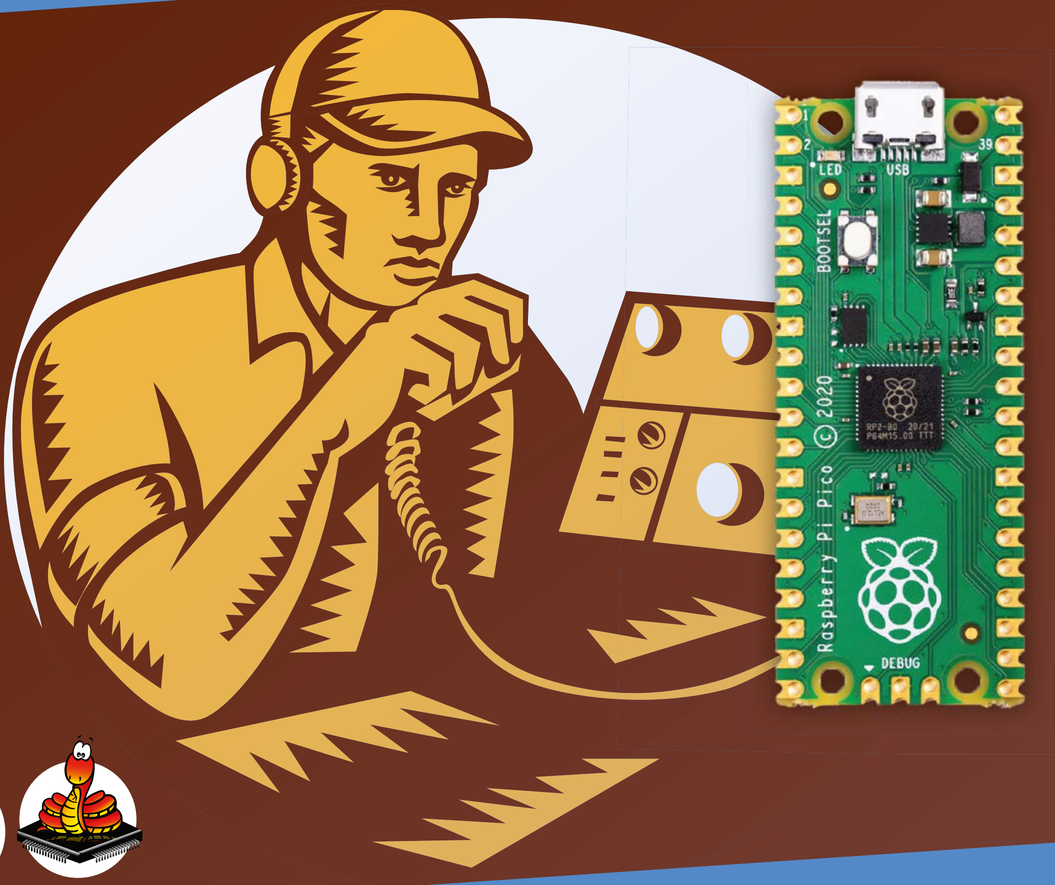
elektor books

Raspberry Pi Pico for Radio Amateurs

Program and build RPi Pico-based ham station utilities, tools, and instruments



Dogan Ibrahim, G7SCU

elektor
design > share > sell

Raspberry Pi Pico for Radio Amateurs

Program and build RPi Pico-based hams station utilities, tools, and instruments

Dogan Ibrahim, G7SCU

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 978-3-89576-481-3** Print
ISBN 978-3-89576-482-0 eBook

● © Copyright 2021: Elektor International Media B.V.
Editor: Jan Buiting
Prepress Production: D-Vision, Julian van den Berg

Preface

In recent years, there have been major changes in the equipment typically used by radio amateurs. Although much classical HF and mobile equipment is still in use by a large number of amateurs, we see the use of computers and digital techniques gaining popularity among amateur radio operators or 'hams'. In the early days of digital communications, personal computers were used by radio amateurs to communicate with each other. Sadly, these PCs have the disadvantage of being rather expensive and bulky. Today though, anyone can buy a 5-euros Raspberry Pi Pico computer and build many interesting amateur radio projects using this device which is smaller than a credit card.

Several authors have produced books and published projects for implementing the Arduino and the Raspberry Pi in amateur radio projects. The Raspberry Pi Pico is a practical alternative to the Arduino because of its low cost, speed, processing power, large memory, many input-output ports, peripheral hardware support, and easy programming. The Raspberry Pi Pico has no operating system, and this makes it easy to use as a general-purpose microcontroller. As a result of these features, the RPi Pico is well suited for use as a "drop-in" computer for amateur radio projects.

This book has three purposes: firstly, it is aimed to teach the basic operating principles and features of the Raspberry Pi Pico to beginners. Secondly, software-only projects are presented that will be of interest to amateur radio operators. Lastly, many hardware-based projects are given using the Raspberry Pi Pico in conjunction with the Python 3 programming language. Although these projects are broad-spectrum in nature, they have been chosen to be interesting and useful to the amateur radio operators.

All the projects used in the book have been assessed and are fully working. The projects are described by giving their block diagrams, circuit diagrams, and full program listings. The program listings are described in detail and readers should find it easy to modify the projects for their own requirements.

The programs discussed in this book are available from the support and resources web page created for the book at the Elektor Store website www.elektor.com. There, the page can be found by searching for "Raspberry Pi Pico for Radio Amateurs". The .zip archive file is under "Downloads". The programs can easily be downloaded, extracted, and stored locally to save the time and effort of typing them.

I hope you enjoy reading the book and find the projects interesting and useful.

