MicroPython für Mikrocontroller

Projekte mit uPyCraft-IDE und ESP32

Dr. Günter Spanner

elektor

1. Auflage 2020

Umschlaggestaltung: Elektor, Aachen

Satz und Aufmachung: D-Vision, Julian van den Berg | Oss (NL)
Druck: WILCO, Amersfoort, Niederlande
Printed in the Netherlands

ISBN 978-3-89576-388-5
Ebook 978-3-89576-389-2
Epub 978-3-89576-390-8

Elektor-Verlag GmbH, Aachen
www.elektor.de

Warnhinweise 9

Programmdownload 10

Kapitel 1 • Einführung 11

1.1 Python, C oder Arduino? 12

1.2 Voraussetzungen 13

Kapitel 2 • Vielfältige Auswahl: ESP-Boards 14

2.1 Inbetriebnahme und Funktionstest 16

2.2 ESP32 im Akkubetrieb 17

Kapitel 3 • Programmier- und Entwicklungsumgebungen 19

3.1 Installation der uPyCraft-IDE 19

3.2 MicroPython für den ESP32 22

3.3 "Hello World" für den Controller 23

3.4 Für Profis: Arbeiten mit dem esptool 27

3.5 Thonny - eine Python-IDE für Einsteiger 32

3.6 Arbeiten mit Thonny 34

3.7 Arbeiten mit Dateien 36

3.8 Tipps zur Fehlerbehebung in der Thonny-IDE 37

Kapitel 4 • Erste Schritte in die Programmierung 40

4.1 Niemals ohne: Kommentare 42

4.2 Die Print()-Anweisung 44

4.3 Einrückungen und Blöcke 45

4.4 Die Hardware im Griff: Digitale Ein- und Ausgänge 46

4.5 Zeitsteuerung und sleep 50

4.6 Wichtige Werte: Variablen und Konstanten 51

4.7 Zahlen und Variablentypen 52

4.8 Konvertieren von Zahlentypen 54

4.9 Little Big Data: Arrays 54

4.10 Operatoren 55

4.11 Bitte mit Format: Ansprechende Text- und Datenausgabe 56

4.12 Zeichen in Ketten: Strings 59

Kapitel 5 • Der Controller im praktischen Einsatz 62

5.1 LED-Blitzer als Alarmanalgensimulator 62

5.2 Nützlich für den Notfall: automatisches SOS-Signal 63

Kapitel 6 • Programmstrukturen 65

6.1 Bedingungen und Schleifen 65

6.2 Lauflichter und Flughafenbefeuerungen 66

6.3 Elektronischer Regenbogen: RGB-LED im Einsatz 68

6.4 SOS kompakt 69

6.5 Versuch und Irrtum: try und except 70

Kapitel 7 • Für fließende Übergänge: Analogsignale 72

7.1 Pulsweitenmodulation 72

7.2 Für romantische Abende: Herzschlagsimulator 75

7.3 Lichtwecker für entspanntes Aufwachen 76

7.4 Mood-Light mit Multicolor-LED 77

7.5 Sauber und glatt: Analogwerte aus dem DAC 78

7.6 Ausgabe von zeitabhängigen Spannungen 80

7.7 Für interessante Kurven: Ein Arbitrary Function-Generator 80

Kapitel 8 • Interrupts und Timer 84

8.1 Unterbrechung erwünscht: Interrupts 84

8.2 Automatisches Nachtlicht 85

8.3 Herren der Zeit: Timer 87

8.4 Ein multifunktionales Blinklicht 90

Kapitel 9 • Sensoren im Einsatz 93

9.1 Erfassung von Mess- und Sensorwerten 93

9.2 Spannungen präzise erfassen: Voltmeter im Eigenbau 95

9.3 Linearitätskorrektur 98

9.4 Linearisierung durch Beschränkung des Wertebereiches 99

9.5 Linearisierung des ADC-Eingangs mittels Ausgleichspolynom 100

9.6 Spannungsmessung 101

9.7 Querempfindlichkeiten: Nebenwirkungen in der Sensortechnik 104

9.8 Anfassen erlaubt: Kapazitive Touch-Sensoren 105

9.9 Gut gekühlt ober überhitzt: Temperatursensoren schaffen Klarheit 108

9.10 Digitale Temperaturerfassung für fehlerfreie Datenübertragung 111

9.11 Der One-Wire-Sensor DS18x20 111

9.12 Datenkrake: Multisensor-Array mit dem DS18x20-Thermosensor 113
9.13 Immer optimal belichtet: Optische Sensoren 115
9.14 Für Film- und Foto-Profis: Elektronisches Luxmeter 117
9.15 Elektronische Fledermäuse: Abstandsmessung mit Ultraschall 119
9.16 Nie mehr Beulen und Kratzer: Abstandswarngerät für Garagen 123
9.17 Optimales Raumklima für Flora und Fauna 124
9.18 Trau – schau wem: Sensorvergleich 128
9.19 Luftdruck- und Höhenmessung 129
9.20 Magnetfelder mit dem Hallsensor erfassen 132
9.21 Alarmmelder überwachen Tür und Tor 134
Kapitel 10 • Bildschirme im Kleinformat: Displaytechnik 135
10.1 Grafische Darstellungen 139
10.2 OLED-Display als Datenplotter 141
10.3 Genaue Uhrzeit bitte: Digitaluhr mit OLED-Display 143
10.4 Nicht nur für Sportler nützlich: Stoppuhr 147
10.5 Berühren genügt: Stoppuhr mit Sensortasten 148
10.6 Prima Klima mit dem BME280-Sensor 151
Kapitel 11 • Großanzeigen für alle Gelegenheiten: LED-Matrizen 153
11.1 LED-Matrix in Aktion 155
11.2 Laufschriften und animierte Graphiken 156
Kapitel 12 • Physical Computing: Servos bringen Bewegung ins Spiel 158
12.1 Servotester 158
12.2 Mega-Display-Servo-Thermometer 162
Kapitel 13 • Drahtlose Datenübertragung: RFID 164
13.1 Auslesen von Karten und Chips 165
13.2 Berührungslos und sicher: RFID-Schloss 167
Kapitel 14 • MicroPython und das Internet of Things (IoT) 171
14.1 Für moderne Detektive: Ein Netzwerkscanner 172
14.2 Auch ohne Kabel gut verbunden: WLAN 174
14.3 Schalten und Walten mit dem Webserver 176
14.4 Der WLAN-Webserver in Aktion 180
14.5 Sensordaten über WLAN auslesen 182

