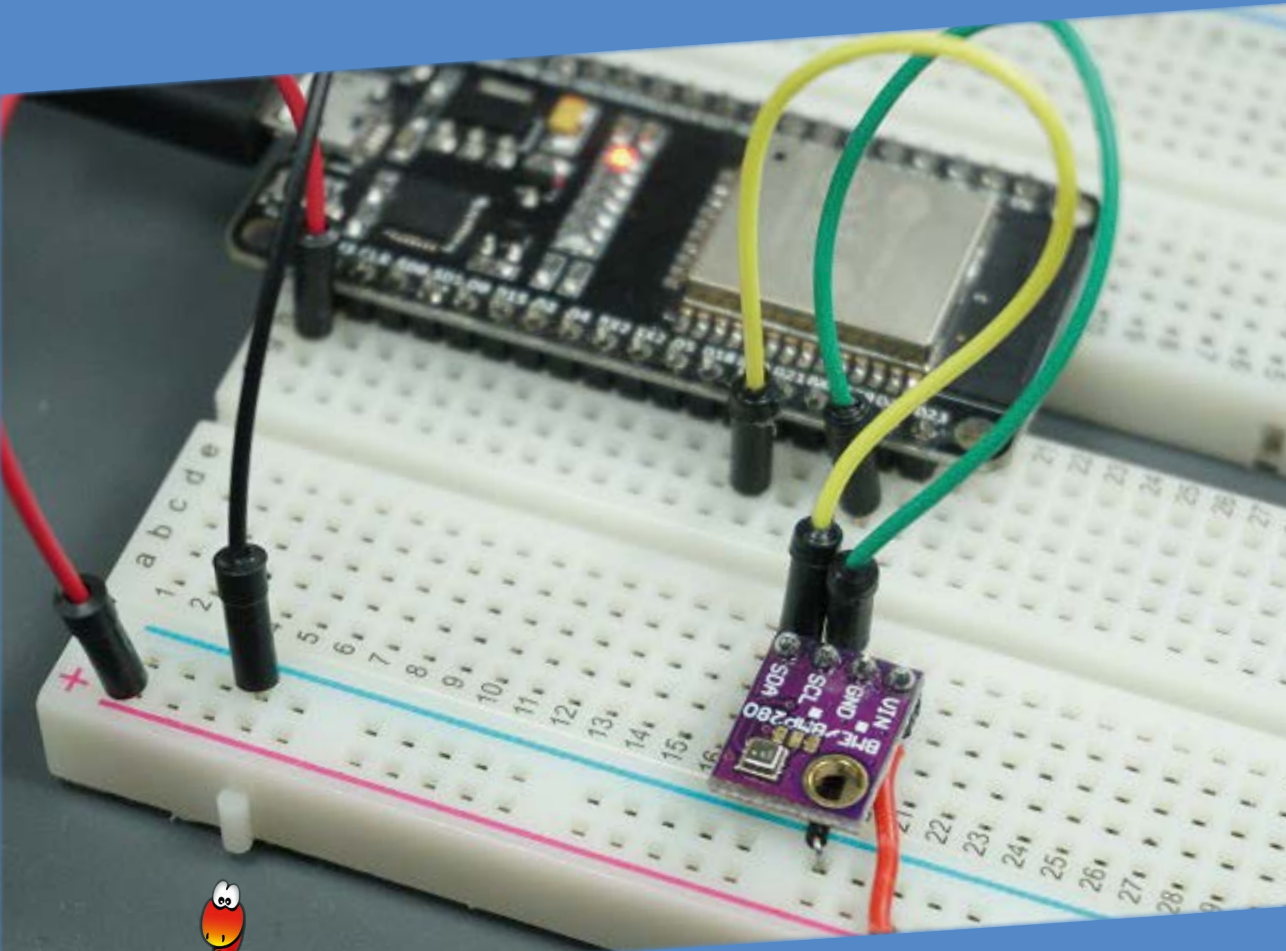


# MicroPython for Microcontrollers

Projects with Thonny-IDE, uPyCraft-IDE, and ESP32



Günter Spinner

---

---

# MicroPython for Microcontrollers

Projects with Thonny-IDE, uPyCraft-IDE, and ESP32



Dr Günter Spanner

---

● 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

● All rights reserved. No part of this book may be reproduced in any material form, including photocopying, or storing in any medium by electronic means and whether or not transiently or incidentally to some other use of this publication, without the written permission of the copyright holder except in accordance with the provisions of the Copyright Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licencing Agency Ltd., 90 Tottenham Court Road, London, England W1P 9HE. Applications for the copyright holder's permission to reproduce any part of the publication should be addressed to the publishers.