Prof Dogan Ibrahim, G7SCU
London, 2021

Chapter 1 • Raspberry Pi Pico Hardware

1.1 Overview

Raspberry Pi Pico is a single-board microcontroller module developed by the Raspberry Pi Foundation. The 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 now on, we will be calling this microcontroller module "Pico" for short.

1.2 Pico hardware module

Pico is a very low-cost, $4 microcontroller module based on the RP2040 microcontroller chip with dual Cortex-M0+ processor. Figure 1.1 shows the front view of the Pico hardware module which is a small board. At 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 forty gold-colored 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.

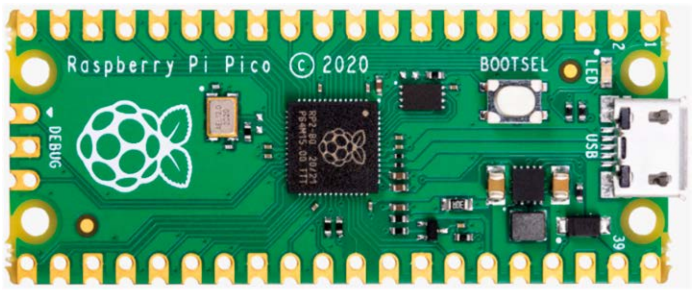


Figure 1.1: Front view of the Pico hardware module.

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 sits an on-board user LED that can be used during program development. Next to this LED there is a button named as BOOTSEL which 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 three connectors that are used to debug 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 (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)

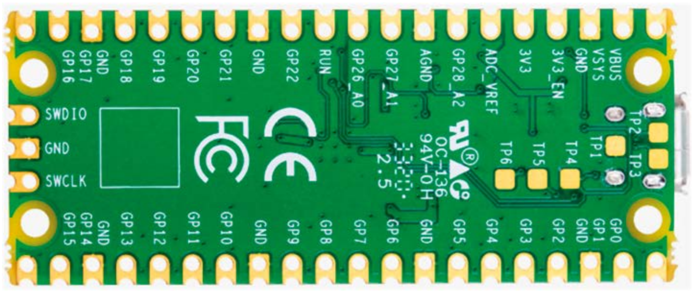


Figure 1.2: Back view of the Pico hardware module.

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

GP29 (input)	—	used in ADC mode (ADC3) to measure VSYS/3
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 (+3.3 V compatible)
- 3 × 12-bit ADC pins
- Serial Wire Debug (SWD) port
- Micro-USB port (USB 1.1) for power (+5 V) and data (programming)
- 2 × UART, 2 x I^{2}C, 2 x SPI bus interface
- 16 × PWM channels

- 1 × Timer (with 4 alarms), 1 x 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 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. A logic level converter module is shown in Figure 1.4. This module can be used to interface the Pico pins to +5 V devices. Connect GND pins to ground, and HV and LV pins to +5 V and +3.3 V, respectively. Use TXI-TXO pins co connect the +3.3 V Pico outputs to +5 V input devices. Similarly, use RXI-RXO pins to connect +5 V output devices to +3.3 V Pico input pins.

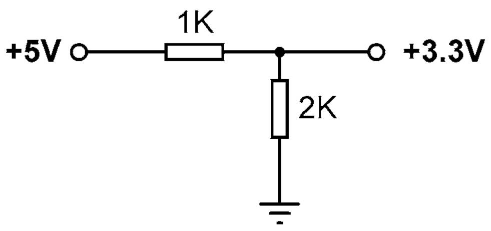


Figure 1.3: Resistive potential divider circuit.

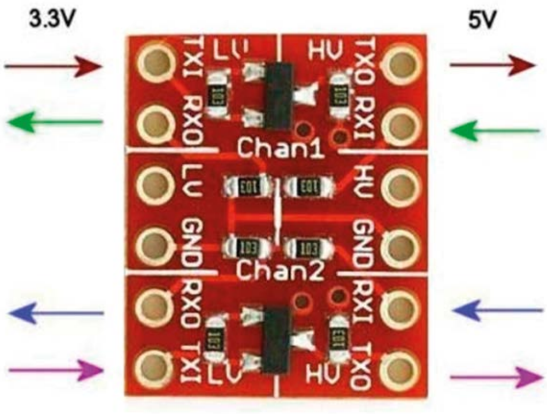


Figure 1.4: Logic converter module.

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. Both the Arduino UNO and Raspberry Pi Pico module are microcontrollers with no operating systems. 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, has larger flash and data memories, provides more digital input-output pins, and has an on-board temperature sensor. The Arduino UNO operates at +5 V and its GPIO pins are +5 V compatible. Some advantages of the Arduino UNO include its built-in EEPROM memory and its ADC with six channels instead of three as in the Pico.

Feature	Raspberry Pi Pico	Arduino UNO
Microcontroller	RP2040	Atmega328P
Core and bits	Dual core, 32-bits, Cortex-M0+	Single-core 8-bits
RAM	264 Kbyte	2 KByte
Flash	2 MByte	32 KByte
CPU speed	48 MHZ to 133 MHz	16 MHz
EEPROM	None	1 KByte
Power input	+5 V through USB port	+5 V 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 I^{2}C	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 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, three 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 1.5) 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 a fixed +3.3 V supply for the MCU and other parts of the board. VSYS is divided by three 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 an 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 allow operation from 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.

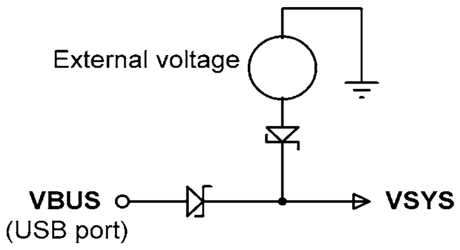


Figure 1.5: Powering the Pico.