14.6 Erfassung von Umweltparametern: WLAN-Thermo/Hygrometer 184
Kapitel 15 • Einfach und gut: Das MQTT-Protokoll. 188
15.1 MQTT via Thingspeak 190
Kapitel 16 • Daten via ThingSpeak ins Internet senden 195
16.1 Regen oder Sturm? Weltweit verfügbare virtuelle Wetterstation. 195
16.2 Grafische Darstellung von Daten in ThingSpeak. 199
16.3 Daten für das Smartphone mit der ThingView-App. 200
16.4 Gegen unerwünschte Besucher: Optische Raumüberwachung 201
Kapitel 17 • Mikropowertechniken und Sleep-Modi. 204
17.1 Stromsparen schont die Umwelt: LowPower-Techniken 204
17.2 Abschalten unnötiger Verbraucher 205
17.3 Wetterstation im Akku- oder Solarbetrieb 206
Kapitel 18 • Für effiziente Kommunikation: Bussysteme. 207
18.1 Grundlagen und Anwendungen des I^2C - Busses 207
18.2 Der SPI-Bus 213
18.3 Die Mitglieder der SPI-Familie 216
18.4 Ansteuern von SD- und µSD-Karten über SPI 216
Kapitel 19 • Aufbau von Schaltungen: Bauelemente und Breadboards. 218
19.1 Breadboards 219
19.2 Drahtbrücken und Jumper-Kabel 220
19.3 Widerstände 222
19.4 Leuchtdioden (LEDs) 223
19.5 Kondensatoren und Elkos 223
Kapitel 20 • Fehlersuche 225
Kapitel 21 • Bezugsquellen 226
Kapitel 22 • Literatur 227
Kapitel 23 • Abbildungsverzeichnis 228
Kapitel 24 • Materialliste "MicroPython" 232
Index 233

Warnhinweise

1. Die Schaltungen in diesem Buch dürfen nur mit geprüften, doppelt isolierten Sicherheitsnetzgeräten betreiben. Isolationsfehler eines einfachen Netzteils könnten zu lebensgefährlichen Spannungen an nicht isolierten Bauelementen führen.

2. Leistungsstarke LEDs können Augenschäden verursachen. Niemals direkt in eine LED blicken!

3. Autor und Verlag übernehmen keinerlei Haftung für Schäden, die durch den Aufbau der beschriebenen Projekte entstehen.

4. Elektronische Schaltungen können elektromagnetische Störstrahlungen aussenden. Da Verlag und Autor keinen Einfluss auf die technischen Ausführungen des Anwenders haben, ist dieser selbst für die Einhaltung der relevanten Emissionsgrenzwerte verantwortlich.

Programmdownload

Die Programme aus diesem Buch können unter

www.elektor.de

heruntergeladen werden. Sofern ein Programm nicht identisch mit dem im Buch beschriebenen ist, sollte die Version aus dem Download verwendet werden, da es sich dabei um die aktuelle Version handelt.

Kapitel 1 • Einführung

Mit dem ESP32-Chips der Firma Espressif Systems steht eine neue Generation von Mikrocontrollern zur Verfügung, die eine hervorragende Performance, Wi-Fi- und Bluetooth-Funktionalität zu einem konkurrenzlosen Preis bieten. Mit diesen Eigenschaften wurde die Maker-Szene im Sturm erobert. Die verschiedensten Anwendungen und Projekte aus den Bereichen Internet of Things (IoT) und Heimautomatisierung lassen sich damit einfach und kostengünstig umsetzen. Der ESP32 ist genauso leicht programmierbar wie die klassischen Arduino-Boards. Im Vergleich dazu bietet der ESP32 jedoch auch unter anderem:

- größeren Flash und SRAM-Speicher
- wesentlich höhere CPU-Geschwindigkeit
- integriertes Wi-Fi / WLAN
- Bluetooth-Funktionalität,
- mehr GPIO-Pins
- Umfangreiche Schnittstellenfunktionalität
- Analog/Digital-Wandler mit höherer Auflösung
- Digital/Analog-Wandler
- Sicherheits- und Verschlüsselungsfunktionen

Er kann daher als aussichtsreichster Arduino-Nachfolger angesehen werden. Häufig hört man in diesem Zusammenhang auch den Ausdruck "Arduino-Killer".

Das vorliegende Buch führt in die Programmierung der modernen Ein-Chip-Systeme (Systems on Chip - SoCs) ein. Neben den technischen Hintergründen soll dabei vor allem die Programmiersprache Python, insbesondere in ihrer Variante "MicroPython" im Vordergrund stehen. Grundlegende Zusammenhänge aus Elektronik und Elektrotechnik werden nur so weit behandelt, wie es für den Aufbau der Schaltungen und Experimente unerlässlich ist.

Die "Hardware" für den Einstieg kann ohnehin sehr einfach gehalten werden. Zunächst sind nur ein Controllerboard und einige Leuchtdioden sowie geeignete Vorwiderstände erforderlich. Der für die Programmierung des Chips erforderliche PC oder Laptop dürfte in jedem Haushalt vorhanden sein. Die passende Programmierumgebung kann kostenlos aus dem Internet geladen werden. Bei der Arbeit mit einer MicroPython-Programmierumgebung treten dann allerdings schnell die ersten Probleme auf. Eine gute Einführung kann also hervorragende Dienste leiten.

Python hat in den letzten Jahren einen enormen Aufschwung erlebt. Nicht zuletzt haben verschiedene Einplatinensysteme wie der Raspberry Pi zu dessen Bekanntheitsgrad beigetragen. Aber auch in anderen Gebieten wie der künstlichen Intelligenz oder dem Machine Learning hat Python weite Verbreitung gefunden. Es ist daher naheliegend, Python bzw. die Variante MicroPython auch für den Einsatz in SoCs zu verwenden.

Allerdings soll es nicht bei einer reinen Einführung in die Programmiersprache bleiben. Vielmehr werden die erlernten Programmierkenntnisse zusammen mit elektronischen Schaltungen in die Praxis umgesetzt. Die vollständig beschriebenen Projekte sind alle für den

Einsatz in Labor oder Alltag geeignet. Neben dem eigentlichen Lerneffekt steht also auch die Freude am Zusammenbau kompletter und nützlicher Geräte im Vordergrund. Durch die Verwendung von Laborsteckboards können Schaltungen aller Art mit geringem Aufwand realisiert werden. Das Erproben und Austesten der Anwendungen wird damit zum lehrreichen Vergnügen.

Durch die verschiedenen Anwendungen wie Wetterstationen, Digitalvoltmeter oder Funktionsgeneratoren sind die vorgestellten Projekte auch für Praktika oder Fach- und Studienarbeiten in den Naturwissenschaften bzw. im Natur- und Technikunterricht bestens geeignet.