● **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 978-3-89576-436-3** Print
- **ISBN 978-3-89576-437-0** eBook
- **ISBN 978-3-89576-438-7** ePub

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

Elektor is part of EIM, the world's leading source of essential technical information and electronics products for pro engineers, electronics designers, and the companies seeking to engage them. Each day, our international team develops and delivers high-quality content - via a variety of media channels (including magazines, video, digital media, and social media) in several languages - relating to electronics design and DIY electronics. [www.elektormagazine.com](http://www.elektormagazine.com)

---

|   |           |
|---|-----------|
| <b>Notices and Disclaimers</b> . . . . .                              | <b>9</b>  |
| <b>Demo Programs Download Archive</b> . . . . .                       | <b>10</b> |
| <b>Chapter 1 • Introduction</b> . . . . .                             | <b>11</b> |
| 1.1 Python, C, or Arduino? . . . . .                                  | 12        |
| 1.2 Requirements . . . . .  | 12        |
| <b>Chapter 2 • A Variety of ESP Boards</b> . . . . .                  | <b>14</b> |
| 2.1 Commissioning and function test . . . . .                         | 16        |
| 2.2 ESP32 on battery power . . . . .                                  | 17        |
| <b>Chapter 3 • Programming and Development Environments</b> . . . . . | <b>19</b> |
| 3.1 Installing the uPyCraft IDE . . . . .                             | 19        |
| 3.2 MicroPython for the ESP32 . . . . .                               | 22        |
| 3.3 "Hello World" for the controller . . . . .                        | 23        |
| 3.4 For professionals: Working with esptool . . . . .                 | 27        |
| 3.5 Thonny — a Python-IDE for beginners . . . . .                     | 31        |
| 3.6 Working with Thonny . . . . .                                     | 34        |
| 3.7 Working with files . . . . .                                      | 36        |
| 3.8 Troubleshooting tips for the Thonny IDE . . . . .                 | 36        |
| <b>Chapter 4 • First Steps in Programming</b> . . . . .               | <b>39</b> |
| 4.1 Never without: Comments . . . . .                                 | 41        |
| 4.2 The Print() statement. . . . .                                    | 42        |
| 4.3 Indentations and blocks . . . . .                                 | 44        |
| 4.4 The hardware under control: digital inputs and outputs . . . . .  | 45        |
| 4.5 Time control and sleep . . . . .                                  | 48        |
| 4.6 Important values: variables and constants . . . . .               | 50        |
| 4.7 Numbers and types of variables . . . . .                          | 50        |
| 4.8 Converting number types . . . . .                                 | 52        |
| 4.9 Little Big Data: Arrays . . . . .                                 | 52        |
| 4.10 Operators . . . . .  | 53        |
| 4.11 With format, please: appealing text and data output . . . . .    | 55        |
| 4.12 Characters in chains: strings . . . . .                          | 58        |
| <b>Chapter 5 • The Controller in Practical Use</b> . . . . .          | <b>60</b> |
| 5.1 LED flasher as alarm system simulator . . . . .                   | 60        |

|  |           |
|--|-----------|
| 5.2 Useful in an emergency: automatic SOS signal . . . . .                       | 61        |
| <b>Chapter 6 • Program Structures . . . . .</b>                                  | <b>63</b> |
| 6.1 Conditions and loops . . . . .   | 63        |
| 6.2 Running lights and airport lighting . . . . .                                | 64        |
| 6.3 Electronic rainbow: RGB LED in use . . . . .                                 | 66        |
| 6.4 SOS compact-style . . . . .  | 67        |
| 6.5 Trial and error: try and except . . . . .                                    | 68        |
| <b>Chapter 7 • Analogue-Signal Generation . . . . .</b>                          | <b>70</b> |
| 7.1 Pulswidth modulation . . . . .   | 70        |
| 7.2 For romantic evenings: heartbeat simulator . . . . .                         | 73        |
| 7.3 Light alarm clock for a relaxed wake-up . . . . .                            | 74        |
| 7.4 Mood-Light with multicolour LED . . . . .                                    | 75        |
| 7.5 Clean and smooth: analogue values from the DAC. . . . .                      | 76        |
| 7.6 Output of time-dependent voltages . . . . .                                  | 77        |
| 7.7 For interesting curves: An arbitrary function generator . . . . .            | 78        |
| <b>Chapter 8 • Interrupts and Timers. . . . .</b>                                | <b>81</b> |
| 8.1 Disruption wanted: Interrupts . . . . .                                      | 81        |
| 8.2 Automatic night light . . . . .  | 82        |
| 8.3 Masters of Time: Timers. . . . .   | 84        |
| 8.4 A multifunctional flashing light . . . . .                                   | 86        |
| <b>Chapter 9 • Using Sensors . . . . .</b>                                       | <b>90</b> |
| 9.1 Acquisition of measurement and sensor values . . . . .                       | 90        |
| 9.2 Precise recording of voltages: a DIY voltmeter . . . . .                     | 92        |
| 9.3 Linearity correction . . . . .   | 95        |
| 9.4 Linearization by limitation of the value range . . . . .                     | 96        |
| 9.5 Linearization of the ADC input by means of compensation polynomial . . . . . | 97        |
| 9.6 Voltage measurement . . . . .  | 98        |
| 9.7 Cross-interferences: side effects in sensor technology . . . . .             | 101       |
| 9.8 Touching permitted: capacitive touch sensors . . . . .                       | 102       |
| 9.9 Well chilled or overheated: temperature sensors provide clarity . . . . .    | 105       |
| 9.10 Digital temperature recording for error-free data transmission . . . . .    | 108       |
| 9.11 The DS18×20 One-Wire sensor . . . . .                                       | 108       |