1.5 Pinout of the RP2040 microcontroller and Pico module

Figure 1.6 shows the RP2040 microcontroller pinout, which is housed in a 56-pin package. The Pico module pinout is shown in Figure 1.7 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.

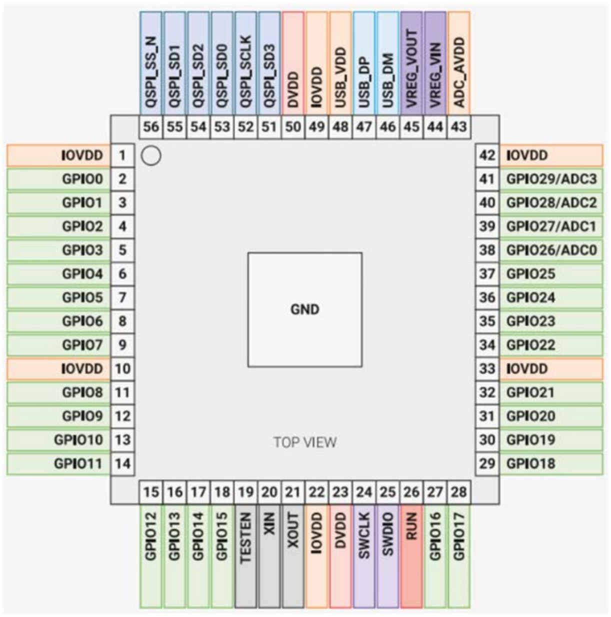


Figure 1.6: RP2040 microcontroller pinout.

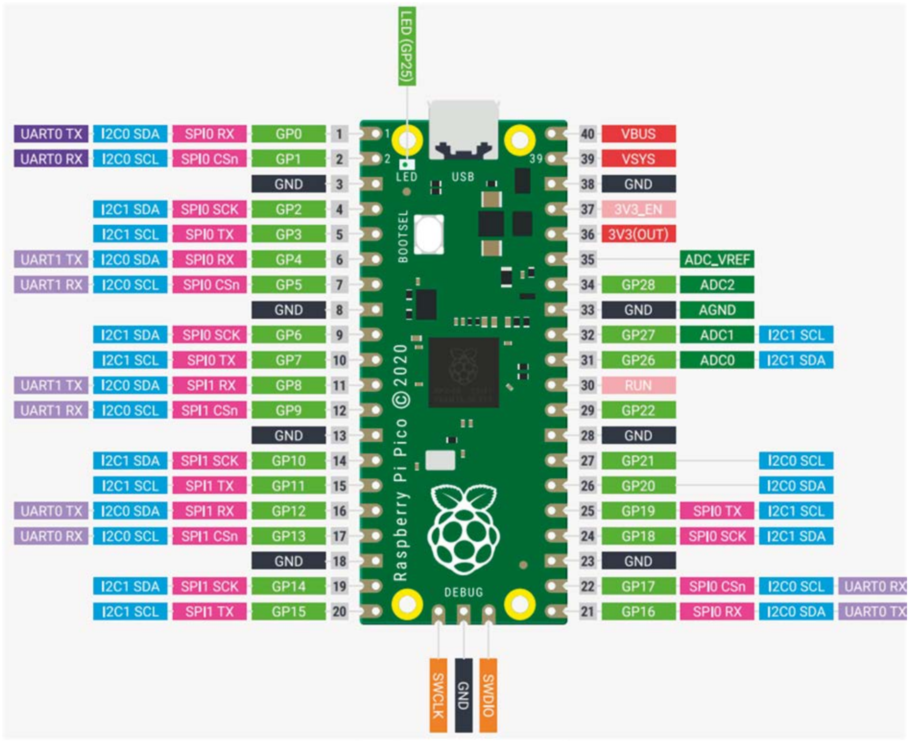


Figure 1.7: Pico pinout.

Figure 1.8 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 numbers 26, 27, 28 can be used either as digital GPIO or as ADC inputs. ADC inputs GPIO26-29 have reverse polarity diodes to 3 Vs 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 safely when the RP2040 is not powered.

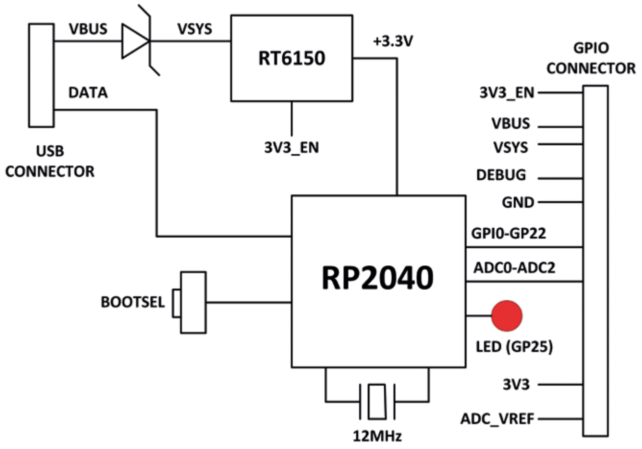


Figure 1.8: Simplified block diagram.

1.6 Other RP2040 microcontroller-based boards

While authoring this book, some third-party manufacturers have been developing micro-controllers based on the RP2040 chip. Some examples are given in this section.