1.1 Python, C oder Arduino?

"Hüte dich vor dem schnellen, dem einfachen Weg!
Er führt auf die dunkle Seite der Macht."

Meister Yoda (Star Wars)

Für Einsteiger ist die Arduino-Programmierumgebung eine der einfachsten Möglichkeiten zur Programmierung des ESP32. Hinter dieser Oberfläche steht die Arduino-Version von C bzw. C++. Die beiden Programmiersprachen-Varianten waren jahrelang eine beliebte Variante für die Entwicklung von "Embedded Systems" also eingebetteter Systeme. Die Arduino-Version von C machte den Einstieg noch einfacher. Darüber hinaus entwickelte sich hierzu eine der größten Technologie-Communities der Welt. Mit neuen Bibliotheken, Software-Fixes und Board-Support konnten Probleme meist rasch gelöst werden. Allerdings ist die Einschränkung, dass Arduino-C ausschließlich in der zugehörigen Umgebung funktioniert, nicht ganz unwesentlich. Insbesondere für die Entwicklung von umfangreicheren Projekten nützliche und wichtige Funktionen fehlen. Daher blieb Arduino-C überwiegend auf Hobby- und Einsteigerprojekte beschränkt.

MicroPython ist relativ neu. Die Community seiner Anwender wächst und es werden zunehmend mehr Plattformen unterstützt. MicroPython ist im Wesentlichen eine schlanke Version von Python, einer der beliebtesten Programmiersprachen der Welt. Daher können spezifische Probleme nicht nur in MicroPython-Communities behandelt werden. Vielmehr tragen auch allgemeine Python-Foren immer häufiger zur Lösungsfindung von MicroPython-Themen bei.

Neben der Community-Unterstützung verfügt MicroPython auch über bestimmte Funktionen, die es deutlich über die Klasse des Arduinos heben. Eines dieser Merkmale ist die sogenannte REPL-Funktion. REPL steht für "Read-Evaluate-Print Loop". Damit sind Programme und Codeabschnitte schnell ausführbar. Das Kompilieren oder Hochladen entfällt. Auf diese Weise können Teile eines Codes bereits während der Entwicklung rasch und effizient ausgetestet werden.

MicroPython enthält eine sehr kompakte Implementierung des Python-Interpreters. Dieser erfordert lediglich 256 kB Flash-Speicher und 16 kB RAM. Trotzdem ist der Interpreter auf maximale Kompatibilität zum Standard-Python ausgelegt. Syntax und Sprachumfang ent-

sprechen weitgehend der Python-Version 3.4. Dementsprechend sollten sich routinierte Python-Programmierer sofort zurechtfinden. Zusätzlich sind ein paar Sprachelemente jenseits des Standards definiert, die den Speicherbedarf gering halten und die Ausführungsgeschwindigkeit steigern.

1.2 Voraussetzungen

Folgende Voraussetzungen sollten für die erfolgreiche Arbeit mit dem vorliegenden Buch erfüllt sein:

- Sicherer Umgang mit dem Betriebssystem Windows
- Grundkenntnisse in einer beliebigen Programmiersprache wie C, Java o. ä.
- Grundlegendes Wissen zum Themenkreis "Elektronik", insbesondere im Bereich "Strom – Spannung - Widerstand" wird vorausgesetzt.

Für spezielle Kenntnisse im Bereich Elektronik sei hier auf die umfangreiche Fachliteratur verwiesen. Hinweise dazu finden sich im Literaturverzeichnis. Die Hardware-Aufbauten wurden bewusst einfach gehalten, da der Fokus auf der Programmierung mit MicroPython liegen soll. Dennoch sind verschiedene Bauelemente und Komponenten erforderlich. Erläuterungen dazu finden sich in den einzelnen Kapiteln, in welchen die Komponenten zuerst benutzt werden. Zudem werden in den letzten Abschnitten des Buches einige grundlegende Bauelemente wie Widerstände oder Leuchtdioden, erläutert. Hier kann man nachschlagen, falls bei einzelnen Komponenten Unklarheiten bestehen sollten.

Darüber hinaus ist

- ein PC oder Laptop mit USB-Schnittstelle
- das Betriebssystem Windows 10
- ein Internetzugang

erforderlich. Zudem ist es sinnvoll, zwischen Rechner und Controller einen aktiven USB-Hub einzusetzen. Dieser hat den Vorteil, dass er einen gewissen Schutz für den PC gewährleistet. Der Hub sollte über eine eigene 5-V-Stromversorgung über ein separates Netzteil verfügen. Dann ist der PC bzw. der Laptop vor Kurzschlüssen hinter dem Hub am besten geschützt, da das "Durchschlagen" eines Kurzschlusses über einen aktiven Hub hinweg bis zum USB-Port des Rechners sehr unwahrscheinlich ist.

Kapitel 2 • Vielfältige Auswahl: ESP-Boards

Der ESP32 ist ein moderner und äußerst leistungsfähiger Mikrocontroller. Neben einer hohen Taktfrequenz und den umfangreichen internen Funktionseinheiten kann der Chip mit integriertem Wi-Fi und Bluetooth aufwarten. Der Controller wurde von Espressif Systems, einem chinesischen Unternehmen mit Sitz in Shanghai, entwickelt und erfreut sich zunehmender Beliebtheit. Die Leistungsmerkmale des ESP stellen die bekannten Arduino-Boards hinsichtlich Preis und Leistungsfähigkeit weit in den Schatten.

Da der ESP32 nur als SMD-Chip verfügbar ist, wird ein sogenanntes Break-out- oder Entwicklungsboard benötigt, wenn man den Controller auch im nicht-professionellem Umfeld einsetzen möchte. Inzwischen ist davon eine nahezu unüberschaubare Vielfalt von verschiedenen Versionen auf dem Markt erhältlich. Die bekanntesten Varianten sind:

Board	Pins	Taster	LiPo charger
ESP32-PICO-KIT	34	EN und BOOT	nein
JOY-iT NodeMCU	30	EN und BOOT	nein
ESP32 DEV KIT DOIT	30/36	EN und BOOT	nein
Adafruit HUZZAH32	28	RESET	ja
ESP32 Thing	40	EN und BOOT	ja
LOLIN32	40	EN und BOOT	nein
Node-ESP-Board	38	EN und BOOT	nein

Die folgende Abbildung zeigt drei verschiedene Ausführungen:

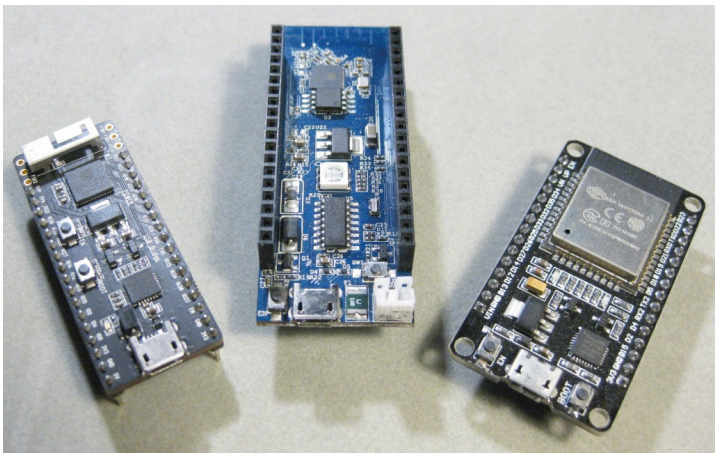

Abbildung 2.1: ESP32-Boards (PICO-KIT, Node-ESP und NodeMCU)

Diese Boards verfügen über die notwendigen Voraussetzungen, um den ESP-Controller in einem lötfreien Steckbrett in Betrieb zu nehmen. Häufig sind neben dem Controller noch weitere Komponenten wie Taster, Li-Ionen-Akku-Lader oder verschiedene LEDs auf den

Boards vorhanden. So können auch ohne externe Beschaltung erste Tests und Experimente durchgeführt werden.

Im folgenden Abschnitt sind die wichtigsten Daten zum ESP32 zusammengefasst. Die Übersicht soll lediglich einen ersten Einblick verschaffen. Ein tieferes Verständnis der einzelnen Leistungsmerkmale und Funktionen wird dann in den jeweiligen Fachabschnitten ermöglicht.

Prozessor:	160 / 240-MHz-Zweikern-Mikroprozessor Tensilica LX6	
Speicher:	520 KByte SRAM / 16 MByte Flashspeicher	
Spannungsversorgung:	2,2 bis 3,6 V	
Stromaufnahme:	Standard:	ca. 50–70 mA
	Wi-Fi-Betrieb:	ca. 80–170 mA
	Tiefschlafmodus:	ca. 2,5 µA
Umgebungstemperatur:	- 40 °C bis + 125 °C	
Ein-/Ausgänge (GPIOs):	32 allgemein verwendbare Ports mit PWM-Funktion und Timer- Logik	
Wi-Fi:	802.11 b/g/n/e/i	
Netzwerkdurchsatz:	135 MBit/s (via UDP-Protokoll)	
Empfängerempfindlichkeit:	- 98 dBm	
Bluetooth:	V4.2 BR/EDR and BLE	
Bluetooth-Funktionalität:	Classic und Bluetooth Low Energy (mit integrierter Antenne)	
Sensoren:	Hallsensor 10 x kapazitiver Touchsensor	
Schnittstellen:	3x UART mit Flusssteuerung über Hardware 3x SPI-Schnittstellen CAN-Bus-2.0-Controller 2x I^2S- und 2xI^2C-Schnittstellen, 18 analoge Eingänge mit 12-Bit-Analog-Digital-Wandler 2 analoge Ausgänge mit 10-Bit-Digital-Analog-Wandler Infrarot (IR) (TX/RX) Motor PWM LED-PWM mit bis zu 16 Kanälen Interface für externen SPI-Flashspeicher für bis zu 16 MByte SD-Karten-Hardware	
Sicherheit:	Wi-Fi: WFA, WPA/WPA2 und WAPI Secure Boot Flash Encryption Cryptographic Hardware Acceleration Elliptic Curve Cryptography (ECC) Zufallszahlengenerator (Random Number Generator – RNG)	

Für die Anwendungsbeispiele im vorliegenden Buch wird meist das ESP32-PICO-KIT verwendet. Alternativ kann auch das ESP32 DEV KIT oder ein anderes Board eingesetzt werden. Welche Variante zum Einsatz kommt, wird jeweils explizit erläutert. Grundsätzlich gilt jedoch, dass die verschiedenen Boards weitgehend kompatibel sind. Sie unterscheiden sich

hauptsächlich in der Größe und der Pin-Anordnung bzw. -Reihenfolge. Die folgende Abbildung zeigt das PICO-KIT mit seinen Funktionseinheiten und Anschlüssen:

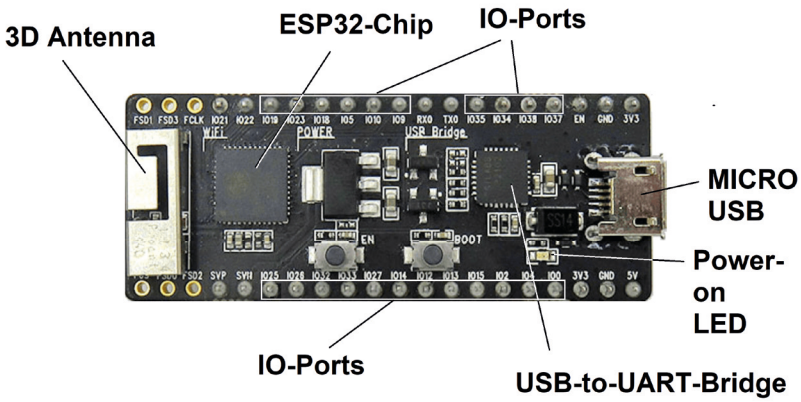

Abbildung 2.2: ESP32-PICO-KIT-Board

Die Breakout-Platinen sind bestens als Experimentier- und Entwicklungsboards geeignet. Über die Port-Anschlüsse können die elektronischen Bauelemente wie LEDs, Temperatursensoren oder sogar kleinere Aktuatoren wie etwa Modellbauservos direkt angeschlossen werden. Das Pico-Kit-Board verfügt unter anderem über die folgenden Leistungsmerkmale:

- ESP-Controller mit zwei 32-bit-Kernen
- Schnelles Wi-Fi- bzw. WLAN Interface (bis zu 150 MBit/s)
- ADC & DAC Funktionalität
- Touch Sensor Einheit
- Host Controller für SD/SDIO/MMC
- SDIO/SPI Controller
- EMAC und PWM-Einheit für die Steuerung von LEDs und Motoren
- UART-, SPI-, I²C- und I²S-Schnittstellen
- Infrarot Fernbedienungscontroller
- GPIO-Interface
- Bluetooth/Bluetooth LE (4.2)
- USB-to-Serial-Chip für den Zugriff über die USB-Schnittstelle

2.1 Inbetriebnahme und Funktionstest

Sobald ein Board zur Verfügung steht, sollte man es einem ersten Funktionstest unterziehen. Hierzu wird ein USB-Kabel mit einem PC oder Laptop und der Micro-USB-Buchse des Boards verbunden. Soweit vorhanden, sollte bereits jetzt ein USB-Hub dazwischen geschaltet werden.