---

|   |            |
|---|------------|
| 9.12 Data power: multi-sensor array with the DS18x20 thermal sensor. . . . .      | 110        |
| 9.13 In full view: optical sensors. . . . .                                       | 112        |
| 9.14 For film and photo professionals: electronic luxmeter. . . . .               | 113        |
| 9.15 Electronic bats: distance measurement with ultrasound . . . . .              | 115        |
| 9.16 No more dents and scratches: distance warning device for garages . . . . .   | 119        |
| 9.17 Optimum indoor climate for flora and fauna . . . . .                         | 121        |
| 9.18 "Trust me ...": sensor comparison . . . . .                                  | 125        |
| 9.19 Air pressure and altitude measurement . . . . .                              | 126        |
| 9.20 Detecting magnetic fields with the Hall sensor. . . . .                      | 129        |
| 9.21 Alarm detectors monitor door and gate . . . . .                              | 130        |
| <b>Chapter 10 • Display Technology and Small-Size Screens . . . . .</b>           | <b>132</b> |
| 10.1 Graphical representations . . . . .  | 135        |
| 10.2 OLED display as data plotter . . . . .                                       | 138        |
| 10.3 The exact time please: digital clock with OLED display . . . . .             | 140        |
| 10.4 Not just for athletes: a stopwatch . . . . .                                 | 144        |
| 10.5 Just touch: stop watch with sensor keys . . . . .                            | 145        |
| 10.6 Great climate with the BME280 sensor! . . . . .                              | 148        |
| <b>Chapter 11 • LED Matrices and Large Displays . . . . .</b>                     | <b>150</b> |
| 11.1 LED matrix in action. . . . .  | 152        |
| 11.2 Running scripts and animated graphics. . . . .                               | 153        |
| <b>Chapter 12 • Physical Computing: Servos Bring Movement into Play . . . . .</b> | <b>155</b> |
| 12.1 A servo tester . . . . .   | 155        |
| 12.2 Mega-display servo thermometer. . . . .                                      | 158        |
| <b>Chapter 13 • RFID and Wireless Data Transmission . . . . .</b>                 | <b>161</b> |
| 13.1 Reading cards and chips . . . . .  | 162        |
| 13.2 Contactless and secure: RFID lock. . . . .                                   | 164        |
| <b>Chapter 14 • MicroPython and the Internet of Things (IoT) . . . . .</b>        | <b>167</b> |
| 14.1 For modern detectives: a network scanner . . . . .                           | 168        |
| 14.2 Connected but no cables: WLAN . . . . .                                      | 169        |
| 14.3 Switch and control with the web server . . . . .                             | 172        |
| 14.4 The WLAN web server in action . . . . .                                      | 176        |
| 14.5 Reading out sensor data via WLAN. . . . .                                    | 177        |

|   |            |
|---|------------|
| 14.6 Recording environmental parameters: WLAN Thermo/Hygrometer . . . . .       | 179        |
| <b>Chapter 15 • Simple and Good: The MQTT Protocol . . . . .</b>                | <b>183</b> |
| 15.1 MQTT via ThingSpeak . . . . .  | 185        |
| <b>Chapter 16 • Sending Data to the Internet via ThingSpeak . . . . .</b>       | <b>190</b> |
| 16.1 Rain or storm? Virtual weather station available worldwide . . . . .       | 190        |
| 16.2 Graphical representation of data in ThingSpeak . . . . .                   | 194        |
| 16.3 Data for the smartphone with the ThingView app . . . . .                   | 195        |
| 16.4 Against unwanted visitors: Optical room surveillance . . . . .             | 196        |
| <b>Chapter 17 • Micropower Techniques and Sleep Modes . . . . .</b>             | <b>199</b> |
| 17.1 Saving power protects the environment: low-power technologies . . . . .    | 199        |
| 17.2 Disabling unnecessary consumers . . . . .                                  | 200        |
| 17.3 Weather station with battery or solar operation . . . . .                  | 201        |
| <b>Chapter 18 • Bus Systems for Efficient Communication . . . . .</b>           | <b>202</b> |
| 18.1 Basics and applications of the I <sup>2</sup> C bus . . . . .              | 202        |
| 18.2 The SPI bus . . . . .  | 207        |
| 18.3 The members of the SPI family . . . . .                                    | 210        |
| 18.4 Controlling SD and µSD cards via SPI . . . . .                             | 210        |
| <b>Chapter 19 • Building Circuits with Components and Breadboards . . . . .</b> | <b>212</b> |
| 19.1 Breadboards . . . . .  | 213        |
| 19.2 Wire jumpers and jumper cables . . . . .                                   | 214        |
| 19.3 Resistors . . . . .  | 215        |
| 19.4 Light-emitting diodes (LEDs) . . . . .                                     | 216        |
| 19.5 Capacitors and electrolytic capacitors . . . . .                           | 217        |
| <b>Chapter 20 • Troubleshooting . . . . .</b>                                   | <b>219</b> |
| <b>Chapter 21 • Hardware Resources . . . . .</b>                                | <b>220</b> |
| <b>Chapter 22 • List of Figures . . . . .</b>                                   | <b>221</b> |
| <b>Chapter 23 • Bill of Materials . . . . .</b>                                 | <b>225</b> |
| <b>Index . . . . .</b>  | <b>226</b> |

## Notices and Disclaimers

1. The circuits in this book are for educational purposes only.
2. The circuits in this book may only be operated with batteries and/or tested, double insulated safety power supplies. Insulation faults of a simple power supply unit can lead to life-threatening voltages on non-insulated components.
3. Powerful LEDs can cause eye damage. Never look directly into an LED!
4. Neither the Author or the Publisher accept any liability for damage resulting from the construction of the described projects, or attempts to do so.
5. Electronic circuits can emit electromagnetic interference radiation. Since neither the Publisher or the Author have any influence on the user's skills in assembling, operating, and controlling electronic circuits, the user alone is responsible for compliance with the relevant emission limit values.



