ESP32: Einstieg in MicroPython

Python – der neue Star am Programmiersprachen-Himmel?

Python ist inzwischen eine der am häufigsten verwendeten, einfachsten und am leichtesten zu erlernenden Programmiersprachen. Das Aufkommen von MicroPython macht die Programmierung von Mikrocontroller-Systemen sehr einfach und unkompliziert. Damit ist die Programmiersprache auch für Einsteiger hervorragend geeignet.

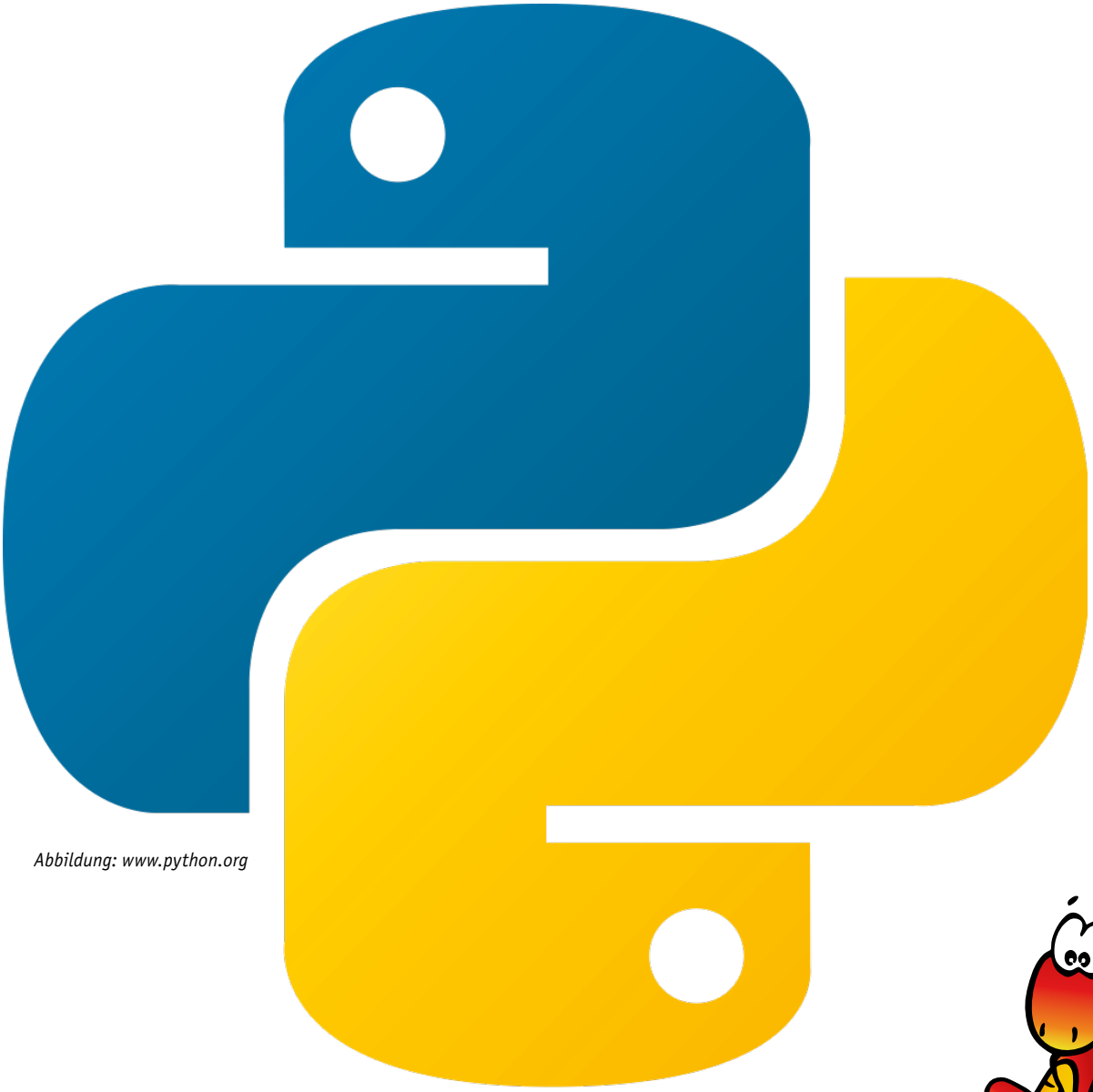

Abbildung: www.python.org

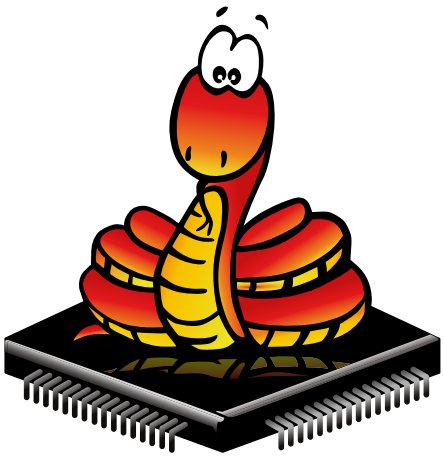

Abbildung: User „Neon22" auf https://github.com

Einfacher Einstieg

Ein Ziel von MicroPython [1] ist es, die Programmierung digitaler Elektronik so einfach wie möglich zu gestalten. Die Sprache soll von einem möglichst großen Anwenderkreis verwendet und nutzbringend eingesetzt werden können. MicroPython wird von Hobbyanwendern und Pädagogen verwendet und kommt sowohl im wissenschaftlichen Bereich als auch bei professionellen Entwicklern zum Einsatz. Bei den Schwergewichten der IT-Branche wie etwa Google zählt Python seit Langem zu den bevorzugten Sprachen – aber auch viele britische Schüler lernen beispielsweise MicroPython mit dem BBC micro:bit.