Bei den meisten Boards leuchtet eine LED auf, sobald die Verbindung mit einen stromführenden USB-Port hergestellt wird. Falls diese sogenannte "Power-on"-LED wider Erwarten nicht leuchtet, sollte die USB-Verbindung unverzüglich getrennt werden. Auf diese Weise kann verhindert werden, dass ein eventuell vorhandener Kurzschluss größeren Schaden

anrichtet. Für die anschließende Fehlersuche finden sich hilfreiche Hinweise im entsprechenden Kapitel am Ende des Buchs.

ESP-Boards sollten stets in einem lötfreien Steckbrett betrieben werden (siehe Abbildung 2.3). Falls doch einmal ohne Breadboard gearbeitet wird, ist darauf zu achten, dass die verwendete Unterlage nicht leitfähig ist, da es sonst zu Kurzschlüssen zwischen den Pins kommen kann. Neben dem ESP-Board selbst kann dies sogar zur Zerstörung des verwendeten USB-Ausgangs des PCs führen.

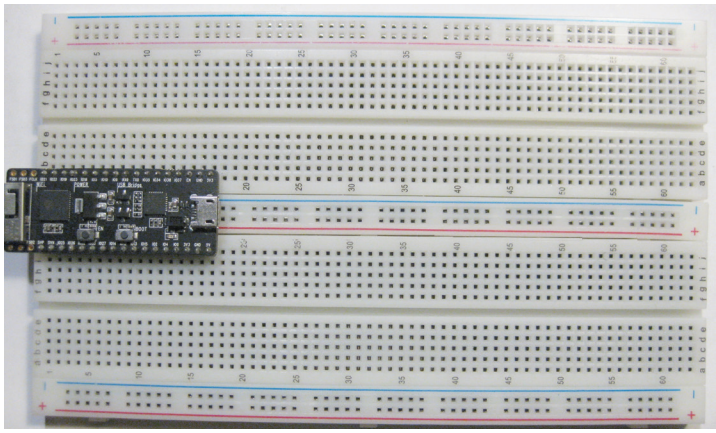

Abbildung 2.3: ESP-Platine im Breadboard

2.2 ESP32 im Akkubetrieb

In den meisten Fällen wird ein ESP-Board über die Micro-USB-Buchse mit Strom versorgt. Da der Controller während der Programm-Erstellung ohnehin mit einem PC oder Laptop verbunden ist, wird keine zusätzliche Spannungsversorgung benötigt.

Falls ein direkter Datenaustausch mit dem PC nicht mehr erforderlich ist, kann das Board auch über ein USB-Netzteil versorgt werden. Dieses sollte mindestens 1.000 mA Strom liefern können, um unerwünschte Spannungseinbrüche zu vermeiden. Zudem ist dann auch noch eine gewisse Reserve vorhanden, um noch einige LEDs, Displays oder Sensoren zu betreiben.

Auch ohne USB-Verbindung kann das Board über Wi-Fi und Bluetooth Daten senden und empfangen. Um eine völlige Unabhängigkeit von Strom- und Datenkabeln zu erreichen, muss das Modul dann nur noch mit Akkus oder Batterien betrieben werden.

Dazu verfügen einige Boards über einen Li-Ionen-Akkuanschluss (siehe Abbildung 2.4). Dort können über den Normstecker passende Zellen direkt angeschlossen werden. Eine interne Spannungsregelung sorgt dann für eine optimale Versorgung der Controllers. Darüber hinaus wird ein angeschlossener Akku geladen, sobald das Board mit einer stromführenden USB-Buchse verbunden wird.

Für diese Anwendung kommen Zellen ab einer Kapazität von etwa 1.500 mAh infrage. Kleinere Akkus unter 300 mAh sollten nicht verwendet werden, da diese unter Umständen vom integrierten Laderegler überlastet werden könnten.

Bei einem typischen Stromverbrauch von ca. 50 mA kann mit der 1.500-mAh-Variante eine Betriebsdauer von etwa 30 Stunden, also etwas mehr als einem Tag erreicht werden. Bei Verwendung von Sleep-Funktionen des Controllers sind sogar noch wesentlich längere Laufzeiten erreichbar.

Einzellige LiPo- oder Lithium-Ionen-Akkus liefern ausreichend Leistung für den ESP32. Ihre Spannung ist jedoch je nach Ladezustand mit 3,7 bis 4,2 V zu hoch für den ESP32. Sie wird daher über einen internen Baustein heruntergeregelt.

Da das Arbeiten mit Li-Ionen-Akkus immer mit einer gewissen Gefahr verbunden ist, darf hier der folgende Hinweis nicht fehlen:

- Lithium-Ionen-Akkus reagieren empfindlich auf falsche Ladeströme oder -spannungen. Unter bestimmten Umständen besteht sogar Brand- oder Explosionsgefahr.
- Jede Anwender ist selbst für seinen Aufbau verantwortlich.
- Verlag oder Autor übernehmen keinerlei Haftung.

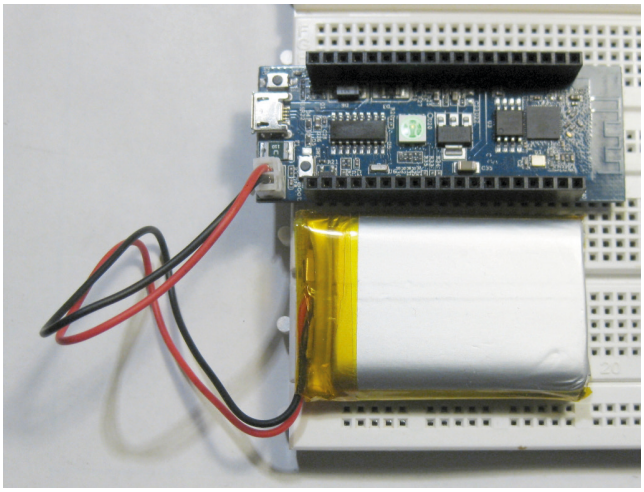

Abbildung 2.4: NodeESP-Board im Akkubetrieb

Kapitel 3 • Programmier- und Entwicklungsumgebungen

Anders als etwa beim Arduino-System stehen für die Arbeit mit MicroPython mehrere Entwicklungsumgebungen (IDEs für engl. Integrated Developing Environment) zur Verfügung. Prinzipiell kann man mit allen IDEs Programme schreiben und auf den Controller laden. Die zwei am weitest verbreiteten Programmierumgebungen sind aktuell

- µPyCraft
- Thonny

Beide verfügen über ihre jeweiligen spezifischen Vor- und Nachteile. Die Unterschiede liegen hauptsächlich in den verschiedenen Verfahren, wie Programmcode für die Anwendungsprojekte entwickelt und verwaltet wird.