1.6.1 Adafruit Feather RP2040

This microcontroller board (Figure 1.9) has the following basic specifications:

- RP2040 32-bit Cortex-M0+ running at 125MHz
- 4 MB Flash memory
- 264 KB RAM
- 4 × 12-bit ADC
- 2 × I^{2}C, 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

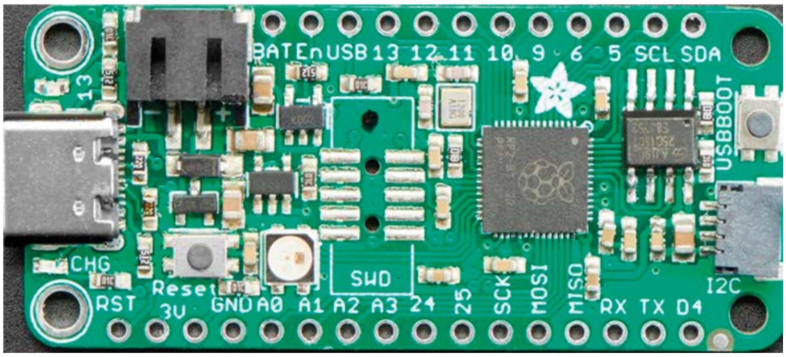


Figure 1.9: Adafruit Feather RP2040.

1.6.2 Adafruit ItsyBitsy RP2040

The ItsyBitsy RP2040 (Figure 1.10) is another RP2040-based microcontroller board from Adafruit. Its basic features are similar to Feather RP2040. It has a USB-micro B connector and provides +5 V output.

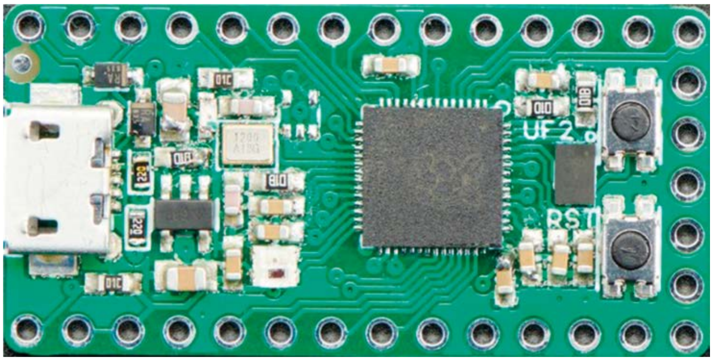


Figure 1.10: Adafruit ItsyBitsy RP2040.

1.6.3 Pimoroni PicoSystem

This is a mini gaming board (Figure 1.11) 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 1.11: Pimoroni PicoSystem.

1.6.4 Arduino Nano RP2040 Connect

This board (Figure 1.12) offers 16 MB flash, a 9-axis IMU, a microphone, plus a very efficient power section equipped with Wi-Fi/Bluetooth. It includes a u-blox NINA-W102 radio module to make the unit IoT compatible. A built-in microphone (MP34DT05) is available for sound activation, audio control, and even AI voice recognition. The 6-axis smart IMU (LSM6DSOXTR) with AI capabilities tells the board which way it is moving and adds fall sensing and double-tap activation. It includes full Wi-Fi 802.11b/g/n connectivity, along with Bluetooth® and BLE v4.2. Supports the Arduino programming language, the IDE 2.0 and all the associated libraries.

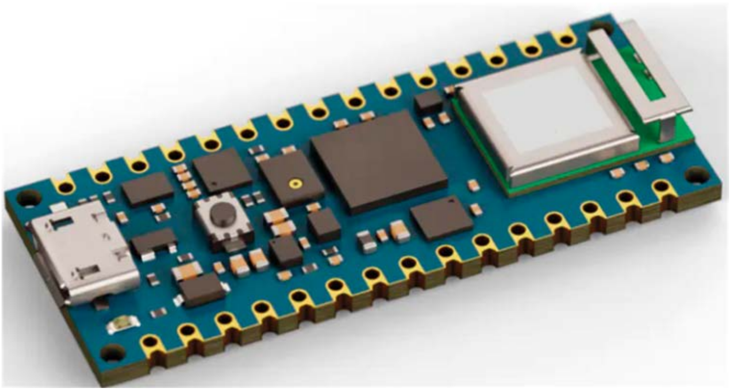


Figure 1.12: Arduino Nano RP2040 Connect.

1.6.5 SparkFun Thing Plus RP2040

This development platform (Figure 1.13) provides an SD card slot, 16MB 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^{2}C, 2 × SPI
- 16 × PWM
- 1 × timer with 4 alarms

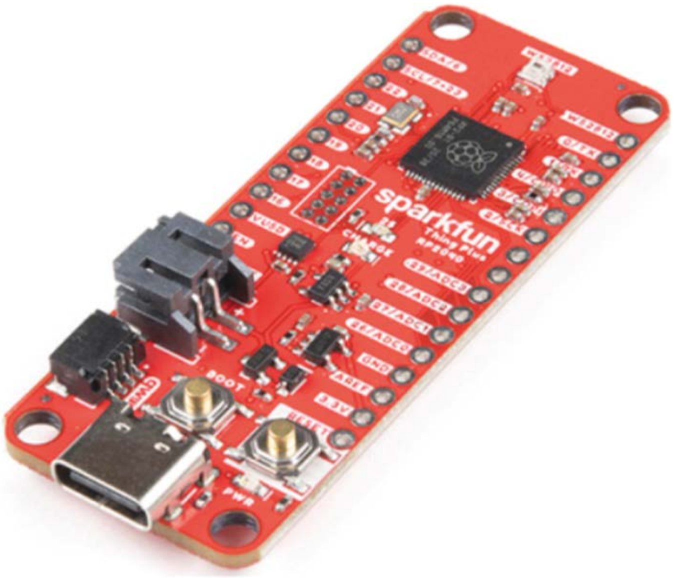


Figure 1.13: SparkFun Thing Plus RP2040.