Durch vielfältige Zusatzmodule wie SciKit [2], NumPy [3] oder SciPy [4] stehen umfangreiche Möglichkeiten zur Verfügung, die von wissenschaftlicher Datenanalyse bis hin zu Machine-Learning und künstlicher Intelligenz reichen. Durch die einfache Syntax ist Python dennoch einsteigerfreundlich und leicht zu verstehen. So ist der Programmcode, um eine LED zum Blinken zu bringen, auch ohne Programmierkenntnisse leicht verständlich:

```
from machine import Pin
from time import sleep

led = Pin(2, Pin.OUT)

while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

Ein besonderes Merkmal von MicroPython ist die Ausstattung mit einer interaktiven REPL-Konsole (**R**ead-**E**valuate-**P**rint-**L**oop). Befehle sind damit direkt auf dem Zielsystem ausführbar. Auf diese Weise kann ein Code schnell getestet werden, ohne dass nach jeder Programmänderung neu kompiliert oder Programme erneut hochgeladen werden müssen.

(Micro)Python vs. C++

Im direkten Vergleich von Python mit C/C++ wird inzwischen immer häufiger Python der Vorzug gegeben. So belegt Python in den Ranglisten der beliebtesten Programmiersprachen häufig den ersten Platz, während C/C++ auf die hinteren Ränge abgerutscht ist. Gründe für diese Beliebtheit sind unter anderem:

- Einsteigerfreundlichkeit
- eine Fülle von Tutorials und Beispielcodes für Python
- umfangreiche Bibliotheken (Libraries)
- aktive Foren

Insbesondere durch die aktive Community finden auch Einsteiger sehr schnell Antworten auf ihre Fragen. Dies ist ein wichtiger Vorteil gegenüber anderen Sprachen, die weniger beliebt sind oder von ihren Benutzern nicht so ausführlich im Internet diskutiert werden.

Unterschiede zwischen Python und C/C++

Sowohl Python als auch C/C++ wurden als universelle Programmiersprachen konzipiert. Dennoch unterscheiden sie sich in vielerlei Hinsicht voneinander. Bei C/C++ muss der Programmcode zunächst in eine für Maschinen verständliche Version übersetzt werden (Kompilierung), die dann vom Prozessor abgearbeitet wird.

Python dagegen wird interpretiert. Das bedeutet, dass der ursprüngliche Programmcode direkt vom Zielprozessor abgearbeitet wird. Eine Kompilierung ist daher nicht erforderlich. Python bietet daher die Möglichkeit, ein einmal geschriebenes Programm auf den verschiedensten Systemen auszuführen. Dazu muss lediglich ein entsprechender Interpreter installiert sein.

Python-Code kann also sowohl auf klassischen Computern unter Windows oder Linux als auch auf kleineren Einplatinensystemen wie dem Raspberry Pi ausgeführt werden. Vor allem die Verwendung auf dem populären Raspberry Pi hat mit zur zunehmenden Beliebtheit von Python beigetragen.

Mit neuen leistungsstarken Controllern wie beispielsweise dem ESP32 ist es nun sogar möglich geworden, Python auch in diesem Bereich effizient und komfortabel einzusetzen. Zu diesem Zweck wurde sogar eine eigene Variante, das sogenannte MicroPython entwickelt.

Python – Pro und Contra

Python ist bekannt für seine Einfachheit und die ausgezeichnete Lesbarkeit des Programmcodes. Fest integrierte Konstrukte sorgen dafür, dass Programme sowohl im kleinen als auch im großen Maßstab problemlos entwickelt werden können. Python ist damit hervorragend skalierbar. Die mögliche Kapselung von Daten und Programmcode in übersichtlichen, wiederverwendbaren Modulen, sogenannten „Objekten", macht Python zu einer vollständig objektorientierten Programmiersprache. C++ wird heutzutage im Allgemeinen vor allem in der hardwarenahen Programmierung verwendet. Klassische Python-Varianten waren dazu bislang nicht gut geeignet. Mit MicroPython wird diese Lücke nun geschlossen.