Die erste Variante, "µPyCraft" bietet eine vergleichsweise einfache Oberfläche für die MicroPython-Entwicklung auf dem ESP32-Controller. Sie arbeitet mit simplen Grafikelementen und erinnert an textorientierte Betriebssysteme. Dafür ist der Umgang mit den einzelnen Funktionen gut verständlich und die Arbeit mit den verschiedenen Menüs ist leicht erlernbar.
Thonny dagegen verfügt über eine vollständig graphische Oberfläche im Windows-Stil. Die IDE ist unter Makern sehr beliebt, insbesondere auch deshalb, weil sie unter dem Raspbian-Betriebssystem auf dem Raspberry Pi verfügbar ist. Viele RasPi-Anwender sind daher bereits mit Thonny bestens vertraut.

Die IDEs stehen für die wichtigsten Betriebssysteme wie

- Windows PC
- Mac OS X
- Linux Ubuntu

kostenlos im Internet zur Verfügung.

Falls bei der Installation oder Verwendung bei einem der beiden Systeme Probleme auftreten sollten, kann die jeweils andere Version als alternatives Programmiersystem verwendet werden. Welche der beiden Varianten gewählt wird, hängt aber natürlich auch von den persönlichen Neigungen und Gewohnheiten des Anwender ab.

3.1 Installation der uPyCraft-IDE

Vor der Installation von uPyCraft IDE sollte die neueste Version von Python 3.7.X auf dem verwendeten Computer installiert sein. Wenn dies nicht der Fall ist, kann die Installation entsprechend den folgenden Anweisungen erfolgen:

1. Download der Installationsdatei von der Python-Downloadseite unter

   www.python.org/downloads

2. Nach dem Download sollte sich auf dem Rechner eine Datei mit dem Namen

   python-3.7.X.exe

   befinden. Ein Doppelklicken auf die Datei startet die Installation.

3. Option "Add Python 3.7 to PATH" auswählen und auf die Schaltfläche "Install Now" klicken.

4. Der Installationsvorgang ist nach einigen Sekunden mit der Meldung "Setup was successful" abgeschlossen. Danach kann das Fenster geschlossen werden.

Nun kann die uPyCraft IDE für Windows unter

https://github.com/DFRobot/uPyCraft

als uPyCraft_V1.x.exe herunter geladen werden. Nach dem Klick auf diese .exe-Datei öffnet sich die uPyCraft-IDE:

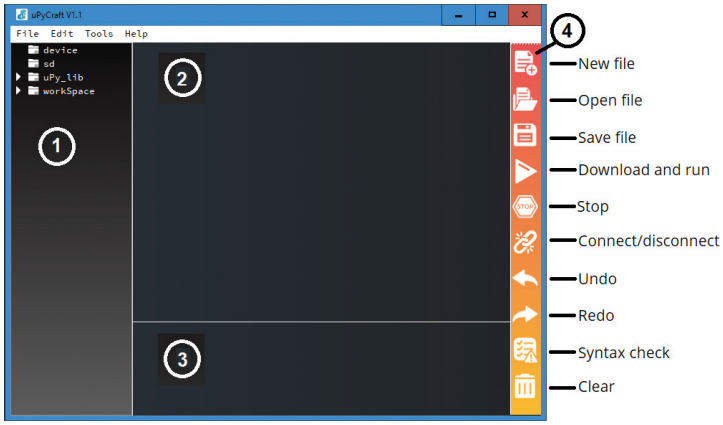

Abbildung 3.1: uPyCraft-IDE

Nachdem die IDE auf dem Computer installiert ist, kann die ESP32-Firmeware auf den Chip geladen werden. Die aktuell Version der MicroPython-Firmware für den ESP32 findet sich unter

http://micropython.org/download#esp32

Dort scrollt man zum Abschnitt "ESP32 modules". Nach dem Anklicken des Links zu "Generic ESP32 module" gelangt man zur Downloadseite der ESP32-BIN-Datei. Diese sieht zum Beispiel so aus:

esp32-idf3-20191220-v1.12.bin

Nun kann man in der uPyCraft-IDE unter

Tools -> Seriell

den ESP32-COM-Port auswählen, hier als COM5:

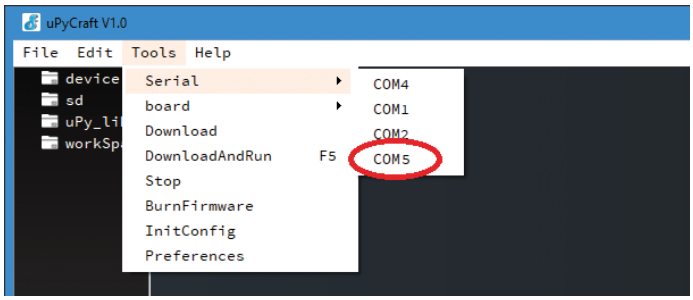

Abbildung 3.2: Auswahl des Ports

Wenn das ESP32-Board an den Computer angeschlossen ist, der ESP32-Port jedoch nicht in der uPyCraft-IDE erscheint, könnte eventuell der passende USB-Treiber fehlen. In diesem Fall muss der Treiber nachinstalliert werden. Ein entsprechender Treiber findet sich unter

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

Danach kann unter

Tools -> Board

die Option "esp32" gewählt werden:

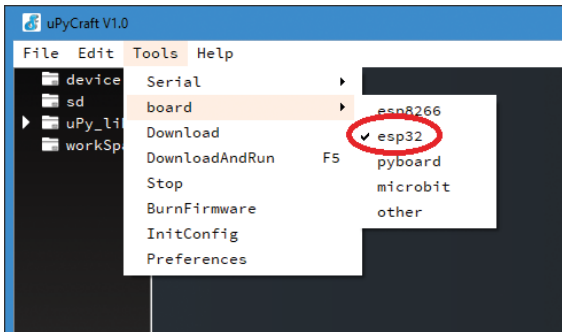

Abbildung 3.3: Auswahl des Boardtyps

Nun kann über

Extras -> Burn Firmware

der MicroPython-Interpreter auf ESP32 geschrieben werden. Die passenden Optionen dazu lauten:

- board: esp32
- burn_addr: 0x1000
- erase_flash: yes
- com: COMX (hier COM5, s. o.)

unter "USERS" wird die heruntergeladene ESP32-BIN-Datei ausgewählt.