## Demo Programs Download Archive

The demo programs mentioned in this book can be downloaded free of charge as a single archive file (.zip) from the book resources web page:

[www.elektor.com/micropython-for-microcontrollers](http://www.elektor.com/micropython-for-microcontrollers)

If a program from the download archive file appears to differ from the version described in the book, the downloaded version should be used as it may reflect updates made by the author since printing the book.

## Chapter 1 • Introduction

The introduction of the ESP32 chip from Espressif Systems marks a new generation of microcontrollers, which offer excellent performance, Wi-Fi, and Bluetooth functionality at an unrivalled price. These features have taken the maker scene by storm. The most diverse applications and projects in the areas of Internet of Things (IoT) and home automation can be implemented easily and cost-effectively. The ESP32 is as easy to program as the classic Arduino boards. In comparison, however, the ESP32 also offers, among other things:

- Larger flash and SRAM memory
- Much higher CPU speed
- Integrated Wi-Fi / WLAN
- Bluetooth functionality
- More GPIO pins
- Extensive interface functionality
- Analogue/digital converter with higher resolution
- Digital/analogue converter
- Security and encryption functions

For these reasons, it can be considered the most promising successor to Arduino. The term "Arduino killer" is often used in this context.

This book introduces the programming of modern single-chip systems (Systems on Chip — SoCs). In addition to the technical background, the focus is on the programming language Python, especially in its variant "MicroPython". Basic relationships between electronics and electrical engineering will only be dealt with to the extent that it is essential for the design of the circuits and experiments.

The "hardware" for getting started can be kept very simple anyway. At first, only a controller board and some light emitting diodes as well as suitable series resistors are required. The PC or laptop required for programming the chip should be available in every household. The appropriate programming environment can be downloaded free of charge from the internet. When working with a MicroPython programming environment, however, the first problems quickly arise. A good introduction can therefore lead to excellent performances.

Python has experienced an enormous upswing in recent years. Various single-board systems like the Raspberry Pi have especially contributed to its popularity. But Python has also found widespread use in other areas such as artificial intelligence or machine learning. Hence it is a logical choice to use Python or the variant MicroPython for the application in SoCs, too.

However, we will not restrict ourselves to a mere introduction to the programming language. In many cases, the learned programming skills are put into practice together with electronic circuitry. The fully described projects are all suitable for use in laboratories or in everyday life. In addition to the educational effect, the pleasure of assembling complete and useful devices is therefore also in the foreground. Through the use of laboratory plug-in boards, circuits of all kinds can be realised with little effort. The testing

and trial of applications thus becomes an educational pleasure.

Due to the various applications such as weather stations, digital voltmeters and function generators, the presented projects are also ideally suited for internships or study courses in the natural sciences or in science and technology lessons.

### 1.1 Python, C, or Arduino?

For beginners, the Arduino programming environment is one of the easiest places to program the ESP32. Behind this interface is the Arduino version of C, as well as C++. These two programming languages have been popular for years for the development of embedded systems. The Arduino version of C made it even easier to get started. In addition, one of the largest technology communities in the world developed for this purpose. With new libraries, software fixes and board support, problems could usually be solved quickly. However, the restriction that Arduino-C only works in its designated environment is not insignificant. Especially for the development of more extensive projects, useful and important functions are missing. Therefore, Arduino-C remained mostly limited to hobby and beginner projects.

MicroPython is relatively new. The user community is growing, and more and more platforms are supported. MicroPython is essentially a slim version of Python, one of the most popular programming languages in the world. Therefore, specific problems can be dealt with not only in MicroPython communities. In fact, general Python forums are increasingly contributing to solving MicroPython issues.

In addition to community support, MicroPython also has certain features that put it well above the class of the Arduino. One of these features is the so-called REPL function. REPL stands for "Read-Evaluate-Print Loop". This allows programs and code sections to be executed quickly. Compiling or uploading is not necessary. In this way, parts of a code can be tested quickly and efficiently during development.

MicroPython contains a very compact implementation of the Python interpreter. It requires only 256 KB flash memory and 16 KB RAM. Nevertheless the interpreter is designed for maximum compatibility with the standard Python. Syntax and language range correspond largely to Python version 3.4, so experienced Python programmers should be able to find their way around immediately. In addition, a few language elements beyond the standard are defined, which keep the memory requirements low and increase the execution speed.

### 1.2 Requirements

In order to work successfully with this book, the following requirements should be met:

- Secure handling of the Windows operating system;
- Basic knowledge of any programming language such as C, Java or similar;
- Basic knowledge in the field of electronics, especially in the area of "current — voltage — resistance" is assumed.

For specialized knowledge in the field of electronics, please refer to the extensive technical literature, especially from Elektor, through their books, magazines, and kits. The hardware

structure was deliberately kept simple, as the focus should be on programming with MicroPython. Nevertheless, different components and parts are required. Explanations can be found in the individual chapters in which the components are used first. In addition, the last sections of the book explain some basic components such as resistors or light emitting diodes. You can consult these if there are any unclarities concerning individual components.

Further requirements are

- a PC or laptop with USB interface,
- the Windows 10 operating system,
- internet access.