C++ umfasst neben Clientanwendungen auch leistungsstarke Serverapplikationen sowie Gerätetreiber und eingebettete Treiberkomponenten. Der Anwendungsbereich erstreckt sich also von der Systemsoftware bis hin zur Anwendungsprogrammierung. Da Python im Vergleich zu C eine noch relativ junge Programmiersprache ist, hat es noch nicht diese universelle Verbreitung in allen Teilbereichen der Informationstechnik gefunden. Allerdings zeigt sich, dass Python auf praktisch allen Gebieten im Vormarsch ist.

Python wird mit einer Vielzahl integrierter Standardbibliotheken geliefert. Diese Funktionen machen es zu einer Sprache mit hoher Benutzerfreundlichkeit. Interpreter für Python sind für praktisch alle wichtigen Betriebssysteme verfügbar

Für den Einsteiger hat Python einige Vorteile zu bieten. So kann eine Variable direkt ohne Deklaration verwendet werden. In den meisten anderen Sprachen muss der Datentyp deklariert werden, bevor die Variable verwendbar ist. Zudem verfügt Python über eine integrierte Speicherbereinigung und einen dynamischen Zuweisungsprozess, der eine effiziente Speicherverwaltung ermöglicht. Man muss daher kaum auf Zeiger (engl. pointer) oder ähnliche, für den Einsteiger meist schwer verständliche Konstrukte zurückgreifen.

Die Tabelle 1 fasst die Vor- und Nachteile der beiden in der Controller-Programmierung vorherrschenden Sprachen zusammen. Der Vollständigkeit halber wurde auch noch die Arduino-IDE mit aufgenommen, da diese eine sehr beliebte Alternative zur Programmierung mit Python darstellt.

MicroPython für den ESP32

MicroPython (µP) ist eine spezielle Implementierung von Python 3 für Mikrocontroller und Embedded Sys-

tems. Da die Micro-Version dem klassischen Python, das auf PCs und Laptops benutzt wird, sehr ähnlich ist, sollten routinierte Programmierer mit dem Umstieg keine Probleme haben.

Um mit µP auf einem Controller arbeiten zu können, muss dort zunächst ein Interpreter installiert werden, da Python ja nicht kompiliert wird. In diesem Abschnitt wird erläutert, wie die MicroPython-Firmware auf den ESP32 geladen werden kann. Mit der hier verwendeten µPyCraft-IDE (Integrated Development Environment – integrierte Entwicklungsumgebung [5]) sind die meisten ESP-basierten Boards programmierbar. Nach dem Durcharbeiten dieses Abschnittes sind bereits viele Funktionen des ESP-Controllers nutzbar. Insbesondere können die einzelnen Ports aktiviert und damit z. B. LEDs angesteuert werden.

Bis auf wenige Ausnahmen sind alle Leistungsmerkmale und Funktionen von Python auch in µP verfügbar. Der größte Unterschied besteht darin, dass die Micro-Version für den Einsatz auf Mikrocontrollersystemen 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 auch auf die GPIOs zugreifen. Speziell für den ESP32 sind außerdem Module zur Unterstützung von Netzwerkverbindungen WiFi und Bluetooth verfügbar.

Folgende Boards werden unterstützt:

- ESP32
- ESP8266
- PyBoard
- Teensy 3.X
- WiPy – Pycom
- andere ESP32-basierte Boards

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 problemlos umgesetzt werden. Zudem schreitet die Implementierung der noch fehlenden Features rasch voran, sodass auch dieser kleine Schönheitsfehler in absehbarer Zeit beseitigt sein wird.

Auf PC-Seite muss zunächst die µPyCraft-IDE installiert werden. Diese ist für die wichtigsten Betriebssysteme wie