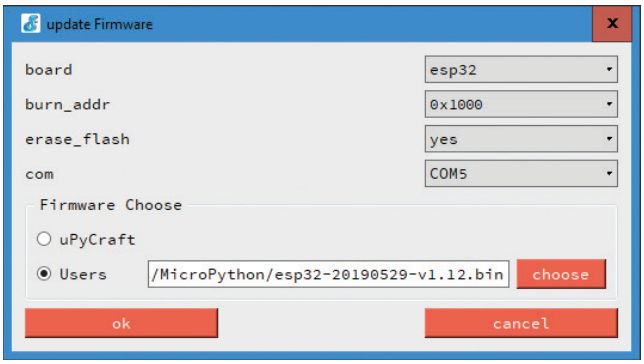

Abbildung 3.4: Die Parameter zum Flashen der Firmware

Sind alle Einstellungen korrekt ausgewählt, muss bei einigen Board-Varianten die Taste "BOOT / FLASH" auf der ESP32-Karte gedrückt werden. Sobald der Vorgang "EraseFlash" beginnt, kann die Taste losgelassen werden. Nach einigen Sekunden sollte die Firmware auf das ESP32-Board geflasht sein. In vielen Fällen beginnt der Download allerdings auch ohne das Drücken der Tasten.

Falls die Anzeige "EraseFlash" nicht startet oder eine Fehlermeldung angezeigt wird, sollten die beschriebenen Schritte wiederholt werden. Auch die Taste "BOOT / FLASH" ist erneut zu drücken, um sicherzustellen, dass der ESP32 in den Flashmodus wechselt.

3.2 MicroPython für den ESP32

Bis auf wenige Ausnahmen sind alle Leistungsmerkmale und Funktionen von Python auch in MicroPython verfügbar. Der größte Unterschied besteht darin, dass die Mikro-Version für den Einsatz auf Ein-Chip-Systemen konzipiert wurde und daher die klassischen, nur für den PC erforderlichen Routinen fehlen.

Aus diesem Grund enthält MicroPython auch nicht die vollständige Standardbibliothek, sondern nur die für Mikrocontroller relevanten Teile. Dafür sind alle für den Zugriff auf die verwendete Hardware erforderlichen Module vorhanden. Mit den entsprechenden Bibliotheken kann man daher sehr einfach auf die GPIO-Pins zugreifen. Speziell für den ESP32 sind außerdem Module zur Unterstützung von Netzwerkverbindungen (WiFi) und Bluetooth verfügbar. Im Speziellen werden u. A. die folgenden Boards unterstützt:

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

Obwohl noch nicht alle Funktionen des ESP-Controllers in MicroPython vollständig verfügbar sind, enthalten die Bibliotheken die wichtigsten Befehle und Routinen. Daher können viele Projekte und Anwendungen reibungslos umgesetzt werden. Zudem schreitet die Implementierung der noch fehlenden Features rasch voran, sodass auch dieser kleine Schönheitsfehler rasch beseitigt sein wird.

Nachdem die MicroPython-Firmware auf dem ESP32 installiert wurde, kann man auch sehr einfach beispielsweise zur Arduino-IDE zurückkehren. Hierzu ist lediglich der neue C-Code mit der IDE auf den Controller zu laden. Ein speziellen Löschverfahren ist nicht erforderlich. Soll anschließend jedoch wieder MicroPython verwendet werden, ist die MicroPython-Firmware neu zu flashen.

3.3 "Hello World" für den Controller

Anders als bei AVR-Controllern, wie sie etwa beim Arduino-System zum Einsatz kommen, kann auf dem ESP32 ein komplettes Dateisystem untergebracht werden. Die ersten Controller-Generationen wurden entweder in Assembler oder in C programmiert. Der Programmcode wurde also in einer Entwicklungsumgebung erstellt und kompiliert. Dann wurde lediglich der fertige "Maschinencode" zum Controller übertragen. Im Speicher des Zielsystems war daher immer nur genau ein Programm vorhanden.

Bei der Programmierung in MicroPython können dagegen mehrere Programme auf dem ESP32-Chip gespeichert werden. Diese können dann direkt vom ebenfalls auf dem System vorhandenen Interpreter abgearbeitet werden. Das Dateisystem kann direkt mit der uPyCraft-IDE verwaltet werden. Es ist daher sinnvoll, dass man sich mit der IDE etwas genauer vertraut macht, bevor das erste Anwendungsprogramm auf den ESP geladen wird. Die Entwicklungsumgebung enthält, ähnlich wie viele andere Programmiertools auch, die folgenden Bestandteile (s. dazu auch Abb. 3.1):

1. Ordner und Dateien
2. Editor
3. MicroPython Shell / Terminal
4. Werkzeuge

Im linken Unterfenster ("Ordner und Dateien") sind im Geräteordner ("device") die Dateien sichtbar, die derzeit auf dem ESP-Board gespeichert sind. Sobald das Board über eine serielle Verbindung mit uPyCraft-IDE verbunden ist, werden beim Öffnen des Geräteordners alle gespeicherten Dateien geladen. Direkt nach der Installation des Python-Interpreters ist hier nur eine "boot.py"-Datei sichtbar. Um den Anwendungscode auszuführen, sollte zusätzlich eine main.py-Datei erstellt werden. Hierzu wird über

file → new

eine neue Datei ("untitled") erstellt. Über das Diskettensymbol im "Tools"-Fenster kann diese Datei lokal unter dem Namen "main.py" auf dem ESP-Chip gespeichert werden.

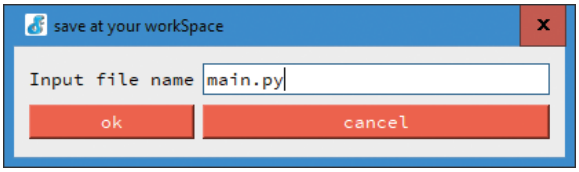

Abbildung 3.5: Erstellen einer neuen Datei "main.py"

Damit stehen die folgenden beiden Files im device-Ordner:

boot.py: wird bei jedem Neustart des Board ausgeführt
main.py: Hauptskript für den Anwendungscode

Unter dem device-Ordner folgt der SD-Ordner. Dieser ist für den Zugriff auf Dateien vorgesehen, welche auf einer SD-Karte gespeichert sind. Einige ESP-32-Boards verfügen über einen SD-Kartensteckplatz. Falls hier eine µSD-Karte eingesteckt wird, erscheinen die auf der Karte vorhandenen Dateien im "sd"-Ordner.