It is also useful to employ an active USB hub between the computer and the controller. This has the advantage that it guarantees a certain protection for the PC. The hub should have its own 5 V power supply through a separate power supply unit. Then the PC or laptop is best protected against short-circuits behind the hub, as it is very unlikely that a short circuit will "blow through" an active hub to the USB port of the computer.

## Chapter 2 • A Variety of ESP Boards

The ESP32 is a modern and extremely powerful microcontroller. In addition to a high clock frequency and the extensive internal functional units, the chip has integrated Wi-Fi and Bluetooth. The controller was developed by Espressif Systems, a Chinese company based in Shanghai, and is enjoying increasing popularity. The performance features of the ESP far exceed the well-known Arduino boards in terms of price and performance.

As the ESP32 is only available as an SMD chip, a so-called break-out board (BoB) or development board is required if the controller is to be used in a non-professional environment. In the meantime, a virtually unmanageable variety of different versions is available on the market. The best-known versions are:

| Board              | Pins  | Buttons     | LiPo Charger |
|--------------------|-------|-------------|--------------|
| ESP32-PICO-KIT     | 34    | EN and BOOT | no           |
| JOY-iT NodeMCU     | 30    | EN and BOOT | no           |
| ESP32 DEV KIT DOIT | 30/36 | EN and BOOT | no           |
| Adafruit HUZZAH32  | 28    | RESET       | yes          |
| ESP32 Thing        | 40    | EN and BOOT | yes          |
| LOLIN32            | 40    | EN and BOOT | no           |
| Node-ESP-Board     | 38    | EN and BOOT | no           |

The following image shows just three different versions:

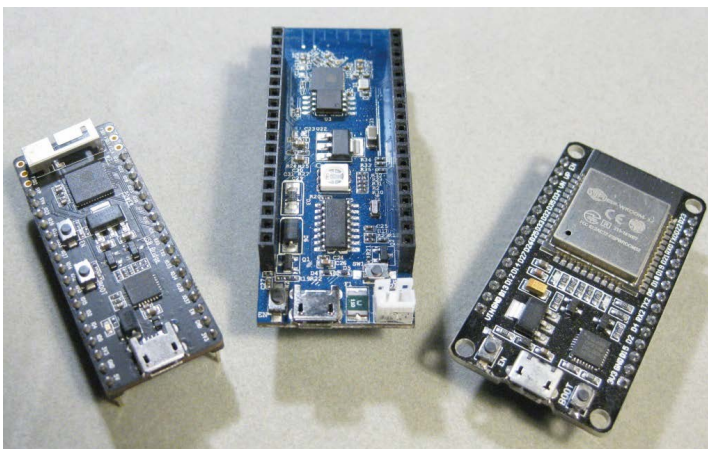


Figure 2.1: ESP32 boards (PICO-KIT, Node-ESP and NodeMCU).

These boards have the necessary prerequisites to operate the ESP controller on a solderless plug-in board or breadboard. Frequently, in addition to the controller, other components such as pushbuttons, a Li-ion battery charger, or various LEDs are mounted on the boards

available. This means that initial tests and experiments can be carried out without external circuitry.

The following section summarizes the most important data about ESP32. The overview should only provide a first impression. A deeper understanding of the individual features and functions is then provided in the relevant sections in the book.

|                          |   |
|--------------------------|---|
| Processor:               | 160 / 240 MHz Tensilica LX6 dual-core microprocessor  |
| Memory:                  | 520 KByte SRAM / 16 MByte Flash memory  |
| Power supply:            | 2.2 V to 3.6 V  |
| Power consumption:       | Standard: approx. 50 - 70 mA<br>Wi-Fi mode: approx. 80 - 170 mA<br>Deep sleep mode: approx. 2.5 $\mu$ A   |
| Ambient temperature:     | -40 °C to +125 °C   |
| Inputs/Outputs (GPIOs):  | 32 general purpose ports with PWM<br>Function and timer logic   |
| Wi-Fi:                   | 802.11 b/g/n/e/i  |
| Network throughput:      | 135 MBit/s (via UDP protocol)   |
| Receiver sensitivity:    | -98 dBm   |
| Bluetooth:               | V4.2 BR/EDR and BLE   |
| Bluetooth functionality: | Classic and Bluetooth Low Energy (with integrated antenna)  |
| Sensors:                 | Hall sensor<br>10 $\times$ capacitive touch sensor  |
| Interfaces:              | 3 $\times$ UART with flow control via hardware 3 $\times$ SPI interfaces<br>CAN bus 2.0 controller<br>2 $\times$ I2S and 2 $\times$ I2C interfaces<br>18 analogue inputs with 12-bit analogue-to-digital converters<br>2 analogue outputs with 10-bit digital-to-analogue converters<br>Infrared (IR) (TX/RX)<br>Motor PWM<br>LED-PWM with up to 16 channels<br>Interface for external SPI flash memory for up to 16 MB<br>SD card hardware |
| Security:                | Wi-Fi: WFA, WPA/WPA2 and WAPI Secure Boot<br>Flash Encryption<br>Cryptographic Hardware Acceleration Elliptic Curve Cryptography (ECC)<br>Random Number Generator (RNG)   |

The ESP32-PICO-KIT is usually employed for the application examples in this book. Alternatively the ESP32 DEV KIT or another board can be used. The variant used is mentioned explicitly in each case. In principle, however, the various boards are largely compatible. They differ mainly in size and in pin arrangement order. Figure 2.2. shows the PICO-KIT with its functional units and connections.