1.6.6 Pimoroni Pico Explorer Base

This development board (Figure 1.14) includes a small breadboard and a 240 × 240 IPS LCD display with four 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 I^{2}C sockets
- Easy access to GPIO and ADC pins
- Mini breadboard
- No soldering required
- Raspberry Pi Pico not supplied

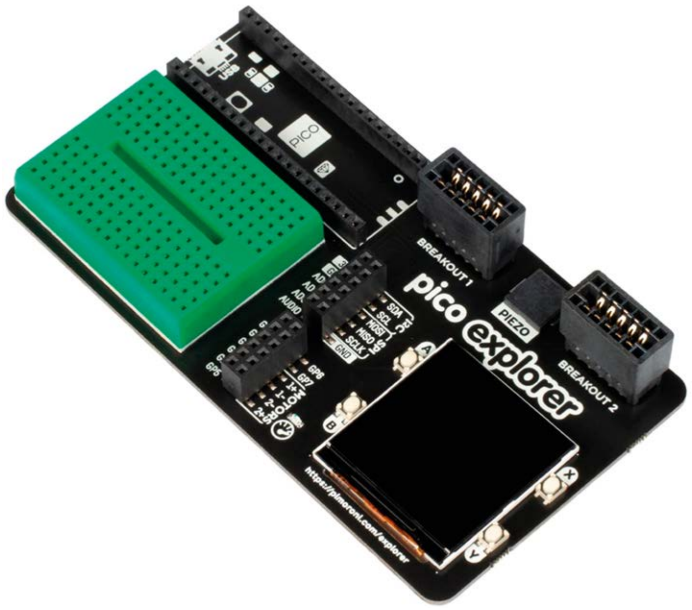


Figure 1.14: Pimoroni Pico Explorer Base.

1.6.7 SparkFun MicroMod RP2040 Processor

This board (Figure 1.15) includes a MicroMod M.2 connector for access to the GPIO pins.

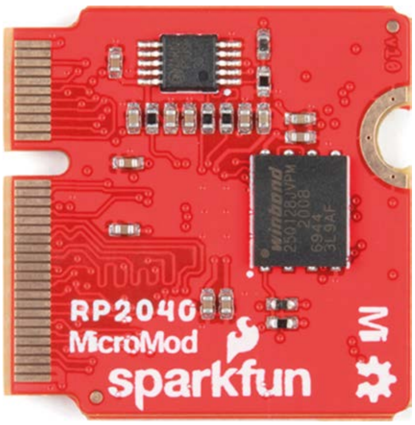


Figure 1.15: SparkFun MicroMod RP 2040 Processor.

1.6.8 SparkFun Pro Micro RP2040

This board (Figure 1.16) includes an ES2812B addressable LED, a boot button, a reset button, a Qwiic connector, a USB-C power interface, a PTC fuse, and castellated GPIO pads.

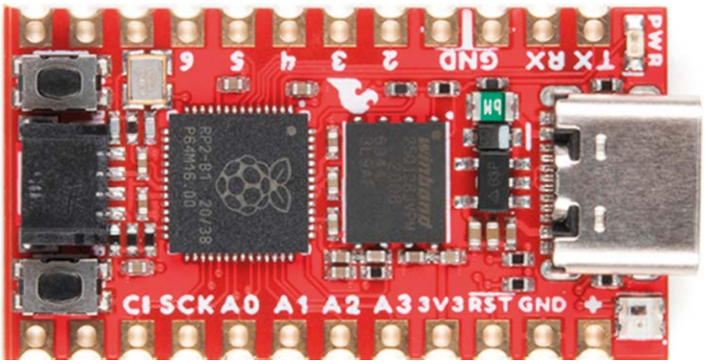


Figure 1.16: SparkFun Pro Micro RP2040.

1.6.9 Pico RGB Keypad Base

This board is equipped with 4 × 4 rainbow-illuminated keypad (Figure 1.17) with APA102 LEDs. The basic features are:

- 4 × 4 keypad
- 16 x APA102 RGB LEDs
- Keypad connected via I²C I/O expander
- labelled GPIO pins

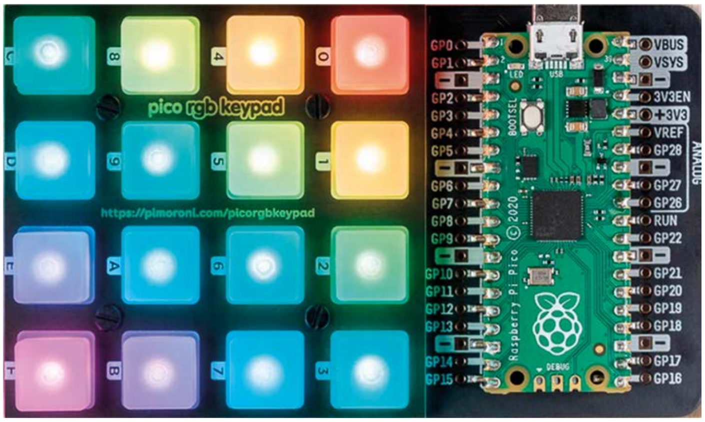


Figure 1.17: Pico RGB Keypad Base.

1.6.10 Pico Omnibus

This is an expansion board for the Pico (Figure 1.18). Basic features of this board are:

- labelled GPIO pins
- 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 mm × 12 mm

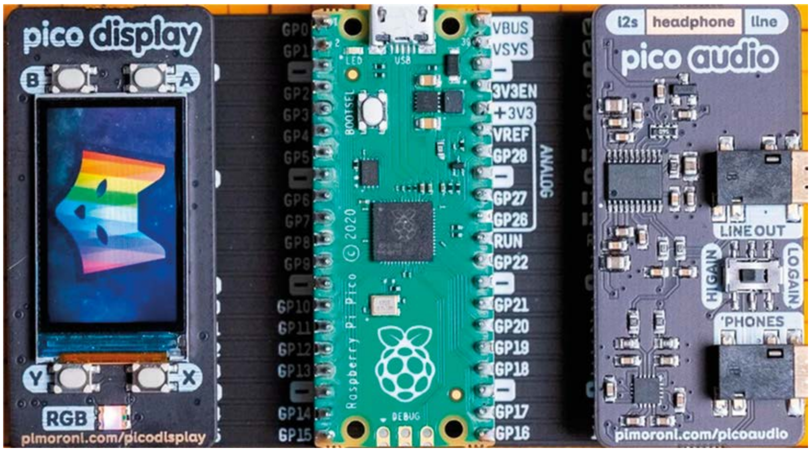


Figure 1.18: Pico Omnibus.