Darunter folgt der uPy_lib-Ordner. Hier werden die integrierten IDE-Bibliotheksdateien angezeigt. Hier finden sich auch bereits direkt nach der Installation des MicroPython-Interpreters verschiedene Dateien. Diese werden als Standard-Libraries zur Verfügung gestellt.

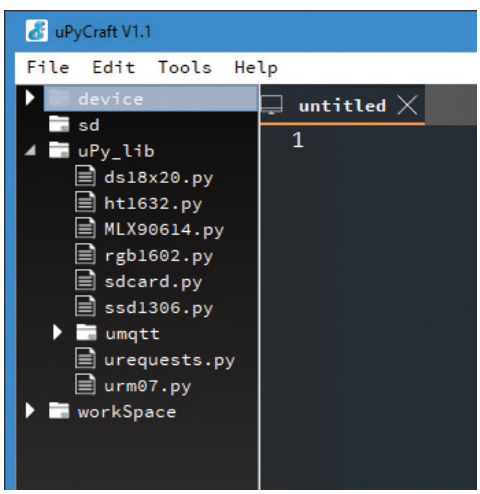

Abbildung 3.6: Standard-Bibliotheken im uPy_lib-Ordner

Der letzte Ordner enthält den sogenannten "workSpace". Hierbei handelt es sich um Verzeichnis zum Speichern von Anwendungsdateien. Diese hier angezeigten Dateien sind auf dem über die Schnittstelle angeschlossenen Computer gespeichert. Hier sollten alle aktiven

Dateien abgelegt werden. Wenn uPycraft zum ersten Mal verwendet wird, empfiehlt es sich daher, ein geeignetes Arbeitsverzeichnis mit dem Namen "workSpace" zu erstellen und dieses für diese dann konsequent für die Arbeit mit dem Controller zu verwenden.

Im Editor-Bereich (2) wird der Code für die .py-Anwendungsprogramme erstellt. Der Editor öffnet für jede Datei eine neue Registerkarte.

Der Abschnitt unter dem Editor-Bereich ist das "MicroPython Shell/Terminal" (3) Alle hier eingegebenen Befehle werden sofort vom ESP-Board ausgeführt. Darüber hinaus zeigt das Terminal auch Informationen über den Status eines laufenden Programms an. Hier erscheinen also eventuelle Syntaxfehler im aktuellen Programm oder Fehlermeldungen beim Hochladen usw.

Mit den Symbolen im Bereich "Werkzeuge" ganz rechts im Hauptfenster (4) können Aufgaben schnell und direkt ausgeführt werden. Die Schaltflächen haben die folgenden Funktionen:

- New File: Erstellt eine neue Datei im Editor
- Open File: Öffnet eine Datei auf dem Computer
- Save File: Speichert eine Datei
- Download and run: Code auf das ESP-Board hochladen und ausführen
- Stop: Beenden der Code-Ausführung. Dies entspricht der Eingabe von STRG + C in der Shell
- Connect/Disconnect: Verbinden oder Trennen der seriellen Schnittstelle. Der serielle Anschluss kann unter Extras -> Serial ausgewählt werden
- Undo: macht die letzte Änderung im Code-Editor rückgängig
- Redo: Letzte Änderung im Code-Editor wiederholen
- Syntaxcheck: Überprüft die Syntax des aktuellen Codes
- Clear: Löscht die Shell / Terminal-Fenstermeldungen

Um mit dem Erstellen eines Programms und dem Ausführen von Code auf dem ESP32 vertraut zu werden, soll im Folgenden ein kurzes Python-Programm entwickelt und ausgeführt werden, welches eine LED blinken lässt. Zunächst muss dazu die Kommunikation mit dem ESP-Board hergestellt werden:

1. Über Tools -> Board das aktuelle Board auswählen
2. In Tools -> Port den COM-Port auswählen, mit dem das ESP-Board verbunden ist
3. Die Connect-Taste stellt dann die serielle Kommunikation mit dem ESP32-Modul her
4. Nach einer erfolgreichen Verbindung mit dem Board wird ">>>" im Shell-Fenster angezeigt.

Nun kann ein Print-Befehl eingeben werden, um zu testen, ob die Kommunikation korrekt funktioniert:

```
>>> print (‚Test')
```

Darauf erscheint die Antwort

```
Test
>>>
```

im Terminal-Fenster. Wenn die Meldung ausgegeben wird, ist alles ok. Andernfalls muss geprüft werden, ob die serielle Kommunikation mit dem Board hergestellt ist und ob die MicroPython-Firmware erfolgreich auf das Board geflasht wurde.

Jetzt kann ein LED-Blink-Skript erstellt werden. Hierzu sind die folgende Schritte erforderlich:

1. Das folgende Programm wird in das Editor-Fenster der weiter oben neu erstellten Datei main.py eingegeben:

```
from machine import Pin
from time import sleep
led = Pin(2, Pin.OUT)
while True:
  led.value(not led.value())
  sleep(1)
```

2. Durch anklicken der Schaltfläche "Stop" kann ein eventuell noch laufendes Skript angehalten werden

3. Mit einem Klick auf die Schaltfläche "Download And Run" wird das Skript auf den Controller geschrieben

4. Im Shell-Fenster sollte nun die Meldung "download ok" angezeigt werden.

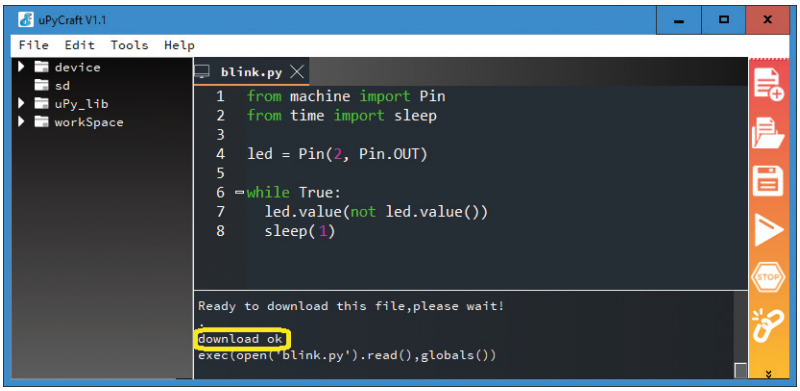

Abbildung 3.7: *Erfolgreicher Download des ersten Programms*