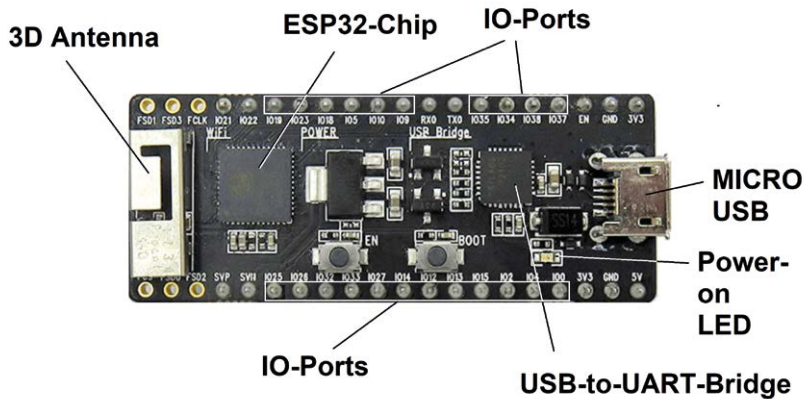


Figure 2.2: ESP32 PICO-KIT board.

The breakout boards are best suited as experimental and development boards. Via the port connections, electronic components such as LEDs, temperature sensors or even smaller actuators such as R/C model servos can be connected directly. The Pico Kit board has the following features, among others:

- ESP controller with two 32-bit cores
- Fast Wi-Fi and WLAN interface (up to 150 MBit/s)
- ADC & DAC functionality
- Touch Sensor Unit
- Host Controller for SD/SDIO/MMC
- SDIO/SPI Controller
- EMAC and PWM unit for controlling LEDs and motors
- UART, SPI, I<sup>2</sup>C and I<sup>2</sup>S interfaces
- Infrared remote-control controller
- GPIO Interface
- Bluetooth/Bluetooth LE (4.2)
- USB-to-Serial-Chip for access via USB interface

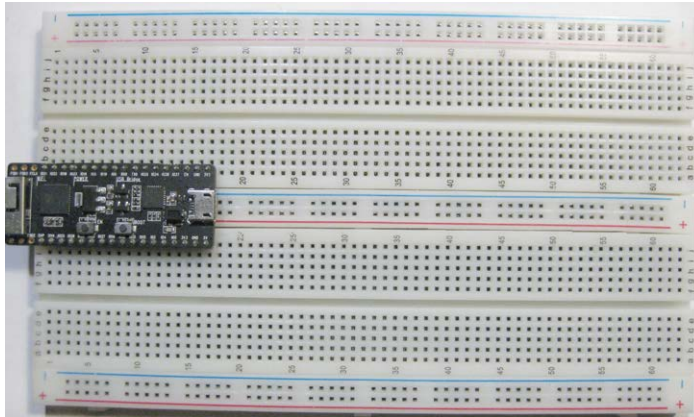
## 2.1 Commissioning and function test

As soon as a board is available, it should be subjected to an initial functional test. For this purpose, a USB cable is connected to a PC or laptop and the micro-USB socket of the board. If present, a USB hub should already be connected in between.

On most boards, an LED will light up when connected to a powered USB port. If, contrary to expectations, this so-called "power-on" LED does not light up, the USB connection should be disconnected immediately. In this way you can prevent a possible short circuit from causing major damage. For further troubleshooting, helpful hints are given in the corresponding chapter at the end of the book.

ESP boards should always be operated in a solderless plug-in board (see Figure 2.3). However, if you do work without a breadboard, make sure that the base used is not conductive,

otherwise short circuits between the pins may occur. Next to the ESP board itself, this can even destroy the USB port of the PC.



*Figure 2.3: ESP board plugged into a breadboard.*

## 2.2 ESP32 on battery power

In most cases, an ESP board is supplied with power via the Micro-USB socket. Since the controller is connected to a PC or laptop during program creation anyway, no additional power supply is required.

If a direct data exchange with the PC is no longer necessary, the board can also be supplied by a USB power supply. This should be able to supply at least 1,000 mA (1 A; 1 amp) of current output to avoid unwanted voltage drops. In addition, there is a certain amount of reserve power to operate some LEDs, displays or sensors.

Even without a USB connection, the board can send and receive data via Wi-Fi and Bluetooth. In order to achieve complete independence from power and data cables, the module then only needs to be powered by (rechargeable) batteries.

Some boards have a Lithium-Ion (Li-Ion) battery connector for this purpose (see Figure 2.4). There, suitable cells can be connected directly via the standard plug. An internal voltage regulation then ensures that the controllers are optimally supplied. In addition, a connected battery is charged as soon as the board is connected to a live USB socket.

For this application, cells with a capacity of about 1,500 mAh or more are suitable. Smaller batteries below 300 mAh should not be used, as they could be overcharged by the integrated charge controller.