1.6.11 Pimoroni Pico VGA Demo Base

This board (Figure 1.19) has VGA output and an SD card slot. The basic features are:

- Powered by Raspberry Pi Pico
- 15-pin VGA connector
- I^{2}S 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

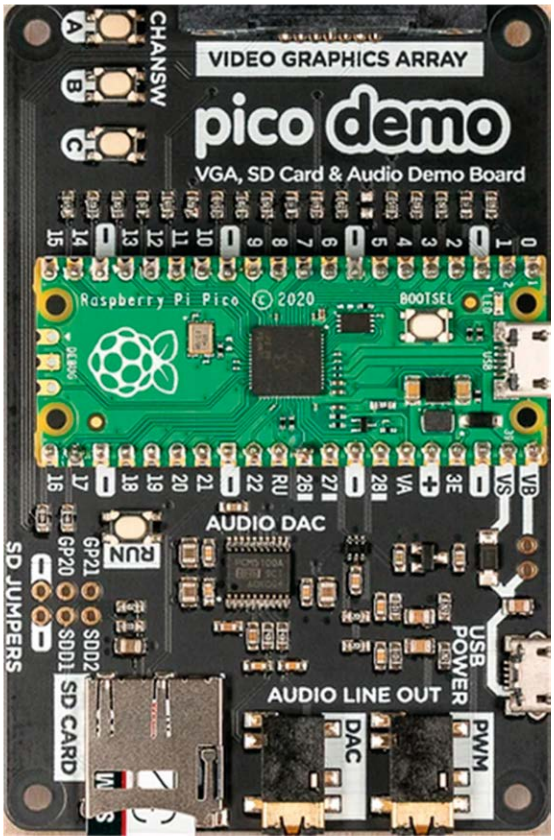


Figure 1.19: Pimoroni Pico VGA Demo Base.

1.6.12 Tiny 2040

This board (Figure 1.20) is a postage stamp sized RP2040 development board with a USB-C connection and 8 MB of flash. The board features:

- 264 KB SRAM
- USB-C connector for power, programming, and data transfer
- 8 MB QSPI flash supporting XiP
- User controllable RGB LED
- 12 IO pins (including four 12-bit ADC channels)
- Switch for basic input (doubles up as DFU select on boot)
- On-board 3V3 regulator (max output current 300mA)
- Input voltage range 3 V to 5.5 V
- Dimensions: approx. 22.9 ×18.2 × 6mm (L x W x H, including the USB-C port)

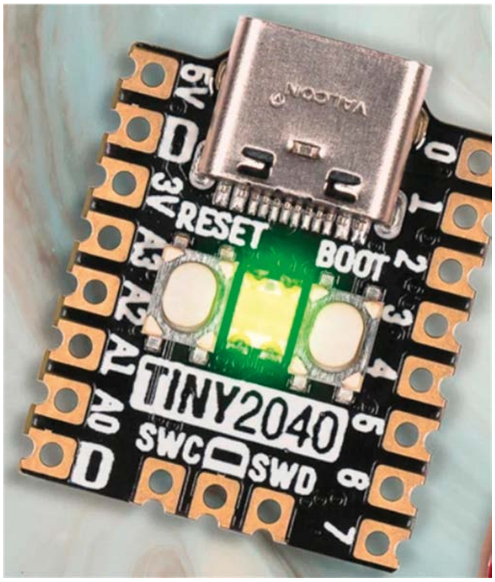


Figure 1.20: Tiny 2040

Chapter 2 • Raspberry Pi Pico Programming

2.1 Overview

At the time of authoring this book, the Raspberry Pi Pico could be programmed using 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.

2.2 Installing MicroPython on Pico

MicroPython must be installed on the Pico before it can be used. Once it is installed it stays on your Pico unless it is overwritten with something else. Installing MicroPython requires an Internet connection, although 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., a Raspberry Pi 4), or by using a PC. In this section we will see how to do the installation using both methods.

2.2.1 Using a Raspberry Pi 4 to help install MicroPython on the Pico

The steps are as follows:

- Boot your Raspberry Pi 4 and login 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 let go 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).

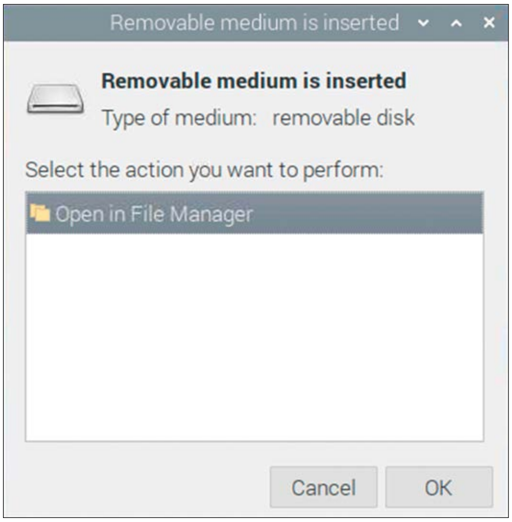

Removable medium is inserted
Removable medium is inserted
Type of medium: removable disk
Select the action you want to perform:
Open in File Manager
Cancel
OK

Figure 2.1: Click OK.