- Windows PC
- Mac OS X
- Linux Ubuntu

verfügbar.

Vor- und Nachteile der in der Controller-Programmierung vorherrschenden Sprachen

Tabelle 1

	Python	Arduino-IDE	C /C++
Aktuelle Version	3.7.3 und 2.7.16	1.8.9	z. B. GCC 9.2
Verfügbarkeit	Windows, Linux, Mac OS, AIX, IBM i, iOS, z/OS, Solaris, VMS	Windows, Linux, Mac OS	Windows, Linux, Mac OS u. a.
Einsteigerfreundlichkeit	Hoch, viele Tutorials Ausführliche Dokumentationen IDE verfügbar	Sehr gut Sehr einfache und intuitiv zu bedienende IDE	Gering, da komplexe Toolchain erforderlich (aufwendige IDE, Linker etc.)
Syntax	Klar und einfach lesbar Keine zusätzlichen Sonderzeichen wie Semikolons und geschweifte Klammern etc. Funktionale Leerzeichen sorgen für konsistente Einrückungen Verwendung natürlicher englischer Wörter wie „und" und „oder", und damit weniger undurchsichtige Symbole	Eng an C angelehnt Vereinfacht durch spezielle Funktionen wie digitalWrite(), loop etc.	Komplexe Syntax Verwendung von Sonderzeichen (++, >>, || etc.), dadurch für Einsteiger schwer lesbarer Code
Bibliotheken und Libraries	Große Standardbibliothek Umfangreiche Applikations-Libraries frei verfügbar	Umfangreiche Applikations-Libraries frei verfügbar	Standardbibliothek verfügbar Professionelle Libraries z. T. kostenpflichtig
Community	Aktive und hilfreiche Communities stehen zur Verfügung	Sehr viele Communities, allerdings meist nur im nicht professionellen Bereich	Communities oft auf professionelle Anwendungen beschränkt
Anwendungsbandbreite	Professioneller und nicht-professioneller Einsatz Z. B. eine der Hauptsprachen bei Google Interfaces mit anderen Sprachen wie C	Überwiegend Hobby und Maker-Szene Für professionelle Projekte weniger geeignet	Industrielle und professionelle Anwendungen
Zukunftssicherheit	Kaum Unterstützung für Parallelität Version 2.x immer noch aktiv, da viele Libraries vorhanden Für zukunftssichere Projekte mit Version 3.x starten	Eingeschränkt, da überwiegend im Hobbybereich eingesetzt	Unsicher Langfristig abnehmende Tendenz
Programmstruktur	Python-Programme sind im Vergleich zu Code in C++ meist deutlich kürzer Schnelles Prototyping und effiziente Codierung möglich	Einfach	Komplex Vergleichsweise langer Code
Ablaufgeschwindigkeit	Python ist langsamer als C++	Deutlich langsamer als C++, aufgrund der aufwendigen Zusatzfunktionen	Sehr schnell und effizient
Installation	Vergleichsweise aufwendige Installation unter Windows	Sehr einfach	Aufwendige und komplexe Installation

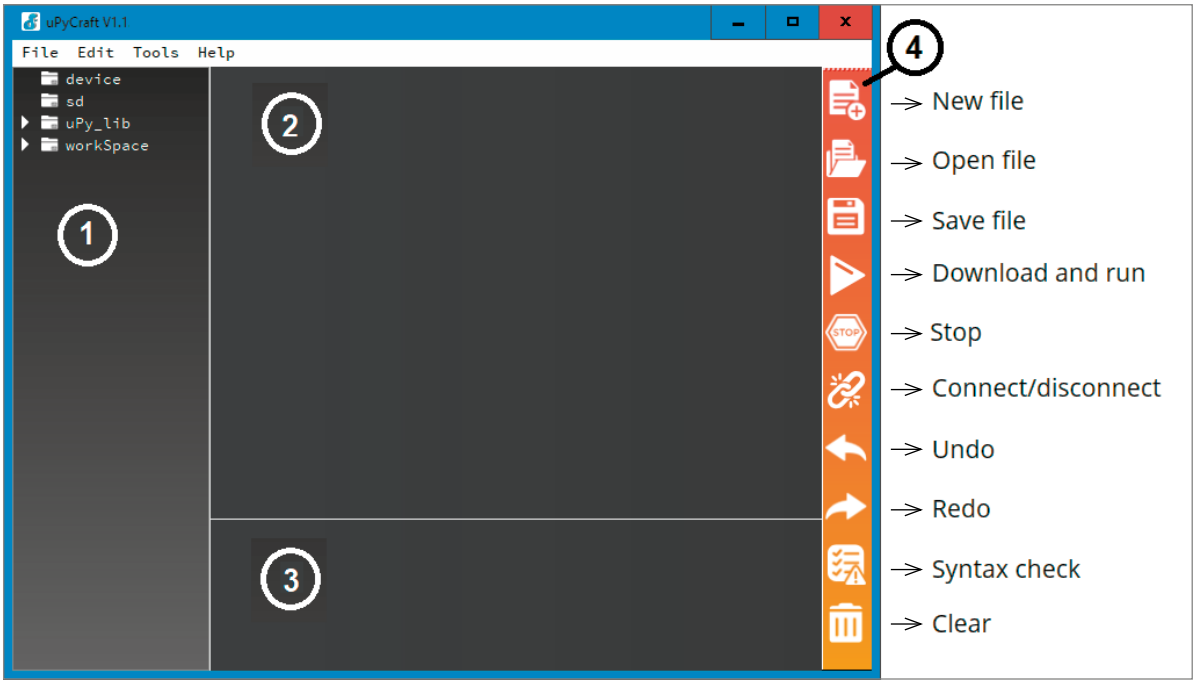

Bild 1: µPyCraft-IDE (Beschreibung der Fenster siehe Text)

Falls es bei der Installation oder Verwendung von µPyCraft Probleme geben sollte, steht mit der Thonny-IDE [6] eine gute Alternative zu Verfügung. Im Folgenden wird jedoch die PyCraft-Version verwendet, da diese bereits weit verbreitet ist. Zudem ist sie einfach und intuitiv zu bedienen und bestens für den Einsatz von ESP-Boards geeignet. Welche der beiden Varianten gewählt wird, bleibt aber letztendlich dem Anwender überlassen.

Der MicroPython-Interpreter ist nicht standardmäßig auf dem ESP32 verfügbar. Daher muss die Anwendung zunächst auf das Board geflasht werden. Hierfür kann die auf einem PC oder Laptop installierte uPyCraft-IDE genutzt werden. Nachdem die MicroPython-Firmware auf Ihrem ESP32 installiert wurde, kann man auch sehr einfach beispielsweise zur Arduino-IDE zurückkehren. Hierzu ist lediglich der neue C-Code mit der Arduino-IDE auf den Controller zu laden. Ein spezielles Löschverfahren ist nicht erforderlich. Soll anschließend jedoch wieder MicroPython verwendet werden, ist die MicroPython-Firmware neu zu flashen.