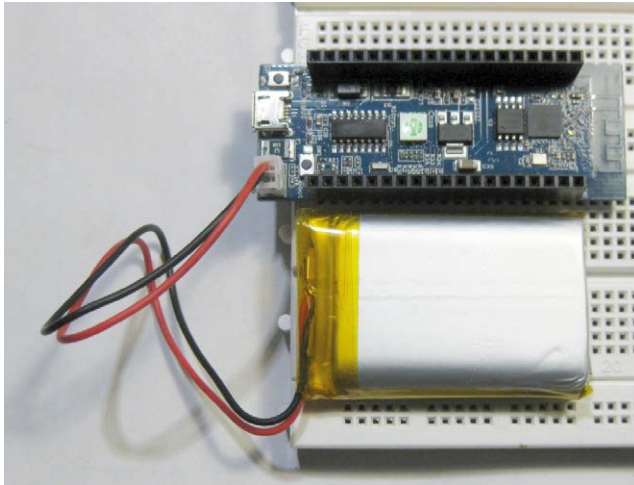
With a typical power consumption of approx. 50 mA, the 1,500 mAh variant can achieve an operating time of around 30 hours, i.e. just over a day. When using the controller's sleep functions, even considerably longer operating times can be achieved.



Single cell LiPo (Lithium Polymer) or Lithium-Ion batteries provide sufficient power for the ESP32. However, their voltage of 3.7 to 4.2 V, depending on the state of charge, is too high for the ESP32. It is therefore regulated down via an internal module.

As working with Li-ion batteries is always associated with a certain degree of danger, the following information should not be omitted:

- Lithium-Ion batteries react sensitively to incorrect charging currents or voltages. Under certain circumstances there is even a risk of fire or explosion.
- Each user is responsible for their own construction and operation of the project.
- Neither the Publisher nor the Author assumes any liability.



*Figure 2.4: NodeESP board in battery mode.*

## Chapter 3 • Programming and Development Environments

Unlike the situation with, say, an Arduino system, there are several Integrated Developing Environments (IDEs) available for working with MicroPython. In principle, you can write programs with all IDEs and load them onto the controller. The two most widespread programming environments are currently:

- $\mu$ PyCraft
- Thonny

Both have their own specific advantages and disadvantages. The differences lie mainly in the different procedures for developing and managing program code for the application projects.

The first variant called  $\mu$ PyCraft offers a comparatively simple interface for MicroPython development on the ESP32 controller. It works with simple graphic elements and resembles text-oriented operating systems. The handling of the individual functions is easy to understand and working with the different menus is easy to learn.

Thonny, on the other hand, has a fully graphical interface in Windows style. The IDE is very popular among makers, especially because it is available under the Raspbian operating system on the Raspberry Pi. Many Raspberry Pi users are therefore already very familiar with Thonny.

The IDEs stand for the most important operating systems such as

- Windows PC
- Mac OS X
- Linux Ubuntu

which are available free of charge on the internet.

If problems occur during installation or use of either system, the other version can be used as an alternative programming system. The version to choose depends of course on the personal inclinations and habits of the user.

### 3.1 Installing the $\mu$ PyCraft IDE

Before installing  $\mu$ PyCraft IDE, the latest version of Python 3.7.X should be installed on the computer you are using. If it is not, the installation can be done according to the following instructions:

1. Download the installation file from the Python download page at:

[www.python.org/downloads](http://www.python.org/downloads)

2. After the download operation, a file named  
`python-3.7.X.exe`  
should reside on your computer. Double-clicking on the file starts the installation.
3. Select "Add Python 3.7 to PATH" and click the "Install Now" button.
4. The installation process is completed after a few seconds and the message "Setup was successful" is displayed. The window can then be closed.

Now the uPyCraft IDE for Windows can be downloaded from

<https://github.com/DFRobot/uPyCraft>

as a file called `uPyCraft_V1.x.exe`. After clicking on this `.exe` file, the uPyCraft-IDE will open:

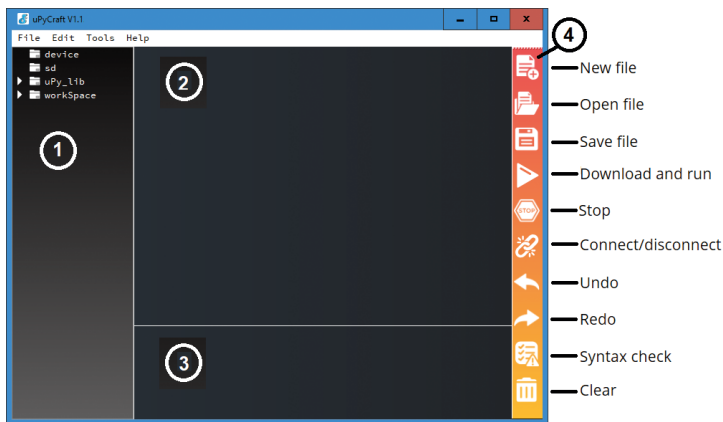


Figure 3.1: uPyCraft-IDE

After the IDE is installed on the computer, the ESP32 firmware can be loaded onto the chip. The current version of the MicroPython firmware for the ESP32 can be found at

<http://micropython.org/download#esp32>

There you scroll to the section "ESP32 modules". After clicking the link to "Generic ESP32 module" you will get to the download page of the ESP32-BIN file. This will look as follows:

`esp32-idf3-20191220-v1.12.bin`

Now you can start the uPyCraft-IDE. Under

Tools -> Serial

select the ESP32-COM port, here as COM5:

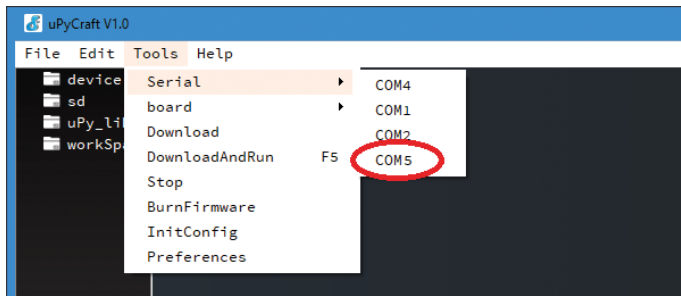


Figure 3.2: Selecting the port.