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

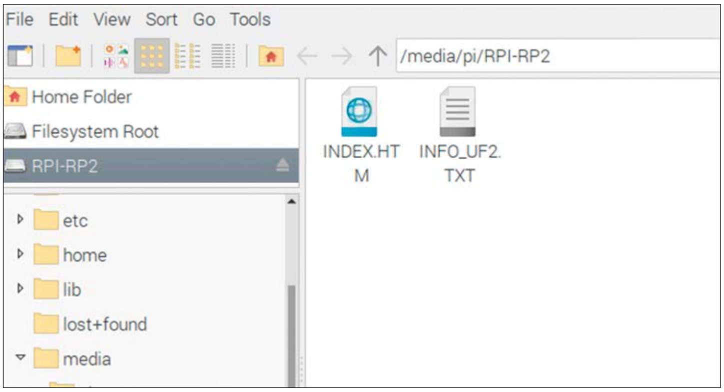


Figure 2.2: You will see two files.

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

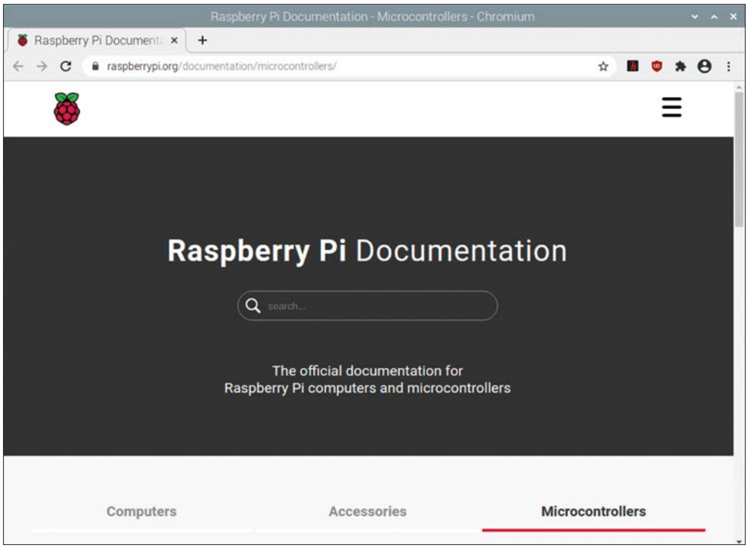

Raspberry Pi Documentation - Microcontrollers - Chromium
Raspberry Pi Document
raspberrypi.org/documentation/microcontrollers/
Raspberry Pi Documentation
search...
The official documentation for
Raspberry Pi computers and microcontrollers
Computers
Accessories
Microcontrollers

Figure 2.3: Displayed message.

- Scroll down and click on the **MicroPython** tab and then click **Download UF2 file** to download the **MicroPython** firmware (Figure 2.4). You should see the downloaded file message at the bottom of the screen (Figure 2.5).

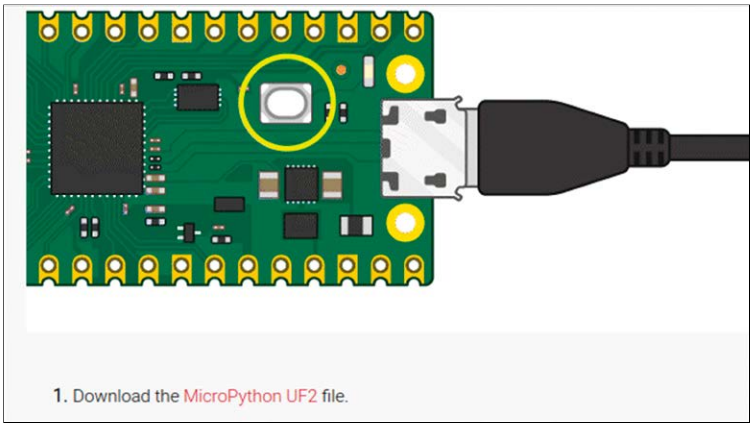
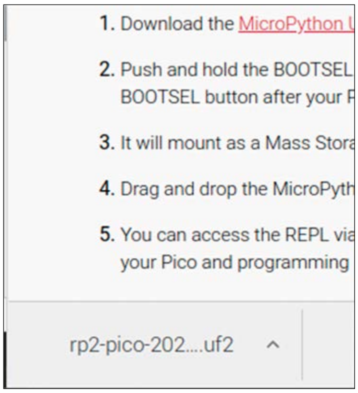


Figure 2.4: Download UF2 file.



1. Download the MicroPython U
2. Push and hold the BOOTSEL
   BOOTSEL button after your P
3. It will mount as a Mass Stora
4. Drag and drop the MicroPyth
5. You can access the REPL via
   your Pico and programming

rp2-pico-202....uf2

Figure 2.5: Downloaded message.

- 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 the name similar to: **rp2-pico-20210902-v1.17.uf2** (Figure 2.6).

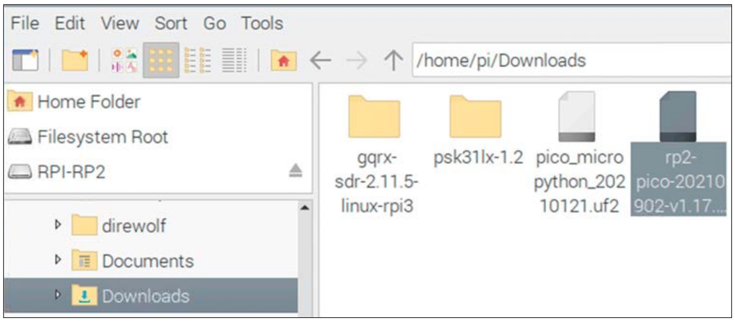

File Edit View Sort Go Tools
/home/pi/Downloads
Home Folder
Filesystem Root
RPI-RP2
direwolf
Documents
Downloads
gqrx-sdr-2.11.5-linux-rpi3
psk31lx-1.2
pico_micropython_20210121.uf2
rp2-pico-20210902-v1.17...