Vor der Installation von µPyCraft-IDE sollte die neueste Version von Python 3.7.X auf dem verwendeten Computer installiert sein. Wenn dies nicht der Fall ist, kann dies mit den folgenden Anweisungen nachgeholt werden:

1. Herunterladen der Installationsdatei von der Python-Downloadseite unter **www.python.org/downloads**
2. Nach dem Download sollte sich auf dem Computer 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 sollte nach einigen Sekunden mit der Meldung „Setup was successful" abgeschlossen sein. Danach kann das Fenster geschlossen werden.

Nun kann die µPyCraft-IDE für Windows oder auch ein anderes Betriebssystem unter

http://docs.dfrobot.com/upycraft/

als uPyCraft_V1.x.exe heruntergeladen werden.

Nach dem Klick auf die .exe-datei öffnet sich die µPyCraft-IDE (Bild 1).

Tipp: Unter Umständen erscheint beim Start des Programms als Fehlermeldung, dass die Schrift SourceCodePro nicht vorhanden ist. Diese kann mit einer Internet-Suche nach SourceCodePro-Fonts gefunden und heruntergeladen werden. Unter Windows muss dann die Datei *SourceCodePro-Regular.ttf* in *SourceCodePro.ttf* umbenannt und per Rechtsklick **für alle Benutzer** installiert werden. Dann verschwindet der Hinweis auf die fehlende Schrift beim nächsten Start der µPyCraft-IDE.

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

http://micropython.org/download#esp32

Dort scrollt man zum Abschnitt „ESP32". Der Link zum Herunterladen der aktuellen ESP32-Datei sieht zum Beispiel so aus:

esp32-20190529-v1.11.bin

Tipp: Im Folgenden arbeiten wir mit der JOY-iT Entwicklungsplatine NodeMCU mit ESP32 (Bild 2, [7]) – andere Boards mit dem ESP32 sollten aber ebenfalls in der beschriebenen Form arbeiten. Gegebenenfalls muss hier aber der Pin, an der die LED angeschlossen ist, geändert werden.

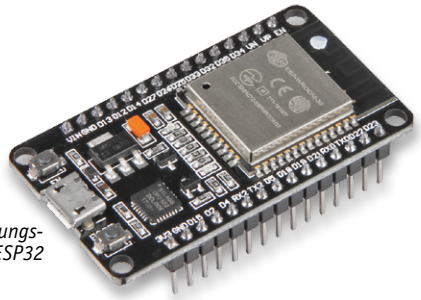

Bild 2: JOY-iT Entwicklungsplatine NodeMCU mit ESP32

Nun kann man in der µPyCraft-IDE unter
Tools → Serial
den ESP32-COM-Port auswählen, hier z. B. als COM12:

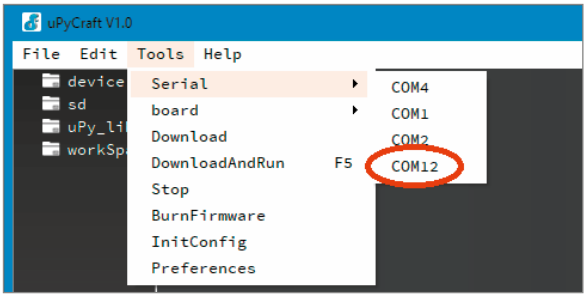

Bild 3: Auswahl des Ports

Wenn das ESP32-Board an den Computer angeschlossen ist, der ESP32-Port jedoch nicht in der µPyCraft-IDE erscheint, könnte eventuell der passende USB-Treiber fehlen. In diesem Fall muss dieser 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:

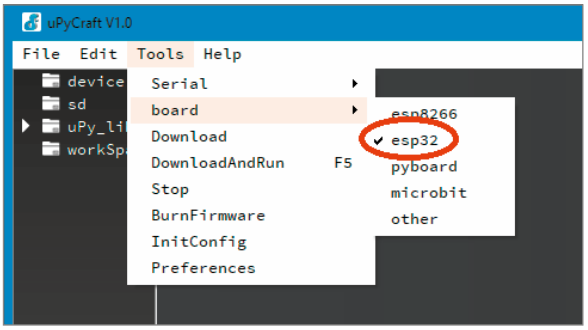

Bild 4: Auswahl des Boardtyps

Nun kann über
Tools → BurnFirmware
der MicroPython-Interpreter auf ESP32 geschrieben werden. Die passenden Optionen dazu lauten:
board: esp32
burn_addr: 0x1000
erase_flash: yes
COM: COMX (hier COM12, siehe oben)

Unter „Users" wird die heruntergeladene ESP32-BIN-Datei ausgewählt, dann auf „ok" geklickt.

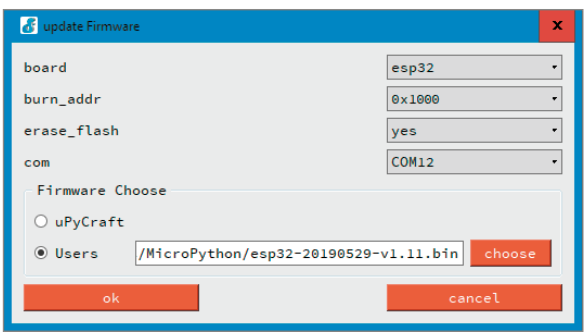

Bild 5: Die Parameter zum Flashen der Firmware

Wenn alle Einstellungen korrekt ausgewählt sind, wird die Taste "Boost" auf dem ESP32-Modul gedrückt – bei manchen Modulen (z. B. LOLIN D32) startet das Flashen automatisch. Sobald der Vorgang „EraseFlash" beginnt, kann die Taste losgelassen werden. Nach einigen Sekunden sollte die Firmware auf das ESP32-Board geflasht sein.