If the ESP32 board is connected to the computer but the ESP32 port does not appear in the uPyCraft IDE, the appropriate USB driver may be missing. In this case, the driver must be reinstalled. A corresponding driver can be found under

<https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers>

Afterwards you can follow

Tools -> Board

Next, the option "esp32" must be selected:

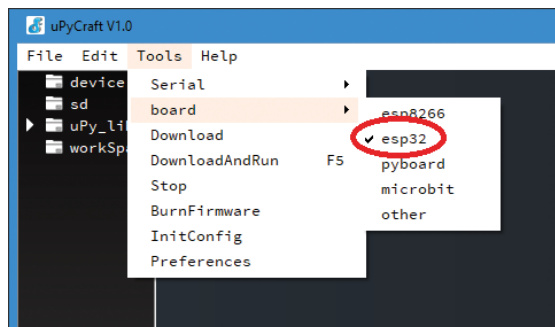


Figure 3.3: Selecting the board type

Now the MicroPython interpreter can be written onto the ESP32 using

Extras -> Burn Firmware

The appropriate options are:

- board: esp32
- burn\_addr: 0x1000
- erase\_flash: yes
- com: COM5 (here COM5, see above)

Under "USERS", select the downloaded ESP32-BIN file, as shown in Figure 3.4.

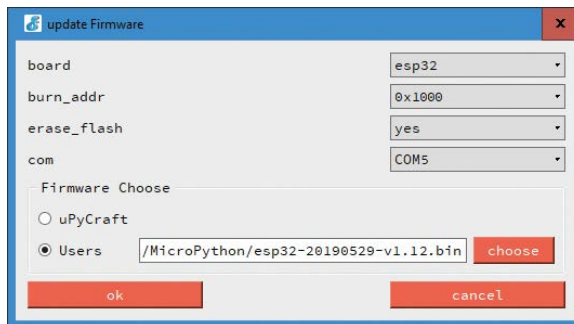


Figure 3.4: The parameters for flashing the firmware

If all settings are correctly selected, the "BOOT / FLASH" button on the ESP32 board must be pressed on some board variants. As soon as the "EraseFlash" process begins, the key can be released. After a few seconds, the firmware should have flashed onto the ESP32 board. However, in many cases the download will start without pressing the buttons.

If the "EraseFlash" display does not start or an error message is displayed, repeat the steps described above. Also press the "BOOT / FLASH" key again to ensure that ESP32 enters the flash mode.

### 3.2 MicroPython for the ESP32

With a few exceptions, all features and functions of Python are also available in MicroPython. The biggest difference is that the micro version was designed for use on single-chip systems and therefore the classic routines required only for the PC are missing.

For this reason MicroPython does not contain the complete standard library, but only the parts relevant for microcontrollers. Therefore, all modules required for accessing the used hardware are available. With the corresponding libraries you can therefore easily access the GPIO pins. Especially for the ESP32 there are also modules available to support network connections (Wi-Fi) and Bluetooth. In particular, the following boards are supported:

- ESP32
- ESP8266
- PyBoard
- Teensy 3.X
- WiPy - Pycom

Although not all functions of the ESP controller are fully available in MicroPython as of yet, the libraries contain the most important commands and routines. Therefore many projects and applications can be implemented smoothly. In addition, the implementation of the missing features is progressing rapidly, so that even this small beauty flaw will be quickly eliminated.

Once the MicroPython firmware has been installed on the ESP32, you can also easily return to the Arduino IDE, for example. To do this, simply load the new C code with the IDE onto the controller. A special deletion procedure is not necessary. However, if you want to use MicroPython again afterwards, the MicroPython firmware must be flashed again.

### 3.3 "Hello World" for the controller

Unlike AVR controllers, such as those used in the Arduino system, the ESP32 can accommodate a complete file system. The first generations of controllers were programmed in either Assembler or C. The program code was therefore created and compiled in a development environment. Then only the finished "machine code" was transferred to the controller. The memory of the target system therefore always contained exactly one program.

In contrast, when programming in MicroPython, several programs can be stored on the ESP32 chip. These can then be processed directly by the interpreter that is also available on the system. The file system can be managed directly with the uPyCraft-IDE. It is therefore advisable to familiarise yourself with the IDE a little more closely before loading the first application program onto the ESP. The development environment contains, similar to many other programming tools, the following components (see also Figure 3.1)

1. Folders and files
2. Editor
3. MicroPython Shell / Terminal
4. Tools

In the left sub-window ("Folders and files"), the files currently stored on the ESP board are visible in the device folder ("device"). As soon as the board is connected to uPyCraft-IDE via a serial connection, all saved files will be loaded when opening the device folder. Directly after the installation of the Python interpreter only a "boot.py" file is visible here. To execute the application code, a main.py file should also be created. You can create a main.py file using:

```
file → new
```

This creates a new file ("untitled"). Through the floppy disk icon in the "Tools" window this file can be saved locally under the name "main.py" on the ESP chip.