Figure 2.6: Locate file with extension: .uf2.

- Drag and drop this file onto the Raspberry Pi Pico's removable drive called: **RPI-RP2** (at the top left side of the screen – see Figure 2.6).
- After a while, the **MicroPython** firmware will be installed onto the internal storage of the Pico and the drive will disappear. Close the window.
- Your Pico is now running **MicroPython**.
- Powering down the Pico will not remove 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 the Thonny by invoking it from the Raspberry Pi. You should leave your Pico connected to the Raspberry Pi. We will create a one-line program to display the message **Hello from Raspberry Pi Pico**:

The steps are:

- Click menu, followed by **Programming** on your Raspberry Pi Desktop and then click **Thonny Python IDE** (see Figure 2.7). The author had version 3.7.3 of Thonny installed on his Raspberry Pi 4.

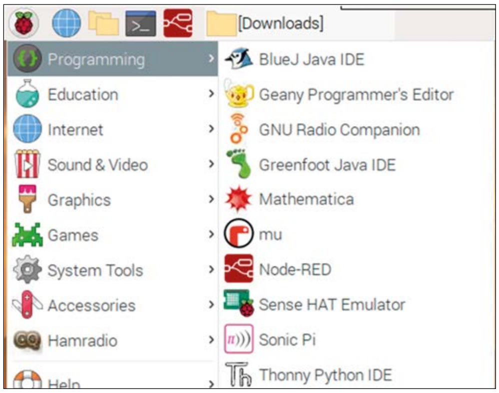


Figure 2.7: Start Thonny on your Raspberry Pi.

- Click on the label **Python** at the bottom right hand corner of Thonny (Figure 2.8).

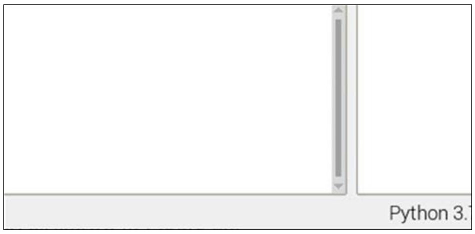


Figure 2.8: Click "Python" in the bottom right-hand corner.

- Click to select **MicroPython (Raspberry Pi Pico)** as shown in Figure 2.9.

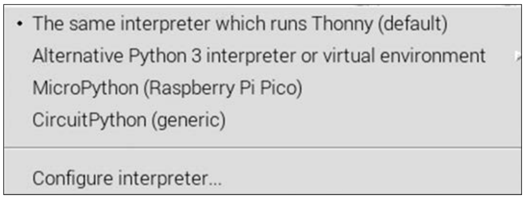
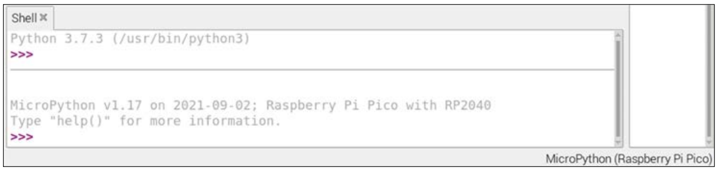

• The same interpreter which runs Thonny (default)
Alternative Python 3 interpreter or virtual environment
MicroPython (Raspberry Pi Pico)
CircuitPython (generic)
Configure interpreter...

Figure 2.9: Select Raspberry Pi Pico.

- You should see the version number of your MicroPython displayed in the bottom part of the screen where the Shell resides (Figure 2.10).



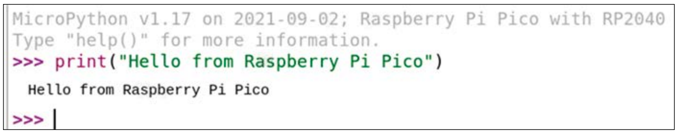
```
Python 3.7.3 (/usr/bin/python3)
>>>

MicroPython v1.17 on 2021-09-02; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

Figure 2.10: Version number of MicroPython is displayed.

- We are now ready to write our simple program. Enter the following line in the lower part of the screen where **Shell** is visible. Program statements written in this part of Thonny are executed online and immediately. This part is normally used to evaluate parts of a program. Enter:

```
print("Hello from Raspberry Pi Pico")
```

and you should see the message **Hello from Raspberry Pi Pico** displayed, as shown in Figure 2.11.



```
MicroPython v1.17 on 2021-09-02; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> print("Hello from Raspberry Pi Pico")
 Hello from Raspberry Pi Pico
>>> 
```

Figure 2.11: Displaying a message.

Icons of the Thonny

At the top of the Thonny screen, you will see a number of icons as shown in Figure 2.12. The functions of these icons are described in this section (notice that letters are used to identify the icons).

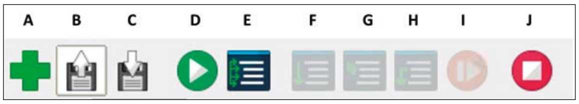


Figure 2.12: Thonny icons.

A: NEW.	This icon is used to create a new file.
B: Open.	This icon is used to Open an existing file
C: Save.	This icon is used to Save a file
D: Run.	This option is used to run the current program
E: Debug.	This icon is used to debug the current program
F: Step Over.	This option is used to step over a function when in Debug mode
G: Step Into.	This option is used to step into a function in Debug mode
H: Step Out.	This icon is used to step out of a function in Debug mode
I: Resume.	This option is used to resume a stopped session
J: Stop/Restart.	This option is used to stop/restart a session