Falls die Anzeige „EraseFlash" nicht startet oder die Fehlermeldung „Erase false" angezeigt wird, ist es oftmals hilfreich, die beschriebenen Schritte zu wiederholen. Auch die Taste "Boost" ist erneut zu drücken, um sicherzustellen, dass der ESP32 in den Flashmodus wechselt.

Laden des ersten MicroPython-Programms

Bevor das erste Anwendungsprogramm auf den ESP geladen wird, sollte man sich mit der µPyCraft-IDE vertraut machen. Diese enthält, analog zu vielen anderen Programmiertools, die folgenden Unterfenster (siehe Bild 1):
1. Ordner und Dateien
2. Editor
3. MicroPython Shell/Terminal
4. Werkzeuge

Die einzelnen Fenster haben die folgenden Funktionen und Aufgaben:

① Ordner und Dateien
Der Geräteordner („device") zeigt die Dateien an, die derzeit auf dem ESP-Board gespeichert sind. Wenn das Board über eine serielle Verbindung mit uPyCraft_IDE verbunden ist, sollten beim Öffnen des Geräteordners alle gespeicherten Dateien geladen werden. Standardmäßig ist hier nur eine boot.py-Datei sichtbar.

Um den Anwendungscode auszuführen, wird empfohlen, eine main.py-Datei zu erstellen.

boot.py: wird ausgeführt, wenn das Board gestartet wird
main.py: Dies ist das Hauptskript, das den Anwendungscode enthält. Es wird unmittelbar nach boot.py ausgeführt.

Der SD-Ordner ist für den Zugriff auf Dateien vorgesehen, die auf SD-Karten gespeichert sind. Dieser ist nur auf Boards mit einem SD-Kartensteckplatz verfügbar.

In der uPy_lib werden die integrierten IDE-Bibliotheksdateien angezeigt.

Schließlich ist unter „workSpace" ein Verzeichnis zum Speichern von Anwendungsdateien vorhanden. Diese Dateien werden auf dem Computer in einem vom Anwender festgelegten Verzeichnis gespeichert. Hier können alle aktiven Dateien griffbereit abgelegt werden. Wenn µPyCraft zum ersten Mal verwendet wird, empfiehlt es sich daher, ein geeignetes Arbeitsverzeichnis zu erstellen.

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

③ MicroPython Shell/Terminal
Die in der MicroPython-Shell eingegebenen Befehle werden sofort vom ESP-Board ausgeführt. Das Terminal liefert auch Informationen über den Status eines laufenden Programms. Hier erscheinen zudem eventuelle Syntaxfehler im aktuellen Programm oder Fehlermeldungen beim Hochladen usw.

④ Werkzeuge
Mit den Symbolen ganz rechts 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

Starten des ersten Python-Programms

Um mit dem Erstellen eines Programms und dem Ausführen von Codes 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 hin sollte die Antwort
   Test
   >>>
   im Terminal-Fenster erscheinen.
   Wenn die Meldung ausgegeben wird, ist alles in Ordnung. Andernfalls muss geprüft werden, ob die serielle Kommunikation mit dem Board hergestellt ist und ob die MicroPython-Firmware erfolgreich auf das Board geflasht wurde.

Nun kann eine main.py-Datei auf dem Board erstellt werden:

1. Auf die Schaltfläche „New File" klicken, um eine neue Datei zu erstellen
2. Die neue Datei wird in main.py umbenannt (Schaltfläche Save) und auf dem Computer im vorher festgelegten Workspace gespeichert
3. Nun wird in der uPyCraft-IDE ein neuer Eintrag mit der Datei main.py angezeigt.

Jetzt kann das Blink-LED-Skript hochgeladen werden. Dazu wird

1. der Code aus der ersten Seite des Artikels in den Editor in der Datei main.py geschrieben
2. dann wird durch anklicken der Schaltfläche „Stop" ein eventuell noch laufendes Skript angehalten
3. Mit einem Klick auf die Schaltfläche „DownloadAndRun" wird das Skript auf den Controller geschrieben
4. Im Shell-Fenster sollte nun die Meldung „download ok" angezeigt werden

```
>>>

Ready to download this file,please wait!
..
download ok
exec(open('boot.py').read(),globals())
```

Bild 6: Erfolgreicher Download des ersten Programms

und die integrierte LED des ESP32-Boards sollte im Sekundentakt blinken.

Damit wurde das erste Python-Programm erfolgreich auf den Controller übertragen und unverzüglich ausgeführt.

Fehlerbehandlung

Insbesondere wenn man neu in die Arbeit mit µP einsteigt, können immer wieder Probleme und Fehlermeldungen auftauchen.

In vielen Fällen behebt ein Neustart des ESP-Controllers über die Tasten EN/RST (enable/reset) das Problem. Alternativ kann in der µPyCraft-IDE die Schaltfläche „Stop" angeklickt und die gewünschte Aktion wiederholt werden.

Falls dies nicht weiterhilft, liegt eventuell einer der folgenden Fehler vor.

Die Meldungen:

```
>>>
Select Serial Port could not open port ‚COM2': FileNotFoundError(2, ‚The system cannot find the file specified.', None, 2)
```

oder

```
>>>
could not open port ‚COM2': PermissionError(13, ‚A device attached to the system is not functioning.', None, 31)
```

deuten auf ein Problem mit der Kommunikationsschnittstelle hin. In diesem Fall sollte das ESP-Board kurz vom USB-Anschluss getrennt und anschließend wieder verbunden werden. Zudem sollte überprüft werden, ob im Menü

Extras → Serial

der richtige serielle Anschluss ausgewählt ist. Über „Verbinden/Trennen", kann die serielle Kommunikation neu gestartet werden.

Dieser Fehler kann auch bedeuten, dass der serielle Anschluss zum ESP in einem anderen Programm (wie z. B. einem seriellen Terminal oder der Arduino-IDE) verwendet wird. Daher sollten alle Programme geschlossen werden, die eventuell eine serielle Kommunikation zum ESP32 aufbauen.

Falls beim Laden eines neuen Skripts die Meldung

```
>>> already in download model, please wait.
```

erscheint, sollte in der uPyCraft-IDE die Schaltfläche „Stop" angeklickt werden. Damit wird der aktuell ausgeführte Code angehalten. Anschließend nochmals auf die Schaltfläche „DownloadAndRun" klicken, um das neue Skript auf den ESP32 zu laden.

Wenn nach dem Hochladen eines neuen Skripts die Meldung

```
>>>
Ready to download this file,please wait!
...
download ok
os.listdir(,.')
Traceback (most recent call last):
File „<stdin>“, line 1, in <module>
NameError: name ‚os‘ isn‘t defined
```

oder

```
>>>
Ready to download this file,please wait!
...
download ok
os.listdir(,.')
OSError: [Errno 98]
```

angezeigt wird, bedeutet dies, dass die neue Datei zwar erfolgreich auf das Board hochgeladen wurde, allerdings wurde das Programm nicht automatisch gestartet. In diesem Fall muss der Downloadmodus des angeschlossenen Boards durch Drücken der Taste EN (enable) oder RST (reset) manuell aktiviert werden.

Erscheint beim Neustart des ESP-Controllers, beim Ausführen eines neuen Skripts oder beim Öffnen der seriellen Schnittstelle die Meldung:

```
>>> Brownout detector was triggered
```

deutet dies auf ein Problem mit der Spannungsversorgung hin. Dies kann folgende Ursachen haben:

- USB-Kabel minderer Qualität haben oft einen zu hohen Kabel- oder Kontaktwiderstand
- Das USB-Kabel ist zu lang
- Auf der ESP32-Platine sind Defekte wie etwa schlechte Lötstellen vorhanden
- Der USB-Anschluss des verwendeten Computers oder Laptops ist fehlerhaft oder liefert zu wenig Strom. Insbesondere Laptops sind oft nicht in der Lage, ausreichend Strom für die zuverlässige Versorgung des ESP32-Boards zu liefern

Die Verwendung von möglichst kurzen USB-Kabeln hoher Qualität behebt hier oftmals das Problem. Auch ein hochwertiger USB-Hub mit einem externen Netzteil kann weiterhelfen. Probeweise sollte man das Board auch an einem anderen Rechner betreiben, um sicherzustellen, dass die Ursache nicht der USB-Port des Computers ist.

Es kommt immer wieder vor, dass beim Aufbau einer seriellen Kommunikation zwischen PC und ESP32 das Fenster „Update Firmware“ angezeigt wird. In diesem Fall wird eventuell ein Skript ausgeführt, das den Kommunikationsaufbau aufgrund hoher Prozessorauslastung verhindert. Hier kann es sinnvoll sein, zu versuchen den COM-Port mehrmals zu öffnen oder den ESP neu zu starten.

Insbesondere wenn eine Wi-Fi-Verbindung aktiv ist, ein Energiesparmodus verwendet wird oder mehrere Tasks parallel ausgeführt werden, empfiehlt es sich, den Verbindungsaufbau probeweise mehrfach zu starten. Falls alle Versuche fehlschlagen, kann es hilfreich sein, die neueste MicroPython-Firmware auf den Controller zu laden.

Ausblick

Nachdem in diesem Artikel erläutert wurde, wie man den ESP32-Controller mit MicroPython programmieren kann, soll im nächsten Beitrag auf verschiedene Anwendungen eingegangen werden. Dabei soll zunächst die Erfassung und Auswertung von Sensordaten im Vordergrund stehen.

Der ESP32-Chip verfügt über zwei unabhängige Analog-Digital-Wandler (ADC), die über Multiplexer bis zu 18 verschiedene Analogsignale simultan erfassen können.

Aber auch Sensoren mit digitalen Ausgängen können über unterschiedliche Bussysteme wie I2C oder One-Wire ausgewertet werden. Kommt noch ein einfaches Display hinzu, entsteht schnell und ohne großen Aufwand ein universell einsetzbares Messsystem, das in der Elektronik-Werkstatt oder in der Hausautomatisierung verschiedene Aufgaben übernehmen kann. ELV

Weitere Infos:

[1] MicroPython: https://www.micropython.org
[2] SciKit : https://scikit-learn.org/stable/
[3] NumPy: https://numpy.org/
[4] SciPy: https://www.scipy.org/
[5] µPyCraft-IDE: http://docs.dfrobot.com/upycraft/
[6] Thonny-IDE: https://thonny.org/
[7] JOY-iT Entwicklungsplatine NodeMCU mit ESP32, Bestell-Nr. 14 51 64